Projektkurs Informatik

Erstellung einer React Fitnessapp

Konrad Geller

Q2, 2019/20

Inhaltsverzeichnis

1 Einleitung **2**

1.1 Aufgabenstellung 2
1.2 Motivation 2

2 Vorstellung der App **2**

2.1 Kurzanleitung für die App 2
2.2 Die Übungen 3
2.3 Das Workout 3
2.3.1 Bearbeiten 3
2.3.2 Durchführen 4
2.3.3 Das Ende 4
2.4 Der Überblick 5
2.4.1 Letzte Trainings 5
2.4.2 Körpergewicht 5
2.4.3 Statistiken 5
2.5 Importieren von Übungen 6
2.6 Einstellungen 6

3 Die Entwicklung **7**

3.1 Verlauf der Entwicklung 7
3.1.1 Aufbau des Grundgerüsts 7
3.1.2 Erste funktionierende Version 7
3.1.3 Bearbeiten von Workouts 7
3.1.4 Bearbeiten von Übungen 8
3.1.5 Aufzeichnung der Daten 8
3.1.6 Neuaufbau der Reducer 9
3.1.7 Das Design 9
3.1.8 Speichern und Löschen von Daten 10
3.1.9 Datenbanksuche von Übungen 10
3.1.10 Text to Speech 11
3.1.11 Bugfixes 11
3.1.12 Test und weitere Funktionen 12
3.2 Interessante Entdeckungen 12

4 Fazit **13**

Anhang **I**

1 Einleitung

1.1 Aufgabenstellung

Die Aufgabenstellung für das Projekt im Rahmen des Projektkurses Informatik bestand darin, mithilfe des Javascript Frameworks *React* eine eigene App zu entwickeln. Diese Aufgabenstellung habe ich mit der Erstellung einer Fitnessapp, entwickelt mit React, erfüllt.

1.2 Motivation

Ich gehe regelmäßig ins Fitnessstudio und erhalte dort auch regelmäßig Coachings, bei denen mir der Trainer neue Übungen empfiehlt. Nach nunmehr zwei Jahren im Fitnessstudio habe ich einen großen Stapel Trainingspläne und keine Übersicht mehr darüber, was ich eigentlich wann wie lange trainiert habe und welche Übungen ich in der Zeit gesammelt habe. Trotz eines großen Angebotes an Fitness-Apps habe ich keine gefunden, die meine individuellen Trainingspläne und - Fortschritte dokumentiert und dabei gleichzeitig werbefrei und benutzerfreundlich ist. Daher habe ich mich entschieden, im Rahmen des Projektkurses eine Fitness-App zu entwickeln, die genau meinen Bedürfnissen entspricht. Zu Beginn hatte ich folgenden Plan, wie meine App aussehen sollte:

> Mein Programm soll es dem Nutzer ermöglichen, sich verschiedene Fitnessübungen einzuspeichern und diese dann in verschiedenen Workout-Plänen zusammenzustellen. Dabei soll der Nutzer zu jeder Übung verschiedene Einstellungen - Name der Übung, Wiederholungen/Zeit, beanspruchte Muskulatur, Beschreibung und ein Bild – einfügen können. Die Workoutpläne können entweder manuell oder zufällig zusammengestellt werden. Jegliche Übungen werden dabei global gespeichert und können somit auch in mehreren Trainingsplänen genutzt werden. Wird der Name einer neuen Übung eingegeben, kann der Nutzer sie zu seiner persönlichen Übungsdatenbank hinzufügen. Auch die verschiedenen Trainingspläne sollen gespeichert werden. Dabei kann der Nutzer vor jedem Training aus den zuvor gespeicherten Workouts sein Training auswählen und starten. Vor Start des Workouts sieht der Nutzer zum Einen alle Übungen in einer Übersicht, kann zum Anderen die einzelnen Übungen mit den jeweiligen Informationen ansehen und bearbeiten. Wird das Training gestartet, soll der Nutzer dann jeweils die aktuelle Übung mit den relevanten Informationen sehen. Nach Beenden des Trainings soll der Trainingserfolg (z. B. Menge und Länge des Trainings) gemessen und die Anzahl der Trainings in einem Zeitraum (z. B. Monatlich) grafisch dargestellt werden.

2 Vorstellung der App

2.1 Kurzanleitung für die App

Beim Öffnen der App befindet man sich auf der Hauptseite des Programms, von der aus man mithilfe des Menüs auf die folgenden Funktionen zugreifen kann:

1. Bearbeiten, Erstellen und Suchen von Fitnessübungen

2. Importieren von Übungen aus einer Datenbank
3. Bearbeiten, Hinzufügen und Starten eines Workouts
4. Ansehen der bisherigen Trainingsdaten in verschiedenen grafischen Übersichten
5. Vornehmen von verschiedenen Einstellungen

Außerdem kann man direkt von der Hauptseite aus ein Workout starten.

2.2 Die Übungen

Um diese Funktion zu nutzen, muss man im Hauptmenü den Punkt *Exercises* auswählen. Dort kann man nun wählen, ob man eine bereits in der Datenbank der zuvor gespeicherten Übungen vorhandene Übung anschauen und bearbeiten oder eine neue hinzufügen möchte. In beiden Fällen wird man auf ein Formular weitergeleitet. Dort muss man zunächst einen Übungsnamen sowie die Dauer der Übung angeben, weitere Felder sind optional. Im nächsten Punkt kann man nun entweder eine oder mehrere beanspruchte Muskelgruppen aus bereits gespeicherten auswählen oder eine neue Muskelgruppe zu der vorhandenen Liste hinzufügen. Dazu tippt man einfach den Namen in das Feld, in dem die Muskulatur angegeben wird, und bestätigt mit Enter. Gleiches gilt für benötigte Materialien, zum Beispiel eine Matte oder ein Gewicht. Diese beiden Punkte helfen später bei der Übungssuche. Im Folgenden kann man nun eine Übungsbeschreibung angeben, dabei kann man normalen oder mit HTML angereicherten Text verwenden, sowie ein Bild für die Übung hinzufügen. Dazu gibt es entweder die Option, die URL eines Bildes anzugeben (mit Enter bestätigen) oder eines in der Datenbank von unsplash.com[3] zu suchen. Man gibt dazu ein Suchwort in das rechte Feld ein und wählt nun ein Bild aus der Liste der Suchergebnisse mit einem Linksklick aus. Um das Bild wieder zu löschen, kann man auf den Papierkorb in der rechten oberen Ecke des Bildes klicken. Um die geänderten Daten zu speichern, das Bearbeiten abzubrechen und geänderte Daten zu verwerfen oder die Übung zu löschen, klickt man auf den jeweiligen Knopf am Ende des Formulars. Eine Unterfunktion beim Menüpunkt der Übungen ist die *Übungssuche*. Man kann in den bereits erstellten Übungen nach einer suchen, wozu man auf den entsprechenden Untermenüpunkt in *Exercises* klicken muss. Dort kann man nun Übungen finden, die je nach Auswahl eine bestimmte Muskelpartie beanspruchen oder bestimmtes Equipment benötigen. Außerdem kann man auch in das letzte Feld den Namen oder einen Teil des Namens einer Übung eingeben und mit Enter bestätigen, um alle zu dem Namen passenden Übungen zu finden.

