technische universität
dortmund

Endbericht

der Projektgruppe CordiAAL

Autoren: Imen Amdouni, Israfil Akman, Andreas Bauer, Rafael Bielen, Henning Brümmer
Ayse Gül Kilic, Artur Krutzek, Egor Kudrjaschow, Stefan Rhein, Andreas Schmidt

Betreuer: Prof. Dr. Heiko Krumm
Dipl.-Inf. Oliver Dohndorf

Datum: 3. Februar 2014

Technische Universität Dortmund
Fakultät für Informatik
Lehrstuhl für praktische Informatik (LS 4)
Rechnernetze und verteilte Systeme (RvS)
http://ls4-www.cs.tu-dortmund.de

In Kooperation mit:
Schüchtermann-Schiller'sche Kliniken
Bad Rothenfelde GmbH & Co. KG

Inhaltsverzeichnis

1. Einleitung

Noch immer gehören Herz-Kreislauf-Erkrankungen, wie Bluthochdruck, Schlaganfall und Herzinfarkt in den Industrieländern zu den häufigsten Erkrankungen. Auch in Deutschland sind sehr viele Todesfälle auf eine Erkrankung des Herzens bzw. des Gefäßsystems zurückzuführen. Am häufigsten sind ältere Menschen im hohen Erwachsenenalter betroffen. Aber auch die Zahl an Erkrankungen im Alter unter 50 Jahren ist in den letzten Jahren gestiegen. Aus diesem Grund werden die möglichen Ursachen aktuell noch in der Gesundheitsvorsorge analysiert. Die Störungen des Herz-Kreislauf-Systems sind unterschiedlich und verlaufen meistens sukzessiv. Zu möglichen Ursachen einer Herz-Kreislauferkrankung zählen Stressfaktoren, ungesunde Ernährung, Rauchen sowie mangelnde Körperbewegung. Wie die Studie Procram aus dem Jahre 1978 bis 2007 am Institut für Arteriosklerosefor-schung an der Universität Münster von Prof. Dr. G. Assmann gezeigt hat, ist Übergewicht ein zusätzlicher Risikofaktor für die oben genannte Erkrankung. Zentrale Ursache für das erhöhte kardiovaskuläre Risiko bei Übergewicht sind die damit verbundenen Faktoren wie z.B. Bluthochdruck, erhöhter Blutzucker und LDL-Cholesterin, welche z.B. durch sportbedingte Gewichtsabnahme zur Verbesserungen führen kann.
Vgl. `http://www.assmann-stiftung.de/procam-studie/`

Wichtig für eine frühe bzw. präventive Diagnose zur Vermeidung von lebensbedrohenden Krankheiten ist es somit, die Patienten vor allem für körperliche Aktivitäten zu motivieren, um so die Risikofaktoren der Erkrankungen zu reduzieren und vorzubeugen.

Aus diesem Grund hat sich die Projektgruppe CordiAAL das Ziel gesetzt, ein Konzept für die Realisierung eines erfolgversprechenden ambienten Systems für Patienten zu entwickeln und umzusetzen. Die Realisierung findet mit Hilfe eines Ergometertrainings in einer virtuellen Umgebung statt. Dies wird mit Hilfe einer 3D-Brille, einem modifizierten Ergometer und einer Vielzahl von Sensoren ermöglicht. Um die Motivation der Patienten zusätzlich zu steigern, besteht auch die Möglichkeit eines virtuellen Gruppentrainings. Dabei wird eine reale Fahrradtour mit bis zu 7 Patienten in einer frei wählbaren virtuellen Strecke simuliert. Eine ausgeklügelte Lastensteuerung sorgt dafür, dass die Patienten ihr Training immer mit der optimalen Herzfrequenz absolvieren können. Die Steuerung kann die Last des Ergometers beeinflussen, so dass es beispielsweise durch Verminderung der Last den Patienten vor Überbelastung schützen kann.

Über die Webplattform RehaWeb können Kardiologen die Vitalparameter für ihre Patienten verwalten, welche als Grundlage für das virtuelle Training dienen. Die Sensoren, welche beim Training aktiv sind, zeichnen nicht nur Daten auf um sie später analysieren zu können, sondern sie dienen auch dazu über die Lastensteuerung direkt während des Trainings auf den Patienten Einfluss zu nehmen zu können. Der Trainierende selbst bekommt ebenfalls seine Vitaldaten auf dem Bildschirm angezeigt. Anhand einer Statusleiste, welche seine aktuelle Herzfrequenz anzeigt, kann er live verfolgen ob er sein Training in dem für ihn optimalen Bereich absolviert. RehaWeb mit seinen Community-Funktionen dient ebenfalls zur Unterstützung der Gesundheitsvorsorge, indem sich Patienten für ein Training anmelden, ihre Statistiken einsehen und mit anderen Teilnehmen Erfahrungen austauschen können.

In Zusammenarbeit mit der Schüchtermann-Schiller'schen Klinik hat die Projektgruppe bei medizinischen Fragen Unterstützung bekommen. Die Schüchtermann-Schiller'schen Klinik gehört zu den fünf größten und modernsten Herzzentren in Deutschland. Sie ermöglicht als integriertes Herzzentrum eine umfassende Versorgung für Patienten. Dabei besitzt die Klinik eigene Abteilungen in den Bereichen für Kardiologie, Herzchirurgie, Anästhesiologie und Rehabilitation. Diese Abteilungen werden jeweils von dem Institut für Prävention und Sportmedizin sowie die Sportwanderwochen unterstützt.

1.1. Aufbau der Arbeit

Der Aufbau der Projektgruppe CordiAAL ist durch seine strukturelle sowie inhaltliche Gestaltung gegliedert, welche zur Erreichung der zuvor formulierten Zielsetzung führt. Die Gliederung des Endberichts und die Inhalte der einzelnen Kapitel werden im Folgenden erläutert.

Im folgenden Kapitel(Kapitel 2) wird der zeitliche Ablauf und das Vorgehensmodell der Projektgruppe erläutert. Hier werden auch die organisatorischen Hilfsmittel vorgestellt, welche wir während des Projekts zur Kommunikation und Datenaustausch verwendet haben. Kapitel 3 gibt einen Überblick über die Grundlagen der verwendeten Technologien. Das Representational State Transfer (REST) ist ein Architekturstil mit dem Webservices realisiert werden. Der Begriff wurde erstmalig in der Dissertation von Dr. Thomas Roy Fielding im Jahr 2000 in Erwähnung gebracht. Das REST-Architekturmodell beschreibt und dient als Referenz für zukünftige Erweiterungen des WWW. Die Technologie Unity3D die für das Projekt eingesetzt wurde, ist eine integrierte Entwicklungsumgebung, die für das 3D-Grafik Anwendungen zuständig ist. Es wurde so ermöglicht, die virtuelle Welt zu modellieren und zu implementieren. Die Anforderungen die für das Projekt gestellt wurden, werden in Kapitel 4 vorgestellt und durch das Pflichtenheft näher erläutert. Kapitel 5 um-

fasst den Entwurf des Projektes. Zuerst wird die Grobarchitektur beschrieben und näher erläutert. Hier wird insbesondere das Zusammenspiel der einzelnen verwendeten Technologien, wie z.B. Tomcat, MySQL, Apache, Java Servlet und UDP Server beschrieben und erklärt. Die UDP Schnittstellen zum UDP Server und die Restschnittstellen werden hier erläutert. Die benötigten und angewendeten Sensoren für das Projekt und die jeweiligen Anbindungen dieser werden ebenfalls in diesem Kapitel erarbeitet. Hierbei wird das Zusammenspiel und die Funktionalitäten der Sensoren zum Beispiel des SPO2 Sensor und ebenso die 3D Brille behandelt. Dabei wird auch die Funktionalität des Ergometers, welches mit einem modifizierten Lenkrad und einem Touch-Display ausgestattet ist beschrieben. Die Implementierungsphase und die dafür benötigten Werkzeuge wie Unity3D, Eclipse und der Quellcode werden in Kapitel 6 vorgestellt. In Kapitel 7 wird die Erprobung beschrieben. Hier wird der Funktionstest bzw. die Probleme, welche bei der Durchführung aufgetreten sind aufgezeigt. Eine Zusammenfassung der Projektgruppe sowie die wichtigsten Erkenntnisse ist Gegenstand von Kapitel 8. Hier wird unter Berücksichtigung der in dieser Arbeit gewonnenen Erkenntnisse, die anfänglich formulierte Zielsetzung der Arbeit noch einmal reflektiert. Das Fazit am Ende dieses Berichts führt Möglichkeiten für die weitere Nutzung unserer Erkenntnisse aus diesem Projekt auf.

2. Organisation

Bevor man mit der Implementierung der Anforderungen starten konnte, wurden alle Teilnehmer der Gruppe in ihrem Wissenstand angegelichen. Dazu eigneten sich die Seminar- und Praktikumsphasen hervorragend.

2.1. Zeitlicher Ablauf und Vorgehensmodell

Innerhalb der ersten zwei Wochen haben die Teilnehmer Informationen zu ihren Themengebieten recherchiert und sich gründlich in diese eingearbeitet, sodass sie im Anschluss den anderen Teilnehmern der Gruppe didaktisch aufbereitete Informationen zu diesem Themengebiet zur Verfügung gestellt haben. Zudem wurden zu den einzelnen Themen knifflige Aufgaben für alle Gruppenteilnehmer bereit gestellt. Die Bearbeitung der Aufgaben fand in den drei bis vier nachfolgenden Wochen statt. Die Bearbeitungszeit der Aufgaben war auf zwei Stunden beschränkt. Aufgrund der Eigenständigkeit der Praktikumsphase und der parallel stattfindenden Gruppentreffen, in denen bereits die Grobarchitektur der Software besprochen wurde, war es nicht erforderlich die Aufgaben und deren Lösungen durch einzelne zu präsentieren.

2.1.1. Seminarphase

In der Seminarphase wurde jedem Gruppenmitglied ein Thema zugewiesen, zu dem eine Präsentation gehalten werden musste. Die Ergebnisse wurden in einer schriftlichen Ausarbeitung dokumentiert. Außerdem wurden Technologien untersucht, die ggfs zum Erreichen des Projektziels verwendet werden konnten. Durch Aufgaben, die zu ausgewählten Technologien erstellt und von jedem Gruppenmitglied bearbeitet wurden, wurde das gewonnene Wissen gefestigt und vertieft. Folgende Aufgaben wurden dann in der Praktikumsphase von den PG-Mitgliedern bearbeitet:

- Android OS
- Web Services (REST)
- OSGi
- IDE Eclipse mit CVS

- 3D Grafikdarstellung
- Google-API

2.1.2. Praktikumsphase

Die Praktikumsphase bildet eine von mehreren Phasen während der Projektgruppenzeit. Sie beginnt im Anschluss an die Seminarphase. Der Sinn dieser Phase liegt darin, die Teilnehmer der Gruppe einerseits auf einen gemeinsamen Wissensstand zu bringen und andererseits bereits gewonnene Kenntnisse zu vertiefen. Die Teilnehmer arbeiten in kleinen Gruppen eigenständig an verschiedenen Aufgaben, die zu den Themen aus der Seminarphase gestellt wurden. Hier wurden insbesondere Themen wie Web Services und 3D Grafikdarstellung vertieft.

2.1.3. Spiralmodell

Bei der Softwareentwicklung haben wir uns für das Spiralmodell entschieden, weil es den Vorteil der Flexibilität bietet. Veränderungen der Anforderungen können während der Entwicklung umgesetzt werden. Das Spiralmodell bietet mit seiner Aufteilung in Festlegung der Ziele, Bewertung der Alternativen, Erstellung eines Prototypen und Planung der Projektfortsetzung die Möglichkeit, immer wieder in einen der Bereiche zu kommen. Zu den Nachteilen zählen wir bei dem Spiralmodell, dass die Verantwortlichen der einzelnen Aufgaben nicht bekannt sind sowie dass parallele Aktivitäten ausgeschlossen werden.

2.1.4. Die vier Phasen des Spiralmodells

1. Ziele	Hier werden die Ziele für das Projekt definiert
1. Risiken	Hier werden die Projektrisiken analysiert und minimiert
2. Entwicklung	Erstellung eines Prototyps
3. Planung	Pläne für die nächste Phase oder Anpassung der Ziele

2.1.5. Organisatorische Hilfsmittel

Während des letzten Jahres wurde zur Unterstützung, Überwachung der Aufgaben, Einhaltung der Meilensteine, Dokumentation und Verwaltung das onlinebasierte Trac-System genutzt.
Trac ist in Python geschrieben und beinhaltet neben Wiki auch noch Subversion und Ticketsystem. Mit dem Ticketsystem kann man leicht die Bug-Fixes verfolgen und Meilensteine definieren. Zudem ermöglicht es die Verbindung mit Subversion, einem Projektmitglied die Veränderungen im Quellcode zu sehen. Gleichzeitig kann das Wiki dazu genutzt werden, Information zu sammeln und an einem Ort gebündelt anzubieten.

3. Grundlagen

Die grundsätzlichen verwendeten Technologien wie z. B. Webservice REST zur Kommunikation zwischen der Webplattform und der Anwendungssoftware oder Unity 3D zur Modellierung von 3D-Objekten in der Anwendungssoftware werden hier vorgestellt.

3.1. REST

REST ist ein Architekturstil, mit dem man Webservice anbieten kann. REST ist kein Standard wie SAOP oder WSDL. Eine REST-Anwendung bietet über das HTTP Protokoll im Web Ressourcen an. Die Ausarbeitung RESTful services, im Anhang zu finden, verdeutlicht erste Eigenschaften von REST und befasst sich mit Themen wie Anwendbarkeit, Ressourcen, Sicherheit und Skalierbarkeit. Hier wird nur kurz erwähnt, dass REST die 4 Grundfunktionen des World Wide Webs unterstützt (siehe Abbildung A.12)

Die da wären:

- GET – Mit der GET-Abfrage kann der Client eine Repräsentation von einem Inhalt, unter REST ist damit die Ressource gemeint, anfordern. Mit einer GET-Anfrage kann an dem Inhalt selbst nichts verändert werden.
- POST – Mit der POST-Abfrage kann man eine Ressource zu einer anderen hinzufügen. Damit ist es möglich in einem Onlineshop einen Artikel in den Warenkorb zu legen oder in unserem Projekt in einem Profil ein neues Bild hoch zu laden.
- PUT – Mit PUT ist es möglich neue Ressourcen anzulegen oder bestehende zu bearbeiten. So können Patienten ihre Profile aktualisieren.
- DELETE – Mit der DELETE-Anfrage ist es möglich einzelne Ressourcen zu entfernen.

3.2. Unity3D

Unity ist eine Spiel-Engine die das Entwickeln von Computerspielen und 3D-Grafik Anwendungen erlaubt. Es ist möglich Anwendungen für PCs, Xbox 360, Playstation 3, Wii, Andorid und iOS zu erstellen. Die Entwicklungsumgebung wurde ursprünglich für Mac OS entwickelt aber inzwischen auch für Windows portiert. Unity besteht aus einem Editor und den C# Editor MonoDevelope.

3.2.1. Entwicklungsumgebung

Die Entwicklungsumgebung besteht aus 3 Fenster wichtigen Fenstern, 3D-Szene, Ressourcen-Verwaltung und Komponenten-Verwaltung. In der 3D-Szene können 3D Objekte mithilfe von „drag-and-drop“ eingefügt, skaliert und positioniert werden. Die Ressourcen-Verwaltung enthält alle Ressourcen wie 3D Objekte, Musik-Dateien oder Bilder die im Projekt benutzt werden. Die Ressourcen können kopiert, neu hinzugefügt, unbenannt und entfernt werden. Unity unterstützt das Konzept von Komponenten. Komponenten sind C#-Klassen die an ein Objekt in der Scene eingefügt werden und Ihm so Eigenschaften verleihen. Im Fenster Komponenten-Verwalten könnten Komponenten hinzugefügt, bearbeitet und entfernt werden.

3.2.2. Technische Eigenschaften

Die grafische Darstellung erfolgt je nach Zielplaftorm in Direct3D oder OpenGl, Unity unterstützt neben Forward-Rendering auch Deffered-Rendering und andere moderene Effekte wie Bumpmapping, Environment Mapping, Parallax Mapping, Umgebungsverdeckung, dynamische Render-To-Texture und Vollbild-Postprocessing Effekte. Eigene Effekt mithilfe von Shadern können auch erstellt werden. Zum Abspielen von Musik und Sound verwendete Unity FMOD, es wird Raumklang und Mehrkanalton unterstützt. Als Programmiersprachen können JavaScript, C# und Boo in Unity benutzt werden. Alle Sprachen sind gleichmächtig und benutzen die gleichen Bibliotheken. Des Weiteren hat Unity noch zusätzliche Werkzeuge im Editor eingebaut: Terrain Modellierer, Baum Pflanzen Editor, Werkzeuge für Explosionen und Bewegungssteuerung für Charaktere Zusammenenarbeit im Netz.

3.3. JSON

Der Datenaustausch zwischen den Anwendungen findet im JSON-Format (JavaScript Object Notation) statt. Genau wie XML ist dieses Datenformat auch für Mensch und Maschine in lesbarer Textform gehalten. Dies ermöglichte es uns auch die Korrektheit der zu sendenden Daten einfach zu überprüfen.
Ein Vorteil gegenüber XML ist, dass bei diesem Format ist der reduzierte erzeugte Overhead, was gerade bei dem Versenden unserer Trainingsstatistiken nicht zu verachten ist.

4. Anforderungen

Um in der Primär- und Sekundärprävention wirksam zu sein und um allgemein das beschriebene Projekt zu ermöglichen müssen gewisse Anforderungen erfüllt werden. Hier wird auf den Anwendungs-Client und auf die *RehaWeb*-Anwendung eingegangen. Detaillierte Informationen, insbesondere zu anderen Anforderungen findet man im Pflichtenheft.

4.1. Anwendungs-Client und Sensoren

Die Sensoren sind in folgenden Geräten integriert:

- Zephyr BioHarness 3.0
- Oculus Rift Virtual-Reality-Brille
- Fahrradergometer
  - Pedale
  - Lenker (nach Modifikation nach links und rechts beweglich)

4.1.1. Verwendete Daten

Sowohl für die Selbstkontrolle des Patienten als auch für eine Visualisierung der Vitaldaten für Arzt und Patient sind die Vitaldaten vom Zepyhr BioHarness 3.0 an den Anwendungs-Client zu übermitteln sowie von diesem Client nach Abschluss des Trainings an den Anwendungs-Server zu senden. Die Herzfrequenz muss im Client verwendet werden, um die Last des Ergomters zu setzen, damit das Training medizinisch Sinn ergibt. Die aktuelle Herzfrequenz des Patienten ist dazu mit den Grenzwerten zu vergleichen, sodass weder Unter- noch Überbelastung auftreten.
Die in CordiAAL verwendeten Daten sind folgende:

- Herzfrequenz (BioHarness)
- Atemfrequenz (BioHarness)
- Körpertemperatur (BioHarness)
- MET-Wert (berechnet aus Wattzahl und Körpergewicht des Patienten)

- Borg-Wert (am Ende des Trainings vom Patienten eingegeben)

Nur die Herzfrequenz wird von der Client-Anwendung tatsächlich für das virtuelle Training gebraucht, die übrigen Daten sind für die Statistiken in RehaWeb. Die Sauerstofsättigung (SpO2) wird im Projekt überhaupt nicht benutzt (siehe 5.4.2).

4.1.2. Fahrradergometer

Pedale Neben einer Vorwärtsbewegung des virtuellen Fahrrads, die durch Betätigung der Pedale des Fahradergometers geschieht, muss auch sichergestellt werden, dass eine Regelung mit dem Ergometer funktioniert. Hierzu muss die aktuelle Herzfrequenz des Patienten im Client verwendet werden, um die Last des Ergometers zu setzen. Dies wird mit einem Regelkreislauf sichergestellt.
Innovativ soll sein, dass während eines Gruppentrainings Patienten mit unterschiedlichen Gesundheitszuständen (und damit jeweils anderen Grenzwerten für ihre Vitaldaten) miteinander trainieren können derart, dass kein Patient innerhalb einer virtuellen Gruppe in der virtuellen Welt verlassen muss. Beispielsweise heißt das für einen Patienten mit schlechtem Gesundheitszutand, dass die Last des Ergometers bei Erreichen seines oberen Grenzwertes für die Herzfrequenz reduziert wird. Das führt dazu, dass die Geschwindigkeit seines virtuellen Fahrrads abnimmt und dieser Patient nach gewisser Zeit von den anderen Patienten in der virtuellen Welt abgehängt werden kann. Dieser Effekt wird in CordiAAL verhindert, indem bei Lastreduktion eines in seinen Grenzbereich eingetretenen Patienten die Tretfrequenz (RPM) erhalten werden kann. Die Geschwindigkeit des virtuellen Fahrrads kann tatsächlich beibehalten werden, da in CordiAAL die RPM benutzt werden, um die Geschwindigkeit zu in der virtuellen Welt vorzugeben.

Lenkung Damit auf dem virtuellen Fahrrad in der virtuellen Landschaft möglichst real gefahren werden kann, muss man das virtuelle Fahrrad nach links und rechts lenken können. Für diesen Zweck wurde der Lenker Fahrradergometers so umgebaut, dass dies mit Hilfe von Sensoren möglich ist. Lenkt man auf dem Fahrradergometer in eine Richtung, so wird das virtuelle Fahrrad in die entsprechende Richtung gelenkt.

4.1.3. Oculus Rift Virtual-Reality-Brille

Um eine reale Fahrradtour im Freien zu imitieren, bedarf es nicht nur eines Fahrradergometers, dessen Lenker zusätzlich beweglich ist, sondern es muss auch das Visuelle real erscheinen. Für diesen Zweck sollte eine Virtual-Reality-Brille (auch: 3D-Brille) benutzt werden, welche in unserem Fall die Oculus Rift (Entwicklerversion) ist. Des Weiteren muss bei Drehung des Kopfes das Bild auf den Bildschirmen der 3D-Brille angepasst werden. Dieses so genannte Head-Tracking wird mittels der Bewegungssensoren der Brille gewährleistet.

4.2. RehaWeb-Anwendung

Die Erweiterung der Community RehaWeb muss, angepasst an das Projekt CordiAAL, folgende Anforderungen erfüllen:

- Verabredung mit anderen Patienten für ein Gruppentraining
- Kommunikation der Patienten untereinander
- Einsicht eines Patienten in seine Statistiken
- Einsicht des Arztes in die Statistiken
- Setzen der Vitalparameter eines Patienten durch den Arzt

4.2.1. Verabredung für ein Gruppentraining

Für die Patienten muss es möglich sein, sich über die Community RehaWeb für ein gemeinsames Gruppentraining zu verabreden.

4.2.2. RehaWeb als Kommunikationsplattform

Weiterhin soll die Möglichkeit für Patienten bestehen sich mit anderen Patienten in der Community zu unterhalten. Dies ist in CordiAAL in einem Forum und mit privaten Nachrichten möglich, sodass sich Patienten beispielsweise über ihren derzeitigen Gesundheits- bzw. Trainingszustand unterhalten können.

4.2.3. Einsicht in Statistiken

Sowohl der Patient als auch der der Arzt müssen Einsicht in Statistiken über Trainings haben können. In CordiAAL ist der Patient autorisiert seine eigene Trainingsstatistiken (textuell) zu sehen. Der Arzt jedoch kann jede Statistik sehen und dies in detaillierter, grafischer Form.

4.2.4. Setzen von Vitalparametern

Dem Arzt muss das Setzen von Vitalparametern, die beim Training eines Patienten nötig sind (z.B. die maximale Trainingsherzfrequenz), ermöglicht werden. In CordiAAL ist zu diesem Zweck eine Eingabemaske vorhanden, in der Werte für Herzfrequenz, Atemfrequenz, Körpertemperatur und Sauerstoffsättigung (SpO2) vom Arzt eingegeben werden können.

5. Entwurf

Bevor mit der Implementierung der Software begonnen werden konnte, stellte sich die Gruppe der Herausforderung, einen Entwurf für die Gesamtarchitektur für die Software zu entwickeln.

5.1. Grobarchitektur

Die Architektur der Projektgruppe CordiAAL unterteilt sich in vier Komponenten. Diese Komponente sind:

- RehaWeb-Anwendung
- Anwendungs-Client
- Anwendungs-Server
- Datenbank

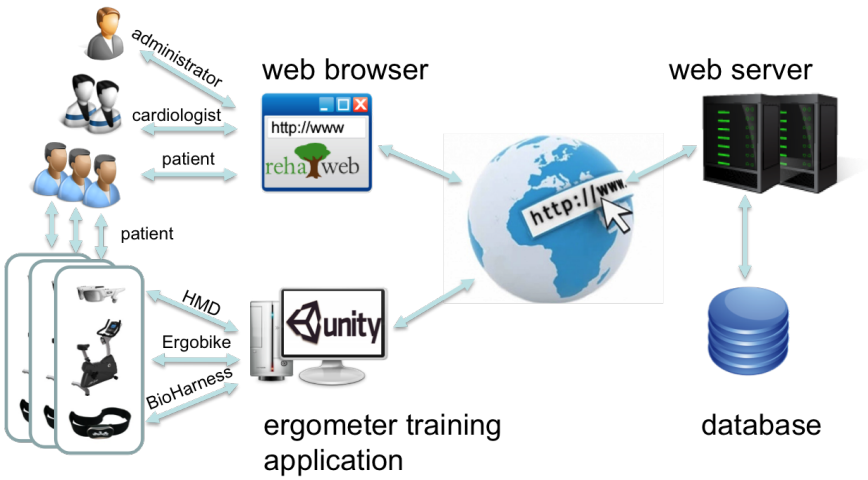

Abbildung 5.1.: Architektur

Die Abbildung 5.1 zeigt die Gesamtarchitektur von CordiAAL.

