

Studienarbeit zur Anwendung „GymJourney“

Alexej Kunz (00193820)
Hochschule für Angewandte Wissenschaften Hof

Moderne App- und Webentwicklung
Sommersemester 2023
23.06.2023

Inhalt

Abbildungsverzeichnis.....	3
1 Einführung	4
1.1 Anforderungsanalyse für GymJourney	4
1.2 Architektur und Entwurf	5
1.2.1 Eingesetzter Technologie-Stack.....	5
1.2.2 Informationsarchitektur.....	6
1.2.3 Datenbankschema	7
2 Implementierung.....	9
2.1 Überprüfung der Authentifizierung des Nutzers	9
2.2 Daten vor unberechtigtem Zugriff schützen.....	10
2.3 Einbindung eines YouTube-Videos	12
3 Walkthrough durch die GymJourney-Website.....	15
4 Fazit	24
5 Selbstständigkeitserklärung	25
Literaturverzeichnis.....	26

Abbildungsverzeichnis

1 React.js Logo.....	6
2 Express.js Logo	6
3 MySQL Logo	6
4 Technologie-Stack	6
5 Informationsarchitektur-Diagramm	7
6 Datenbankschema	8
7 authToken-Middleware in „backend/middlewares/authenticateToken.js“	9
8 workoutsAllowed-Middleware in „backend/middlewares/workoutsAllowed.js“	11
9 postExercise-Funktion in „backend/controllers/exercise.js“	12
10 Hilfsvideo-Abschnitt in „components/Accordion/Accordion.jsx“	13
11 YTEmbedded-Komponente in „components/YTEmbedded/YTEmbedded.jsx“	14
12 Login-Seite	16
13 Register-Seite	16
14 Home-Seite.....	17
15 Trainingsplan erstellen	17
16 Der erstellte Trainingsplan	18
17 Trainingsplan bearbeiten	18
18 Der bearbeitete Trainingsplan	19
19 Trainingsplan löschen	19
20 Erfolgreiche Löschung eines Trainingsplans	20
21 Workouts-Seite	20
22 Workouts-Seite mit gespeicherten Workouts	21
23 Übungen-Seite.....	22
24 Übungen-Seite mit gespeicherter Übung	22
25 Informationen zu einer Übung	23
26 Fehlerseite	23

1 Einführung

Die Website GymJourney richtet sich an sportbegeisterte Personen. Sie soll ihnen dabei helfen, möglichst einfach und unkompliziert die Übersicht über ihre Trainingspläne zu behalten. Diese Studienarbeit dient der Dokumentation der zentralen Inhalte der Implementierung, der Architektur und des Aufbaus der GymJourney-Website.

1.1 Anforderungsanalyse für GymJourney

In diesem Abschnitt wird eine Anforderungsanalyse durchgeführt. Durch eine Anforderungsanalyse werden vor Start eines Projekts die Anforderungen an dieses Projekt festgehalten, sodass jeder, der am Projekt beteiligt ist, genau weiß welche Anforderungen festgelegt wurden. Dadurch wird verhindert, dass Projekte scheitern, weil sich Anforderungen nachträglich geändert haben oder dass behauptet wird, dass vorher besprochene Anforderungen nicht umgesetzt wurden [vgl. 1].

- Bei der Anwendung soll es sich um eine Single Page Application (SPA) handeln
- Die Anwendung soll es dem Nutzer ermöglichen, einen Account zu erstellen und sich mit diesem Account anzumelden
- Die Anwendung soll eine sichere Aufbewahrung des Passworts des Nutzers gewährleisten, indem das Passwort gehasht und gesalted wird
- Die Sitzung des angemeldeten Nutzers soll über Cookies verwaltet werden
- Der Nutzer soll sich über einen Button von der Anwendung ausloggen können
- Es sollen nur angemeldete Nutzer Zugriff auf die Anwendung haben
- Der Nutzer soll in der Lage sein eigene Trainingspläne, Trainingseinheiten (auch Workouts genannt), und Übungen zu erstellen, angezeigt zu bekommen, zu bearbeiten und zu löschen
- Die Anwendung soll bei den Übungen YouTube-Videos einbinden können, sodass der Nutzer diese durch einen Link abspeichern kann und auf der Website anschauen kann