2.3 Das Workout

2.3.1 Bearbeiten

Beim ersten Klick auf den Menüpunkt *Workout* kann man entweder ein bestehendes Workout auswählen oder ein neues erstellen. Zudem kann man, nach dem Öffnen der

Auswahl, einen Eintrag mit Klick auf den Papierkorb komplett aus den Nutzerdaten entfernen. Des Weiteren kann man das gerade ausgewählte Workout mit Klick auf den untenstehenden Knopf löschen. Möchte man ein neues Workout erstellen, kann man in der Auswahl dessen Namen eingeben, mit Enter bestätigen und es zu der Liste seiner Workouts hinzufügen. Wurde nun ein Workout ausgewählt sieht man, sofern bereits vorhanden, eine Auflistung aller Übungen des Workouts mit einer Option zum Löschen und einer zum Bearbeiten, die man mit einem Linksklick auf das jeweilige Symbol auswählen kann. Auch kann man hier eine Übungsdauer angeben, die nur genau für diese Übung in diesem Workout gelten soll. Möchte man die Reihenfolge der Übungen verändern, kann man dies per *Drag and Drop* der jeweiligen Übung an deren neue Position erreichen. Am Ende dieser Auflistung sieht man ein weiteres Auswahlmenü. Aus diesem kann man entweder durch einen Klick eine Übung auswählen oder durch Tippen eines Übungsnamen und Bestätigens durch Enter eine neue Übung erstellen. Außerdem kann man Einträge aus der Liste entfernen, indem man auf den Papierkorb auf der rechten Seite des jeweiligen Eintrags klickt. Damit wird die Übung aus allen Workouts und der Liste entfernt. Bei jedem Workout wird angezeigt, welche Muskulatur beansprucht und welches Equipment benötigt wird. Zuletzt gibt es noch die Option, die Pausenzeit für dieses Workout zu verändern. Wurde nun mindestens eine Übung hinzugefügt, kann man mit einem Klick auf den Startknopf das Workout starten.

2.3.2 Durchführen

Nun wird man zum wirklichen Workout weitergeleitet. Es startet mit der ersten Übung des Workouts, wobei diese Seite in zwei Teilen aufgebaut ist:
Auf der linken Seite sieht man den Namen und Restzeit der jeweiligen Übung. Anhand eines Fortschrittsbalkens kann man erkennen, welcher Anteil der Übungszeit bereits vergangen ist. Unter diesem Kreis gibt es die Optionen, das Workout sofort zu beenden, zur nächsten weiterzugehen sowie die Übung zu pausieren und danach wieder zu starten.
Auf der rechten Seite sieht man Informationen zu der aktuellen Übung. Oben steht der Name der Übung, darunter das ausgewählte Bild und darunter wiederum die Beschreibung der Übung.
Wenn die Übung vollendet wurde, beginnt, nach Ansage eines Countdowns, eine Pause, deren Länge man in der Bearbeitungsfunktion des Workouts (2.3.1) anpassen kann. Auch hier sieht man wieder auf der linken Seite die Zeit bis zum Pausenende und einen Fortschrittsbalken. Auf der rechten Seite erscheint die Übung, welche nach der Pause beginnen wird. Dabei kann man durch einen Klick auf diese Information auch direkt zu dieser Übung vorspringen.

2.3.3 Das Ende

Wenn die letzte Übung des Workouts beendet wurde, wird man auf eine Übersichtsseite weitergeleitet. Dabei sieht man zum einen eine Übersicht über bereits geleistete

Erfolge (siehe 2.4.3), zum Anderen Links zu anderen Teilen der App.

2.4 Der Überblick

Auch aus dem Hauptmenü heraus kann man die Überblickseite anzeigen. Dazu muss man den jeweils gewünschten Unterpunkt aus dem Menüpunkt *Overview* auswählen.

2.4.1 Letzte Trainings

Auf der Überblick-Seite sieht man als Erstes eine Übersicht über die zuletzt durchgeführten Trainings, angezeigt mithilfe einer Timeline. Bei Klick auf eines der Daten in der Timeline sieht man dann in einer kleinen Beschreibung, welche Trainings man wie oft an dem betreffenden Tag durchgeführt hat.

2.4.2 Körpergewicht

Als nächstes sieht man nun einen Graphen, der die Entwicklung des Körpergewichts an verschiedenen Tagen aufzeichnet. Um direkt das Gewicht am heutigen Tag hinzuzufügen, kann man auf das Plus-Symbol klicken und dort sein aktuelles Körpergewicht hinzufügen.

2.4.3 Statistiken

Nun kommt man zu den Statistiken, die sich bei jedem Neuladen anders anordnen:

1. *different workouts*: Die Anzahl der von dem Nutzer hinzugefügten Workouts.
2. *kg weight*: Das zuletzt gemessene Körpergewicht. Dabei gibt es bei einem Klick auf die beiden Pfeile die Option, das aktuelle Gewicht einzustellen.
3. *weight loss*: Das Gewicht, welches man zwischen dem maximalen und dem aktuellen Körpergewicht verloren hat.
4. *today's workouts*: Die Anzahl der Workouts, die man an dem heutigen Tag bereits durchgeführt hat.
5. *different exercises*: Die Anzahl der in der Datenbank gespeicherten Fitnessübungen.
6. *workouts done*: Die Anzahl der Workouts, die man seit dem letzten Zurücksetzen durchgeführt hat.
7. *training minutes*: Die Gesamtzeit, die man bereits trainiert hat.
8. *Ø minutes workout time*: Die Zeit, die man im Durchschnitt für jedes Training aufgewendet hat.
9. *workouts/week*: Die Anzahl an Workouts, die man im Durchschnitt pro Woche durchgeführt hat.