Die RehaWeb-Homepage ist ein Community, die für Patienten eine Internetplattform bietet, in der sie sich zuerst registrieren und dann ein Profil erstellen können. Bei der Erstellung eines Profils stehen dem Patienten diverse Funktionalitäten zur Verfügung, wie

zum Beispiel die Möglichkeit einen Trainingsplan zu errichten, bei der sie ebenso Einzeltrainings oder aber auch Gruppentrainings erstellen können. Bei dem Gruppentraining können sich die Patienten über das Internet zu einem gemeinsamen virtuellen Wandertraining verabreden. Dabei kann der Arzt z.B. Patientendaten einsehen, Grenzwerte für die Patienten über das Webinterface setzen und Statistiken auswerten. Die Kommunikation von RehaWeb mit dem Anwendungs-Server erfolgt über eine REST-Schnittstelle bzw. HTTP. Der Anwendungs-Client involviert die für jeden einzelnen Patienten vorgesehene Dienste, mit der das Training durchgeführt werden kann. Die Software Unity3D läuft auf dem Betriebssystem Microsoft Windows. Der Client bzw. der Patient meldet sich bei der Unity3D an, daraufhin kann er sich Trainingseinstellungen vornehmen und das eigentliche Training starten. An den Client ist der Fahrradergometer und die 3D-Brille angeschlossen. Der Brustgürtel stellt dagegen eine Verbindung per Bluetooth her. Mittels einer REST-Schnittstelle werden die Funktionalitäten der Trainingsvorbereitung und des Trainingsendes zum Anwendungs-Server vorgenommen. Die Kommunikation bzw. der Trainingsablauf findet über UDP statt. Dabei werden beispielsweise die Positionsangaben vom Client zum Server gesendet, sodass dieser seinerseits diese Daten an die anderen Gruppentraining teilnehmenden Clients senden kann. Daraus folgt, dass die aktuelle Position eines Patientens in der virtuellen Welt auf den anderen Clients dargestellt werden. Die gesamte Kommunikation läuft über den Anwendungs-Server zwischen den verschiedenen Clients. Darüber hinaus verwaltet der Server die gesamte Trainings-Session und leitet die zu speichernden Daten an die an ihn angeschlossene Datenbank weiter. Auf dem Server ist als Betriebssystem Linus installiert. Der Administrator übernimmt die Verwaltung des gesamten Systems CordiALL.

5.2. Server

Am Anfang der Softwareentwicklung wurde von den PG-Mitgliedern ein Webserver eingerichtet. Es gibt viele Webserverlösungen, doch die am meisten verbreiteten sind Apache Webserver und IIS. Während die Apache Lösung, von der Apache Software Foundation, für die meisten frei verfügbar ist, muss man bei der IIS-Lösung von Microsoft für die Lizenzen bezahlen. Daher fiel die Entscheidung in der Auswahl der Webserveranwendung leicht und schnell zu Gunsten von Apache. Apache ist modular aufgebaut und kann leicht durch zusätzliche Module erweitert werden. Ob PERL, PHP, Datenbank-Konnektor oder andere Weberweiterungen, alles kann über ein Modul leicht integriert werden. Zur Unterstützung von Apache 2.2 wurde auf dem Server neben Tomcat 6.0, MySQL 5.5, PHP 5.3 und eine JAVA-Basierte Datenbank namens Derby installiert. Apache 2.2 bietet unter anderem die Möglichkeit, statische Webseiten aus dem Internet von Verteilten-Systemen abzurufen. Hierzu kommt, dass Apache ab der Version 2.0 ein Multiprozessing-Modul anbietet. Dieses ermöglicht eine gleichzeitige Bedienung mehrerer Clientanfragen. Tomcat

6.0 ist ein „open source Applications-Server“ und wird auf dem Server eingesetzt, um eine dynamische Web-Plattform namens Reha-Web für Patienten, Ärzte und Administratoren zu bieten. Zudem verwendet Tomcat die Java Servlet and JavaServer Pages Technologien. Reha-Web wurde bereits so entwickelt, dass der Einsatz von verschiedenen Datenbanken möglich ist. Ursprünglich lief die Plattform auf ihrem eigenen Server unter der JAVA-Basierten Datenbank, Derby. Die PG-Gruppe hatte sich hier dazu entschlossen MySQL einzusetzen, das den Vorteil des Webzugriffs mit sich brachte. Mit PHPMYADMIN ist es möglich mit einem beliebigen Browser auf die MySQL Datenbank zugreifen zu können. Der Vorteil hierbei liegt klar auf der Hand: man muss weniger SSH-Konten für User erstellen, sondern lediglich nur ein paar MySQL-User, die nur Leserechte besitzen. Derby Datenbank Software, die ebenfalls von der Apache Software Foundation in JAVA entwickelt wurde, ist im Gegenteil zu anderen Datenbanken sehr leicht. Sie ist nach der Installation nur wenige Megabyte groß. Da sie in Java geschrieben ist, ist die Datenbanksoftware unabhängig und läuft unter allen Plattformen, die auch JAVA unterstützen. Zudem unterstützt Derby die SQL und JDBC Standards.

Anwendungsserver

Neben dem Webserver, welcher für die RehaWeb Anwendung sowie die REST-Schnittstelle verantwortlich ist, benötigen wir noch einen Anwendungsserver. Dieser hat zum einen die Aufgabe ein Echtzeitverhalten im Multiplayerbereich zu gewährleisten, sowie eine Echtzeitkommunikation zwischen Patienten in einem Training bereitzustellen. Der Anwendungsserver ist in Java implementiert und Kommuniziert über UDP-Nachrichten mit dem Anwendungsclient. Im folgenden werden der Positionserver als auch der Voicechatserver näher erläutert.

Positionserver

Ziel unserer Arbeit war es, eine Motivationssteigerung durch das Trainieren in einer virtuellen Gruppe zu erreichen. Um diese Eigenschaft zu erreichen, muss das System Realzeitbedingungen aufweisen. Des Weiteren interagieren die einzelnen Anwendungen nicht direkt miteinander, sondern unter Nutzung des Positionservers. Das verteilte System ist in Abbildung 5.2 abgebildet. Die 3D-Welt ist weiterhin statisch auf jedem Rechner geladen, nur die Anzeige der Mitspieler findet dynamisch über die Informationen der einzelnen Mitspieler in einer Gruppe statt. Damit diese Informationen in Echtzeit am Positionserver anliegen, wird jede Sekunde ein UDP-Datenpaket wie in Abbildung 5.3 abgebildet an den Server geschickt.

Kommt eine solche Nachricht am Positionserver an, wird über die Gruppen ID im Datenpacket überprüft, ob diese Gruppe schon am Server anliegt. Ist dies nicht der Fall, also eine noch unbekannte neue Trainingsgruppe wurde geöffnet, wird eine solche Gruppe

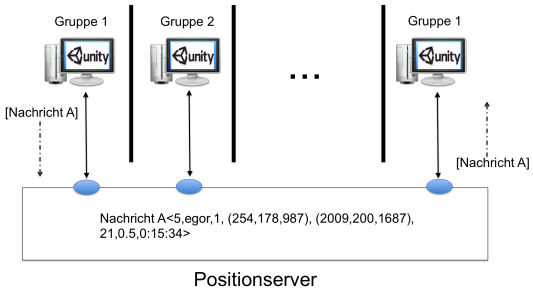

Abbildung 5.2.: Verteiltes System

Abbildung 5.3.: UDP-Datenpackt

am Server angelegt und sämtliche Daten aus unserer UDP-Nachricht dort abgespeichert. Besteht diese Gruppe aber schon, müssen wir prüfen ob dieser Patient schon am Server anliegt oder neu ins Training eingestiegen ist. Ist der Patient schon am Server bekannt, sendet der Positionserver diesem Patienten sämtliche aktuelle Daten der anderen Patienten im Training über UDP zu. Treten wir einem schon bestehenden Training bei und die Patienten ID liegt noch nicht am Server an, werden alle Daten vom Patienten aus der Nachricht in der Gruppe abgespeichert, und die aktuellen Daten sämtlicher Mitspieler werden über UDP-Pakete zugesendet. Der Ablauf des Positionserver ist in Abbildung 5.4 abgebildet.

Voicechatserver

Neben dem Positionserver benötigen wir noch einen weiteren Anwendungsserver für die Echtzeitkommunikation der Patienten in einer Trainingsgruppe. Dafür entwarfen wir den Voicechatserver, dieser kommuniziert ebenfalls über UDP-Nachrichten mit dem Anwendungsclient. Die Idee ist ähnlich wie beim Positionserver. Auf dem Voicechatserver werden die Gruppen und deren Patienten verwaltet. Zwischen diesen Patienten in einer Gruppe werden die UDP-Datenpakete versendet, siehe Abbildung 5.5.

Zu beginn eines Gruppentrainings wird eine Registrierung/Anmeldung über die Gruppen ID am Server getätigt. Dieser legt, so fern eine solche Gruppe noch nicht existiert, eine solche Gruppe an und speichert den Absender anhand der IP-Adresse ab. Alle 100ms werden nun UDP-Datenpacket mit den Sprachsamples an den Server versendet. Der Server leitet diese Pakete nun einfach an alle sich in dieser Gruppe befindlichen Patienten über die IP-Adresse weiter. Am Ende eines Trainings werden die Patienten aus der Gruppe entfernt und schließlich die Gruppe gelöscht. Dieser Ablauf ist in Abbildung 5.6 skizziert.

5.3. Client (Unity3D)

5.3.1. Grobarchitektur des Clients

Für die Realisierung der Software haben wir das Model View Controller (MVC) Muster verwendet. Dabei wird es in drei Komponenten unterteilt: in Verarbeitung(engl. model), Ausgabe (engl. view) und Eingabe (engl. controller). Das Model repräsentiert und enthält

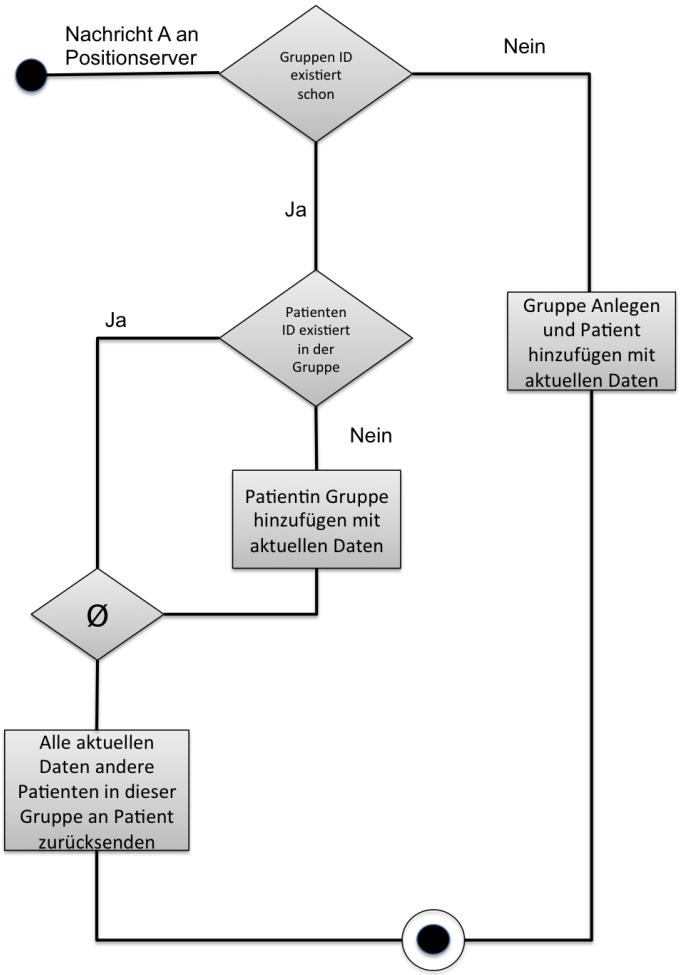

Abbildung 5.4.: Anwendungsfall ankommende Nachricht

die Daten einer Anwendung. Die View Komponente sorgt für die grafische Darstellung der Daten. Sie ist der sichtbare Teil der GUI-Komponente im Modell View Controller-Muster. Der Controller verwaltet die Benutzeraktionen, indem er sie verarbeitet und die Daten entsprechend an die Model und View Komponenten weiterleitet. In der Abbildung 5.7 ist unser MVC-Modell dargestellt.

Wir haben die Software in drei große Komponenten unterteilt. Die Unterteilung besteht aus einen SingelPlayer, MultiPlayer und der Statistic. Angefangen haben wir damit den SingelPlayer zu implementieren. Der Lastenkontroller, Vitalkontroller und der Movekontroller wurden zusammen mit dem Playerkontroller eingebunden. Der Lastenkontroller steht für die Lastensteuerung, mit welcher wir das Ergometer und durch die Software angepasst haben. Der Vitalkontroller ist für den Bioharness Brustgürtel zuständig. Er liefert uns die Vitaldaten, wie zum Beispiel die Herzfrequenz, die Atemfrequenz, Blutdruck, Beschleunigungswerte. Der Movekontroller dagegen bezieht sich auf die Strecke in Form von Karten welche ebenfalls mit eingebunden wurden. Diese Kontroller wiederum sind mit dem

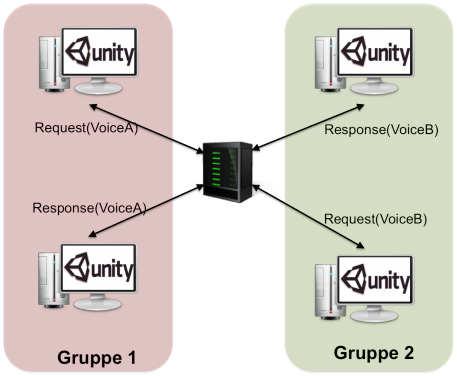

Abbildung 5.5.: Voice Chat zwischen zwei Gruppen

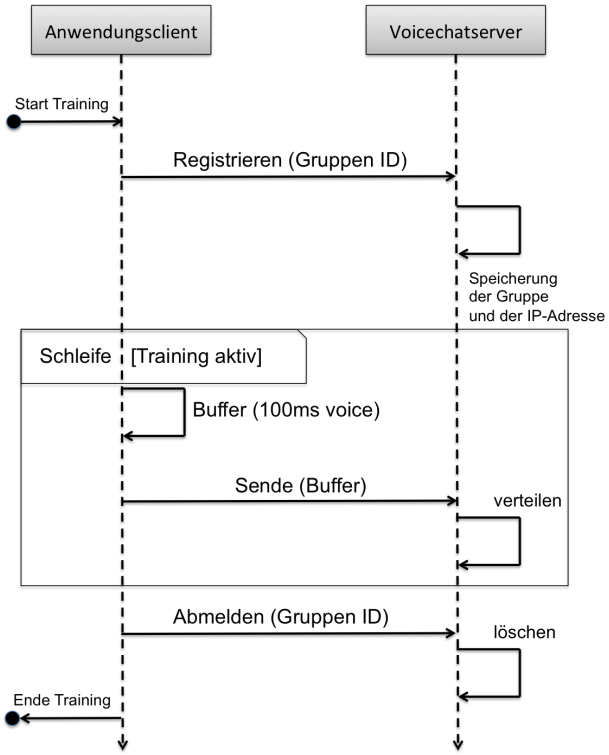

Abbildung 5.6.: Ablauf der Echtzeitkommunikation über den Voicechatserver

Game Kontroller, welcher die Verbindung zu Unity3D herstellen verbunden. Das gleiche gilt auch für den Multiplayer. Jeder Spieler bekommt von seinen jeweiligen Mitspielern Daten übermittelt. Diese Daten sind zum einen Informationen zu seiner Person und zum anderen der aktuellen Position seines Standortes. Die Komponente Statistic ist zuständig für alle Belange rund um das Aufzeichnen, verwalten und versenden von Trainingsdaten. Die jeweiligen Trainingsdaten eines Patienten, welche während des Trainings aufgezeichnet werden, bekommt der Patient am Ende seiner Trainingseinheit als kurze Zusammenfassung dargestellt. Die komplette Datenmenge wird an den Server übermittelt und dort in einer

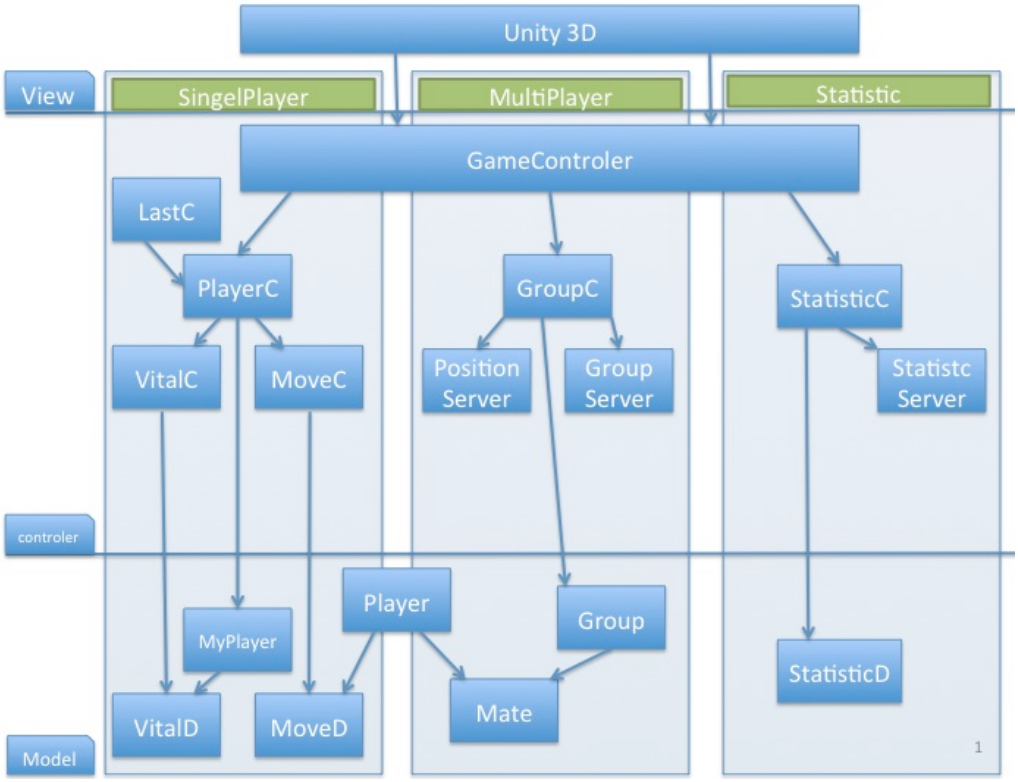

Abbildung 5.7.: Modell View Controller

Datenbank abgelegt. Die Patienten und Ihre behandelnden Ärzte haben die Möglichkeit, diese über die RehaWeb Webseite zu betrachten und auszuwerten.

5.3.2. Das Protokoll UDP

Das User Datagram Protocol (UDP) ist als Transportprotokoll im TCP/IP–Schichtenmodell der dritten Schicht zugeordnet. Es stellt grundlegende Funktionen zur Verfügung um zwischen kommunizierenden Prozessen mit geringem Aufwand Daten austauschen zu können und nutzt dabei den Vermittlungsdienst IP. UDP ist ein verbindungsloses Protokoll, d.h., es garantiert keine sichere Ankunft der versendeten Datenpakete, die so genannten Datagramme. Diese Datagramme können in unterschiedlicher Reihnfolge den Empfänger erreichen. Dabei können auch Pakete verloren gehen. Hierbei werden weder der Sender noch der Empfänger über den Verlust informiert. Wir als Projekt Gruppe CordiAAL haben uns für das Protokoll UDP entschieden, statt das TCP/IP. Zwar ist TCP/IP ein verbindungsorientiertes Protokoll und bietet mehr Kontrolle und Sicherheit beim Datentransfer aber dennoch war für unseren Datenaustausch dies nicht von großer Bedeutung. In unserem vorliegendem Projekt hat es ausgereicht, dass UDP die Datagramme bzw. die aktuellen Werte jede Sekunde hintereinander sendet. Dabei sind die Verluste von Datagrammen für uns nicht relevant, da jede Sekunde immer aktuelle Daten nacheinander kontinuierlich ver-

schickt werden und wir so durch den Erhalt von den Daten gewährleisten können. UDP ist ein pragmatisches Mittel, um Nachrichtenaustausch, die keine zuverlässige Übertragung erfordern, zu führen. In der folgenden Abbildung 5.8 wird eine UDP PDU dargestell.

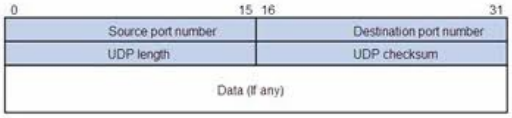

Abbildung 5.8.: UDP PDU

Als Programmierschnittstelle benutzen Java-Programme so genannte Sockets für den Nachrichtenaustausch über UDP/IP und TCP/IP. Sockets sind als Endpunkte einer Kommunikationsbeziehung zwischen Prozessoren zu betrachten, was zur Folge hat, dass eine Verbindung zwei Sockets definieren. Ein UDP-Socket wird auf den lokalem Rechner an eine Portnummer gebunden und durch eine IP eindeutig definiert. Somit können Datagramme an jeden anderen UDP-Socket gesendet und von jedem beliebigen UDP-Socket empfangen werden. Der Unterschied zwischen Client und Server ist sehr gering. Der wesentliche Unterschied des Servers besteht darin, dass dieser die bind() Methode aufrufen muss, um nicht einen zufälligen Port zu bekommen, sondern genau den, den wir erfordern. Im weiterem Verlauf dieser Arbeit wird die Klasse UDPCommunication() näher erläutert. In der folgenden Abbildung 5.9 wird der Client/Server Socket dargestellt:

5.3.3. REST Schnittstellen

Für die Kommunikation zwischen dem CordiAAL-Server und der Unity-Client Anwendung haben wir uns für die Verwendung von REST-Schnittstellen entschieden. Diese Schnittstellen haben wir beispielsweise für den Login, das Abrufen von Benutzerdaten inklusive Vitalparameter, das Erstellen und Abrufen von Gruppen sowie das Senden und Empfangen der statistischen Daten, welche bei den Trainings der Patienten aufgezeichnet werden genutzt.
Im Folgenden werden die einzelnen Schnittstellen des Clients detailliert aufgeführt und beschrieben. Eine tabellarische Übersicht der verwendeten Client Schnittstellen ist in Tabelle 5.1 dargestellt.

Login mit Übermittlung der Userdaten (Vitalparameter): Als Methode wird GET benutzt. Die Funktion dieser Schnittstelle ist das Überprüfen der Login Daten der Patienten beim Start des Unity-Clients. Ist die Kombination aus Benutzername und Passwort korrekt, werden die Profildaten inklusive der Vitalparameter, welche über RehaWeb von dem jeweils zuständigen Arzt eingestellt worden sind als Antwort an den Client übermittelt. Welche Daten genau?

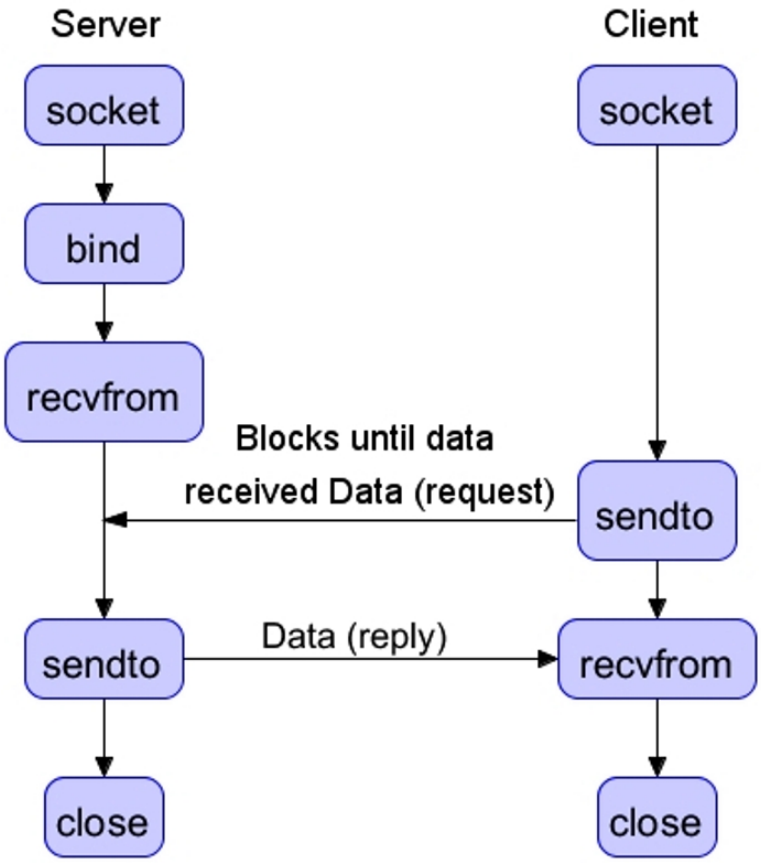

Abbildung 5.9.: UDP Socket

Virtuelle Routen abrufen: Als Methode wird GET benutzt. Über diese Schnittstelle werden die vorhandenen virtuellen Strecken, inklusive den wichtigsten Eckdaten abgerufen. Bei der Erstellung eines Ergometer Trainings können die Patienten sich zwischen den vorhanden Strecken entscheiden, welche sie für ihre Trainingseinheit nutzen möchten. Die Eckdaten enthalten Informationen zu der Schwierigkeit und eine Beschreibung der virtuellen Trainingswelt.

Virtuelle Gruppen abrufen: Hat sich der Patient für ein Gruppentraining entschieden, ist es möglich sich über diese Schnittstelle alle aktuellen virtuellen Gruppen in Form einer Liste anzeigen zu lassen.

Vitueller Gruppe beitreten: Über diese Schnittstelle kann ein Patient zu einem virtuellen Training beitreten. Profil ID des Patienten wird an den Server übermittelt. Daher wird hier die Methode POST verwendet. Als Antwort erhält man die Gruppen ID. Die Profil ID des Users wird in der Datenbank dem jeweiligen Training zugeordnet.

Virtuelle Gruppe erstellen: Die Schnittstelle dient der Erstellung einer neuen virtuellen Trainingsgruppe. Als Methode wird POST verwendet, da hier die Informationen für

Funktion	URL	Methode
Login	BaseURL+/checklogin	GET
virtuelle Routen abrufen	BaseURL+/VirtualRoute	GET
virtuelle Gruppen abrufen	BaseURL+/VirtualGroup	GET
virtuelle Gruppe beitreten	BaseURL+/VirtualGroup/+{VGroupID}	POST
virtuelle Gruppe erstellen	BaseURL+/VirtualGroup	POST
Statistiken abrufen	BaseURL+/user/+{UserId}/statistic	GET
Statistik senden	BaseURL+/statistic	POST

Tabelle 5.1.: Client REST-Schnittstellen

die neue Gruppe per JSON String an den Server übermittelt werden. Als Antwort erhält man die ID der neu erstellten virtuellen Gruppe.

Statistiken abrufen: Über diese Schnittstelle werden die Statistiken der einzelnen Patienten abgerufen. Patienten können so auch in der Ergometer Client Anwendung ihre gesamten Statistiken einsehen. Allerdings handelt es sich hierbei um eine gekürzte Version, damit es nicht zu längeren Wartezeiten kommt. Detailtiere Statistiken können über RehaWeb eingesehen werden. Die verwendete Methode ist GET. Bei der Anfrage wird die Profil ID des Patienten übergeben. Als Antwort vom Server erhält man eine Liste mit allen bisher gespeicherten Trainingsdaten zu dieser Profil ID.