- Die Anwendung soll eine Fehlerseite anzeigen, sollte eine Route nicht gefunden werden
- Die Anwendung soll responsive sein und somit sowohl auf mobilen Endgeräten als auch auf Desktop-Geräten erfolgreich dargestellt werden können
- Die Anwendung soll nach der Hierarchiereihenfolge Trainingspläne, Trainingseinheiten und Übungen aufgebaut sein – wird ein Trainingsplan gelöscht so sollen auch alle Trainingseinheiten und Übungen dieses Trainingsplans gelöscht werden
- Der Nutzer soll nur auf seine eigenen Daten zugreifen können – Zugriffe auf Daten anderer Nutzer sollen vermieden werden
- Die Anwendung soll lediglich im Dark Mode gestaltet werden
- Das Frontend der Anwendung soll mit einem Framework wie beispielsweise React.js umgesetzt werden
- Es soll Routing mit Master-Detail-Views und parametrisierten URLs verwendet werden

1.2 Architektur und Entwurf

In diesem Abschnitt soll die Architektur der Anwendung erläutert werden, wobei zunächst auf den eingesetzten Technologie-Stack eingegangen wird. Daraufhin werden die Informationsarchitektur und das Datenbankschema näher beleuchtet.

1.2.1 Eingesetzter Technologie-Stack

Das Frontend der Anwendung wurde mit React.js geschrieben. React.js ist ein JavaScript-Framework, das verwendet wird, um User Interfaces zu erstellen. Es ist das beliebteste komponentenbasierte JavaScript-Framework [vgl. 2]. Alternativen hierzu wären unter anderem Angular.js, Vue.js oder Hof.js, jedoch wurde sich aufgrund persönlicher Vorerfahrung mit React.js für React.js entschieden. Für das Backend wurde Node.js mit dem Framework Express verwendet. Node.js an sich ermöglicht es unter anderem serverseitige Applikationen mit JavaScript zu schreiben. Node.js hat von sich aus jedoch nicht die Möglichkeit HTTP-Anfragen wie GET, PUT, POST, DELETE und Co. zu verarbeiten, weshalb Express genutzt wird, da es sonst nötig wäre, eine eigene Logik für

das Verarbeiten von HTTP-Anfragen zu schreiben. Express ist das beliebteste Node-Webframework [vgl. 3]. Alternative Backend-Frameworks wären unter anderem Laravel oder Ruby on Rails, jedoch wurde sich auch hier aufgrund von persönlichen Vorerfahrungen mit JavaScript für Express entschieden. Um Daten aus dieser Anwendung zu speichern, wird eine Datenbank benötigt. Hier wurde sich für MySQL entschieden, da man hier einfach voneinander abhängigen Entitäten abbilden kann und es eines der beliebtesten Datenbankmanagementsysteme ist. Die Beliebtheit lässt sich wohl vor allem damit begründen, dass MySQL häufig in WordPress-Anwendungen zum Einsatz kommt [vgl. 4]. Mögliche Alternativen wären hier unter anderem PostgreSQL, SQLite oder auch MariaDB. Eine Übersicht über den Technologie-Stack bietet Abbildung 4.

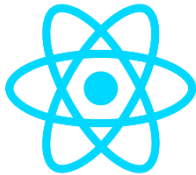


Frontend	Backend	Datenbank
<p>React.js</p>  <p>Abbildung 1: React.js Logo [5]</p>	<p>Express.js</p>  <p>Abbildung 2: Express.js Logo [6]</p>	<p>MySQL</p>  <p>Abbildung 3: MySQL Logo [7]</p>

Abbildung 4: Technologie-Stack

1.2.2 Informationsarchitektur

In diesem Abschnitt wird ein Informationsarchitektur-Diagramm abgebildet. Die Informationsarchitektur gibt an, wie die Informationen einer Website organisiert sind [vgl. 8]. Das Diagramm aus Abbildung 5 stellt hier also dar, von welcher Seite aus der Nutzer welche anderen Seiten der GymJourney-Website erreichen kann.