2.5 Importieren von Übungen

Für das Importieren wird die Datenbank von wger.de[1] benutzt. Bei Auswahl des Eintrags im Menü (*import exercises*) wird man auf ein Suchformular weitergeleitet. Werden keine weiteren Parameter aufgelistet, werden einem dabei sofort alle alphabetisch sortierten Übungen aus der von wger.de bereitgestellten Datenbank angezeigt. Zusätzlich kann man über den Ergebnissen auch noch Parameter für die Suche setzen. So kann man nach der Sprache der Übung, den beanspruchten Muskeln sowie den notwendigen Materialien filtern. Außerdem kann man den Namen der Übung direkt eingeben und erhält dann, nach Bestätigung mit Enter und falls in der Datenbank vorhanden, alle Einträge mit diesem Namen. Dabei ist, aufgrund des Aufbaus der API, der exakte Name notwendig.

2.6 Einstellungen

Zuletzt kann man noch über den Menüpunkt *Settings* verschiedene Daten einstellen. Zunächst kann man die Übungsdauer als Standardwerte für neu erstellte Übungen einstellen, die dann bereits direkt nach Erstellung einer neuen Übung in den Feldern stehen werden. Außerdem kann man die standardmäßige Pausenzeit zwischen zwei Übungen einstellen, diese kann man dann später aber zusätzlich auch für jedes Workout unterschiedlich wählen. Des Weiteren kann man in den Einstellungen verschiedene Einstellungen zu den gespeicherten Daten treffen. So kann man alle vom Nutzer hinzugefügten Daten als *JSON*-Datei herunterladen und auf anderen Geräten wieder hochladen. Dazu klickt man auf das betreffende Feld, wobei man die Auswahl hat, die in der Datei gespeicherten Daten zu den bereits vorhandenen Daten hinzuzufügen oder die alten Daten zu überschreiben (dazu muss das entsprechende Feld abgehakt werden). Außerdem kann man noch alle Daten löschen. Dabei kann man aus den folgenden zu löschenden Feldern auswählen, um gegebenenfalls nicht alle zu löschen:

- Übungen
- Workouts
- Gespeicherte Muskelgruppen und Equipment
- Aufzeichnung des Körpergewichts
- Durchgeführte Trainings
- Trainingszeit

Zuletzt kann man die ganze App auch zurücksetzen, um wieder alle Standarddaten, die auch beim ersten Laden der App angezeigt wurden, wieder zu laden.

3 Die Entwicklung

3.1 Verlauf der Entwicklung

3.1.1 Aufbau des Grundgerüsts

Zu Beginn des Entwicklungsprozesses habe ich ein Grundgerüst geschaffen. So habe ich bereits einige der in Abschnitt 2 genannten Funktionen als zunächst leere Dateien erstellt, um diese später mit Funktionalität zu füllen. Außerdem habe ich bereits das für React-Router erforderliche Grundgerüst erstellt. Zuletzt habe ich Reducer mit ersten Actions implementiert. Dabei habe ich mich jedoch zunächst für einen Reducer für die aktuelle Übung sowie einen für das aktuelle Workout entschieden, was ich später noch geändert habe.

3.1.2 Erste funktionierende Version

Im nächsten Schritt wollte ich dann eine erste funktionierende Version mit einem jedoch sehr eingeschränkten Funktionsumfang implementieren. So konnte man in dieser Version bereits aus einem festen Vorrat an Übungen ein zufällig generiertes Workout starten. Außerdem habe ich erste Funktionen für den Überblick (2.4) erstellt, welche ich aber in späteren Versionen größtenteils überarbeitet habe.

3.1.3 Bearbeiten von Workouts

Im Folgenden habe ich daran gearbeitet, dass man Workouts auch manuell erstellen und Übungen hinzufügen kann. Dabei hatte ich anfangs einige Schwierigkeiten mit der Auswahl für das jeweilige Workout und dem Hinzufügen von Übungen zu diesem Workout. Es war schwierig, richtig mit dem mithilfe eines npm-Pakets hinzugefügten Auswahlfeldes umzugehen und dieses für meine Zwecke anzupassen. So musste ich es zum Beispiel schaffen, dass bei einem Klick auf den Papierkorb lediglich das jeweilige Feld gelöscht, nicht jedoch ausgewählt, wird. Nachdem ich dies zunächst mithilfe einer Variable, die jeweils gesetzt wird, wenn die Maus sich über dem Papierkorb befindet, gelöst habe, merkte ich später, dass dies noch einfacher mithilfe der Funktion *stopPropagation* zu lösen war. Außerdem musste ich nun einen weiteren Reducer hinzufügen, in dem die Daten des Nutzers gespeichert werden konnten. Dabei musste ich es auch schaffen, die jeweils ausgewählten Workouts zu diesem Reducer hinzuzufügen und vor allem die gespeicherten Übungen zu dem Workout hinzuzufügen. Zuletzt habe ich dann die Funktion implementiert, dass man mithilfe von Drag-and-Drop[4] die Reihenfolge der Übungen verändern kann. Dabei musste ich, nachdem die jeweilige Übung an der neuen Stelle losgelassen wurde, anhand der neuen Reihenfolge der IDs auch die Übungen neu anordnen. Dies war mithilfe des Javascript Array.map-Befehls möglich (Abb. 2).

3.1.4 Bearbeiten von Übungen

Des Weiteren wollte ich eine Möglichkeit bieten, dass man seine gespeicherten Übungen einfach bearbeiten kann. Nachdem ich zunächst mit einem normalen *HTML-Formular* gearbeitet hatte, stieg ich dann auf das einfacher zu handhabende *Formik*[6] um. Bei der Erstellung der einzelnen Formularfelder erwies sich nun das bereits vorher für die Workouterstellung implementierte Auswahlfeld als nützlich. So konnte ich dieses relativ einfach auch für die Angabe von Muskeln und Equipment der Übungen verwenden. Eine Besonderheit in dem Übungsformular ist das Suchen von Bildern auf unsplash.com. Inspiriert von der Bildersuche im Udemy Kurs von Stephen Grider[11] wollte ich eine ähnliche Suche auch in meiner App implementieren. Dabei gab es Folgendes zu beachten: Ich musste es zum Einen durch ein Weitergeben des *onChange-Handlers* schaffen, dass eingegebene Daten auch in Formik gespeichert werden. Zum Anderen beabsichtigte ich, dass sofort bei Eintippen eines Buchstaben die Suche gestartet wird. Dabei habe ich jedoch erst nach einigem Testen gemerkt, dass dann bei Eintippen des nächsten Buchstaben die vorherige Suche gestoppt werden muss, um unvorhergesehene Ergebnisse zu vermeiden und die Anfrage auf die unsplash-API zu beschleunigen. Dazu erwies sich der von *Axios*[2] bereitgestellte *CancelToken* als sehr nützlich, da mithilfe dessen die zuvor gestellte Anfrage einfach abgebrochen werden konnte (Abb. 3).