Statistiken senden: Am Ende einer Trainingseinheit werden die während des Trainings aufgezeichneten Daten an den Server übermittelt, damit sie dort in der Datenbank gespeichert werden können. Dazu dient diese Schnittstelle. Sie verwendet die Methode POST. Im Request Body wird der JSON String

5.4. Sensoren und Anbindungen

Für unser reaktives System stehen uns Sensoren aus dem medizinischen und technischen Bereich zu Verfügung. Zur medizinischen Überwachung verwenden wir den Zephyr BioHarness Sensor sowie den Corscience NiBP2010 SpO2 Sensor. Als technische Sensoren verwenden wir das ergo_bike 2002 der Firma Daum electronic sowie eine 3D-Brille zur Anzeige der immersiven 3D-Welt und Trecking der Kopfbewegungen. Im folgenden werden die einzelnen Sensoren mit ihren zugrunde liegenden Schnittstellen beschrieben.

5.4.1. Zephyr BioHarness 3.0

Der BioHarness 3.0 (auch *BioHarness 3*) der Firma Zephyr ist ein kompaktes Modul für das Aufzeichnen von physiologischen Daten und wird beispielsweise für die biomechani-

sche und physiologische Forschung verwendet. Er wird an ein schmales Band angebracht, an dem Sensoren für Elektrokardiogramm (EKG) und Atemfrequenz integriert sind. Der BioHarness 3 bietet weitaus mehr Funktionalität, wie z. B. die Erkennung der Beschleunigung in Richtung der X-, Y- und Z-Achse. Insgesamt machen wir in unserem Projekt CordiAAL Gebrauch von folgenden Vitaldaten, die vom BioHarness 3 geliefert werden:

- EKG und damit die **Herzfrequenz** (Heart rate = HR)
- **Atemfrequenz**
- **Körpertemperatur** (errechnet aus Herzfrequenz)

Für die Körpertemperatur ist im Standardumfang *kein* Sensor enthalten, jedoch wird dieser mit Hilfe eines Algorithmus aus der Herzfrequenz errechnet.([**?**, S. 6])
Für die Installation ist das SDK nötig. Dieses beinhaltet u. a. folgende Dateien:

- Treiber für die Installation
- Die Dokumentation (u. a. die Protokollspezifikation ([**?**]))
- Das Konfigurationstool *Zephyr Config Tool.exe*
- Die Test-Applikation
- Den *BioHarness Log Downloader*

Das SDK kann, sofern nicht im Lieferumfang vorhanden, von der Webseite des Herstellers heruntergeladen werden [1]. Näheres zur Installation und Inbetriebnahme ist in 6.4 zu finden.
Der BioHarness 3 besitzt vier LED-Anzeigen. Diese haben, je nachdem, ob das Modul in der Ladeschale ist oder nicht, grundsätzlich eine andere Bedeutung. Es gibt jeweils eine Anzeige in eigener Farbe für die Datenübertragung bzw. Konnektivität (Bluetooth), das Logging, die Batterie und die Erkennung der Herzfrequenz ([**?**, S. 33]). In Tabelle 5.2 ist eine Beschreibung dieser LED-Anzeigen für den Fall, dass das Modul *nicht* in der Ladeschale ist:

Aufgeladen wird der Akku, indem das Modul in eine Ladeschale gesetzt wird, welche per USB-Kabel z. B. mit einem PC verbunden werden kann. Die Ladezeit beträgt 1 Stunde für eine Ladung von 90% der Gesamtkapazität. Für eine volle Aufladung sind 3 Stunden Ladezeit nötig. Ein voller Akku des Brustgürtels hält 24 Stunden im Standbymodus und 18 Stunden im Betrieb ([**?**, S. 17]).
Der BioHarness 3 kann in drei Übertragunsmodi betrieben werden ([**?**, S. 5]):

1.) Übertragungsmodus: Der BioHarness 3 sendet die ermittelten Daten per Bluetooth an den Empfänger (z. B. einen PC)

[1] http://www.zephyr-technology.com/

	blinkend	**leuchtend**	**aus**
Bluetooth	verbunden	Fehler	deaktiviert
Logging	aktiviert	Fehler	deaktiviert
Batterie	Ladung $> 30\%$	Ladung $< 30\%$	Ladung $< 10\%$
HR Detektion	HR detektiert	Gurt umgeschnallt, aber HR nicht detektiert	Gurt nicht umgeschnallt

Tabelle 5.2.: Die LED-Anzeigen des BioHarness 3 (nicht in Ladeschale)

2.) Log-Modus: Der BioHarness 3 schreibt die ermittelten Daten in den internen Speicher.

3.) Übertragungs- und Log-Modus: Beide Modi werden gleichzeitig betrieben.

Die Default-Konfiguration ist gleichzeitiger Übertragungs- und Logmodus.
Es ist möglich den BioHarness 3 sowohl im Übertragunsmodus als auch im Log-Modus zu konfigurieren. Im ersten Fall sendet man entsprechende Befehle zum Modul, für den zweiten Fall ist das *Zephyr Config Tool.exe* benutzbar, wobei das Modul in der Ladeschale per USB-Kabel mit dem Rechner verbunden sein muss ([**?**, S. 22]). Falls das Endgerät, mit dem sich der BioHarness 3 per Bluetooth verbinden soll, keine integrierte Bluetooth-Schnittstelle besitzt, so ist dies auch mit einem Bluetooth-Adapter am Endgerät möglich. Sobald eine Verbindung über Bluetooth mit Software auf einem Endgerät (z. B. mit der Bluetooth Test-Applikation) hergestellt ist, blinkt die LED-Anzeige für Bluetooth. Für die Kommunikation über Bluetooth nutzt das BioHarness 3 das **Bluetooth Serial Port Profile** (SPP). Dabei wird folgendes Low-Level-Protokoll zugrunde gelegt ([**?**, S. 5]):

- 115,200 baud
- 8 Datenbits
- 1 Stopbit
- Keine Parität

Aus Sicherheitsgründen ist ein PIN-Code vorhanden. Werkseitig lautet dieser: "**1234**"([**?**, S. 5]). Dieser kann beim Bluetooth-Pairing unter Windows direkt eingegeben werden. Danach muss er normalerweise nicht wieder eingegeben werden. Für Details zur Installation sei auf 6.4 hingewiesen.
Der BioHarness 3 besitzt eine Vielzahl an Funktionalitäten, wie z. B. eine Echtzeituhr oder das Speichern patientenbezogener Daten. In unserem Projekt wird allerdings hauptsächlich Gebrauch gemacht vom periodischen Datenpaket *Summary Data Packet*, welches uns die oben genannten Vitaldaten Herzfrequenz, Atemfrequenz und Körpertemperatur liefert.

Das Übertragunsintervall des *Summary Data Packet* kann per Befehl im Übertragunsmodus sekundengenau gesetzt werden, wobei ein Minimum von 1 Sekunde und ein Maximum von 65534 Sekunden ($\approx$ 18 Stunden) möglich sind [**?**, S. 6].

5.4.2. Corscience NiBP2010/ChipOx

Wichtiger Hinweis: In unserem Projekt sollte das *Corscience NiBP2010/ChipOx* lediglich für das Auslesen des **SpO_2**-Wertes verwendet werden, jedoch war die Anbindung des Moduls an unsere Client-Applikation **nicht** erfolgreich, sodass der SpO_2-Wert im Projekt **nicht** verwendet werden kann. Grund hierfür könnte sein, dass die Client-Applikation nicht so viele COM-Ports untersützt.
Pulsoxymetrie ist ein nichtinvasives Verfahren zur Bestimmung der arteriellen Sauerstoffsättigung. Hierbei wird ein Clip an den Finger angebracht, durch den die Lichtabsorption bzw. Lichtremission bei Durchleuchtung der Haut gemessen wird.
Das *Corscience Non invasive Blood Pressure (NiBP2010)* Modul mit der integrierten SpO2 Platine *ChipOx* ist ein kompaktes, nichtinvasiv messendes automatisches Blutdruckmessmodul sowohl für Erwachsene als auch für Neugeborene. Zusammen mit der SpO_2 Platine *ChipOx* ist es auch in der Lage die funktionelle Sauerstoffsättigung (SpO_2) sowie die Pulsfrequenz zu ermitteln ([**?**, S. 4]). Für die Pulsoxymetrie stehen hiermit drei Antwortmodi des Geräts zur Verfügung: *sensitiv*, *normal* und *stabil*. Der Default-Modus ist *normal*. Der sensitive Modus liefert beste Genauigkeit mit sensitiver Artefaktunterdrückung. Für sehr stabile Werte kann der stabile Modus verwendet werden. In allen drei Modi werden die aktuellen Werte für SpO_2 und Pulsfrequenz sekündlich übermittelt. Der Wertebereich für diesen reicht von 0 bis 100, gemessen in Prozent. Für Details sei auf die Dokumentation verwiesen ([**?**]).
Das *Corscience NiBP2010/ChipOx* wird per USB-Kabel an den Rechner angeschlossen, sodass automatisch ein virtueller COM-Port eingerichtet wird.

5.4.3. Ergobike

Um einen medizinischen Trainingseffekt, sowie eine individuelle Lastkontrolle der Patienten zu gewährleisten, benötigen wir eine Aufzeichnung der Lastparameter. Hierzu verwenden wir das ergo_bike 2002 der Firma Daum electronic. Dieses Fahrrad besitzt eine serielle RS232 Schnittstelle mit der wir Lastparameter abrufen können. Die Lastdaten werden in Form von einzelnen Datenpaketen übertragen, die einen Mindestabstand von 50 ms aufweisen müssen. Zum Anschluss an die serielle Schnittstelle eines Rechners muss das Update-Kabel der Firma Daum verwendet werden. Folgende Daten können zwischen den Komponenten PC und Fahrrad ausgetauscht werden, siehe Tabelle 5.3

Daten vom fahrrad zum PC	Daten vom PC zum Fahrrad
aktuelle Wattleistung in 5-Schritten	die Person (Limit-Daten,Cardio-Puls,Drehzahl und Programm)
aktuelle RPM-Drehzahl	das gewünschte Programm
RPM-Kennlinien Zustand	die gewünschte Watt-Leistung
die Momentan-Geschwindigkeit	den aktuellen Gang
die zurückgelegte Distanz	den möglichen Ziel-Puls
die gefahrene Zeit	die Soll-Geschwindigkeit
die verbrauchten KJoule	
das eingestellte Programm	
die angegebene Person	
den aktuellen Pulswert	
den Pulszustand (Up-Ok-Down-Alarm)	
den Relax-Wert	
den Relax-Modi	
den aktuellen Gang	
die Cockpit-Version	

Tabelle 5.3.: Lastdaten die zwischen Fahrrad und PC ausgetauscht werden können

Die grün unterlegten Daten werden für das Projekt CordiAAL benutzt. Jede Sekunde werden durch das Run_Daten Packet des Fahrrads, welches 19 Bytes groß ist, neben den benötigten Daten (Watt, RPM) auch die folgenden Größen mitgesendet:

- Cockpit-Adresse
- das eingestellte Programm
- die Person
- der Tret-Zustand
- die Geschwindigkeit
- zurückgelegte Distanz
- verbrauchte KJoule
- Puls
- Gang

Für die Einstellung der gewünschten Wattleistung während des Trainings wird das Set_Watt Paket verendet, welches 3 Byte groß ist. Diese Kommunikation zwischen unserer Anwendung und dem Fahrrad wird benötigt, um ein realistisches Fahrverhalten im virtuellen Training zu gewährleisten. Einstellungen wie aktuelle Steigungen sowie die mögliche Unterscheidung des gefahrenen Untergrunds, können so realistische und variable Watt Einstellungen annehmen. Es können so Watteinstellungen im Bereich von 25Watt bis 400Watt vorgenommen werden. Wichtig für das Einstellen der Wattleistung ist, dass der gewünschte Watt-Wert bei der Übertragung an das Fahrrad durch 5 geteilt wird. Also für einen gewünschten Watt-Wert von 25Watt, wird über das Set-Watt Datenpaket der Wert 5 versendet. Für nähere Informationen der Schnittstellenbeschreibung, siehe (ref. Daum).

5.4.4. Lenker

Damit sich ein Patient in der Virtuellen Welt auch bewegen kann, benötigten wir neben den Informationen des Fahrrads auch noch die Möglichkeit das Fahrrad zu lenken. Daher wurde der Fahrradlenker soweit modifiziert, dass der Lenker in der Horizontalen beweglich ist. Die Informationen bezüglich der aktuellen Lenkerposition mussten nun noch an unsere Anwendung übermittelt werden. Hierzu haben wir einen Schubregler eines Joysticks verwendet. Dieser Regler gibt uns die Möglichkeit, Werte zwischen 1 und -1 zu erfassen und zu verarbeiten. Durch eine Kalibrierung liefert die mittlere Lenkerposition eine 0, eine eingeschlagene Lenkerposition nach ganz links eine -1 und eine eingeschlagene Lenkerposition nach ganz rechts eine 1. Durch diese Modifizierung, haben wir nun die Möglichkeit, in der virtuellen Welt zu fahren.

5.4.5. Virtual-Reality-Brille mit Head Trecking

Als nächste Herausforderung, neben der Bewegung des Fahrrads, stand die Anzeige der immersiven 3D-Welt, sowie die Möglichkeit sich in dieser umzusehen im Fokus. Hierzu stand zu Beginn des Projekts die Cinemizer OLED der Firma Zeiss zur Verfügung. Die Cinemizer OLED besteht aus zwei hochauflösenden OLED-Displays mit je eine Auflösung von 870 x 500 Pixel. Die Bildgröße ist eine Simulation eines 40 zoll großen Bildschirms aus 2m Entfernung. Als 3D-Technologien werden sowohl Side-by-side, Top-Bottom und Line interleaved unterstützt. Das zuspielen von Video Inhalten wird über eine HDMI Schnittstelle gewährleistet. Durch den schlechten Blickwinkel sowie der schlechten 3D Anzeige entschlossen wir uns für ein Hardware Wechsel hin zur Oculus Rift Virtual-Reality-Brille . Die Oculus Rift ist ein Head-Mounted Display, bestehend aus einem 7 Zoll großen Bildschirm, das mit eine Auflösung von 1.280 x 800 Pixel arbeitet. Zusätzlich ist diese Brille mit einem Bewegungs- bzw. Neigungssensor, für das Motion-Tracking, verbunden. Das Motion-Tracking erfolgt mit einer Abtastfrequenz von 1.000 Hz und wird mittels eines 3-Achsen-Gyroskop sowie eines 3-Achsen-Beschleunigunssensors ausgeführt. Erkannt werden allerdings nur Drehungen

des Kopfes (Rotationen) in beiden Achsen, nicht aber lineare Bewegungen. Das Bildsignal wird ebenfalls über eine HDMI oder DVI Schnittstelle an die Brille weitergeleitet. Bei einer Auflösung von 1.280 x 800 Pixel für das gesamte Display stehen pro Auge also 640 x 800 Pixel zur Verfügung. Durch die Optik entsteht ein Sichtfeld mit einem Öffnungswinkel von 110° diagonal und 90° horizontal. Durch dieses große Sichtfeld bekommt der Nutzer das Gefühl sich in der dargestellten Szene real zu befinden. Damit eine grafische Anwendung mithilfe der Oculus Rift richtig dargestellt wird, müssen zahlreiche Anpassungen vorgenommen werden. Nicht nur das Motion-Tracking spielt hier eine Rolle, sondern auch die Aufteilung des Bildes im Display in eine Darstellung für das linke und das rechte Auge. Oculus bietet hierzu ein quelloffenes Software-Developement-Kit (SDK) an, das von den Anwendungsentwicklern genutzt werden kann. Zum Vergleich dieser beiden Virtual-Reality-Brillen mit ihren Vor- bzw. Nachteilen, siehe Tabelle 5.4

Cinemizer OLED Zeiss		**Oculus Rift Virtual-Reality-Brille**	
Pro	*Contra*	*Pro*	*Contra*
leicht und bequem zu tragen	Motion-Tracking nicht serienmäßig	gutes Motion-Tracking	schwer und klobig
leichte Entwicklung dank Side-by-side bzw. Top-Bottom Technologie	sehr schlechter Blickwinkel	gute und realistische 3D-Anzeige	Schwierige Entwicklung der 3D-Anzeige
Stereo-Sound	nicht realistische 3D-Anzeige	großes Sichtfeld	Auflösung
	Preis	Preis	kein Sound

Tabelle 5.4.: Vor und Nachteile der Virtuell-Reality Brillen

Aufgrund des größeren Sichtfelds in Verbindung mit der realistischen Anzeige, welches für uns ein wichtiger Aspekt für die Motivation einzelner Patienten darstellt, haben wir uns für die Oculus Rift und gegen die Cinemizer OLED entschieden.

5.4.6. Touch-Monitor

Neben der Anzeige und Durchführung des Trainings in unserer Anwendung benötigen wir noch eine leichte und intuitive Eingabemöglichkeit für:

- Login
- Filterung der aktuellen Trainings

- Filterung der eigenen Statistiken

- sowie einer Bedienung der Anwendung

Da eine Interaktion mit Maus und Tastatur auf dem Fahrrad mit den angeschlossenen Sensoren umständlich und schwierig ist, brachten wir einen Touch-Monitor an das Fahrrad an. Durch die Technik des kapazitiven Touchscreen haben wir die Vorteile der einfachen und sicheren Eingabe für eine Interaktion mit unserer Anwendung ohne zusätzliche Hardware gewährleistet.

5.5. Reha-Web

Die Projektgruppe RehaWeb[2] befasste sich im Wintersemester 2010/11 mit dem immer wichtiger werdenden Bereich Ämbient Assisted Living"(AAL). AAL beschreibt die Entwicklung, Erprobung und Einführung von IT-basierten Assistenzsystemen auf Basis von Sensorik. Die Gruppe konzeptionierte und entwickelte ein AAL-Sensor- und Servicesystem, in dem die Funktionen von Sensoren als Services bereitgestellt und mit diesen geeignete flexible Applikationen realisiert wurden. Daneben realisierte die Gruppe in Zusammenarbeit mit Partnern aus Industrie und Medizin eine zielgruppengerechte Healthcare Community im Sinne des Web 2.0, in die das zuvor konstruierte AAL-System integriert wurde. Auf der Webplattform können Patienten sich Informationen zu verschiedenen Wanderrouten einholen, mit anderen Patienten Erfahrungen austauschen und sich zu gemeinsamen Wanderungen verabreden. Mittlerweile wurde die Community um ein weiteres funktionsreiches Feature ergänzt. Das OSAMI-Projekt[3] beschäftigt sich mit der Idee „eine SOA-fähige Komponentenplattform bereit zu stellen“. Hiervon können Patienten und Ärzte gleichermaßen profitieren. Wie in der Kurzdarstellung des OSAMI-Projekts ersichtlich, kann diese Software in folgenden Anwendungsfeldern integriert werden:

[2] `http://ls4-www.cs.tu-dortmund.de/cms/de/lehre/vorherige_semester/2011_ss/rehaweb/index.html`

[3] `http://ls4-www.cs.tu-dortmund.de/cms/Medienpool/forschung/rvs/OSAMIkurz.pdf`

Klinik	Von Klinikaufenthalten herrührende Diagnose- und Behandlungsdaten werden externen Nutzern elektronisch zugänglich gemacht und ermöglichen somit einen zeitnahen Zugriff auf die vollständige Krankengeschichte eines Patienten. Existierende Klinikinformationssysteme und elektronische Patientenakten lassen sich nahtlos integrieren. Dadurch ist eine herstellerneutrale und formatunabhängige Nutzung der Informationen realisierbar. Die weiterführenden, rehabilitativen Behandlungen, die in der häuslichen Umgebung des Patienten stattfinden sollen, werden vom Klinikpersonal in Form eines Behandlungsplans zusammengestellt. Erforderliche, medizinische Geräte (z. B. EKG-Messgerät) werden dem Patienten zur Verfügung gestellt und konfigurieren sich in Entsprechung zum individuellen Behandlungsplan in der häuslichen Umgebung automatisch.
Arztpraxis	Der niedergelassene Arzt ist Bindeglied zwischen Klinik und Patient. Er überwacht den Heilungsverlauf, passt bei Bedarf den Behandlungsplan an oder weist den Patienten zurück in die Klinik.
Häusliche Umgebung	Die häusliche IT-Infrastruktur umfasst eine Vielzahl miteinander interagierender, handelsüblicher (z.B. PCs, Smartphones) und spezialisierter, medizinischer Geräte. Von Sensoren erfasste medizinische Parameter werden analysiert. Sind Schwellwerte erreicht oder verläuft der Genesungsprozess ungünstig, ergreift das System vordefinierte Maßnahmen. Je nach Konfiguration werden die Daten direkt an den zuständigen, medizinischen Supervisor geschickt oder für eine spätere Nutzung lokal hinterlegt.
Gerätehersteller	Spezialisierte Dienste unterschiedlicher Hersteller lassen sich flexibel kombinieren und anwendungsübergreifend einsetzen. Dadurch entstehen Markteinstiegschancen auch für solche KMUs, die derzeit noch nicht im medizinischen Bereich aktiv sind.

Auch die CordiAAL-Gruppe hat einen Teil Ihrer Anwendung in RehaWeb eingebunden. Auf der Webplattform werden alle nötigen Informationen zu den virtuellen Strecken aus der CordiAAL-Anwendung dargeboten. Die nachfolgende Tabelle beschreibt die Anwendungsmöglichkeiten durch Administratoren, Ärzte und Patienten:

Administratoren	Die Administratoren der Plattform können unter bestimmten Voraussetzungen neue virtuelle Strecken in dem System anlegen, Rollen anderer Community Mitglieder ändern und Foren bearbeiten.
Ärzte	Ärzte können auf der Webplattform Trainingsfortschritte der Patienten einsehen und gegebenenfalls für das nächste Training die Gesundheitswerte, wie die Herzfrequenz, anpassen.
Patienten	Patienten können ihre Trainingsstatistiken nach jedem Training einsehen. Damit das virtuelle Training auch in der Gruppe stattfinden kann, gibt es die Möglichkeit sich mit anderen aus der Community zu verabreden. Diese Funktion ermöglicht auch einen Trainingsplan anzulegen. Der Trainingsplan beinhaltet wann der Patient an welcher Strecke wie lange trainieren wird. Patienten können auf der Plattform nach der Auswahl der Strecke auf der Sie tranineren wollen eine Gruppe bestehend aus einem und maximal neun Mitgliedern erstellen und andere befreundete Patienten in die Gruppe einladen. Einen Überblick über alle eigenen Gruppen bekommt der Patient im persönlichen Bereich auf der Plattform dargestellt.

Die Möglichkeit einen Trainingsplan anzulegen oder eine Trainings-Statistik online abzurufen, erleichtert es dem Arzt, aber auch den Patienten sich einen Überblick über den Trainingsfortschritt zu verschaffen. Dabei muss man folgende Möglichkeiten unterscheiden:

Ärzte	Ärzte können auf der Webplattform Trainingsfortschritte der Patienten anhand der Statistikeinträge ansehen. Dabei wird dem Arzt ein Überblick über alle bereits absolvierten Trainingseinheiten jeden Patientes angeboten. Aus dieser Übersicht kann sich ein Arzt zu einem Training alle vorhandenen Trainingsdaten wie ausgewählte Strecke, Trainingsdauer und die Vitalwerte anzeigen lassen.
Patienten	Patienten können ihre Trainingsstatistiken nur in gekürzter Form sehen. Dazu gibt es im persönlichen Bereich auf der Plattform eine Übersicht der einzelnen Termine, an denen Sie trainiert haben.

Die Statistik ist eine webbasierte Anwendung, in der sich im Browser alle nötigen Werte der abgeschlossenen Trainingseinheiten im Detail betrachten lassen können. Der Zugriff auf die Daten der Datenbank erfolgt über eine REST-Schnittstelle. Für jede Trainingssession wird ein individueller Rest-Befehl gesendet, um den Traffic möglichst gering zu halten. Die Darstellung der Statistik erfolgt mit flot[4], einer JavaScript basierten Anwendung.

[4] `http://www.flotcharts.org`

6. Implementierung

In der Implementierungsphase der Projektgruppe wurde mit Hilfe von Hilfswerkzeugen wie MonoDevelope und Eclipse die Software für das virtuelle Training und die Erweiterungen auf RehaWeb umgesetzt.

6.1. Werkzeuge

Zur Softwareentwicklung gibt es viele nützliche Werkzeuge. Diese sind zum Teil lizenzfrei. Die PG-Gruppe hat aus der großen Vielfalt MonoDevelope, Eclipse und VisualStudio ausgewählt.

6.1.1. Unity3D und MonoDevelope

Eine 3D Scene wird im Unity3D Editor erstellt und bearbeitete. Die Objekte können mit der Maus frei platziert werden und ihn ihrer Größe und Rotation manipuliert werden. Sollen die Objekte in der Scene eine Logik enthalten, muss eine entsprechende Komponente angefügt werden. Diese Komponenten sind aus Programmiere Sicht Klassen. Für das Entwickeln von Komponenten haben wir uns für MonoDevelope entschieden. MonoDevelope wird von Unity3D als Standard Entwicklungsumgebung verwendet. Es hat einen Funktionsumfang wie Visual Studio kann aber als Debugger für Unity3D verwendet werden, MonoDevelope ist die einzige Entwicklungsumgebung die Debugging ohne zusätzliches Plugin unterstützt. MonoDevelope wurde aber nicht zum kompilieren des Projektes benutzt, Unity3D benutzt einen eigenen internen Compiler um das Projekte zu erstellen, somit wurde MonoDevelope nur zum programmieren und Debuggen benutzt.

6.1.2. Blender

Blender ist eine Open Source Software zur Modellierung und Animation von 3D-Modellen. Alternativen wie Cinema4D oder Maya sind mit erheblichen Kosten verbunden. Hauptgrund für den Einsatz von Blender war es, eine Animation unseres Radfahrers zu erstellen. Dieser sollte auf unserem Fahrrad sitzen und in Abhängigkeit von der aktuellen Geschwindigkeit in die Pedalen treten. Eine Animation in Blender ist ein Bewegungsablauf eines 3D-Modells, das für jeden Frame die aktuellen Positionen des 3D-Modells abspeichert. Für dieses Ziel haben wir Blender in der Version benutzt. Ein großer Vorteil von Open Source

Projekten ist, dass man im Internet über Foren, Youtube oder Webseiten zahlreiche Tutorials und Hilfe von anderen erfahrenen Anwendern findet. Das erleichtert den Einstieg in die Anwendung der Software erheblich. Für unser Projekt haben wir nur einen kleinen Teil von Blender in Anspruch genommen. Da wir zeitlich eingeschränkt waren, haben wir uns im Internet eines 3D-Modells für den Radfahrer und das Fahrrad bedient. Blender erleichtert auch hier dem Benutzer die Arbeit. Durch die Importfunktion können problemlos auch Formate von anderen 3D Modellierungsprogrammen eingefügt werden. Im Internet gibt es ein Fülle von 3D-Modellen, die kostenlos durch andere User zur Verfügung gestellt werden.