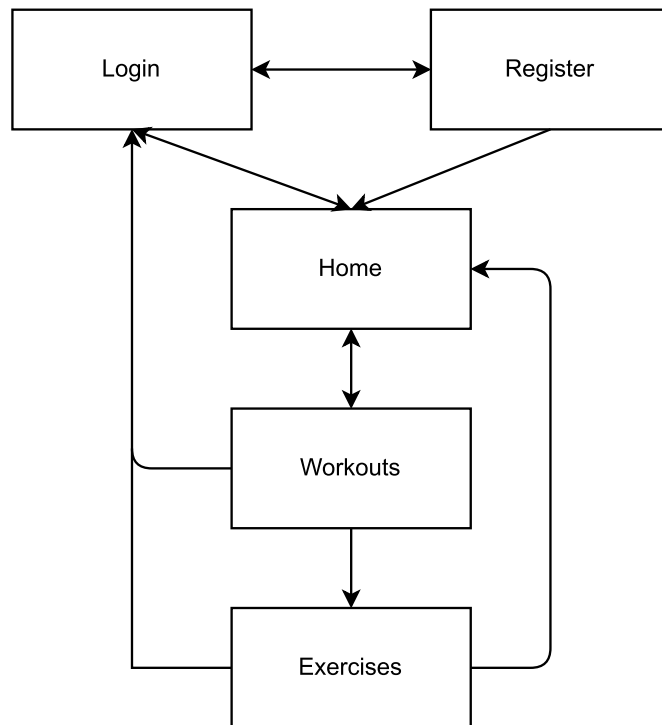


Abbildung 5: Informationsarchitektur-Diagramm

Wie zu sehen ist, kann die Home-Seite, die die Trainingspläne des Nutzers auflistet, aufgrund der Navigationsleiste von jeder anderen Seite aus aufgerufen werden. Auch die Login-Seite kann von jeder anderen Seite aus aufgerufen werden, weil der Nutzer sich von jeder Seite aus über die Navigationsleiste abmelden kann und daraufhin auf der Login-Seite landet. Die Informationsarchitektur der GymJourney-Website kommt der linearen Informationsarchitektur am nächsten, weil der Nutzer hier zumindest auf den Hauptseiten „Home“, „Workouts“ und „Exercises“ strukturiert durch die Website geführt wird [vgl. 8].

1.2.3 Datenbankschema

In Abbildung 6 wird das Datenbankschema der GymJourney-Website visualisiert. Das Datenbankschema veranschaulicht, wie die Daten in der Datenbank strukturiert sind [vgl. 9].