3.1.5 Aufzeichnung der Daten

Daraufhin habe ich mich an das Problem der Aufzeichnung der Daten, also der in der Vergangenheit durchgeführten Workouts sowie des eingegebenen Körpergewichts, gekümmert. Dazu habe ich zunächst einmal die verschiedenen Reducer, die vorher alle in einer Datei waren, in verschiedene Dateien ausgelagert, da diese sonst zu lang und zu unübersichtlich geworden wären. Nun wurde ein Workout nach seiner Beendigung im Reducer der Nutzerdaten gespeichert. Dabei habe ich mich dazu entschieden, die letzten Trainings nicht als IDs zu speichern, sondern deren Titel, da so auch nach der Löschung einer Übung auf diese korrekt zugegriffen werden kann. Außerdem konnte man bereits im Graphen, der die Entwicklung des Körpergewichts anzeigt (siehe 2.4.2), weitere Einträge vornehmen. Später erreichte ich dann, dass Trainings in einer Timeline angezeigt wurden. Dabei musste ich es zunächst schaffen, dass gleiche Daten in verschiedenen Feldern des Arrays in eines zusammengefasst werden. Da die Daten der letzten Trainings bereits sortiert waren, konnte ich dazu einfach nebeneinanderliegende daraufhin vergleichen, ob sie am selben Tag liegen. Außerdem musste ich es unter Anderem mithilfe der Array.reduce Funktion schaffen, dass die Anzahl einer Art des Trainings an einem Tag gezählt werden können (Abb. 4). Eine weitere Besonderheit in der Darstellung der Daten war das Anzeigen der Statistiken (2.4.3). Ich wollte erreichen, dass sich die Statistiken bei jedem Neuladen der Seite neu anordnen. Dazu musste ich die in den Statistiken angezeigten Daten in einem Array speichern und dieses mithilfe einer selbst implementierten Mischfunktion in der componentDidMount-Methode mischen. Schwierigkeiten gab es dabei bei der Anzei-

ge des Gewichts, welche immer auf den aktuellen State zugreifen muss, um auch nach einer Änderung des States durch die beiden Pfeile noch das richtige Körpergewicht anzuzeigen. Dazu musste ich eine Methode erstellen, welche diese aktuellen Werte berechnen und in dem Array speichern kann. Somit wird die Methode dann bei jedem Rendern auch wirklich aufgerufen und berechnet das aktuelle Körpergewicht richtig.

3.1.6 Neuaufbau der Reducer

Als mein Projekt schon weiter fortgeschritten war, habe ich gemerkt, dass meine aktuelle Methode, Übungen zu speichern und zu verwalten, bei einem weiteren Ausbau des Projekts zu Problemen führen würde. So hatte ich diese zunächst in der Liste der Übungen jeweils mit passenden Informationen gespeichert, in den Workouts dann jedoch nur die Namen gespeichert. Dies veränderte ich insofern, als nun nur noch die IDs der jeweiligen Übungen in den Workouts gespeichert werden. Damit verbunden hatte ich dann aber noch einigen Aufwand, in vielen Komponenten zunächst wieder die eigentlichen Informationen der Übungen mithilfe der IDs herauszufinden und somit anzuzeigen. Dies konnte vorher einfacher geschehen, da damals die Namen direkt gespeichert wurden. Nun benutzte ich dazu die *mapStateToProps-Methode*, da dadurch ein besseres Trennen vom Extrahieren der Daten aus dem Redux-Store und dem wirklichen Anzeigen innerhalb des Komponenten geschehen konnte. Dabei hatte dies vor allem den Grund, die Klassen möglichst übersichtlich zu halten, was sich im späteren Verlauf des Projekts tatsächlich als wichtig erwiesen hat. Später, als ich noch die Idee hatte, dass man jeder Übung auch in nur einem Workout eine bestimmte Zeit zuweisen könnte, musste ich die Struktur der Übungen in einem Workout ein weiteres Mal verändern. So musste jeweils die ID und die Zeit gespeichert werden. Zudem erstellte ich einen Reducer, in dem alle aktuellen Daten gespeichert werden können. So werden in diesem die aktuelle Übung sowie das aktuelle Workout gespeichert. Des Weiteren wird gespeichert, an welcher Stelle und zu welcher Zeit des aktuellen Workouts sich der Nutzer befindet. Zuletzt wird die Zeit gespeichert, die die Pause noch dauern wird. Durch diesen Reducer wird es dem Nutzer möglich, auch nach Verlassen und sogar nach einem Neuladen der Seite das Workout an der Stelle weiterzuführen, an der er es beendet hat. Dazu musste ich jeweils die Zeit nach dem Verlassen der Seite, also in componentDidUpdate, und nach dem Neuladen, in einem *beforeunload-listener*, neu im Reducer speichern.

3.1.7 Das Design

Als bereits eine erste laufende Version meiner App vorhanden war, habe ich damit begonnen, mithilfe von *CSS* und den Klassen von *SemanticUI*[9] meine App zu designen. Für einzelne Icons, die nicht von SemanticUI bereitgestellt wurden, benötigte ich Icons von Fontawesome[5]. Dabei hat sich als vorteilhaft herausgestellt, dass ich zunächst einmal die wichtigsten Funktionen entwickelt hatte, und mich dabei mehr auf diese konzentrieren konnte, und mich erst danach wirklich mit dem Design zu beschäftigen. Somit hatte ich durch bereits vorhandene Funktionen bereits einen

Überblick, wie die App später aussehen und gestaltet sein sollte. Ich habe jedoch auch im weiteren Verlauf der Projektentwicklung immer mal wieder an dem Design gearbeitet und auch vorher schon erste Komponenten mit SemanticUI gestaltet. Dabei hat sich insbesondere der *Grid-Container* als hilfreich zur Darstellung von verschiedenen Komponenten erwiesen, zum Beispiel den Statistiken (2.4.3, siehe Abb. 5) und den einzelnen Übungen im Workout.