Als ersten Schritt für unsere Fahrradanimation haben wir die 3D-Modelle in Blender importiert. Danach haben wir die Modelle in eine Datei zusammengepackt. Menschliche 3D-Modelle werden überwiegend in der sogenannten T-Pose zur Verfügung gestellt. Diese 3D Modelle werden auch Mesh genannt und bestehen aus einem Netz von Polygonen. Wir könnten nun anfangen, unsere Modell zu animieren. Da wir je nach Pose mehrere Objekte gleichzeitig verschieben müssten, würde diese Vorgehensweise eine Menge Zeit in Anspruch nehmen. Deswegen haben wir mit unserem Radfahrer erst ein sogenanntes Rigging vor der eigentlichen Animation vorgenommen. Rigging bedeutet, dass wir unserer Mesh eine Armature hinzufügen. Diese Armature ist ein Knochenskelett, das aus mehreren Knochen(bei Blender auch Bones genannt) besteht. Mit Hilfe dieses Skeletts ist es möglich, den Radfahrer im letzten Schritt in die Pedalen treten zu lassen. Im nächsten Schritt könnten wir unser eigenes Skelett konstruieren und es auf unseren Radfahrer anpassen. Wir haben uns mit Hilfe eines Add-ons die mühselige Arbeit erleichtert. Das Add-on stellt uns ein fertiges menschliches Skelett zu Verfügung. Einzelne Knochen die wir nicht gebraucht haben, konnten einfach entfernt werden. Es ist auch möglich, sich ein animiertes Skelett herunterzuladen und es anschließend an ein 3D-Körper anzupassen. Dabei handelt es sich um .bvh Dateien, die problemlos in ein Projekt importieren werden können. Anschließend ist es wichtig, die Knochen unseres Skelettes anatomisch korrekt in unseren Radfahren zu platzieren. Dabei sollte man präzise arbeiten, um am Ende ein realistisches Ergebnis zu erzielen.

Nachdem jeder Knochen an der richtigen Stelle sitzt, wählen wir unseren Radfahren aus und weisen ihn das fertige Skelett zu. Somit weiß Blender, dass das Skelett zu unserem Radfahren gehört. In den „Weight Paint- Mode“ wird jedem einzelnen Knochen ein Gebiet der Mesh zugeteilt. Dabei soll der Oberschenkelknochen nur das Gebiet um den Oberschenkel verformen und keine Auswirkungen auf andere Körperteile haben. Die Zugehörigkeit wird dabei farblich gekennzeichnet. Rot steht dabei für den Wert 1.0 und bedeutet, dass das Mesh dem zugehörigen Knochen folgt und stark beeinflusst wird. Blau hat den Wert 0.0 und bewirkt das genaue Gegenteil

Als letzten vorbereitenden Schritt zur Animation fasst man mittels Inverse Kinematik die Gelenke an den Körper zusammen. Hierzu weist man den Oberschenkel sowie Unter-

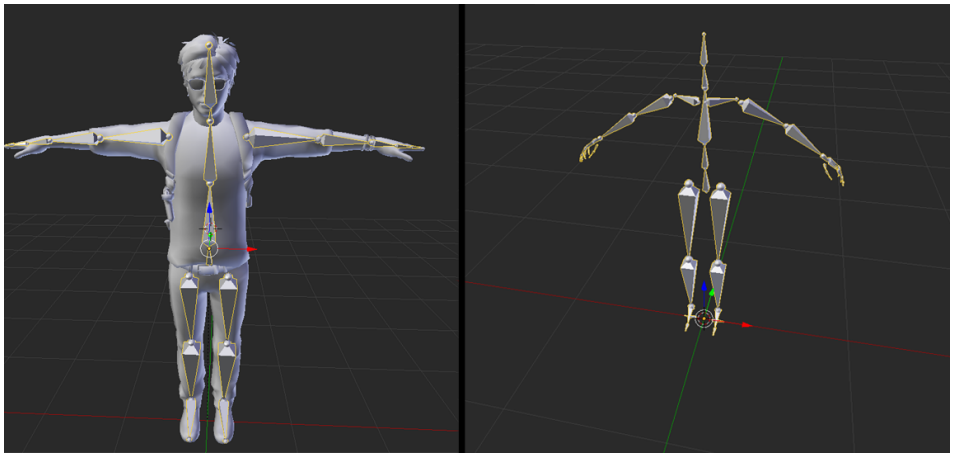

Abbildung 6.1.: Auf der rechten Seite sieht man unser Skelett, dass auf unser 3D-Modell angepasst worden ist.

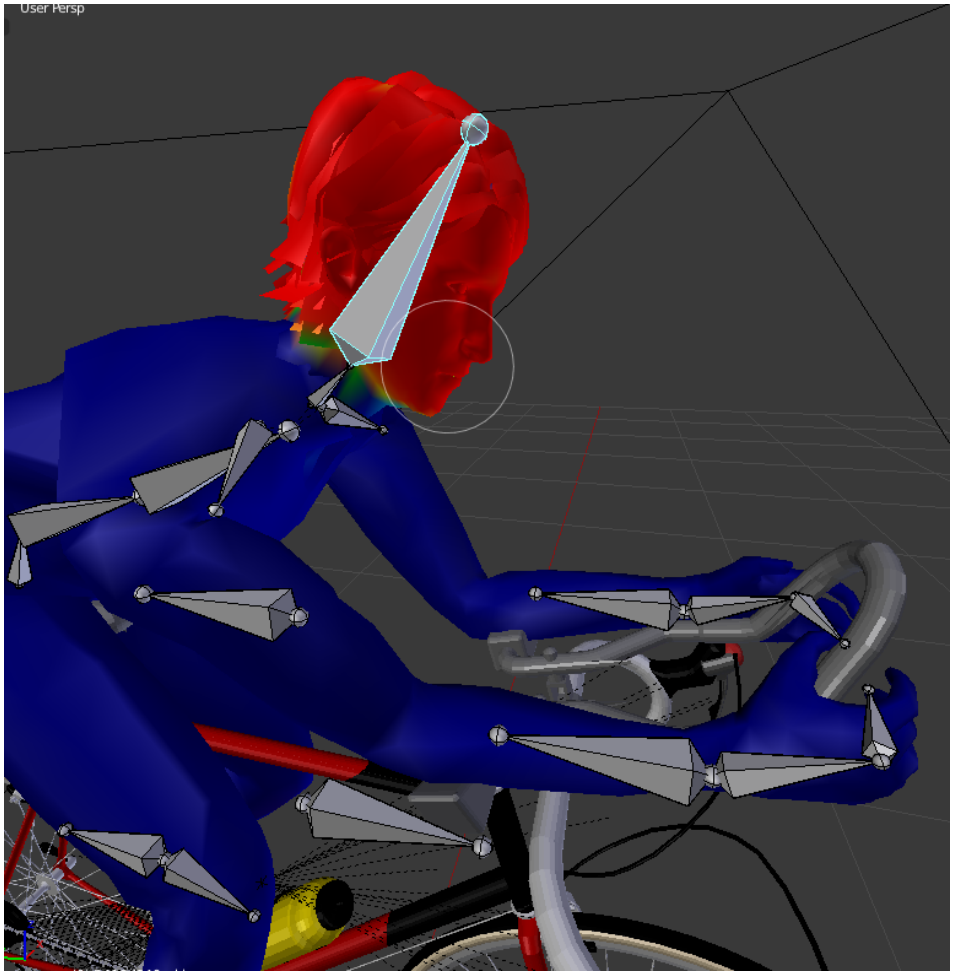

Abbildung 6.2.: Der Weight Paint Modus in Blender

schenkelknochen einem neu erzeugten Knochen zu. Mittles dieser neu erzeugten Knochen kann man das Kniegelenk in eine beliebige Position bringen.Dieser neu erzeugte Knochen

ist das letzte Glied der Kinematischen Kette und beeinflusst alle anderen Knochen der Kette. In Blender kann man die beeinflussten Knochen in der Kette selbst auswählen. Die Inverse Kinematik wird auch in der Robotik genutzt und sorgt dafür, dass sich die Gelenke in einem anatomisch korrekten Winkel befinden. Falls wir unser Knie nun in einen bestimmte Position bringen wollen, sorgt die Inverse Kinematik, dass sich der Oberschenkel und Unterschenkelknochen automatisch mitbewegen und eine realistische Stellung einnehmen.
Sobald das Modell fertig geriggt ist, kann man mit der Animation anfangen. Hierzu wechseln wir in den Pose-Mode von Blender. Im unteren Bereich des Bildschirmes befindet sich eine Zeitachse. Diese wird nicht in Sekunden, sondern in Frames gemessen. Als Standard haben wir 250 Frames zur Verfügung. Auf der rechten Seite des Bildschirmes können wir uns aussuchen, wie viele Frames eine Sekunde dauern soll. Für unsere Animation haben wir uns für 24 Frames pro Sekunde entschieden. Wir legen unser Modell in die Startposition und setzen dann bei Frame 0, für alle benötigten Knochen, Keyframes ein. Bei einem gesetzten Keyframe merkt sich Blender die Koordinaten des jeweiligen Knochens auf den aktuellen Frame. Damit das mehrmalige hintereinander Abspielen unserer Animation einen flüssigen Ablauf hat, müssen wir unsere Animation loopfähig gestalten. Dazu kopieren wir die Anfangspose unseres Modells und fügen sie am letzten Frame unser Animation wieder ein. Danach wird wieder ein Keyframe gesetzt. Für die anderen Frames gehen wir ähnlich vor. Dazu bringen wir unser Knochen in die gewünschte Position und setzten ein Keyframe, bis wir einen flüssigen Bewegungsablauf haben. Man muss nach jedem einzelnen Frame ein Keyframe setzen. Blender berechnet die Position der Knochen von Keyframe zu Keyframe automatisch und erzeugt so einen sanften Übergang. Das Modell ist fertig animiert wenn nach einem Durchlauf die Beine wieder in der Ausgangsstellung sind. Für unser Modell haben wir insgesamt neun Keyframes für die kreisförmige Bewegung gesetzt.

Als letzten Schritt wird das Projekt exportiert und bei Unity3D in den Projektordner kopiert. Dort haben wir abschließend unseren Radfahrer auf seinen Fahrrad. In dem Modell befindet sich eine Animationsdatei. Diese Kann je nach Bedarf abgespielt werden. Die Geschwindigkeit der Animation kann bei Unity3D dynamisch zu Laufzeit mittels eines einfachen Skripts reguliert werden. So ist es möglich, den Radfahrer in Abhängigkeit von der Geschwindigkeit in die Pedalen treten zu lassen.
Der Einstieg in Blender war zu Beginn sehr schwer und man hat sich bei der hohen Anzahl der teilweise komplexen Funktion sehr verloren gefühlt. In dem Projekt haben wir nur ein kleinen Teil von Blenders Funktionsumfang genutzt und so Einblick in die Welt der 3D Animation gehabt. Nahezu für alle Funktion gibt es ein Videotutorial, was einem die Bearbeitung eigener Animationen erheblich erleichtert. Dabei ist das Herzstück der Animation ein gut geriggtes 3D-Modell.

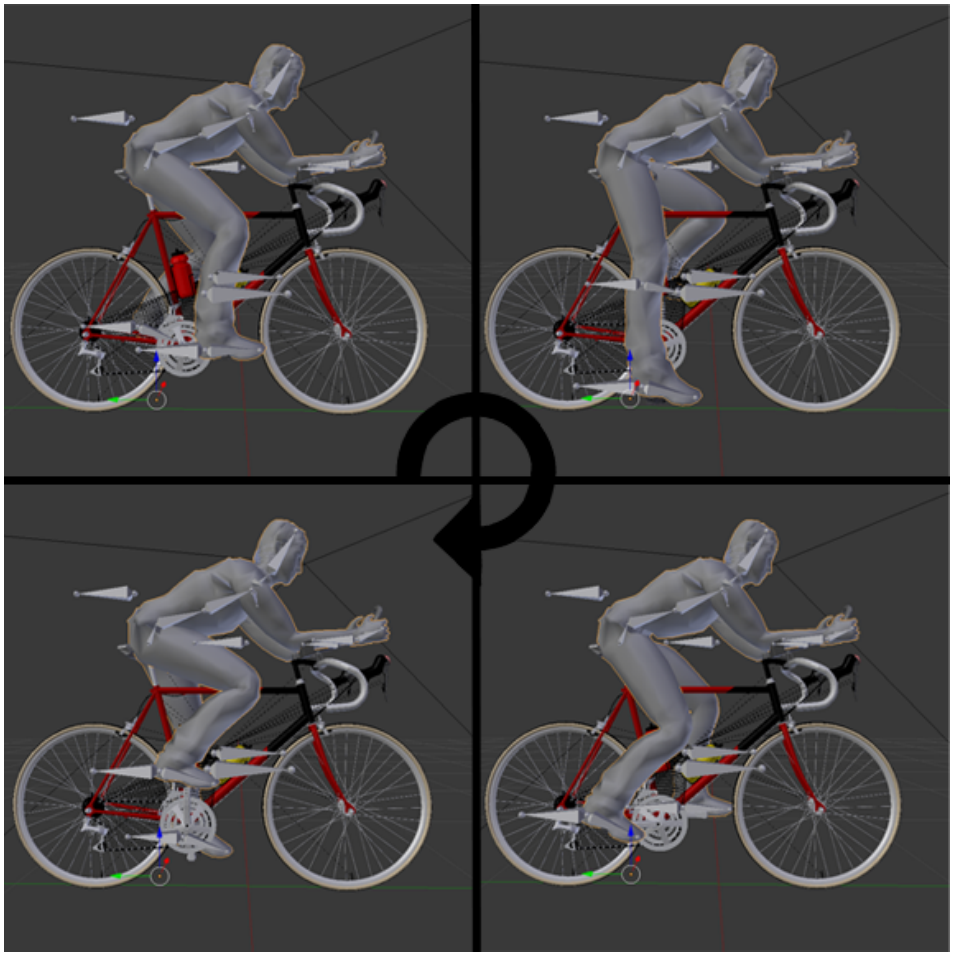

Abbildung 6.3.: Der Lifecycle unser Animation.

6.1.3. Eclipse

Als Programmierwerkzeug zur Erweiterung der Webplattform, diese wurde bereits durch eine Vorgänger Projekt-Gruppe entwickelt, wurde Eclipse eingesetzt. Mit Eclipse kann man leicht Software verschiedener Art entwickeln. Es wird gerne als Entwicklungsumgebung für JAVA und PHP Projekte genommen. Da die Teilnehmer der Gruppe sich bereits mit Eclipse auskannten, war die Entscheidung gegen die Alternativen, wie z. B. NetBeans, gefallen. Mit einer Vielzahl an Erweiterungen und Plug-Ins bietet Eclipse eine breite Möglichkeit an Hilfen für den Entwickler. Es gibt sowohl ein Subversion Plug-In als auch einen Compiler. Ebenso ist es möglich einen Application-Server mit Eclipse zu verbinden und damit JAVA-Webprojekte direkt in Eclipse starten zu können.

6.1.4. Maven

Das primäre Ziel, Maven einzusetzen, war es, schnell aus dem Java-Projekt eine WAR-Datei erstellen zu können, um es in Tomcat einbinden zu können. Vorteile von Maven in der Übersicht:

- Einfaches Erstellen des Build-Prozesses
- Die Bereitstellung eines einheitlichen Build-Systems
- Zulassen transparenter Migration auf neue Features

6.1.5. MS Visual Studio und SVN

Als Entwicklungswerkzeug für jegliche C# Anwendungen haben wir uns für Microsoft Visual Studio in Version 2012 entschieden. Diese Entscheidung haben wir nicht bereut. Es erfüllte zu jederzeit unsere Erwartungen und brachte alle notwendigen Funktion die wir benötigten mit. Es unterstützt und vereinfacht die Entwicklung durch Funktionen wie Autoersetzung und Vorschlägen. Darüber hinaus gibt es im Internet eine große Community, welche bei eventuellen Fragen meistens eine Lösung bietet. Für uns Studenten an der TU-Dortmund ist es zudem möglich das Microsoft Visual Studio kostenfrei im Rahmen unseres Studiums zu nutzen.
Die Wahl für unsere Versionskontrolle fiel auf AnkhSVN für Microsoft Visual Studio, da einige unserer PG-Teilnehmer bereits mit diesem Programm vertraut waren und es es sich sehr gut in Visual Studio integrieren lässt. Es ist kostenfrei verfügbar und bietet viele nützliche Funktionen wie intuitive Konfliktbeseitigung, eine Echtzeit Übersicht aller Projektveränderungen und übersichtliche Integration in die Entwicklungsumgebung.

6.2. Quellcode

Dieses Kapitel beschreibt, wie die im Entwurf festgelegten Strukturen im einzelnen umgesetzt wurden.

6.2.1. RehaWeb

Die Weiter-Entwicklung des RehaWeb-Portals ergänzte die Anmwendung um neue Entitiäten, Klassen und Ressourcen, sodass neue Funktionen auf der Webplattform entstanden sind. Die neuen Funktionen und Ressourcen sind aus den Anforderungen für ein virtuelles Training entstanden. So kann man auf der Webplattform die virtuellen Strecken im Detail ansehen, neue Trainingszeiten anlegen und andere Patienenten zum Gruppentraining einladen. Bevor ein Patient ein virtuelles Training überhaupt starten kann, muss ein Artzt seine Vitalwerte auf der Plattform eintragen. Dazu wurde auf der Plattform eine spezielles

Formular entwickelt *RehaWeb-Arzt-Maske* (siehe Abbildung A.14). Nachfolgende Kaptel beschreiben die einzelnen neuen Eigenschaften von RehaWeb.

User-Rolle RehaWeb hatte bereits für einzelne User auf der Plattform eindeutige Rollen implementiert. Diese wurden um eine neue Rolle, CORDIAALPATIENT, erweitert. Gibt es einen User, der an einem virtuellen Training teinehmen will, so muss dieser die Rolle CORDIAALPATIENT haben. Die Rollen können nur von Administratoren der Plattform im administrativen Bereich der Plattform zugewiesen werden.

Patienten-Profil Mit der neuen CORDIAALPATIENT Rolle erweitert sich auch das Profil jedes Patienten. Neben der Anschrift und den Kontaktdaten jedes Patienten ist es für das virtuelle Training nötig, von dem Patienten bestimmte Vitalwerte zu kennen. Diese werden im Vorfeld durch einen Arzt im administrativen Bereich der Platform für den Patienten eingetragen. Nur mit diesen Werten kann ein Patient ein optimales virtuelles Training durchführen. *RehaWeb-Arzt-Maske* (siehe Abbildung A.14 Der Arzt kann die Werte wie Herzfrequenz, Atemfrequenz, Sauerstoffsättigung oder Körpertemperatur in dem Patienten-Profil- Formular bearbeiten.

Virtuele Routen Mit virtuellen Routen wurden neue Ressourcen in RehaWeb eingefügt. Eine virtuelle Route wird durch die Klasse VirtualRoutes beschrieben. Die Klasse wird auf der Plattform unter dem Menüpunkt *(*Virtuelle-Strecken) aufgerufen. Jede virtuelle Strecke ist eine Klasse und hat folgenden Eigenschaften:
Neben einem Titel und einer Beschreibung hat jede Route die Angabe zu der Schwierigkeit und Distanz der Strecke, zudem kann jeder Patient Bilder zu den einzelnen Strecken hochladen, Kommentare schreiben, die Strecke als *(*Gefällt mir) anklicken und über die Plattform auf der Strecke sein nächstes Training einplanen.

Virtuelles Training Um ein Training besser planen zu können, eignet sich ein Trainingsplan. Solch einen Trainingsplan kann man auf der Plattform anlegen. Die Klasse VirtualGroup ermöglich diese Funktion auf der Plattform. Der Patient kann eine Gruppe, die aus einem bis maximal neun Mitgliedern besteht, anlegen und die Trainingszeiten eintragen. Die Klasse ist mit dem Patienten und der ausgewählten virtuellen Strecke verknüpft.

6.2.2. Client

Der Client besteht aus einer agierenden und einer reagierenden Komponente. Die agierende Komponente besteht aus den Klassen der Unity3D Umgebung und ist für die Darstellung der Umgebungselemente und das Anzeigen, der durch die reagierende Komponente bereitgestellten Vitaldaten sowie die Positionen und Bewegungen der Mitspieler.

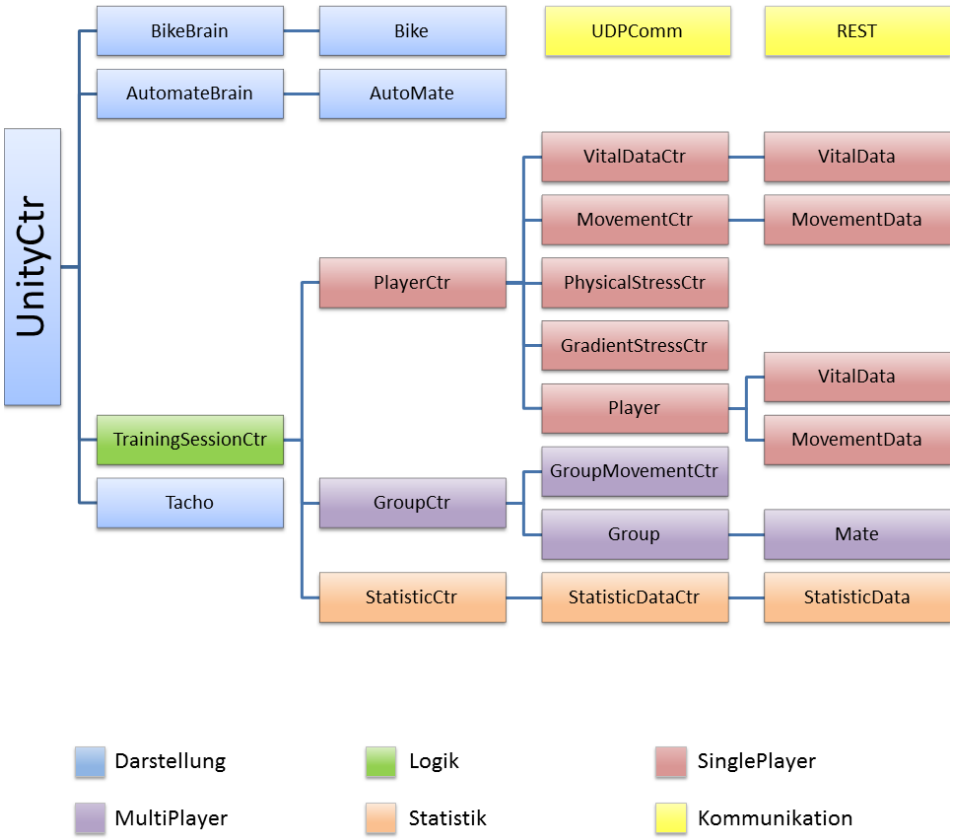

Abbildung 6.4.: Klassendiagramm der Logik des Clients

Unity3D

In diesem Teil werden die Klassen beschrieben die in Unity3D als Komponenten benutzt wurden. Die Komponenten werden alle von der Klasse MonoBehaviour abgeleitete und haben eine Update Methode die jeden Frame ausgeführt wird und eine Start Methode die äquivalent zum Konstruktor ist, der Konstrutkor und Destruktor sollte laut Unity3D Anleitung nicht benutzt werden. Die gesamte Logik der Klassen befindet in der Update Methode, einige Klasse enthalten noch Getter Methoden die aber selbsterklärend sein sollten. In unserem Projekt gibt es zwei Strecken, track-map und race-map. Auf beiden Strecken befinden sich das Objekt UnityCtrl das für die Steuerungs-Logik verantwortlich ist. Das Objekt ist ein unsichtbarerer Würfel, an diesem Würfel ist zusätzlich noch ein Bike Objekt angeheftet. Am Würfel befindet sich noch die Komponenten OcolusRift die für das darstellen des 3D Bildes verantwortlich ist. Der unsichtbare Würfel hat folgende Komponenten: UnityCtrl, GroundHpfReader, Tacho und AutoMateBrain

UnityCtrl Diese Klasse ruft die Update Methode vom TrainingSeassionController auf und übergibt ihm die aktuelle Position und Geschwindigkeit des virtuellen Fahrrads. Danach werden die Positionen der anderen Mitspieler, Daten des Brustgurts und Ergometer mithilfe der Klasse TrainingSeassionController ausgelesen und an die passenden Klassen mithilfe einer Update Methode gesendet. Hält der Patient den Bildschirm länger gedrückt beenden UnityCtrl die Klasse TrainingSeassionControlle, speichert die Statistiken in der statischen Klasse Data und ruft den Screen BorgWert auf.

AutoMateBrain AutoMateCtrl bekommt die Anzahl und Position der Mitspieler von UnityCtrl und speichert eine Liste mit Typ Mate Objekt. Sollte sich die Spieler Anzahl verringern oder erhähen wird eine Mate Objekt instanziiert oder entfernt. Nachdem die Anzahl der Mitspieler justiert wurde werden alle Mate Objekte mit der Methode SetPos auf ihre aktuelle Position gesetzt.

AutoMate Die Klasse AutoMate enthält ein 3D Modell eines Fahrrads und eines Avatars. Beim Instanziieren dieser Klasse werden die 3D Modelle erstellt und die Farbe des virtuellen Avatars geändert um die einzelnen Mitspieler unterscheiden zu können. Die Klasse AutoMate besitzt keine zusätzliche Logik. Sie dient als Schnittstelle zum 3D Modell um seine Position und Farbe zu verändern.

Tacho Tacho bekommt den Puls des Patienten, Geschwindigkeit des Ergometers und Warnungen der Lasten-Kontrolle von der UnityCtrl Klasse. Diese Daten werden auf dem Bildschirm als 2D Objekte gezeichnet. Zusätzlich wird am oberen Bildschirm eine Leiste gezeichnet die anzeigt ob der Puls des Patienten gerade zu hoch oder zu niedrig ist.