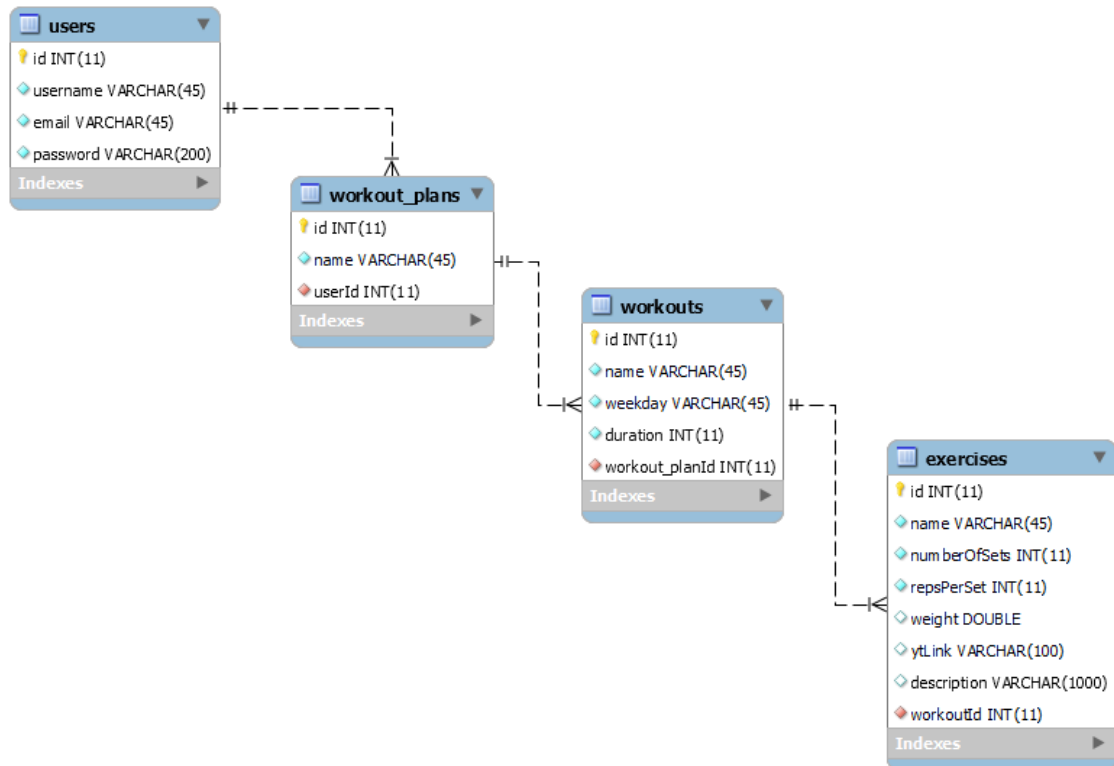


Abbildung 6: Datenbankschema

Wie hier zu sehen ist, liegen in der Datenbank die vier Tabellen `users`, `workout_plans`, `workouts` und `exercises`. Zwischen diesen Tabellen gibt es drei 1:N-Beziehungen, nämlich zwischen `users` und `workout_plans`, zwischen `workout_plans` und `workouts` und zwischen `workouts` und `exercises`. Das Datenbankschema ist eine Folge der aus der Aufgabenstellung dieser Studienarbeit geforderten Master-Detail-Beziehung. Die Tabelle `users` beinhaltet die Informationen zu den Nutzern der Anwendung inklusive deren Passwörter, die vor der Speicherung durch Hashing mit einem Salt geschützt werden. Entsprechend deren Tabellennamen beinhaltet die Tabelle `workout_plans` die Informationen zu Trainingsplänen, die Tabelle `workouts` die Informationen zu den Trainingseinheiten und die Tabelle `exercises` die Informationen zu den Übungen.

2 Implementierung

In diesem Abschnitt werden wichtige Aspekte der Implementierung der GymJourney-Website dokumentiert.

2.1 Überprüfung der Authentifizierung des Nutzers

Hier wird die Implementierung dokumentiert, die dafür sorgt, dass erfolgreich überprüft werden kann, ob ein Nutzer authentifiziert ist oder nicht. Dies ist vor allem für die GymJourney-Website von großer Bedeutung, da nur authentifizierte Nutzer die Website nutzen sollen. Um dies zu gewährleisten, wird für jede Route die Middleware „authToken“ dazwischengeschaltet. Eine Ausnahme ist hier die Datei „backend/routes/auth.js“, da hier die Routen für das Anmelden, das Registrieren und das Abmelden des Nutzers hinterlegt sind und ein Nutzer nicht vor der Anmeldung bzw. Registrierung authentifiziert sein kann. Die Implementierung der authToken-Middleware ist in Abbildung 7 zu sehen.

```
import jwt from "jsonwebtoken";

const authToken = async (req, res, next) => {
  const token = req.cookies.accessToken;

  if(!token){
    return res.status(401).json({msg: "Nutzer ist nicht angemeldet!"});
  }

  try{
    const user = jwt.verify(token, "verysecretkey");
    req.userId = user.id;
    next();
  }catch (error) {
    res.status(403).json({msg: "Der Token ist nicht gültig!"});
  }
}

export default authToken;
```