3.1.8 Speichern und Löschen von Daten

Anschließend habe ich mich um das Speichern der Daten gekümmert. Es war mir wichtig, dass der Nutzer alle Daten ohne viel Aufwand auf seinem Gerät speichern und auch als Datei exportieren kann, um seine Daten mit anderen zu teilen oder auf ein anderes Gerät zu überspielen. Begonnen habe ich dabei mit dem Speichern im lokalen Speicher (*localstorage*) des Browsers[10]. Was zunächst einfach erschien - es musste einfach bei Veränderung eines Reducers sein Inhalt gespeichert werden - war dann später doch mit einigen Problemen verbunden. So gab es einige Schwierigkeiten beim Speichern des Javascript *Date*, wie es in den Trainingsaufzeichnungen verwendet wird, da dieses in einem falschen Format gespeichert wurde. Somit stellte sich erst nach einiger Fehlersuche heraus, dass das Datum jeweils einzeln konvertiert werden musste, um es dann auch wieder richtig abrufen zu können (Abb. 6). Zusätzlich wollte ich in meiner App eine Funktion haben, mithilfe derer man seine Daten auch auf verschiedenen Geräten laden kann. Dabei entschied ich mich jedoch vor allem aufgrund des Sicherheitsaspekts gegen das Aufsetzen eines Servers, der diese Daten speichern soll, und außerdem einen zu hohen Zeitaufwand erfordert hätte. Stattdessen implementierte ich eine Möglichkeit des Speicherns in einer JSON-Datei. So baute ich ein System zum Hoch- und Runter laden von Dateien ein, in denen jeweils die in der App gespeicherten Daten „transportiert“werden können. Dabei stellte sich zwar die Funktion, Daten aus einer Datei zu bereits vorhandenen Daten hinzuzufügen als etwas aufwändiger heraus, wobei es doch mit dem *Javascript Spread-Operator* jedoch recht einfach ging. In Verbindung mit dem Speichern und Laden von Daten baute ich auch ein Löschen derselben ein. Dabei wollte ich, dass man auch einzelne zu löschende „Felder“(z. B. Workouts oder Übungen) angeben konnte. Somit ergab sich vor allem in den Reducern ein gewisser Aufwand, nach all diesen Feldern zu filtern und nur diese auch wirklich zu löschen. Auch merkte ich, dass meine App zu diesem Zeitpunkt noch einige Probleme machte, wenn noch keine Daten vorhanden waren, sodass ich an verschiedensten Stellen in meinem Code Änderungen vornehmen musste.

3.1.9 Datenbanksuche von Übungen

Nachdem alle für ein Funktionieren notwendigen Funktionen bereits implementiert waren, entschied ich kurzfristig, dass es hilfreich wäre, wenn man mithilfe meiner App auch externe Übungen aus einer Datenbank importieren könnte und somit eine noch größere Vielfalt an Übungen zur Verfügung hätte. Nachdem ich nach einer dafür geeigneten Datenbank gesucht hatte, entschied ich mich für die von wger.de[1]. Dabei

stellte sich bei dieser Datenbank als besonders hilfreich heraus, dass die Suchkriterien (Muskulatur, Equipment und Sprache der Übungen) ebenfalls in der Datenbank gespeichert sind. Somit habe ich mithilfe des Datenbankzugriffs ein Anzeigen der Übungen sowie eine Suchfunktion nach verschiedenen Kriterien implementiert. Dazu werden zunächst die Suchkriterien beim ersten Start der App in einem Reducer gespeichert. Nun müssen diese bei Aufruf der Importier-Funktion nicht jedes Mal geladen werden und sind sofort erhältlich. Außerdem werden Muskulatur und Equipment in den Nutzerdaten gespeichert und können so auch bei eigenen Übungen schnell und ohne die Notwendigkeit, neue Muskelgruppen oder Zubehör hinzuzufügen, eingefügt werden. Dabei merkte ich, dass die Löschfunktion die Standardwerte nicht löschen darf, da sonst bei Hinzufügen einer Übung aus der Datenbank die falschen IDs für Muskeln und Equipment vergeben werden, was letztlich zu Fehlern führt. Da nun die Grundstruktur solch einer Suche bereits vorhanden war, konnte ich ohne großen Aufwand diese auch als interne Suche benutzen, wobei dann entsprechend die Anfrage an die API durch eine an die in den Reducern gespeicherten Übungen ersetzt werden musste.

3.1.10 Text to Speech

Als eine der letzten Funktionen implementierte ich nun die Funktion, dass ein Countdown sowie der Übungsname der nächsten Übung mithilfe einer Text-to-speech-Funktion angesagt werden. Dabei musste ich zunächst ein gut funktionierendes und vor allem, da beim Countdown kaum Verzögerungen auftreten dürfen, schnelles npm-Paket finden, welches solch eine Funktion in React ermöglicht. Ich fand es erst nach einigem Suchen; auch fand ich erst nach einigem Ausprobieren heraus, dass man die Text-to-speech-Instanz im State der jeweiligen Komponente nach ihrem Laden speichern muss. Zudem musste ich es schaffen, dass immer die richtige Zeit angesagt wird. So soll, falls in der Bearbeitung des Workouts eine eigene Zeit für die Übung angezeigt und angesagt werden. Wurde diese jedoch nicht angegeben, soll die in der Übungsbearbeitung angegebene Zeit benutzt werden. Zudem musste ich es schaffen, dass, nach einem Neuladen, nur noch die Restzeit angesagt wird. Nach einigen Schwierigkeiten hat aber auch das funktioniert.

3.1.11 Bugfixes

Schon im gesamten Entwicklungsprozess, doch vor allem am Ende, hatte ich viel mit verschiedensten Fehlern zu tun. Teilweise hatten sie ihren Ursprung in einer fehlenden Beachtung von Ausnahmefällen (z. B. der Fall, wenn noch keine Trainings durchgeführt wurden). Andere wiederum traten durch erst nach der Implementierung des jeweiligen Codes neu eingeführte Funktionen auf (z. B. habe ich erst später eine Funktion zum Speichern der beanspruchten Muskulatur von Übungen implementiert). So hatte ich zunächst nicht an diese neuen Funktionen gedacht und musste dementsprechend später mit den dadurch verursachten Fehlern umgehen.