GroundHpfReader Die Komponente GroundHpfReader bekommt von UnityCtrl die aktuelle Position des virtuellen Fahrrads übergeben. Mithilfe dieser Position überprüft die Klasse auf welcher Textur sich das Fahrrad gerade befindet und schickt diesen Wert an UnityCtrl zurück. Dieser Wert wird von UnityCtrl an TrainingSeassionController übergeben.

Das Objekt Bike besteht einem 3D Modell und den folgenden Komponenten: BikeCtrl und Bike

BikeBrain UnityCtrl übergibt dieser Klasse die Geschwindigkeit und Wattzahl des Ergometers. In der Update Methode wird die Position des Lenkrad ausgelesen und die verstrichene Zeit vom letzten Update bis zum aktuellen Update gemessen. Mithilfe der Werte wird eine Interpolation berechnet, dessen Ergebnis die neue Position des Fahrrads ist. Anschließend wird die Methode SetPosition vom Objekt Bike ausgerufen die das virtuelle Fahrrad auf die neue Position setzt.

Bike Die Klasse Bike hat selber keine Logik. Sie dient nur als Schnittstelle zum 3D Modell um seine Position zu verändern.

Logik

Wie bereits beschrieben ist die Logik-Ebene ein reagierendes System, welches von der Unity3D-Umgebung aufgerufen wird um die zu visualisierenden Daten zu erhalten. Beim Aufruf werden von der darstellenden Schicht auch Positionsdaten des Spielers an den TrainingsSessionController übermittelt, damit diese an alle Mitspieler übergeben werden können um zu gewährleisten, dass der Spieler bei allen Mitspielern in Echtzeit dargestellt werden kann.

TrainingsSessionCotroller Der TrainingsSessionController dient als Schnittstelle zwischen der Unity3D-Umgebung und der Logik-Umgebung. Die Abbildung 6.5 stellt die Grundfunktion dieser Klasse dar. Bei jedem Update-Aufruf aus der Unity3D-Umgebung werden die Übertragenen Positionsdaten an den PlayerController weitergeleitet, worauf dieser ein Player-Objekt zurückgibt, welcher die aktuellen Vital- und Ergometerdaten enthält. Diese Daten werden jetzt an den Group- und StatisticController weitergeleitet um die Anzeige des Spielers bei den Mitspielern zu ermöglichen und um die Statistik zu vervollständigen. Des weiteren werden von dem Player,- Group- und StatisticController Methoden zur Verfügung gestellt die das abfragen der aktuell erstellten Objekte ermöglichen um diese dann an die Unity3D-Umgebung zu senden. Um einen reibungslosen Ablauf zu gewährleisten werden der Player- und der GroupController in Threads ausgelagert, da deren Funktionen nicht in Echtzeit abgearbeitet werden.

PlayerController Der PlayerControlle ist die Verwaltungsinstanz des SinglePlayer Bereiches. Diese Klasse besitzt das Player-Objekt, welches die aktuellen Vital- und Ergometerdaten enthält und diese dem TrainingSessionController und damit auch der Unity3D-Umgebung zur Verfügung stellt. Wie die Abbildung 6.6 zeigt, werden als erstes nach dem Update-Aufruf des PlayerControllers die Vitaldaten vom VitalDataController und die Ergometerdaten vom MovementController bezogen und im Player-Objekt gespeichert. Die bezogenen Vitaldaten werden nun an den PhysicalStressController übergeben, der mit den aktuellen Daten errechnet ob eine Warnung ausgegeben werden soll oder ob die Watteinstellungen des Ergometer-Bikes geändert werden müssen. Mit den aus der Unity3D-Umgebung übergebenen Positionsdaten werden im GradientStressController nun die Steigung und die damit verbundene Erhöhung oder Minderung der Belastung errechnet. Durch die nun vorliegenden Daten wird durch den MovementController die benötigten Einstellungen am Ergometer-Bike vorgenommen.

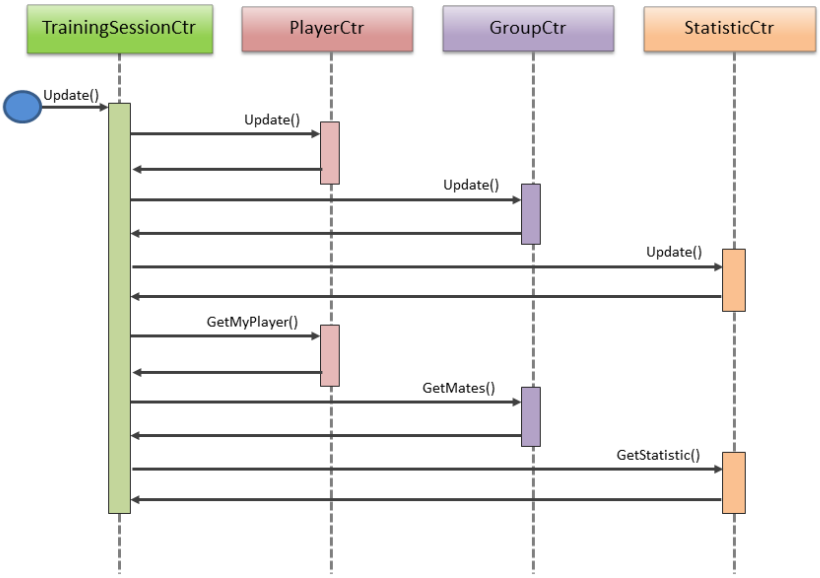

Abbildung 6.5.: Sequenzdiagramm des TrainingSessionController

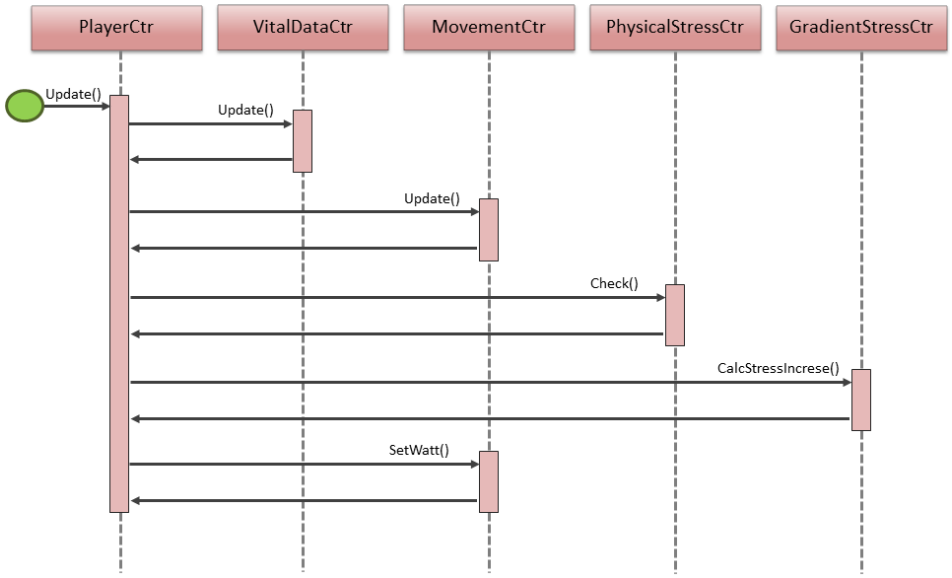

Abbildung 6.6.: Sequenzdiagramm des PlayerController

VitalDataController Der *VitalDataController* verwaltet einerseits die Vitaldaten und andererseits die Anbindung zum Zephyr BioHarness 3 und SpO_2-Modul. Der *VitalDataCon-*

troller befindet sich in der Hierarchie direkt unter dem *PlayerController* (siehe Abbildung 6.4). Für die Verbindung mit dem Bioharness 3 bzw. dem SpO_2-Modul ist die Klasse *VitalDataBioharness* bzw. *VitalDataChipOx* verantwortlich. Sobald die *Update*-Methode des *PlayerController* aufgerufen wird, wird auch die *Update*-Methode des *VitalDataController* aufgerufen (siehe Abbildung 6.6). Letztere sorgt dafür, dass das Vitaldaten-Objekt (*MyPlayer.VitalData*) des *PlayerController* aktualisiert werden und die aktuelle Werte enthalten. Wie allerdings bereits in 5.4.2 erwähnt, funktioniert die Integration des SpO_2-Moduls in die Client-Applikation nicht, sodass der SpO_2-Wert nicht aktualisiert wird und immer dem Wert 0 (%) entspricht.
Hinweis: Die Vorgehensweise bei der Implementierung des Codes für die Anbindung mit dem BioHarness 3 basiert stark auf der Arbeit der Projektgruppe 559 GlobalSensing. Für Informationen sei auf den Endbericht der Projektgruppe GlobalSensing ([**?**]) und auf deren Programmcode verwiesen.

MovementController Diese Klasse stellt die Verbindung zum Ergometer-Bike her um für das Training relevante Daten auszulesen und Einstellungen vorzunehmen. Bei der Implementierung stellten wir fest, dass die Bibliotheken von C# die für den Datenaustausch mit einer Seriellen Schnittstelle zu Verfügung gestellt sind einige Probleme bei dem modifizierten Kabel der Firma Daum electronic aufwies. Dieser wurde gelöst, indem die Kommunikation zum ergo_bike 2002 über eine in Java geschriebene Applikation hergestellt wird und diese über ein Socket diese Daten an den MovementController sendet. Durch diese Änderung ist der MovementController zu einer Schnittstelle zwischen der Java-Applikation und dem PlayerController geworden.

PhysicalStressController Der PhysicalStressController ist die medizinische Überwachung des Trainierenden und wird bei jedem Aufruf des PlayerControllers durchgeführt. Die Methode gibt eine PhysicalStressInstuction zurück, die Zustände annehmen kann um eine Warnung auszugeben, um eine Anpassung der Watt-Zahlen anzufordern oder um einen Alarm und den Trainingsabbruch zu fordern. Da der Aufruf des PhysicalStressControllers unregelmäßig ist werden Zeitmesser verwendet um einen korrekten zeitlichen Ablauf zu gewährleisten. Der Herzfrequenzbereich der Trainierenden wird in Bereiche unterteilt und jeder Bereich außer dem Normal-Bereich mit einem Timer versehen. Wenn jetzt die Herzfrequenz in einen nicht normalen Bereich fällt oder steigt wird der entsprechende Timer aktiviert und wenn innerhalb einer fest definierten Regenerierungsperiode von 10 Sekunden keine Normalisierung der Herzfrequenz stattfindet wird die jeweilige Anweisung (Warnung, Anpassung der Watt-Zahl oder Alarm) ausgegeben. Sollte sich innerhalb der Regenerierungsperiode die Herzfrequenz normalisieren (durch Befolgen der Warnung oder durch die Anpassung der Watt-Zahl) wird der entsprechende Timer zurückgesetzt.

GroupController Der GroupController ist dazu da um die Liste der Mitspieler zu verwalten und kontrollieren. Bei einem Aufruf der Update Methode wird ein Player-Objekt übergeben um die eigenen Daten an den Server zu senden. Dieser wird an den GroupMovementController übergeben und im Gegenzug wird eine Liste von Mitspielern zurückgegeben. Diese Liste wird nun mit der aktuell vorhandenen Liste abgeglichen, da in der zurückgegebenen Liste des GroupMovementControllers auch mehrere oder kein Objekt eines Mitspielers vorhanden sein kann. Die Objekte der vorhandenen Mitspieler werden jetzt mit dem neusten Objekt aktualisiert und neue hinzugefügt. Bei jeder Aktualisierung oder Hinzufügung wird ein Zeitstempel gesetzt, da ein Austritt aus dem Training oder ein Abbruch eines Mitspielers nicht vom Server gesendet wird, wird ein Mitspieler nach einer Toleranzperiode von 3 Sekunden in der keine Nachricht vom entsprechenden Mitspieler angekommen ist aus der Liste gelöscht.

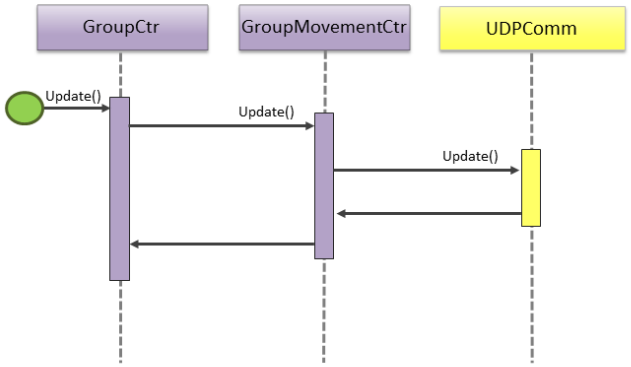

Abbildung 6.7.: Sequenzdiagramm des GroupController

GroupMovementController Die Klasse GrupMovementController beinhaltet drei Methoden. Die Methode update von Typ List<Mate> bekommt die aktuellen myPlayer Daten. Es wird sozusagen in die List<Mate> hinzugefügt. Diese Daten sind zum Beispiel ID, Name, Training Time, Positionen, Directionen, Accelrations und Distance. Die Daten von Typ myPlayer werden zu einem String konvertiert und über die UDP Kommunikation versendet. Solange Nachrichten empfangen werden können, werden diese Strings zu ConvertTouserData umgewandelt und in die Mate Liste aufgenommen. Die ConvertTouserData Methode bekommt einen String. Dieser String wird gesplittet und geparst. Die Methode ConvertToString bekommt von Typ MyPlayer myplayer und wandelt diese in String um.

UDPCommunication Die Klasse UDPCommunication impliziert Methoden zum Senden und zum Empfangen. Zum einen die Mehode send() und zum anderen die Methode receive(). Ein UDP Client kann Nachrichten senden und empfangen. Dabei wird in der

Funktion	Name der Methode	noch frei
Login	checklogin()	GET
virtuelle Routen abrufen	GetRoutes()	GET
virtuelle Gruppen abrufen	GetGroups()	GET
virtuelle Gruppe beitreten	AddIdToGroup()	POST
virtuelle Gruppe erstellen	CreateGroup()	POST
Statistiken abrufen	GetStatistik()	GET
Statistik senden	PostStatistik()	POST

Tabelle 6.1.: Client REST-Schnittstellen und ihre Methoden

Methode send() das String Message in Byte umgewandelt und verschickt. Der Empfänger erhält diesen Byte und wandelt ihn wieder in einem String um. Es ist zu beachten, dass die ReceiveVoice() Methode die Nachrichten, die in der Warteschlange sind und nicht aus der Warteschlange rauskommen können, durch die Ausnahmebehandlung (Try/Catch Block) vermieden werden. Wir haben einen ReceiveTimout für 10 Sekunden gesetzt, d.h. wenn der Empfänger innerhalb von 10 Sekunden keine Nachricht erhält, kommt er aus der ReceiveVoice() Methode raus.

REST Dieser Abschnitt der Implementierung beschreibt die Klasse Rest der Client-Anwendung inklusive ihrer Methoden. Die Aufgabe dieser Klasse besteht darin, den Datenaustausch zwischen der Ergometer Clientanwendung und dem Datenbankserver zu realisieren. Eine grundlegende Funktionsbeschreibung der einzelnen Schnittstellen ist im Vorfeld schon beschrieben worden. Im Folgenden wird nun die Implementierung der zugehörigen Methoden im Detail erläutert. Um die Übersichtlichkeit zu gewährleisten ist in Tabelle 6.1 jeder Schnittstelle ihre dazugehörige Methode zugeordnet.
Der Datenaustausch zwischen Ergometer Clientanwendung und dem Datenbankserver basiert auf REST-Schnittstellen. Um diese Aufgabe zu erfüllen und die Daten senden und empfangen zu können, ist es notwendig C# Objekte der Clientanwendung zu serialisieren und ins JSON-Format zu konvertieren und anderseits Daten, welche vom Server gesendet werden vom JSON-Format in Objekte zu deserialisieren. Um diesen Teilschritt zu bewältigen gibt es eine Vielzahl von fertigen Bibliotheken für C#. Die bekanntesten und meist empfohlenen sind JSON.net oder fastJSON. In unserem Fall war es leider nicht möglich einfach eines dieser Programme auszuwählen. Von Unity3D gibt es zwei wesentliche Einschränkungen, welche die Auswahl solcher Hilfsprogramme auf eine sehr kleine Auswahl reduziert. Zum einen darf die verwendete .Net Version nicht größer als 2 sein und zum anderen ist es in der kostenfreien Version von Unity3D nicht möglich fertige Bibliotheken über dlls einzubinden. Diese Möglichkeit besteht erst mit der Pro Version. Die meisten, besten und weitverbreitesten JSON-Konverter liegen aber meistens nur als dll-Bibliothek

vor oder nutzen eine weit höhere Version von .Net. Ein Konvertierungsprogramm welches diese Bedingungen aber dennoch erfüllt und auch in der Unity3D Community empfohlen wurde ist LitJSON. Die Klassen des Programms finden sich im Ordner LitJSON in der Projektumgebung. Um ein JSON-String zu einem Objekt zu deserialisieren ruft man die Methode *LitJson.JsonMapper.ToObjec()* auf und übergibt den zu konvertierende JSON-String. Als Rückgabewert erhält man das deserialisierte C#-Objekt. Müssen Daten an den Server übermittelt werden, welche als Objekt vorliegen müssen diese vor der Versendung mit dem Befehl *LitJson.JsonMapper.ToJson()* in einen JSON-String serialisiert werden. Bei dem Aufruf der Methode wird in diesem Fall das zu sendende Objekt übergeben und als Rückgabewert erhält man den fertigen JSON-String. Allerdings sind beim serialisieren und deserialisieren von komplexeren Objekten und einigen Datentypen Fehler aufgetreten. Um die Probleme zu beseitigen, ist es notwendig gewesen, zum einen Objekte vor der Serialisierung zu bearbeiten und einige Datentypen bei denen es zu Problemen gekommen ist im Vorfeld über Hilfsklassen für die Konvertierung vorzubereiten. Auf der anderen Seite sind auch die JSON-Strings, welche vom Datenbankserver gesendet worden sind teilweise auch in einem nicht direkt deserialiesrbaren Format angekommen, so dass auch hier vor der Deserialisierung zu Objekten Vorarbeiten geleistet werden mussten. Im folgenden werde ich daher die einzelnen notwendigen Aufbereitungen der jeweiligen Methoden für die einzelnen Schnittstellen zusätzlich erläutern, denn teilweise sind unterschiedliche Anpassungen von Nöten gewesen. Eine grafische Darstellung der Schnittstellen für die GET-Schnittstellen ist in Abbildung 6.8 dargestellt und analog in Abbildung 6.9 für die POST-Schnittstellen.

Rest.checklogin() Diese Methode benötigt zum Aufruf einen Benutzernamen und ein Passwort als String. Die Daten werden von dem jeweiligen Patienten in der Login-Eingabemaske der Ergometer Clientanwendung eingegeben. Diese Informationen werden in Form von *Network Credentials* eines *http.web.requests* über die checklogin REST-Schnittstelle an den Server übertragen, welcher diese im weiteren Überprüft. Ist die Kombination aus Benutzername und Passwort korrekt, erhält man als Antwort den HTTP-Statuscode 200 und im Messagebody die Userdaten in Form eines JSON-Strings. Der JSON-String wird ausgelesen und mit Hilfe von LitJSON in ein Objekt vom Typ UserDaten deserialisiert. Dieses Objekt enthält alle relevanten Informationen, welche für das spätere Ergometertraining notwendig sind. Fällt die Prüfung negativ aus oder ist der Server nicht erreichbar, erhält man als Antwort einen Statuscode ungleich 200 (HTTP 500) und der Rückgabewert ist *null*.

Da wie bereits erwähnt, der Datenbankserver alle Werte als Strings interpretiert und sendet, wird der JSON-String von LitJSON zuerst in ein Hilfsobjekt deserialisiert, in welchem eben alle Attribute als Strings vorhanden sind. über die Methode *UserDatenAufbereiten()* wird dann das Hilfs-Objekt in das gewünschte Objekt vom Typ UserDaten überführt.

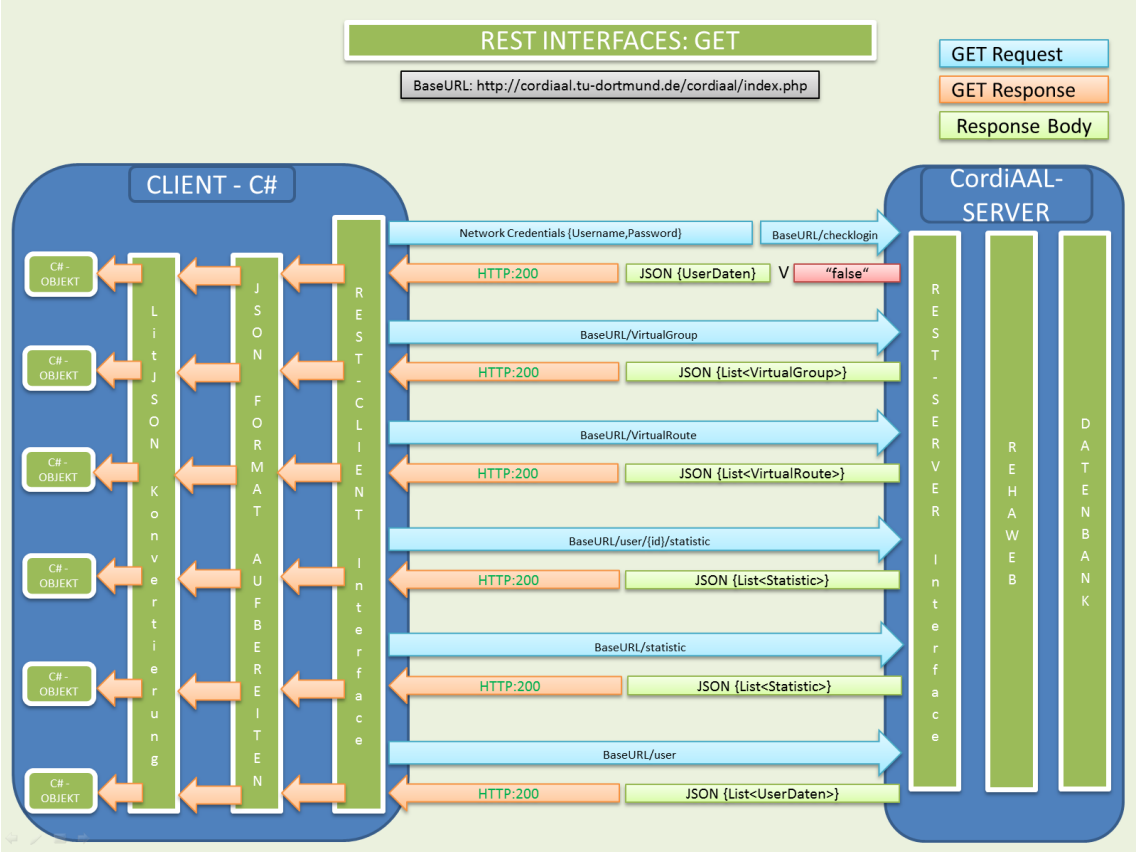

Abbildung 6.8.: REST-Schnittstellen für GET

Rest.CreateGroup() Wird von einem Patienten ein neues Gruppentraining erstellt, wird diese Methode aufgerufen. Als Übergabewert wird ein Objekt vom Typ *VirtualGroup* erwartet. Dieses Objekt kann von LitJSON aber ohne weitere Anpassung nicht serialisiert werden. Innerhalb des Objekts sind zwei Attribute vom Typ DateTime. Diesen Datentyp kann LitJSON nicht korrekt serialisieren. Zusätzlich wird der Wert für die GruppenID entfernt, da die ID erst nach erfolgreichen erzeugen in der Datenbank zugewiesen wird. Aus diesem Grund wird vor der Serialisierung durch LitJSON mit der Methode *VirtualGroupUmbereiten()*, das Objekt so angepasst, dass es Fehlerfrei von LitJSON umgewandelt werden kann. Der fertig konvertierte JSON-String wird dann in den Request Body des *http.web.requests* geschrieben. Als REST-Methode wird in diesem Fall POST benutzt, da bei dieser Methode Daten an den Server versendet werden. Bei erfolgreicher Erstellung der Gruppe erhält man den HTTP Statuscode 200 und im Response Body die ID der erstellten Gruppe. Andernfalls bekommt man als Antwort -1 im Response Body gesendet.

Rest.AddIdToGroup() Hat sich ein Patient entschieden an einem bestehenden virtuellen Gruppentraining teilzunehmen, wird über diese Methode die ID des Patienten zu einer virtuellen Trainingsgruppe hinzugefügt und in der Datenbank vermerkt. Der

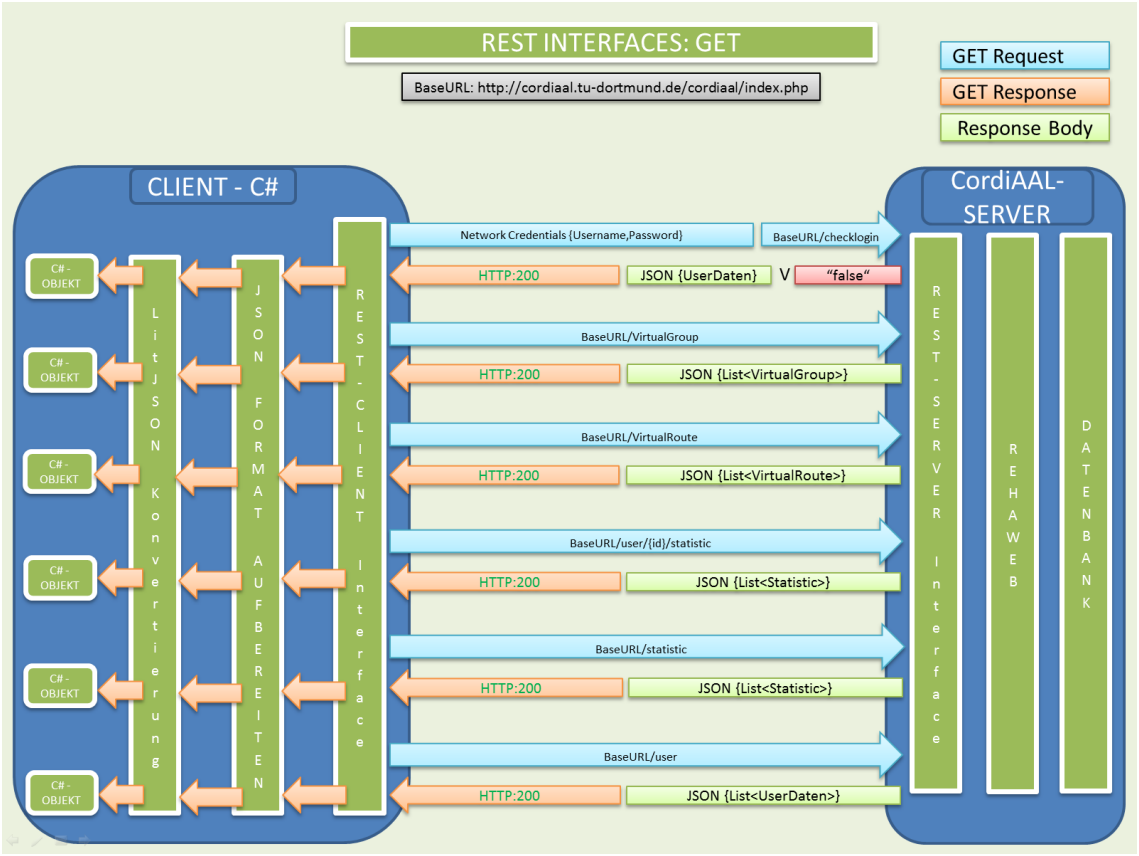

Abbildung 6.9.: REST-Schnittstellen für POST

Aufruf erfolgt mit der gewünschten *VirtualGroupID* und der ID des Users. Bei erfolgreichem Hinzufügen des Patienten zu der gewünschten Gruppe, wird die ID der Gruppe als String und der HTTP-Statuscode 200 zurückgegeben. Eine Serialisierung von LitJSON ist hier nicht erforderlich, da der JSON-String in diesem Fall direkt erstellt werden kann und auch kein Objekt für die Serialisierung zur Verfügung steht.