3.1.12 Test und weitere Funktionen

Zum Abschluss der Entwicklung habe ich die ganze App getestet. Dabei fielen mir vor allem noch einige Bugs, aber auch noch fehlende Features auf. Beispielsweise merkte ich, dass man vor dem Löschen einer Übung oder eines Workouts gewarnt werden müsste, was sich mit der React-Router Komponente *Prompt* sowie der *window.confirm*-Funktion lösen ließ. Damit verbunden sollte der Nutzer auch beim Verlassen des Übungsformulars ohne Speicherung gewarnt werden, was sich mit einem *beforeunload-Event* lösen ließ. Auch verbunden mit dem Übungsformular war ein Schutz, dass nur Übungsnamen bis zu einer Länge von 30 Zeichen möglich sein sollten. Dadurch wollte ich verhindern, dass durch zu lange Namen das Layout der App nicht mehr funktioniert. Gleiches implementierte ich für Namen neuer Workouts. Des Weiteren merkte ich, dass es gut wäre, wenn für jede individuelle Übung eine eigene Zeit eingegeben werden könnte, die sich von der im Übungsformular (2.2) angegebenen Zeit unterscheidet. Das war insofern schwierig, als ich die Struktur der Workouts ändern musste. So musste nun im Array, in dem die Übungen gespeichert werden, nicht mehr nur noch die IDs, sondern zusätzlich die Übungsdauer gespeichert werden. Also musste ich nochmals den Reducer und einige Komponenten umändern, was sich jedoch größtenteils in der mapStateToProps-Methode lösen ließ. Zudem wollte ich, da die wger-API die Beschreibung oft in Form von HTML anbietet, diese in den jeweiligen Workouts so auch anzeigen.

3.2 Interessante Entdeckungen

Während der Entwicklung machte ich sehr viele interessante Entdeckungen, vor allem da ich vor dem Projektkurs noch nicht mit React und großen Projekten und kaum mit Javascript gearbeitet hatte. Im Folgenden werde ich einige dieser Entdeckungen auflisten.

Übergeben von Properties Beim Modularisieren meines Codes in verschiedene Komponenten merkte ich, dass es möglich ist, mithilfe des Spread-Operators alle Properties einrs Komponente an dessen Unterkomponenten zu übergeben.(Abb. 8)

Arrays und Komponenten Beim Aufbau der Statistiken merkte ich zudem, dass man auch React-Komponenten in Arrays speichern kann. So lernte ich, dass Komponenten in JSX-Syntax einfach normale Javascript Objekte sind.

Das Javascript Set Als ich, um doppelte Trainings an einzelnen Tagen nur einmal anzuzeigen, ein Array mit nur einzigartigen Werten brauchte, lernte ich das *Set* kennen. Dieses speichert eindeutige Werte und kann zudem mithilfe des Spread-Operators in ein Array umgewandelt werden.

Umwandeln des Datums Als ich mit dem Speichern von Daten im localstorage arbeitete, merkte ich, dass es dabei beim Javascript-Date Probleme gab. So wird

dieses, wenn man es mit JSON.stringify in einen String umwandelt, so gespeichert, dass es auch nach einem Rückverwandeln mit JSON.parse weiterhin ein String bleibt. Somit kann man entsprechende Methoden nicht mehr aufrufen und muss den String zunächst in ein Datum umwandeln.

Setzen des Redux-State in mapStateToProps Eine weitere Entdeckung machte ich durch Zufall, als ich eine Variable des Redux-States in der mapStateToProps-Funktion aus Versehen direkt setzte. So wird eine Variable, wenn man diese mit *state.variablenname = wert* in der mapStateToProps-Funktion setzt, dauerhaft im State auf diesen Wert gesetzt.

4 Fazit

Insgesamt bin ich sehr zufrieden mit dem Projekt und seiner Entwicklung. So konnte ich alle zu Beginn geplanten Funktionen auch tatsächlich realisieren, wobei sich die App letztendlich trotzdem stark von der ursprünglichen Idee (Abb. 9) unterschied. Auch die finale Projektstruktur unterschied sich grundlegend von der zu Beginn skizzierten (Abb. 10). Des Weiteren denke ich, dass man durch eine noch bessere Planung des Ablaufs der Programmierung sicherlich einigen Aufwand hätte verhindern können, so zum Beispiel das Problem mit dem Speichern von IDs in den Reducern (3.1.6). Gleichzeitig habe ich gemerkt, dass viele Funktionen erst während ihrer Implementierung zu Problemen führten, die ich zu Beginn noch nicht erahnen konnte. Außerdem denke ich, dass eine insgesamt bessere und übersichtlichere Projektstruktur die Entwicklung des Projekts sicherlich vereinfacht hätte. Somit wäre auch an dieser Stelle bessere Planung zu Beginn vorteilhaft gewesen. Als ich die App auf verschiedenen Geräten getestet habe, fiel mir auf, dass sie optimal auf größeren Bildschirmen funktioniert. Auf kleinen Mobilgeräten dagegen wird die Bedienung und Darstellung schwierig. Die Mobiloptimierung würde aber den Rahmen dieser Projektarbeit sprengen, sie ist vielleicht ein Plan für die Weiterentwicklung
Sehr zufrieden bin ich im Nachhinein damit, dass mir während der Entwicklung weitere Ideen gekommen sind, welche, neben den geplanten Funktionen, die App in meinen Augen auf jeden Fall noch deutlich verbessert haben. Letztlich gefällt mir die App nach ihrer Entwicklung noch besser als zu dem Zeitpunkt der Planung und trotz einiger Probleme hat mir die Entwicklung sehr viel Spaß gemacht. Zudem entspricht die App meinen Bedürfnissen, weshalb ich denke, dass ich sie in Zukunft auch benutzen werde.

Anhang

Literatur

[1] API für Übungen. http://wger.de/de/software/api. Eingesehen am 29.02.2020.

[2] Axios. https://github.com/axios/axios. Eingesehen am 29.02.2020.

[3] Bildersuche. https://unsplash.com/developers. Eingesehen am 29.02.2020.

[4] Drag and Drop Liste. https://codesandbox.io/s/k260nyxq9v. Eingesehen am 29.02.2020.

[5] Fontawesome. https://fontawesome.com/icons. Eingesehen am 29.02.2020.

[6] Formik. https://jaredpalmer.com/formik/docs/overview. Eingesehen am 29.02.2020.

[7] React-Logo. https://upload.wikimedia.org/wikipedia/commons/thumb/a/a7/React-icon.svg/2000px-React-icon.svg.png.

[8] Redux-Logo. https://upload.wikimedia.org/wikipedia/commons/4/49/Redux.png.

[9] SemanticUI. https://semantic-ui.com/. Eingesehen am 29.02.2020.

[10] Speichern im Localstorage. https://medium.com/@jrcreencia/persisting-redux-state-to-local-storage-f81eb0b90e7e. Eingesehen am 29.02.2020.

[11] Udemy-Kurs. https://www.udemy.com/course/react-redux/. Eingesehen am 29.02.2020.

[12] Veröffentlichung React-App auf Github. https://dev.to/yuribenjamin/how-to-deploy-react-app-in-github-pages-2a1f. Eingesehen am 29.02.2020.

[13] Warnung bei Verlassen einer Seite. https://luxiyalu.com/react-how-to-prompt-user-of-unsaved-data-before-leaving-site/. Eingesehen am 29.02.2020.