Rest.GetGroups() Diese Methode fordert eine Liste aller aktuellen und zukünftigen Gruppen vom Datenbankserver an. Die Methode erwartet daher keinen Wert und liefert vom Datenbankserver eine Liste von Objekten vom Typ *VirtualGroup* zurück. Da hier die die Methode GET für den *http.web.request* verwendet wird muss dieses nicht explizit angegeben werden, da bei keiner Angabe GET als default gewählt wird.
Nachdem der JSON-String vom Datenbankserver im Respone Body gesendet worden und ausgelesen worden ist, muss er vor der Deserialisierung bearbeitet werden, da er in diesem Format nicht von LitJSON korrekt interpretiert werden kann. Zum einen sind wie erwähnt alle Werte als String gekennzeichnet und es sind von der Datenbank Zeichen hinzugefügt worden, welche vorerst entfernt werden müssen. Zu diesem Zweck wird der String an die Methode *Aufbereiten()* übergeben. Die Methode wird im Folgenden noch genauer erläutert. Ist der JSON-String aufbereitet worden kann dieser

von LitJSON in eine Liste von Hilfs-Objekten deserialisiert werden. Hilfs-Objekte, da alle Attribute von der Datenbank als Strings ausgeben worden sind. Diese Liste von Hilfs-Objekten vom Typ *VirtualGroupHelper* wird daraufhin von der Methode *VirtualGroupAufbereiten()* in eine Liste von *VirtualGroup* Objekten konvertiert. Schliesslich kann die Liste von als Rückgabewert übergeben werden.

Rest.GetRoutes() Den Patienten stehet eine Auswahl verschiedener virtueller Strecken für ihr Training zur Verfügung. Um im Ergometer Client eine Liste aller virtuellen Strecken abzurufen, wird diese Methode benutzt. Diese Methode funktioniert vom Ablauf her ähnlich wie die Methode GetGroups(). Die Unterschiede bestehen darin, das der Rückgabewert eine Liste mit Objekten vom Typ *VirtualRoute* sind und für die Aufbereitung nach der Deserialisierung die Methode *VirtualRouteAufbereiten()* genutzt wird.

Rest.GetStatistic() Diese Methode ruft eine Liste aller Statistiken für einen bestimmten Patienten ab, daher wird bei dieser Methode auch die ID des Patienten mit übergeben, welche dann in der URL für die REST-Schnittstelle eingesetzt wird. Da es sich hier bei dem Rückgabewert wiederum um eine Liste von Objekten handelt, ist die Vorgehensweise genau so wie bei den beiden anderen Methoden, welche zuvor erläutert worden sind. Der Methodenaufruf um die Liste von Hilfsobjekten zu konvertieren lautet in diesem Fall aber "*StatistikAufbereiten()*" um aus der Liste von *StatisticHelper* Objekten eine Liste von *Statistic* Objekten zu machen. Zusätzlich wäre noch zu erwähnen, dass es sich bei den Statistik Daten nicht um die Vollständen Werte handelt welche nach Trainingsabschluß übermittelt werden, sondern nur um eine abgespeckte Version, damit im laufe der Zeit, wenn sich eine Vielzahl von Statistiken angesammelt haben, der Abruf nicht allzu lange dauert.

Rest.PostStatistic() Um die Daten welche bei einem Training aufgezeichnet worden sind in der Datenbank zu speichern muss diese Methode aufgerufen werden. Bei ihrem Aufruf muss das Objekt vom Typ *Statistic* mit übergeben werden. Dieses Objekt beinhaltet alle Daten die während des Trainings aufgezeichnet worden sind plus zusätzliche Informationen wie bestimmte Maximal, Minimal und Durchschnittswerte. Da hier Daten an den Server gesendet werden, wird hier als Request-Methode POST verwendet. Das Objekt wird zuerst von LitJSON zu einem JSON-String serialisiert und in den Request-Body des *http.web.request* geschrieben. Darüber hinaus wird der String zur Sicherheit auf der lokalen Festplatte gespeichert (C:\CordiAAL\Statistik). Wenn die gesendete Statistik erfolgreich in der Datenbank gespeichert werden konnte, gibt es als Response den HTTP-Statuscode 200 und im Response-Body wird die ID der gespeicherten Statistik übermittelt. Diese ID ist auch gleichzeitig der Rückgabewert der Methode. Bei Misserfolg gibt die Methode null zurück.

Wie die praktische Abfolge beim beim Senden einer Statistik abläuft wird im Folgenden erläutert: Sobald das Training startet, werden ab der ersten Sekunde, von der Anwendung die Daten für die Statistik des Users erhoben. Die Statistik speichert Daten sowohl von den extern angeschlossenen Sensoren sowie virtuelle Informationen aus der Trainingsanwendung selbst, wie zum Beispiel Höhenmeter und gefahrene Distanz.
Nach Beendigung des Trainings wird der User aufgefordert den Borg-Wert einzugeben. Dies ist der letzte Wert, welcher für die Statistik benötigt wird. Nach Eingabe des Werts durch den User, kann die Statistik an den Server zur Speicherung in der Datenbank versendet werden.
Sollte bei dem Versenden der Statistik-Daten ein Fehler auftreten, wodurch die Daten nicht an den Server übermittelt werden können, werden sie lokal auf dem PC des Users gespeichert, damit sie zu einem späteren Zeitpunkt übermittelt werden können. Auf diese Weise gehen die wichtigen Trainingsdaten des Users nicht verloren.

Aufbereiten() Die Methode wird genutzt um JSON-Strings, welche von der Datenbank kommen zu bearbeiten, damit sie von LitJSON interpretiert und deserialisiert werden können. Zuerst werden zusätzlichen Zeichen entfernt ("1":, "2":, ...). Dies geschieht über einen regulären Ausdruck. Daraufhin werden geschweifte Klammern am Anfang und Ende des Strings entfernt. Im letzten Schritt werden eckige Klammern an die erste und letzte Position geschrieben.
Diese Prozedur muss nur bei JSON-Strings angewendet werden, welche eine Liste von Objekten beinhalten.

6.2.3. Anwendungsserver

Der Anwendungsserver besteht aus zwei Servern die jeweils eine Kommunikation übernehmen. Der PositionServer empfängt und sendet die Position und Bewegungsdaten und sendet diese an die jeweiligen Gruppenmitglieder. Der VoiceChatServer ist in seiner Funktionalität recht ähnlich aufgebaut und empfängt anstatt von Positionsdaten die gebufferten VoiceChat-Daten und sendet diese an die zugehörigen Gruppenmitglieder weiter.

PositionServer

Der PositionServer ist wie gerade beschrieben dazu da um die Daten die für den Ablauf eines Gruppentrainings nötig sind auf alle Gruppenmitglieder in Realzeit zu verteilen. Aus diesem Grund wurde auf alles unnötige verzichtet um keine größeren Auswirkungen auf die Antwortzeit zu erzeugen. Das Abbild 6.10 zeigt den kompakten Aufbau des PositionServer.

Server Die Klasse Server startet die Anwendung und gibt die Möglichkeit den Port auf dem gearbeitet wird und die Paketgröße der einzelnen UDP-Pakete einzustellen. Diese

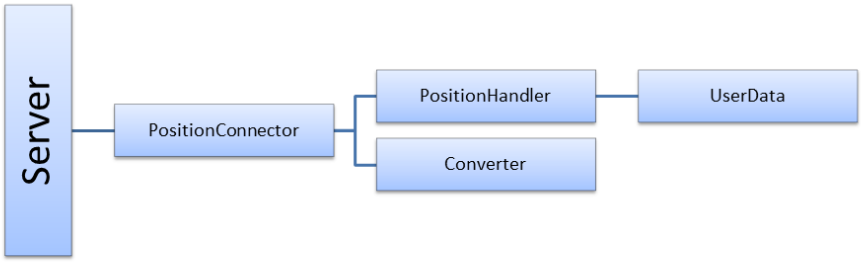

Abbildung 6.10.: Klassendiagramm des PositionServer

werden durch Kommandozeilenargumente übergeben oder wenn nichts beim Ausführen eingegeben wurde mit Standardwerten versehen.

PositionConnector Der PositionConnector hat die Funktion die Verbindung zum Socket herzustellen und auf eingehende Nachrichten von Benutzern abzuwarten. Jede eingehende Nachricht wird abgefangen und der empfangene String durch die Converter-Klasse in das UserData-Objekt umgewandelt. Danach wird das UserData-Objekt an den PositionHandler übergeben, der darauf eine Liste von UserData-Objekten zurückgibt, die Daten von allen Gruppenmitgliedern enthält, die zum Benutzer passen von dem die Nachricht eingegangen war. Diese Liste von Daten wird nun durch die Converter-Klasse wieder in einen String umgewandelt und per UDP an den Benutzer geschickt.

PositionHandler Der PositionHandler ist dazu da um die UserData-Objekte zu verwalten. Jedes Objekt enthält als Kennung eine eindeutige ID und eine GruppenID. Sobald ein UserData-Objekt übergeben wird, wird die Liste nach Objekten mit der gleichen GruppenID durchsucht und zurückgegeben. Damit veraltete oder ausgeloggte Benutzer nicht weiterhin gesendet werden, wird bei Eingang jedes Objektes die Liste nach Objekten überprüft, die seit einem Zeitintervall von drei Sekunden nicht aktualisiert wurden und löscht diese aus der Liste. Bei jeder Aktualisierung eines Objektes wird der Zeitstempel neu gesetzt.

VoiceChatServer

Der VoiceCahtServer ist sehr ähnlich aufgebaut und der einzige Unterschied besteht darin, dass bevor die VoiceChat Kommunikation beginnt eine Nachricht mit der GruppenID gesendet wird anhand dieser und der dazugehörigen IP wird das UserData-Objekt erstellt

und ab diesem Zeitpunkt werden nur noch die VoiceChat-Daten übertragen und immer nach der IP einer Gruppe zugeordnet.

6.3. 3D Modellierung

Für die Entwicklung unseres Projektes haben wir uns für Unity3D entschieden. Hauptgrund für das Arbeiten mit Unity3D war, dass ein Projektteilnehmer bereits erste Erfahrungen mit der Spiel-Engine sammeln und somit der Gruppe einen Überblick über die Funktionen und Möglichkeiten der Entwicklungsumgebung vermitteln konnte. Ein großer Vorteil von diesem Programm ist, dass es weit verbreitet ist und mittlerweile eine große Comunity besitzt. Aufgrund der zahlreichen Seiten, Foren und Videos erweist sich der Einstieg in die Entwicklungsumgebung sehr angenehm und schnell. Man findet nahezu für jedes Thema fertige Tutorien und kann somit viele Ideen und fertige Programmcode direkt in sein Projekt übernehmen und nach Belieben anpassen. Nachdem man ein Projekt in Unity erstellt hat, wird automatisch eine Ordnerstruktur erzeugt. Dort befindet sich auch der Asset-Ordner. In diesen Asset-Ordnen können bspw. 3D-Modelle, Animationen, Audiodateien und vieles mehr eingefügt werden. Ein Projekt besteht aus mindestens einer oder mehreren Szenen. Diese Szenen können wiederum miteinander verknüpft werden. Zur Gestaltung der Szenen liefert Unity3D eine Menge Werkzeuge und nützliche Skripte. Zusätzlich besteht die Möglichkeit, sich im Asset-Store oder auf verschiedenen Seiten der Comunity, Add-Ons zu installieren (Z.B. EasyRoad3D oder TerrainToolkit). Für die Logik und den Ablauf des Spiels wird von Unity3D standardmäßig die Entwicklungsumgebung von MonoDevelop mitgeliefert. Hier kann man mit JavaScript oder C# seinen eigenen Code schreiben und dadurch das Spiel mit Funktionen ergänzen. Dort erzeugte öffentliche Variablen werden auch in der Unity3D Entwicklungsumgebung angezeigt. Diese können wiederum mit Objekten und Komponenten aus der virtuellen Welt verknüpft werden. Im weiteren Verlauf dieses Abschnitts wird ein kleiner Einblick in die Welt von Unity3D gewährt und gezeigt, was wir im Detail mit Unity3D gemacht haben.

6.3.1. Erstellen einer Karte

In Unity wird eine 3D-Landschaft auch Terrain genannt. Standardmäßig ist eine Größe von 2000x2000 angegeben. Eine Längeneinheit in Unity entspricht dabei einem Meter in der realen Welt. Zum Terrain muss als nächstes eine Lichtquelle hinzugefügt werden. Hierbei hat man die Wahl zwischen folgenden Komponenten:

- Directional Light
- Spotlight
- Area Light

- Point Light

Letzteres ist dabei am besten geeignet, um die Sonne für unser 3D-Terrain zu simulieren. Fügt man nun diese Lichtquelle in unser Projekt ein, erscheint diese neben unserem Terrain in der Hierarchie. Dort finden sich alle Objekte, die man seiner Szene hinzugefügt hat. Rechts neben der Hierarchie befindet sich die Ordnerstruktur unseres Projektes. Hier findet man die in Unity bereitgestellten Standard Assets, Skripte, Materiale und vieles mehr. Des Weiteren ist es möglich, einen eigenen Ordner anzulegen und seine Dateien zu speichern. Nachdem wir unser Terrain und eine Lichtquelle erstellt haben, können wir anfangen, unser Terrain nach Belieben zu gestalten. Wählen wir in der Hierarchie unser Terrain aus, erscheinen im Inspector eine Reihe von Werkzeugen, um das Terrain zu bearbeiten.

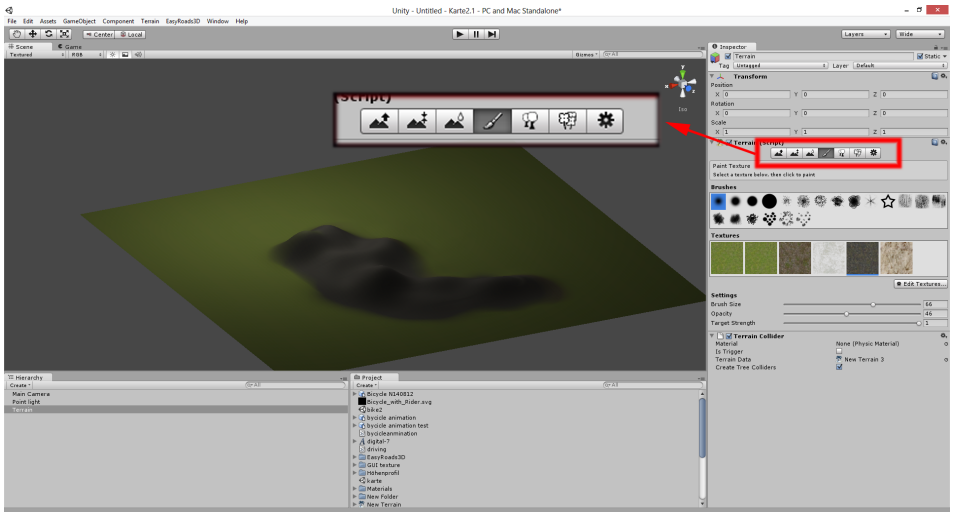

Abbildung 6.11.: Werkzeugleiste zum Bearbeiten der Landschaft in Unity3D.

Als Erstes ist es sinnvoll, dem Terrain mit Hilfe des Paintwerkzeuges eine Textur zu verleihen. Diese hinzugefügte Textur benutzt Unity als Untergrund für das Terrain. Es ist möglich, beliebig viele Texturen in sein Terrain einzufügen und so eine Landschaft seiner Wahl zu gestalten. Mit den ersten drei Werkzeugen in der linken Spalte kann man das Höhenprofil bearbeiten. Mit einem weiteren Werkzeug kann man ganze Wälder, einzelne Bäume oder Pflanzen platzieren. Dabei stellt Unity für den Anfang eine Reihe von Bäumen und Pflanzen zur Verfügung. Diese können durch Modelle aus dem Asset-Store oder im Web ergänzt werden. Zur Gestaltung von Straßen oder Wegen kann man das Paintwerkzeug benutzen oder sich zur Hilfe ein Add-On installieren. Wir haben uns beim Einfügen von Straßen eines Add-Ons namens EasyRoad3D entschieden. Möchte man nun in seinem Terrain Häuser, Brücken oder sonstige 3D-Modelle einbauen, gibt es zwei Möglichkeiten. Entweder man installiert ein 3D-Modellierungsprogramm und erstellt mühselig seine eigenen Modelle oder man bedient sich der zahlreichen Modelle im Web. Diese findet man problemlos auf verschiedenen Internetseiten. Unity unterstützt dabei eine Menge von

Formaten (.3ds, .fbx, .3ds). Nach dem Download müssen die 3D-Modelle lediglich in den Asset- Ordner des Projektes gespeichert und in die jeweilige Szene geschoben werden. Sie erscheinen nun direkt im Terrain und können auf die benötigte Größe skaliert werden. Hin und wieder kommt es vor, dass bei dem Import das Material verloren geht und nur graue Körper dargestellt werden. Für unser Projekt haben wir zwei Teststrecken erstellt und alle Funktionen ausprobiert. Jedoch sind wir zu dem Entschluss gekommen, ein fertiges Terrain für die Präsentation aus dem Internet zu benutzen, da das Erstellen einer detailreichen Karte mit einem sehr hohen zeitlichen Faktor verbunden ist und wir dennoch nicht auf das 3D-Erlebnis eines detailreichen Terrains verzichten wollten. Abschließend lässt sich konstatieren, dass es zwar nicht besonders schwierig ist, sein eigenes Terrain zu erstellen, es allerings mit einem hohen zeitlichen Aufwand einhergeht und viel Kreativität erforderlich ist.

6.3.2. Avatar

Nachdem wir unser Terrain fertiggestellt haben, begannen wir mit der Entwicklung eines Avatars. Dieser sollte die physikalischen Eigenschaften eines Fahrrads besitzen und sich in unserem Terrain frei bewegen lassen. Die Beschleunigung sollte über das Ergometer und die Lenkung über unser eingebautes Fahrradlenkrad erfolgen. Für unseren Avatar mussten wir zunächst ein geeignetes 3D-Modell finden, ihn in den Asset-Ordner ablegen und anschließend in unser Terrain einfügen. Damit unser Modell den Gesetzen der Schwerkraft unterliegen kann, fügten wir ihm einen Rigidbody hinzu. Nun sollte unser Modell nicht mehr in der Luft schweben. Leider blieb er nicht auf dem Terrain, sondern flog durch das Terrain hindurch. Das kommt dadurch, dass unser Modell zwar in Unity sichtbar ist, aber keine Kollisionen erkennt. Auch hier können wir uns den fertigen Komponenten von Unity bedienen und unserem Modell ein Box Collider hinzufügen. Dieser sollte auf die Größe unseres Modells angepasst werden. Durch diesen Schritt fällt unser Modell nicht mehr durch das Terrain, sondern erkennt Kollisionen. Für die Simulation von jeglichen Modellen mit Rädern, stellt Unity sogenannte Wheel Collider zur Verfügung. Auch diese sind unter Component zu finden. Hierzu erstellen wir zunächst ein leeres Gameobjekt und fügen diesem zwei Wheel Collider hinzu. Diese werden auf unser Modell gezogen. Dabei platzieren wir die Wheel Collider so, dass sie in den Rädern des Fahrrads liegen.

Die Wheel Collider können über ein Skript oder C# Code angesprochen werden. Da wir uns in unserem Projekt für eine objektorientierte Programmiersprache entschieden haben, benutzen wir einen C# Code. Als Programmierumgebung stellt uns Unity, MonoDevelop zur Verfügung. Es legt uns automatisch zwei leere Methoden an:

- Start()- Methode: Wird beim Start der Unityanwendung aufgerufen, um Methoden und Variablen zu initialisieren

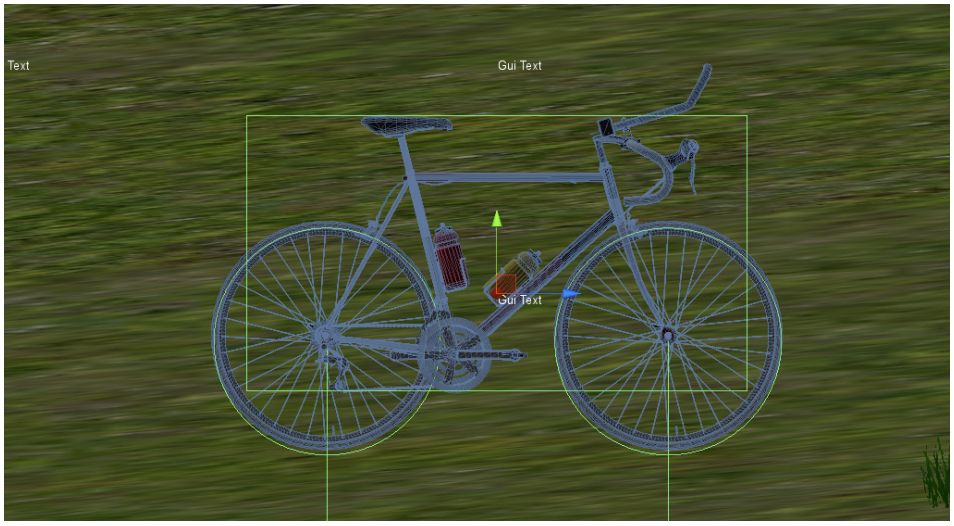

Abbildung 6.12.: Die Wheel Collider wurden exact in unser 3D-Modell eingefügt. Die beiden Veritkalen Striche unter den Rädern zeigen sie Stärke unser Stoßdämper. und das Reechteck um das Fahrrad ist unser Box Collider. Dieser ist für Kollisionen zuständig

- Update()- Methode: Wird nach dem Start 50 mal pro Sekunde aufgerufen und aktualisiert unsere Parameter

In die Update-Methode schreiben wir unseren Code zum Steuern der Wheel Collider. Alle öffentlich erzeugten Variablen werden uns in der Entwicklungsumgebung von Unity direkt angezeigt. Diese findet man auf der rechten Seite im Inspector des aktuell ausgewählten Objektes. Wir erzeugen zwei Wheel Collider Variablen. Dafür stellt uns Unity eine vorgefertigte Bibliothek zum Steuern zur Verfügung. Über bestimmte Methoden und Parameter können wir die Umdrehungsgeschwindigkeit der Räder steuern. Dazu holen wir uns die Geschwindigkeit des Ergometers aus unserer Controller Klasse. Dort wird die Verknüpfung zwischen dem Fahrrad und unserem virtuellen Training in Unity hergestellt. Für die Lenkung bedienen wir uns einer Variable der Wheel Collider Klasse. Über die Notation FrontWheel.steerAngle können wir den Winkel des Richtungswechsels eingeben, wobei wir bei dem Wert 0 geradeaus fahren und mit 90 nach rechts lenken. Wiederum greifen wir die zuständige Variable in unsere Controller Klassen, die den Winkel des Lenkrades abfängt. Auf der Homepage von Unity findet man ein Liste, der zur Verfügung gestellten Bibliothek. Unser Modell bewegt sich nun, aber wenn wir beschleunigen oder lenken, bleiben die Räder und das Lenkrad unseres Modells starr. Um das Drehen der Räder beim Beschleunigen an unsere Geschwindigkeit anzupassen, kann man bei den Wheel Collidern auf die rpm Variable zugreifen. Diese gibt uns an, wie oft sich unser Rad pro Minute dreht. Diesen Wert müssen wir für unser Vorhaben noch ein wenig anpassen. Wir wollen wissen, um wieviel Grad sich unser Rad pro Sekunde dreht. Dazu erstellt man eine Transform Variable und setzt dort das Rad vom 3D-Modell ein. Mittels der Transform Variable können wir

wiederum bestimmte Werte während der Laufzeit verändern. Interessant ist für uns der Rotationswert. Wir können mit dem Befehl TransformObjekt.rotate unser ausgewähltes Objekt um die X,Y und Z Koordinaten rotieren lassen. Das gleiche Prinzip benutzen wir bei der Lenkung. Wir fassen alle relevanten Teile zu einem Objekt zusammen (Vorderrad, Lenkrad, Gabel) und erzeugen dafür eine Transform Variable. Diesmal benutzen wir allerdings für den Winkel unserer Lenkung die Steer Angle Variable von unserem Wheel Collider für das Vorderrad.

Die Animation des Radfahrers haben wir mit Blender vorgenommen. Theoretisch würde es auch mit Unity funktionieren, doch Blender erwies sich hier als vorteilhafter, da es für die Entwicklung einer Animation wesentlich mehr Funktionen zur Verfügung stellt und uns somit viel Arbeit beim Animieren abnimmt. Der genaue Ablauf findet sich im Kapitel von Blender. Als Nächstes widmen wir uns der GUI im Spiel.

6.3.3. GUI

In unserer GUI wollen wir eine Reihe von Vitalparametern während des Spiels anzeigen lassen. Diese sollen den Patienten helfen, sich an die vorgegebenen Werte vom Arzt zu halten und somit eine optimale Auslastung während des Trainings zu haben. Hierzu müssen folgende Werte auf dem Bildschirm angezeigt werden:

- Puls des Patienten mit seinen vorgeschriebenen Grenzen
- SpO2 Wert
- aktuelle Wattleistung des Ergometers

Des Weiteren werden Werte zur Überwachung der aktuellen Leistung angezeigt.