[14] Zugriff auf die App. https://konradge.github.io/be-fit/. Eingesehen am 29.02.2020.

Bilder

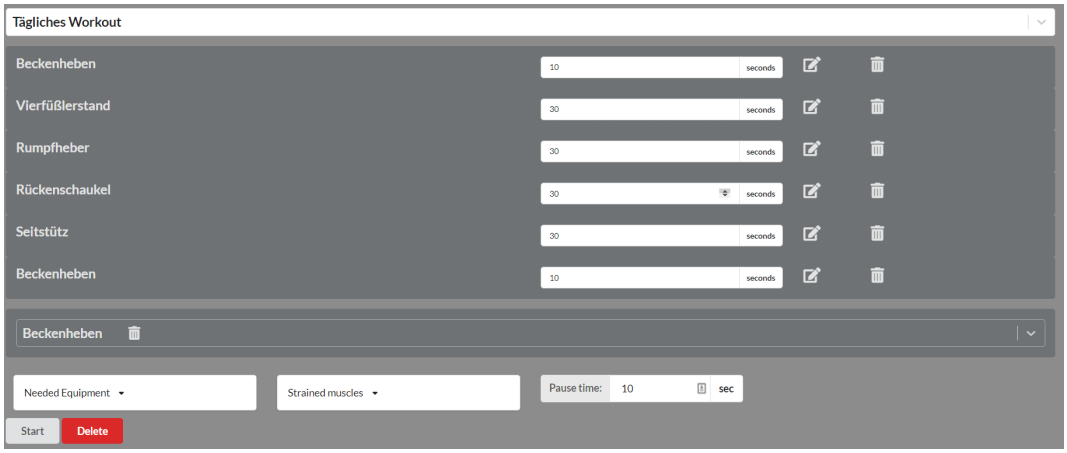

Abbildung 1: Erstellen und Bearbeiten eines Workouts

```
onDragEnd={newIdOrder => {
  const newOrder = newIdOrder.map(id =>
    this.props.workout.exercises.find(ex => ex.id === id)
  );
  this.props.editWorkout(this.props.workout.id, {
    exercises: newOrder
  });
```

Abbildung 2: Bearbeitung der Übungsreihenfolge nach Drag-and-Drop

```
async searchImages(keyword) {
  //Suche nach eingegebenem Wort auf Unsplash. Sobald ein neuer Buchstabe
  // eingegeben wird, cancele die alte Anfrage und sende eine neue
  if (this.cancel !== undefined) {
    this.cancel();
  }
  try {
    const request = await axios.get(this.baseURL + "&query=" + keyword, {
      cancelToken: new axios.CancelToken(c => {
        this.cancel = c;
      })
    });
```

Abbildung 3: Anfrage an unsplash.com

```
let filteredTrainings = [];
if (this.uniqueDaysWithTraining.length > 0) {
  //Zeige ausgewählten oder heutigen Tag an
  let index = this.state.index || this.uniqueDaysWithTraining.length - 1;
  //Finde Traings-Titel an diesem Tag
  let trainingsOnSelectedDay = this.props.lastWorkouts
    .filter(t => isSameDay(t.date, this.uniqueDaysWithTraining[index]))
    .map(t => t.title);
  //Reduziere diese auf einzigartige Trainings
  filteredTrainings = unique(trainingsOnSelectedDay);
  //Zähle, wie oft diese vorkommen
  filteredTrainings = filteredTrainings.map(name => {
    return {
      name,
      count: trainingsOnSelectedDay.reduce((acc, val) => {
        return acc + (val === name ? 1 : 0);
      }, 0)
    };
  });
}
```

Abbildung 4: Erstellung der Anzeige des ausgewählten Timeline-Eintrags

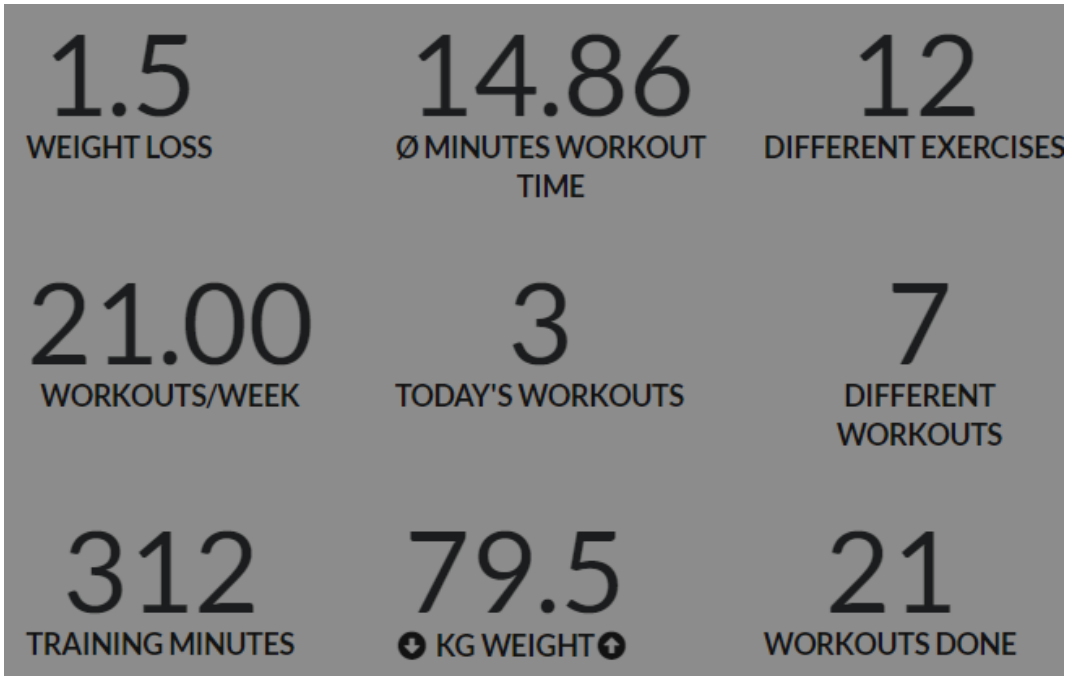

Abbildung 5: Darstellung der Statistiken in einem Grid

```
export function prepareDateInHistory(persistedState) {
  let { lastWorkouts, weight } = persistedState;
  for (let key in lastWorkouts) {
    lastWorkouts[key].date = new Date(lastWorkouts[key].date);
  }
  for (let key in weight) {
    weight[key].date = new Date(weight[key].date);
  }
}
```