- die aktuelle Geschwindigkeit
- gefahrene Kilometer
- absolvierte Zeit im Training
- ein Höhenprofil, das den Patienten die absolvierten Höhenmeter anzeigt

Falls der Patient sich außerhalb seiner optimalen Werte befindet, wird eine Hilfe auf dem Bildschirm erscheinen und den Patienten darauf hinweisen, je nach Situation, entweder langsamer oder schneller zu fahren. Für das Gestalten der GUI bietet Unity auch hier bereits fertige Objekte an, die uns viel Arbeit abnehmen.

- GUI Text: Zeigt einen Text auf dem Bildschirm an
- GUI Lable: Füllt eine Fläche mit einer Farbe

- GUI Texture: Erlaubt das Einfügen von Bildern oder Texturen

Wir können diesen Spielobjekten einen statischen oder dynamischen Platz auf dem Bildschirm zuweisen. Bei der statischen Variante platzieren wir die Objekte über die Pixel. Bei der Dynamischen werden sie über die aktuelle Breite bzw. Höhe unseres Bildschirms angezeigt. Wir verwenden in unserem Projekt die dynamische Variante. Diese passt sich jeder möglichen Auflösung an. Somit würden die Vitalparameter bei einer neuen 3D-Brille, die mit einer höheren Auflösung läuft, die vorgesehenen Positionen beibehalten. Da wir nur über den C#-Code an unsere benötigten Variablen kommen, müssen wir auch hier GUI-Text Variablen erzeugen. GUI-Text Variablen können nur Strings anzeigen. Da unsere anzuzeigenden Parameter überwiegend nur als Intenger oder Float vorliegen, müssen wir diese mit der toString()-Methode umwandelt. Die Werte unserer Vitalparameter bekommen wir aus unseren Controller Klassen. Diese fangen die gesendeten Werte des Brustgürtels, SpO2 Sensors und des Ergometers ab. Für die Geschwindigkeit, die zurückgelegten Kilometer und Trainingsdauer müssen wir uns Hilfsvariablen anlegen. Die Geschwindigkeit und die zurückgelegten Kilometer fangen wir von unserem Ergometer ab. Zum Erstellen des Höhenprofils haben wir als Hilfsvariablen die aktuelle Höhe unseres Fahrrads. Dazu kann man eine öffentliche Transform-Variable erstellen. Für Transform-Variablen kann man mit der Punktnotation die aktuellen X, Y und Z Positionen des Objektes abrufen. Für die Darstellung des Höhenprofils haben wir ein Array erzeugt, das alle paar Meter die aktuelle Höhe abspeichert und anschließend einen Balken auf der GUI zeichnet. Wenn wir am Ende des Arrays ankommen, verschieben sich alle Werte um eine Position nach links. Somit wird der erste Wert aus dem Array überschrieben und der aktuellste Wert am Ende gespeichert.

6.4. Installation

Die Verwendung der Client-Applikation unter dem Betriebssystem Windows 7 statt. Es ist auch möglich, dass die Applikation mit Windows 8 funktioniert, getestet ist alles allerdings mit Windows 7.
Für die ersten Schritte zum Start des virtuellen Trainings benötigt man folgende Komponenten:

- Ergometer inklusive eines eingebauten USB Lenkrads und Touch-Display
- Brustgürtel *Zephyr Bioharness 3*
- SpO2 Sensor (enthalten im *Corscience NiBP2010/ChipOx*)
- *Occulus Rift 3D* Brille mit eingebautem Headtracker

- Die *Unity3D* Software für den Client
- Zugangsdaten, um das virtuelle Training zu starten (Username, Passwort)

Des Weiteren werden folgende Anschlüsse an den Computer benötigt:

- 4* USB (Touch-Display, Headtracker 3D Brille, SpO2 Sensor, Lenkung des Ergometers)
- 1* COM (Ergometer)
- 1* HDMI (3D-Brille)
- 1* Bluetooth Schnittstelle für den Brustgürtel (auch per Bluetooth-USB-Adapter möglich)

6.4.1. Schritt 1: Zugangsdaten

Bevor man mit dem Training beginnen kann, benötigt man als Erstes die Zugangsdaten, welche man entweder vom Arzt oder dem Administrator bekommt. Der Administrator legt dabei einen neuen Patienten an und ordnet diesem einen Usernamen und ein Passwort zu. Der angelegte Patient wird wiederum vom zuständigen Arzt untersucht, wonach der Arzt die Grenzen der Vitalparameter für den Patienten bestimmt, um eine optimale Auslastung während des Trainings zu gewährleisten. Wenn alle relevanten Daten eingetragen sind, ist es dem Patienten möglich, sich einzuloggen und am virtuellen Trainingsbetrieb teilzunehmen.

6.4.2. Schritt 2: Unity3D installieren und einrichten

Nach dem Anlegen des Patienten sollte die Client-Anwendung von Unity3D auf einen Rechner seiner Wahl installiert werden. Die Software ist standardmäßig für ein Windows Betriebssystem ausgelegt. Für die Installation benötigt man ca. 120Mb freien Festplattenspeicher. Die Installation ist selbsterklärend. Man benötigt einen Account bei Unity, den man während der Installation erstellen kann, sofern man noch keinen hat.

Wichtiger Hinweis: Es könnte folgende Fehlermeldung im Unity Editor erscheinen: *Assets/CordiAALClient/VitalDataBioHarness.cs(26,24): error CS0246: The type or namespace name 'SerialPort' could not be found. Are you missing a using directive or an assembly reference?*

Mögliche Lösung: Man wählt im Unity Editor *Edit* → *Project Settings* → *Player* aus. Daraufhin öffnet sich im Unity Editor im *Inspector* das Menü *PlayerSettings*. Hier wählt man unter *Other Settings* → *Optimization* → *Api Compatibility Level* den Eintrag *.NET 2.0* aus, falls dies nicht schon eingestellt ist. Es könnte nötig sein, dass man anschließend das Skript reimportieren muss. Dies geschieht, indem man oben in der Leiste auf *Assets* → *Reimport* klickt. Es könnte jedoch auch nach dieser Einstellung der Fehler vorhanden sein, sodass man eine andere Lösung finden muss. In unserem Projekt hat die beschriebene Lösung funktioniert.

6.4.3. Schritt 3: Sensoren

Als Nächstes müssen alle Sensoren zur Überwachung der Vitalparameter angeschlossen werden.

BioHarness 3

Installation Der BioHarness 3 besitzt eine Bluetooth Schnittstelle. Falls kein Bluetooth am Computer vorhanden ist, funktioniert es auch mit einem Bluetooth Dongle. Den Dongle sollte man von Windows selbst erkennen und installieren lassen, was in unserem Projekt auch problemlos geschehen ist. Man sollte den Dongle im Idealfall nicht mit einem eigens mitgelieferten Treiber installieren ([**?**, S. 7]).
Nützliche Informationen zur Installation des BioHarness Treibers enthält [**?**, S. 9 ff]. Für die Installation des Treibers sind wir jedoch einer anderen Installationsanleitung von Zephyr gefolgt ([**?**]).

Einrichten Falls der BioHarness 3 zuvor eingerichtet war und man ihn erneut einrichten möchte, beispielsweise weil die Kommunikation nicht mehr funktioniert, dann sollte man ihn zunächst im Ordner *Geräte und Drucker* entfernen: Dazu geht man wie folgt vor:

1. Gehe zu *Systemsteuerung* → *Hardware und Sound* → *Geräte und Drucker*)
2. Wähle den BioHarness aus der Liste und wähle *Gerät entfernen* oben aus der Leiste.

Für ein erneutes Hinzufügen in Windows muss der BioHarness eingeschaltet sein. Nun geht man wie folgt vor:

1. Klicke in der oberen Leiste auf *Gerät hinzufügen*. Es wird nach Geräten gesucht.
2. Falls der BioHarness in der Liste der gefundenen Geräte erscheint, wählt man ihn aus und klickt auf *Weiter*.
3. Anschließend wird man möglicherweise darauf hingewiesen einen Kopplungscode einzugeben. Hier wählt man *Kopplungscode des Geräts verwenden* aus und gibt den Code des BioHarness ein. Dieser ist in der Voreinstellung "**1234**".

4. Nun werden zwei virtuelle COM-Ports eingerichtet, von denen man für unser Projekt nur den ausgehenden virtuellen COM-Port betrachten muss.

Programmierung Für die Anbindung des BioHarness 3 an externe Software sei auf die Dokumentation, insbesondere auf die Protokollspezifikation ([**?**]), verwiesen.

Hinweise Falls es zum Fehlschlagen der Kommunikation mit dem BioHarness 3 kommt (aus einem externen Programm oder aus dem Konfigurationstool von Zephyr), könnte es zweckmäßig sein einen *Reset* am BioHarness 3 auszuführen. Dieser geht komfortabel mit Hilfe des Konfigurationstools *Zephyr Cfg Tool* ([**?**, S. 14]). Im Anschluss könnte es sein, dass man das oben beschriebene Bluetooth-Pairing (siehe Einrichten) erneut ausführen muss.

Corscience NiBP2010/ChipOx

Für den Anschluss des SpO2 Sensors wird nur ein USB Port benötigt. Nach Anschluss des Sensors an den Rechner wird anschließend automatisch ein virtueller COM-Port eingerichtet.
Wichtiger Hinweis: Die Anbindung des Moduls an unsere Client-Applikation war **nicht** erfolgreich, sodass der SpO_2-Wert im Projekt **nicht** verwendet werden kann (vgl. 5.4.2).

Ergometer

Zum Anschließen des Ergometers werden zwei USB Steckplätze und ein COM Port benötigt. Einer der USB Anschlüsse wird zum Anschließen des Lenkrads benötigt. Der andere ist für die Steuerung des mitgelieferten Touchscreens auf dem Ergometer. Über den seriellen COM-Port werden die relevanten Daten vom Ergometer an das Programm geschickt.

Occulus Rift 3D Brille

Zum Anschließen der 3D Brille wird die Krontoll-Box mittels USB an den Computer verbunden. Dieser steuert den eingebauten Headtracker in der 3D-Brille und folgt automatisch der Blickrichtung des Benutzers im Training. An der Knotroll-Box ist es möglich, das Bild über einen DVI oder HDMI Eingang zu übertagen. Es dürfen jedoch nicht beide Anschlüsse gleichzeitig benutzt werden. Mit der 3D-Brille werden zudem drei verschiedene Linsenstärken mitgeliefert.

Damit Das Ergometer und der Brustgürtel automatisch im Spiel erkannt werden, ist es vor dem Start notwendig, im Installationsordner die ComConfig.txt Datei umzuschreiben. Dort werden die Ports für die Schnittstelle zum Bluetooth und Ergometer definiert. Nur so kann gewährleistet werden, dass der Brustgürtel und das Ergometer automatisch beim Trainingsbeginn erkannt werden.

1. COM-Port: Bluetooth Schnittstelle
2. COM-Port: Ergometer

Sobald alle Geräte an den Computer angeschlossen sind, kann das Training beginnen. Nachdem man die Client-Anwendung gestartet hat, gibt man die Daten zum Einloggen ein. Danach gelangt man in das Hauptmenü und kann sich einen Trainingsmodus aussuchen.

7. Erprobung

7.1. Funktionstest

Für den Funktionstest werden mehrere Komponenten der Unity-Anwendung getestet. Diese Komponenten werden in verschiedenen Szenarien auf ihre Funktionalität überprüft. Es ist dabei wichtig, dass die gesammelten Patientendaten während des Trainings nicht verloren gehen und ordnungsgemäß auf unsere Datenbank abgespeichert werden. Eine einfache Anmeldung und Handhabung der Anwendung spielt dabei auch eine große Rolle. Der Test wird mit der Anmeldung bei Reha-Web anfangen. Dort soll sich der Patient mit anderen Patienten zum Training verabreden und dann ein virtuelles Gruppentraining auf den Ergometer absolvieren. Nach dem Training bekommt der Patient und der Arzt via Reha-Web oder der Unity-Anwendung, die Möglichkeit sich die gespeicherten Daten anzeigen zu lassen. Diese Daten können dann abschließend vom Arzt ausgewertet werden.

Für den Funktionstest werden wir in einzelnen alle vorhandenen Funktionen testen und alle aufgetretenen Probleme Protokollieren.

Wir unterteilen dabei den Test in drei Kategorien:

– Test von Rehaweb
– Funktionalität während des Trainings
– Funktionalität der Client-Anwendung vor und nach dem Training.

7.1.1. Test von Rehaweb

Diese Tests beziehen sich nur auf die Erweiterungen zu den bereits vorhandenen Rehaweb-Projekt. Es werden ausschließlich nur unsere eigenen Erweiterungen getestet, die für CordiALL von Bedeutung sind:

– Benutzer Registrieren
– Benutzer Einloggen
– Grenzwerte des Patienten setzen und einsehen(Arzt)
– Trainingshistorie einsehen(Arzt)
– Trainingshistorie einsehen(Benutzer)
– Trainingshistorie einsehen(Admin)
– Strecken hinzufügen/ ändern/ entfernen(Admin)
– Trainingstermin für ein Gruppentraining festlegen(Benutzer)

7.1.2. Funktionalität der Client-Anwendung vor und nach dem Training

Patient Einloggen:

– Das Einloggen ist nur möglich, wenn die richtige Kombination aus Passwort und Benutzernamen eingegeben wird. Bei einer inkorrekten Eingabe erscheint eine Fehlermeldung.

Patient Ausloggen:

– In den Menü ist es im Hauptmenü und im Ende-Screen (erscheint im Anschluss an ein gerade abgelaufenes Training nach Betätigen des *Weiter*-Buttons in der Anzeige der aktuellen Statistik) möglich sich aus dem Spiel zu loggen. Hierzu muss der Patient auf "Ende" drücken. Danach erscheint ein Fenster um das ausloggen zu bestätigen. Somit wird sichergestellt, dass der Patient sich nicht unabsichtlich ausloggt.

Menü-Naviagtion:

– Die Navigation durch das Menü funktioniert fehlerfrei. Es ist aus fast jedem Menüfenster möglich mittels des *Zurück*- bzw. *Hauptmenü*- Knopfes auf das Vorherige Menüfenster bzw. Hauptmenü zu springen. Aus dem Screen zur Eingabe eines Borg-Wertes und aus dem Screen mit der Statistik direkt im Anschluss an ein Training ist dies nicht möglich. Jedes Untermenü wird nach dem anwählen ohne Verzögerung korrekt angezeigt.

– Einzeltraining starten (Virtuelle Strecke und Dauer auswählen)

– Nachdem der Patient zum Menüpunkt "Einzeltraining" navigiert ist, ist es im Anschluss möglich die Dauer und die Strecke des Trainings zu bestimmen. Die Dauer kann in Stunden und Minuten angegeben werden. Hierbei ist es möglich ein Training zwischen 1 Minute und 5 Stunden zu wählen. Unter der Dauer befindet sich die Kartenauswahl. Mittels einer Dropdown- Liste kann man zwischen zwei Strecken wählen. Hat man sich für eine Strecke entschieden erscheinen Informationen über die Schwierigkeit, Länge und eine kurze Beschreibung der Strecke. Abschließend wird nach dem Betätigen des OK Buttons die ausgewählte Strecke geladen. Das Training wird automatisch nach der ausgewählten Dauer beendet und es wird einen Bildschirm mit der Wahl des Borg-Wertes angezeigt.

– Gruppentraining erstellen und starten(Gruppenname, Anzahl Personen, Dauer und Virtuelle Strecke auswählen)

– In den Menüpunkt "Gruppe erstellen" kann der Patient ein Gruppennamen und eine Strecke seiner Wahl aussuchen. Darüber hinaus kann er die Größe der Gruppe bestimmen. Die maximale Gruppengröße ist auf 7 Teilnehmer beschränkt. Genauso wie beim Einzeltraining ist es auch hier möglich die Dauer des Trainings zu bestimmen. Nachdem alle Parameter ausgewählt sind, werden die Daten an den Server gesendet. Dieser speichert die erstellte Gruppe in der Datenbank. Die zuvor erstellte Gruppe ist nach dem Betätigen des OK Buttons für alle Teilnehmer unter Lobby aktuell sichtbar.

Gruppentraining beitreten:

– Hierzu haben wir die zwei Möglichkeiten ausprobiert. “Lobby - Aktuell“ ermöglicht den Patienten sich bereits gestartete Trainingseinheiten anzeigen zu lassen.Während “Lobby-Künftig“ die in Zukunft anstehenden Trainingseinheiten anzeigt.

– Statistik anzeigen lassen: Auch hier gibt es zwei Möglichkeiten sich die Statistik anzeigen zu lassen.

– Um uns eine Statistik anzeigen zu lassen, haben wir zuvor ein Training erstellt.

– - Web-Statistik (kurze Übersicht über vergangene Trainingseinheiten)

– - Statistiken-Kompakt

7.1.3. Funktionalität während des Trainings

– Kommunikation (Voice Chat)

– Training beenden (F1 = vorzeitiges beenden und Statistik abschicken, F10 beenden ohne abschicken der Statistik, automatisches beenden des Trainings nach ablaufen der Trainingszeit)

– Bei diesen Test haben wir alle Drei Möglichkeiten getestet ein Spiel zu Beenden.

– 1. Training beenden mit F1

– Der Patient hat die Möglichkeit mittels der F10 Taste das Training vorzeitig zu beenden ,ohne dass die geplante Trainingszeit erreicht worden ist und die aktuell gespeicherte Statistik verloren geht. Wir haben einen Spiel gestartet und es zweimal vorzeitig abgebrochen. Um zu schauen, ob unserer Test erfolgreich war, haben wir nach dem Spiel jeweils in unsere Statistik nachgeschaut , ob unsere Daten an den Server abgeschickt und gespeichert worden sind. In beiden Fällen war der Test erfolgreich.

– 2. Training beenden mit F10

– Bei dieser Variante haben wir getestet , ob die Daten Tatsächlich nicht an den Server gesendet worden sind. Auch hier haben wir zweimal ein Training gestartet , um diese Funktion zu testen. In beiden Fällen haben wir kein Eintrag in der Datenbank verursacht.

– 3. Training automatisch nach ablaufen der Trainingszeit beenden.

– Auch bei diesem Test haben wir zwei Trainingseinheiten erstellt und im Anschluss nachgeschaut , ob die Statistik an den Server gesendet worden ist. Das Training wurde automatisch beendet und es befand sich nach dem Training eine Statistik mit unseren Daten.

– Korrekte Darstellung der 3D Landschaft

– Unsere zwei zur Auswahl stehenden Karten wurden beim starten korrekt geladen.

– Korrekte Darstellung der Gruppenteilnehmer

– Die Gruppen teilnehmen wurden bei einem Gruppentraining korrekt darstellt .Jeden Teilnehmern wird automatisch eine Farbe zugewiesen, die sich von den anderen Teilnehmern unterscheidet. Auch die Aktuelle Position aller Gruppenteilnehmer wird ordnungs-

gemäß angezeigt. Ab und zu kommt es zu kleinen Datenverlusten bei der Übertragung. Dadurch wirkt die Bewegung der Gruppenteilnehmer sprunghaft und nicht so flüssig wie geplant. Wir haben beim Test eine Gruppe mit insgesamt vier Patienten erstellt.

– Funktionalität der 3D Brille mit dem eingebauten Headtracker

– Bei dem Test der Brille ist uns aufgefallen, dass man nur eine korrekte Darstellung bekommt, wenn man eine Auflösung von 1280*800 bei den Client einstellt. Nur auf dieser Auflösung arbeitet die 3D Brille korrekt. Falls man eine andere Auflösung benutzt, kann es vorkommen, das Teile des Sichtfeldes abgeschnitten werden. Bei der Benutzung des Headtrackers mussten wir die Blickrichtung vor dem Start erst richtig einstellen. Hierzu haben wir mit der Brille gerade aus geguckt und die Position der Kamera mit der Maus manuell korrigiert. Nachdem wir dies vorgenommen haben, funktionierte der eingebaute Headtracker sehr gut. Er reagierte sehr empfindlich und reaktionsschnell auf jede Bewegung unseres Kopfes.

– Lenkung des Ergometers

– Die Lenkung des Ergometers funktionierte wie geplant. Dabei wurde der Winkel des realen Lenkers direkt in dem virtuellen Fahrrad übernommen. Als Grenzen für das virtuelle Fahrrad haben wir ein Einschlag von maximal 35 Grad gewählt. Aufgefallen ist, dass sich das Lenkrad nach einer einigen Zeit verstellt und wieder neu justiert werden muss.

– Beschleunigung des Ergometers

– Zum testen der Beschleunigung haben wir im erstellten Training eine längere Distanz zurückgelegt und so beobachtet ob unser Client die Daten des Ergometers korrekt empfängt. Nach einer kurzen Reaktionszeit hat unser Client die Daten des Ergometers empfangen und die Beschleunigung bzw. die Geschwindigkeit in virtuellen Training der realen Geschwindigkeit angepasst.

– Anzeige der Vitalparameter (Herzfrequenz, SpO2)

– Die Übertragung der Herzfrequenz vom Brustgürtel an den Client lief ohne Probleme. Beim Starten des Clients wurde unser Brustgürtel direkt erkannt und beim starten des Trainings auch korrekt angezeigt. Einerseits als Zahl in den rechten oberen Rand des Displays und andererseits in unseren Balken mit den vorgegebenen Grenzen des Patienten. Dieser Balken hat einen Kursor der sich dynamisch mit der veränderten Herzfrequenz auf den Balken bewegt.

– Bei der Anzeige des SpO2 Wertes hatten wir Probleme mit Unity festgestellt. In unseren Testcode konnten wir die Werte auslesen, diese wurden aber nicht in Unity angezeigt. Leider ist es uns zur zeit noch nicht Gelungen diese während des Trainings anzeigen zu lassen.

– Anzeige der absolvierten Leistung(KmH, Km, Zeit, Höhenmeter)

– Die Geschwindigkeit und absolvierten Kilometer im Training werden während des Trainings von Ergometer empfangen. Beide Werte wurden uns korrekt angezeigt. Die verbrachte Zeit im Training und das Höhenprofil wird direkt von Unity berechnet. Auch hier konnten

wir feststellen das die Parameter korrekt angezeigt worden sind und sich das Höhenprofil dynamisch mit unserer Höhe verändert hat.

– Lastensteuerung des Patienten (mittels der Herzfrequenz)

– Steuerung der Wattleistung bei einer Steigung/ Senkung /keine Steigung

– Steuerung der Wattleistung bei unterschiedlichen Untergründen(Asphalt, Erde, Sand)

– Borgwert nach dem Training setzen

– Nach dem Beenden des Trainings wurde uns ein Bildschirm mit ?? verschiedenen Werten angezeigt, um unsere absolvierte Trainingseinheit zu bewerten. Der abgegebene Wert wurde ordnungsgemäß übermittelt und in der Statistik gespeichert.

– Trainingsdaten/Statistik an den Server senden und einsehen

– Nach jedem Training wurde die gesamte Statistik des Trainings an den Server gesendet. Diese konnten wir unmittelbar nach dem absolvierten Training einsehen. Einerseits direkt im Menü des Spiels oder über einen externen Link. Dieser Links verweist auf eine kurze Übersicht der vergangenen Trainingseinheiten.

– Zusammenfassung nach dem Training anzeigen

– Die Zusammenfassung des Trainings erfolgt nach der Eingabe des Borgwertes. Nach jedem Training haben wir auf unseren Bildschirm eine kompakte Zusammenfassung unser Trainingsdaten bekommen. Dort wurde der auch zuvor erfasste Borgwert aufgelistet. Des weiteren konnten wir auch die zurückgelegte Distanz, Dauer des Training, ...

– ablesen.

7.2. Probleme

– Bei der Nutzung der 3D- Brillen (Carlr Zeiss, Occulus Rift 3D) wurden bei einigen Nutzern Schwindelgefühle ausgelöst. Diese traten erst nach einigen Minuten beim Tragen der Brille auf. Die Hersteller beider Brillen haben auf dieses Problem bereits aufmerksam gemacht.

– Bei der Einbindung des SpO2 Sensors in Unity3D haben wir große Probleme gehabt. Unity3D hat uns nur Nullen geliefert. Eventuell besteht eine Inkompatibilität des Sensors mit Unity3d. Bei einem Test des Sensors außerhalb Unity3D konnten wir korrekte Werte auslesen.

– Probleme bei RehaWeb

– Bei Leistungsschwachen Computern kommt es zu erheblichen Problemen in der Darstellung. Das Spiel läuft nicht flüssig

8. Schluss

Das Ende dieser Dokumentation wird durch dieses Kapitel gebildet. Zunächst wird ein Fazit aus dem in der Projektgruppe Geleistetem gezogen. Es wird u. a. auf das zu Beginn besprochene Minimalziel eingegangen. Anschließend werden im Ausblick Verbesserungsvorschläge genannt, die unser Projekt noch interessanter gestalten könnten.

8.1. Fazit

Der Projektgruppe CordiAAL ist es gelungen verschiedene Technologien zu verschmelzen und diese innovativ in einem Konzept der Primär- und Sekundärprävention nutzbar zu machen. Auch wenn derartige Ideen bereits realisiert worden sind, so ist in unserem Projekt das Prinzip der dynamischen Lastkontrolle während eines virtuellen Gruppentrainings das, was unser Projekt von denen anderer abgrenzt. Dadurch ist es Patienten in einer virtuellen Gruppe möglich nahe anderer leistungsstärkerer Patienten virtuell mit zu fahren, wo es in konventionellen Umsetzungen nicht so ist. Dafür sorgt unser Algorithmus für die dynamische Lastkontrolle. Es ist erstaunlich, zu welchen Produkte sich bei Verschmelzung verschiedenster Technologien wie z. B. der 3D-Brille und einem modifizierten Fahrradergometer ergeben, um am Ende einem Zweck zu dienen: der menschlichen Gesundheit. Dass sich die Umsetzung dieses Projekts einen Platz in der Realität sichern kann hat nicht zuletzt die Annahme unseres wissenschaftlichen Artikels bei der internationalen Konferenz *HEALTHINF 2014* gezeigt. In einer Zeit, in der Zugriff auf das Internet selbstverständlich ist sowie viele im Besitz modernster Technologien sind, scheint es in naher Zukunft möglich, dass Patienten wie gesundheitsbewusste Menschen die nötige Hardware erwerben und in der virtuellen Welt trainieren. Alles in allem ist eine motivationssteigernde Wirkung mit Hilfe unserer Umsetzung sehr wohl denkbar.
Unsere Projektgruppe hat das Minimalziel erreicht:

1. Es wurde ein ambientes System entworfen und entwickelt. Dieses integriert mehrere Sensoren. In der Client-Applikation ist eine Belastungskontrolle implementiert genauso wie die Gruppenkommunikation per Voice-Chat.

2. Es wurde ein Regelsystem entwickelt, welches das Training einer Gruppe an Hand der Werte personenbezogener Sensoren steuert.

Des Weiteren wurden sowohl technische als auch soziale Kompetenzen erworben. Als Mitglied der Projektgruppe musste man sich beispielsweise oft in neue Themengebiete einarbeiten, während der Planungs- wie der Entwicklungsphase, sodass das technische Können erweitert worden ist. Abgerundet wird das Ganze mit dem verpflichtenden Leistungsnachweis, der uns einen Schritt weiter im Studium bringt.

8.2. Ausblick

Die Gestaltung und der Ablauf eines Trainings in der virtuellen 3D-Welt ist der Kernpunkt der Motivationssteigerung. Eine Erweiterung in diesem Bereich wäre es, einen eigenen Avatar der Patienten zu erstellen und diesen auch als 3D-Objekt im Spiel anzeigen zulassen. Durch das gezielte ansteuern eines solchen Avatars wäre es möglich, die Sprachausgabe direkt an einen solchen Avatar zu knüpfen. Dies hätte den Vorteil, dass die Stimme lauter wird, wenn man sich auf eine Person zubewegt, und leiser, wenn man sich von ihr wegbewegt. Um einen noch höheren Realismus zu gewinnen, wird in Zukunft eine Rüttelplatte mit mechanischem Steigungswinkel unter das Fahrrad angebracht. Dies bietet die Möglichkeit, dass Steigungen realistischer wahrgenommen werden können und die Beschaffenheit des Untergrundes realistischer übertragen werden. Durch die endgültige Version der Oculus Rift Brille und der verbesserten Auflösung, wird eine noch bessere Verschmelzung von Realität und virtuellen Welt geschaffen.

A. Weitere Informationen

A.1. Anleitung

Diese Anleitung bezieht sich hauptsächlich auf die Client-Applikation. Sofern nicht anders gesagt, ist der Benutzer immer der Patient. Zum Schluss wird kurz auf die Web-Statistik eingegangen, auf die der Arzt Zugriff hat.
Hinweis: Es sind nicht zu jedem Eintrag Screenshots vorhanden.

A.1.1. Benutzerhandbuch

Login: Hier müssen die persönlichen Log-In Daten eingegeben werden. Unter **Benutzer** gibt man den Benutzernamen und unter **Passwort** das Passwort ein. Bestätigt wird die Eingabe mit einem Klick auf **OK**.
Man hat im Login Menü und im Hauptmenü die Möglichkeit das Programm mit **Ende** zu beenden. Außerdem besteht in jedem Menü (außer Login) mit **Zurück** die Möglichkeit in das vorherige Menü zu gelangen. Das Login Menü ist in Abbildung A.1 dargestellt.

Hauptmenü: Nach erfolgreichem Login wird man zum Hauptmenü weitergeleitet. Im Hauptmenü hat man mit einem Klick auf *Training* die Möglichkeit ein neues **Training** zu starten, über den Button *Statistiken - Kompakt* seine **Statistiken** in kompakter Form an-

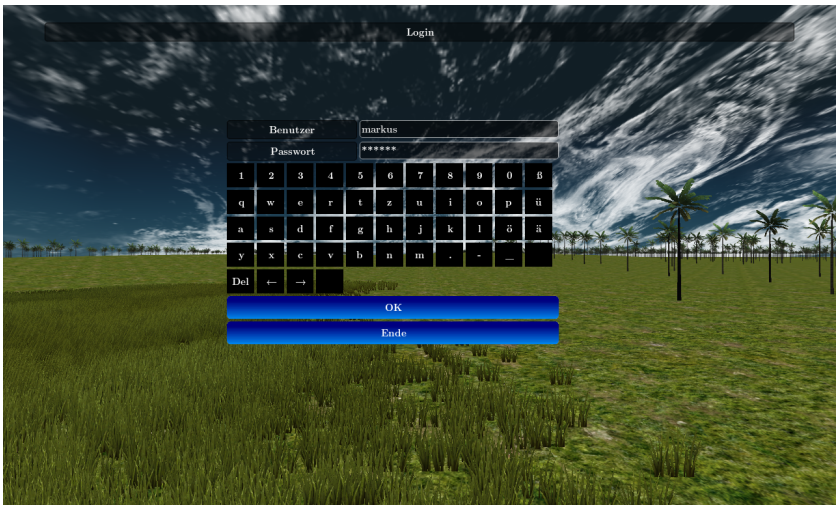

Abbildung A.1.: Das Login Menü

Abbildung A.2.: Das Hauptmenü nach dem Login

zuschauen. *Kompakt* heißt, dass die Statistiken in textueller Form und nicht mit jedem Detail dargestellt werden. Über *Web-Statistik* können bisherige Trainings (**Trainingshistorie**) auf der RehaWeb Community angezeigt werden. Zusätzlich ist es möglich per Klick auf *RehaWeb Community* auf die **Hauptseite der Community** zu gelangen. Sollten in den letzten beiden Fällen Ihre Anmeldedaten der Community im Browser *nicht* gespeichert sein, so müssen Sie sich zunächst einloggen. Das Hauptmenü ist in Abbildung A.2 zu sehen.

Hauptmenü → Training: Falls man sich für ein Training entschieden hat, so wird man im folgenden Bildschirm darauf hingewiesen, alle Sensoren korrekt verbunden zu haben, wie in Abbildung A.3 zu sehen ist. Mit einem Klick auf *Weiter zum Training* wird man zum Menü *Training* weitergeleitet (Abbildung A.4). Hier kann man zwischen einem Einzeltraining und einem Gruppentraining wählen.

Hauptmenü → Training → Einzeltraining: Hat man sich für ein Einzeltraining entschieden, so hat man die in Abbildung A.5 gezeigten Auswahlmöglichkeiten für:

Dauer: Die Dauer des Trainings in Stunden und Minuten.

Karte: Die gewünschte virtuelle Welt, in der man trainieren möchte. Hier werden außerdem neben einer kurzen **Beschreibung** der Karte auch die **Schwierigkeit** und die **Länge** (in km) angezeigt.

Mit einem Klick auf *OK* startet das Training nach einem kurzen Ladeprozess.

Hauptmenü → Training → Gruppentraining: In diesem Menü (Abbildung A.6) sind zwei Möglichkeiten des Trainings gegeben. Man kann mit *Gruppe erstellen* eine eigene

Abbildung A.3.: Ein Hinweis auf das korrekte Verbinden aller Sensoren

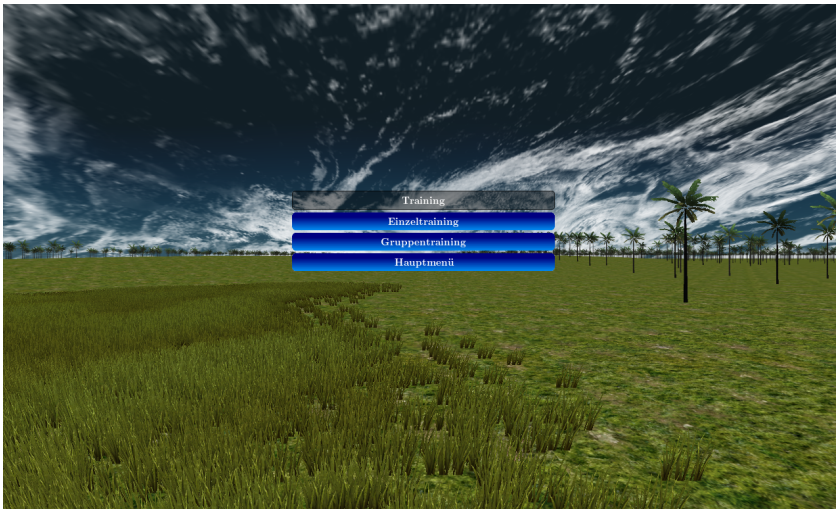

Abbildung A.4.: Das Menü Training

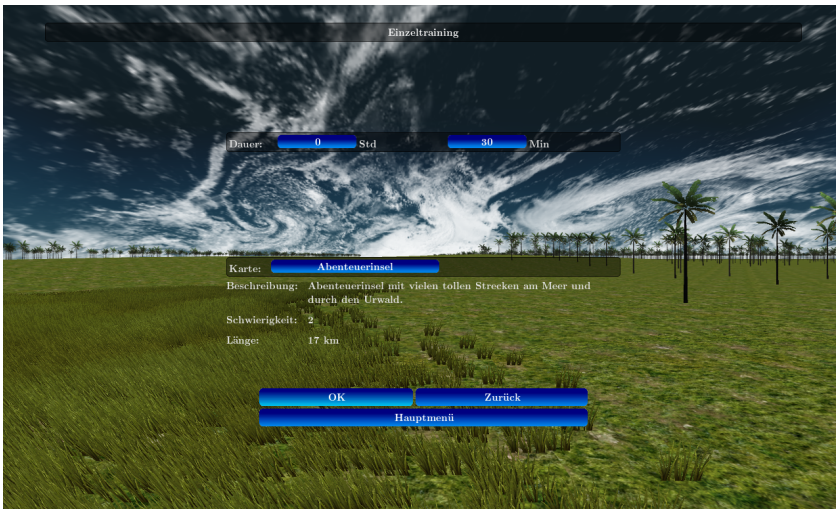

Abbildung A.5.: Das Menü Einzeltraining

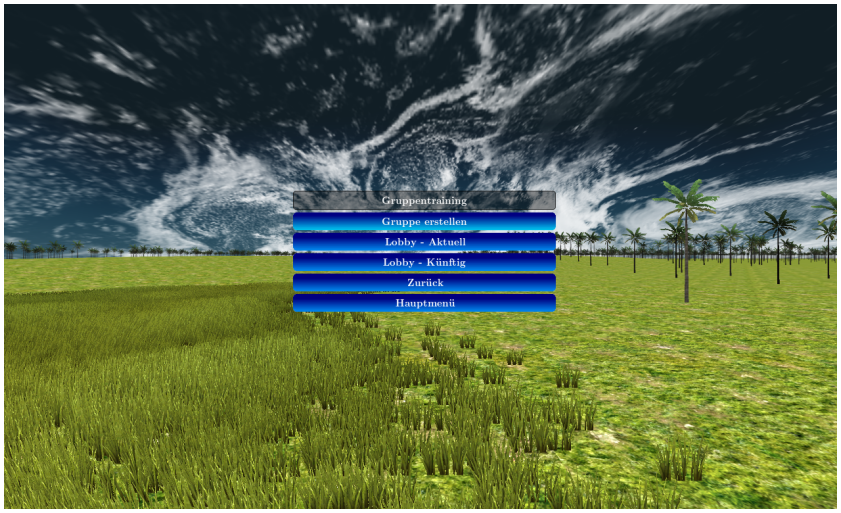

Abbildung A.6.: Das Menü Gruppentraining

Gruppe erstellen oder mit *Lobby - Aktuell* die aktuell trainierenden Gruppen einsehen und einer vorhandenen Gruppe beitreten. Des Weiteren sind mit *Lobby - Künftig* die geplanten Gruppentrainings einsehbar.

Hauptmenü → Training → Gruppentraining → Gruppe erstellen: Hat man sich dafür entschieden eine eigene Gruppe zu erstellen, so hat man die in Abbildung A.7 gezeigten Auswahlmöglichkeiten für:

Max. Personen: Die maximale Anzahl an Personen, die in der virtuellen Gruppe gemeinsam trainieren können sollen.

Gruppenname: Der gewünschte Name für die neue Gruppe.

Dauer: Die Dauer des Trainings in Stunden und Minuten.

Karte: Die gewünschte virtuelle Welt, in der man trainieren möchte. Hier werden außerdem neben einer kurzen **Beschreibung** der Karte auch die **Schwierigkeit** und die **Länge** (in km) angezeigt.

Mit einem Klick auf *OK* startet das Training nach einem kurzen Ladeprozess. Dieser Gruppe können im Verlauf des Trainings Patienten beitreten.

Hauptmenü → Training → Gruppentraining → Lobby - Aktuell: Hier ist eine Übersicht aller aktuell trainierenden Gruppen. In der Gesamtübersicht werden Informationen wie Gruppenname und Startzeit der jeweiligen Gruppe angezeigt. Mit einem Klick auf einen Eintrag aus der Liste kann man sich nähere Details zur Gruppe anzeigen lassen. Ein Klick auf *Aktualisieren* aktualisiert die Übersicht, sodass beispielsweise neue Gruppen angezeigt

Abbildung A.7.: Eine eigene Gruppe erstellen

werden. Das Aktualisieren erfolgt jedoch auch automatisch im 10-Sekunden-Takt. Klickt man auf *Beitreten*, so tritt man dieser Gruppe bei und das Gruppentraining beginnt nach einem kurzen Ladeprozess.

Hauptmenü → Training → Gruppentraining → Lobby - Künftig: Dieses Menü ist analog zum Menü Lobby - Aktuell und bietet eine Übersicht aller für die Zukunft geplanten Gruppentrainings, welche über die RehaWeb Community geplant worden sind. Nur kann man hier keiner Gruppe beitreten.

Training Beenden: Um ein laufendes Training frühzeitig abzubrechen, müssen Sie auf den Bildschirm drücken und ihn einige Sekunden gedrückt halten. Dies gilt für Einzel- sowie für Gruppentrainings.

Hauptmenü → Statistiken - Kompakt: In diesem Bereich erhält man eine kompakte Übersicht über die bisher absolvierten Trainingseinheiten (Trainingshistorie) innerhalb der Client-Applikation. In der Gesamtübersicht werden Informationen wie Startzeit und Dauer des jeweiligen Trainings angezeigt. Mit einem Klick auf einen Eintrag aus der Liste kann man sich nähere Details zum Training anzeigen lassen.

Hauptmenü → Web-Statistik: Mit einem Klick auf *Web-Statistik* gelangt man direkt auf die persönliche RehaWeb Seite *Mein RehaWeb → Tagebuch → Virtuelles-Training* und erhält eine kurze Übersicht über die bisher absolvierten Trainingseinheiten. Sofern man nicht in RehaWeb eingeloggt war, muss man sich mit Benutzername und Passwort anmelden und wird anschließend auf die genannte Seite weitergeleitet.

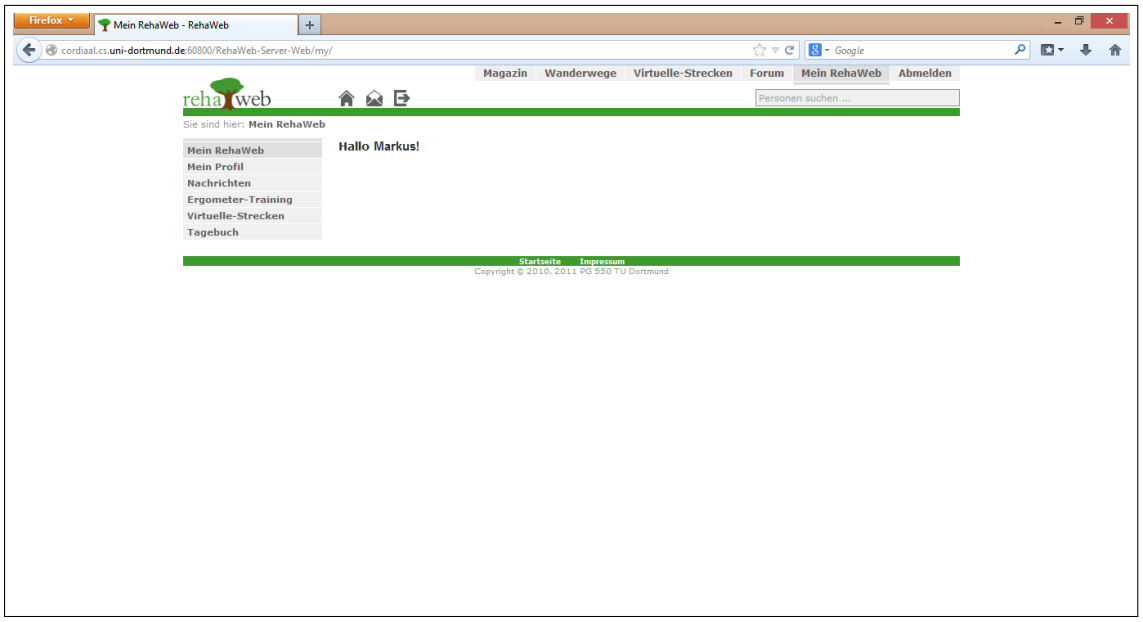

Abbildung A.8.: Die Seite *Mein RehaWeb*

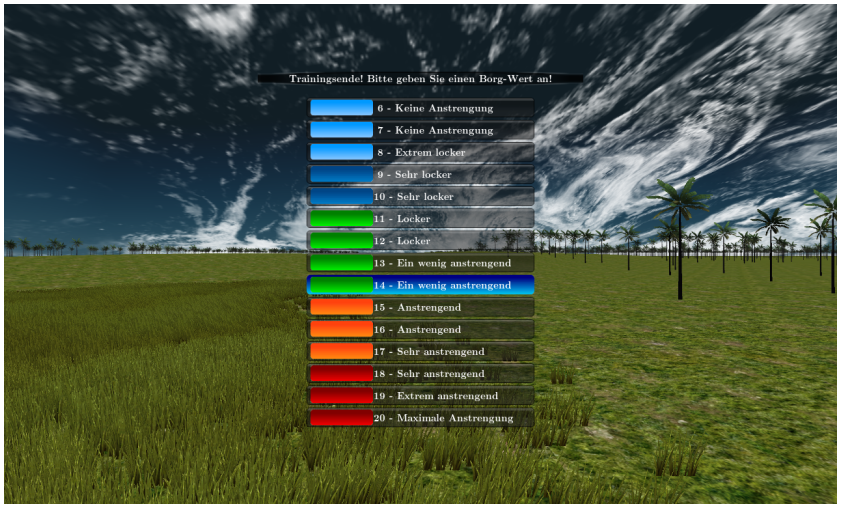

Abbildung A.9.: Eingabe des Borg-Wertes

Hauptmenü → RehaWeb Community: Mit einem Klick auf *RehaWeb Community* gelangt man direkt auf die persönliche Seite *Mein RehaWeb* (Abbildung A.8). Sofern man nicht in RehaWeb eingeloggt war, muss man sich mit Benutzername und Passwort anmelden und wird anschließend auf die genannte Seite weitergeleitet.

Nach dem Training: Nachdem das Training abgeschlossen wurde, erscheint ein Menü zur Eingabe des Borg-Werts (Abbildung A.9). Mit einem Klick auf die entsprechende Schwierigkeitsstufe erscheint eine Statistik für die soeben absolvierte Trainingseinheit (Abbildung A.10).

Abbildung A.10.: Die Statistik der beendeten Trainingseinheit

Web-Statistik für den Arzt: Der Arzt hat über RehaWeb die Möglichkeit eine detaillierte grafische Statistik über all seine Patienten einzusehen. Hier werden beispielsweise die Herzgrenzen und Herzfrequenz angezeigt (Abbildung A.11).

REST-Methoden REST unterstützt die 4 Grundfunktionen des 3WC: GET, POST, PUT, DELETE damit kann man jede Ressource im mit GET aanfordern, mit POST erstellen, mit PUT bearbeiten und mit DELETE löschen.

RehaWeb-Strucktur RehaWeb-Strucktur beschreibt den Aufbau der Webplattform in den einzelnen Schichten.

RehaWeb-Arzt-Maske Damit ein Patient in der Virtuellenwelt trainieren kann, muss ein Artzt zunächst

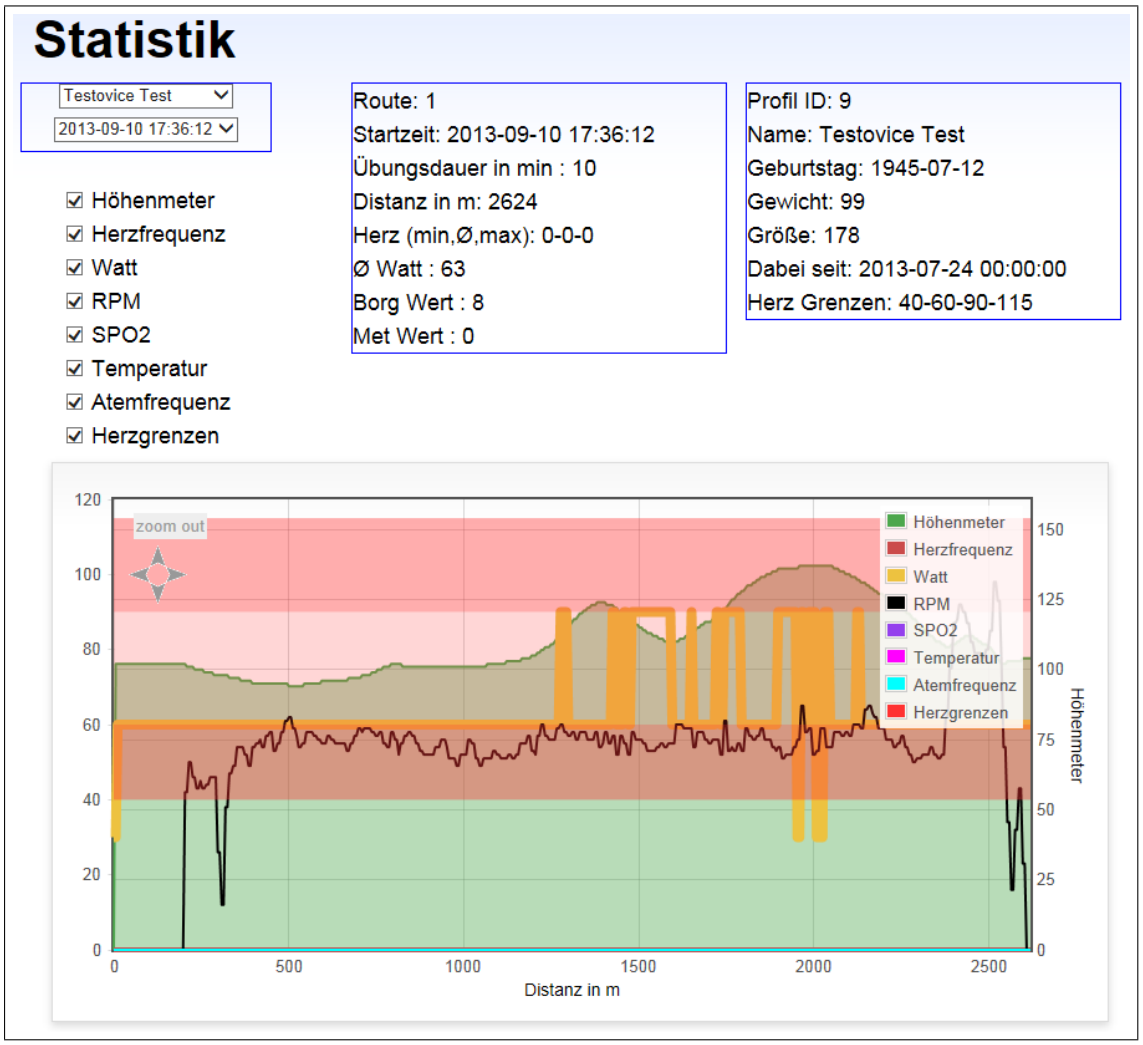

Abbildung A.11.: In RehaWeb: Eine detaillierte Statistik für den Arzt über seine Patienten

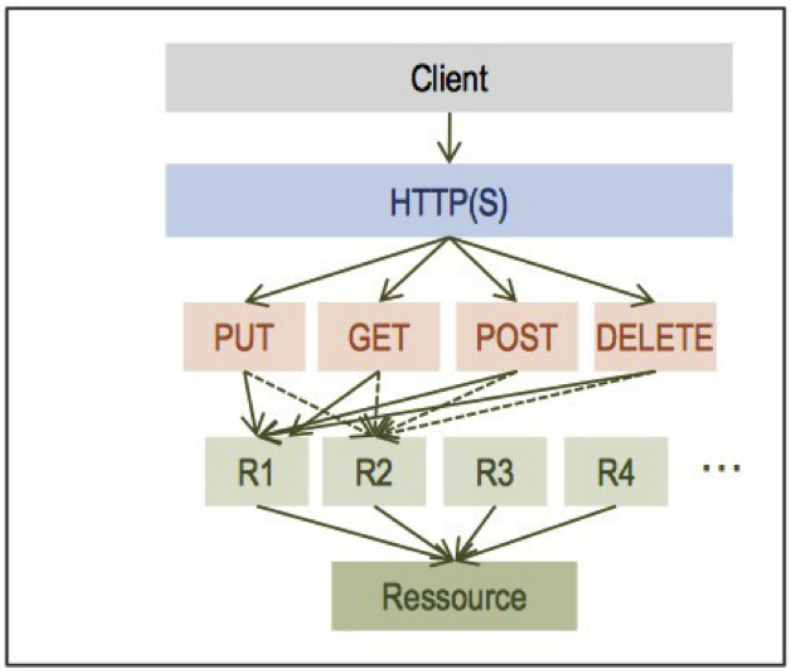

Abbildung A.12.: REST-Spezifikation

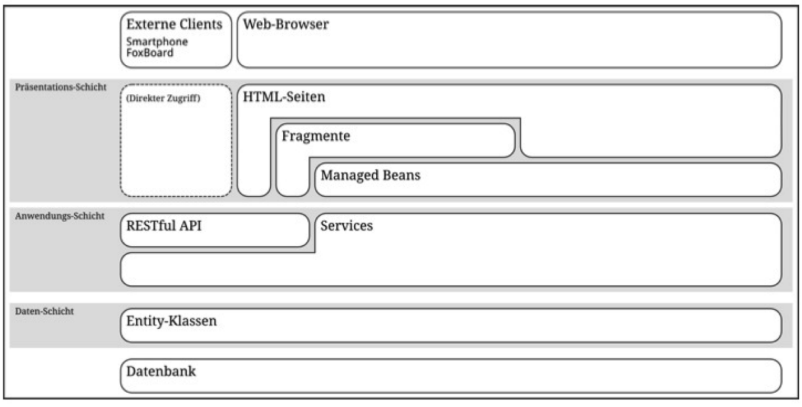

Abbildung A.13.: RehaWeb-Strucktur

Abbildung A.14.: RehaWeb-Arzt-Maske

Abbildungsverzeichnis

Tabellenverzeichnis