Abbildung 6: Darstellung der Statistiken in einem Grid

```
export function announceExercise(index, workout, exerciseList, speech) {
  let exercise = findById(exerciseList, workout.exercises[index].id);
  speech.speak({
    text:
      (workout.exercises[index].duration || exercise.duration) +
      " seconds " +
      exercise.name,
    queue: false
  });
}
```

Abbildung 7: Hilfsfunktion zur Ansage der Übungen

```
return (
  <ExerciseScreen
    {...this.props}
    isRunning={this.state.isRunning}
    next={this.next.bind(this)}
    previous={this.previous.bind(this)}
    pauseTimer={this.pauseTimer.bind(this)}
    runTimer={this.runTimer.bind(this)}
  />
);
```

Abbildung 8: Übergeben der Properties an Unterelement

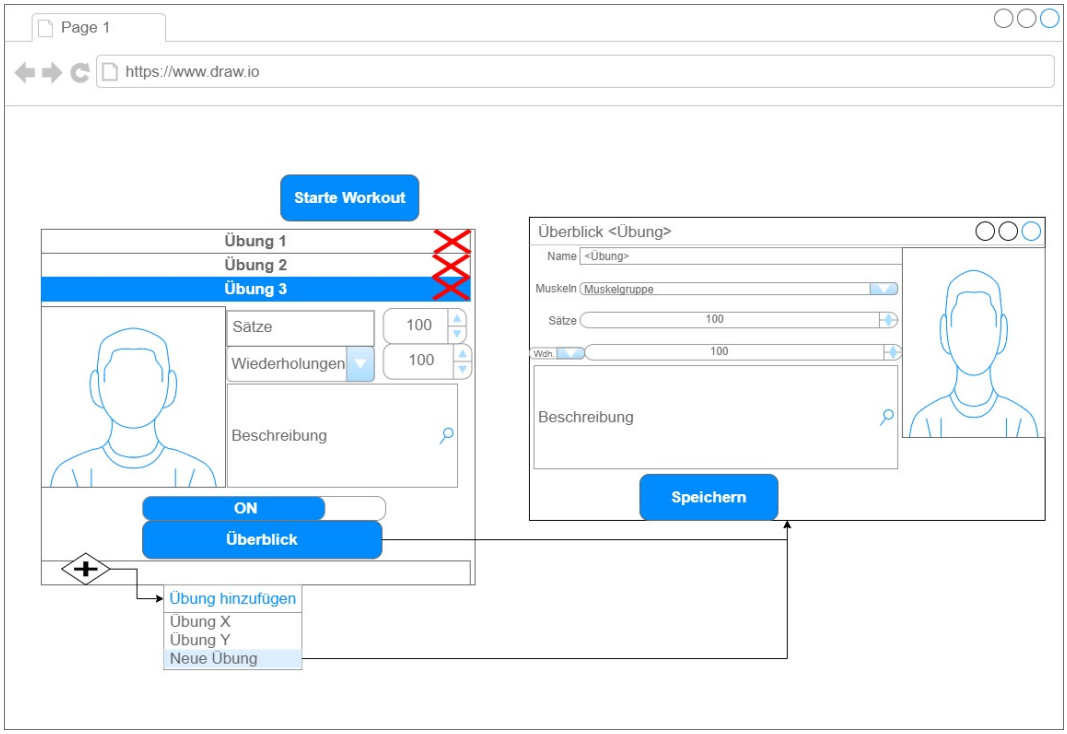

Abbildung 9: Das geplante Layout der App

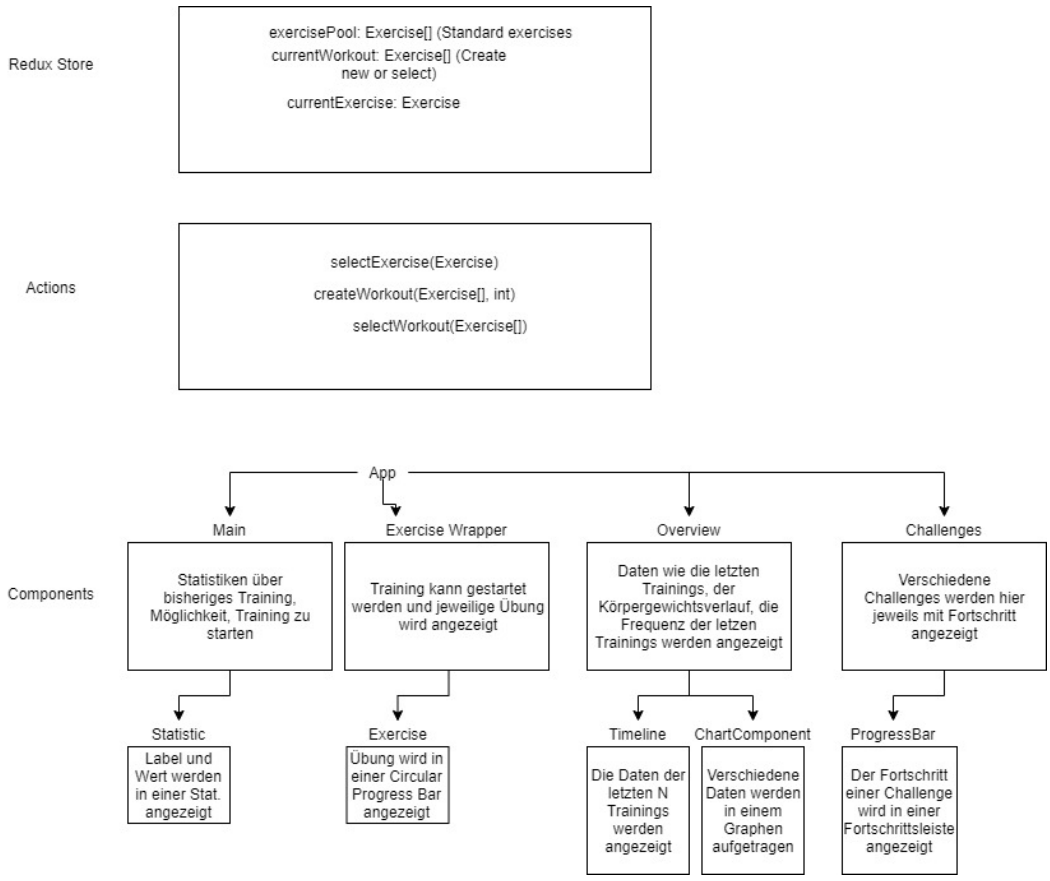

Abbildung 10: Die geplante Projektstruktur