Abbildung 7: authToken-Middleware in „backend/middlewares/authenticateToken.js“

Hier wird zunächst der JSON Web Token (JWT) mit dem Namen „accessToken“ ausgelesen, der in den Cookies gespeichert wurde, nachdem sich der Nutzer angemeldet bzw. registriert hat. Ist dieser Token nicht in den Cookies abgespeichert, so ist der Nutzer

nicht angemeldet. Um den Nutzer über diesen Fehler zu informieren, wird als Antwort der Status-Code 401 (Unauthorized, [vgl. 10]) mit der Nachricht „Nutzer ist nicht angemeldet!“ zurückgeliefert. Ist jedoch ein Token vorhanden, so wird dieser durch die `jwt.verify`-Funktion überprüft. Durch diese Funktion wird mit dem „`verysecretkey`“, der auch verwendet wurde, um den Token zu erstellen, sichergestellt, dass der Token nicht manipuliert wurde. Dieser Key sollte normalerweise in den Umgebungsvariablen gespeichert werden, ist hier aber öffentlich einsehbar, weil es sich um keine produktive Website handelt. Ist der Token gültig, so wird der entschlüsselte Payload zurückgeliefert und in der Variable „`user`“ gespeichert [vgl. 11]. In diesem Fall besteht der Payload aus der Id des Nutzers und seinem Benutzernamen. Da die Id des Nutzers in vielen Anfragen benötigt wird, wird sie hier unter „`req.userId`“ abgespeichert und kann von den nachfolgenden Funktionen verwendet werden. Der Funktionsaufruf `next()` sorgt dafür, dass nun die nächste Funktion aufgerufen wird. Diese kann in dieser Applikation ebenfalls eine Middleware oder ein Controller sein. Ist der Token ungültig, so wird nach der `jwt.verify`-Funktion der `catch`-Block ausgeführt und der Status-Code 403 (Forbidden, [vgl. 10]) mit der Nachricht „Der Token ist nicht gültig!“ zurückgeliefert.

2.2 Daten vor unberechtigtem Zugriff schützen

Da ein Nutzer der GymJourney-Website lediglich Zugriff auf seine eigenen Daten haben soll und sich nicht beispielsweise die Trainingseinheiten eines anderen Nutzers ansehen darf, muss das über eine Middleware verhindert werden. Insgesamt gibt es zwei verschiedene Middlewares dafür, wobei sich eine um die Trainingspläne und um die Trainingseinheiten (`workoutsAllowed`) und eine um die Übungen (`exercisesAllowed`) kümmert. In diesem Abschnitt wird lediglich die Middleware `workoutsAllowed` betrachtet, welche in Abbildung 8 zu sehen ist. Die andere Middleware ist jedoch sehr ähnlich aufgebaut.

```

import { db } from "../connect.js";

const workoutsAllowed = async (req, res, next) => {
  const workoutPlanId = req.params.workout_planId;
  const userId = req.userId;

  const q = "SELECT userId FROM workout_plans WHERE id = ?";

  db.query(q, [workoutPlanId], (error, data) => {
    if(error) return res.status(500).json(error);

    const workoutPlan = data[0];

    if(workoutPlan?.userId !== userId){
      return res.status(403).json("Forbidden");
    }else{
      next();
    }
  })
}

export default workoutsAllowed;

```

Abbildung 8: workoutsAllowed-Middleware in „backend/middlewares/workoutsAllowed.js“

Zu Beginn wird die Id des Trainingsplans aus dem Request-Header ausgelesen und in der Variable „workoutPlanId“ gespeichert. Des Weiteren wird die Id des Nutzers, der die Anfrage stellt, in der Variable „userId“ abgelegt. Wie zu sehen ist, kann auf die Id des Nutzers über req.userId zugegriffen werden, was daran liegt, dass diese Middleware nach der authToken-Middleware aus Abschnitt 2.1 aufgerufen wird. Der Nutzer versucht hier auf Trainingseinheiten zuzugreifen. Aus diesem Grund wird zunächst eine Query aufgebaut, die die Id des Nutzers abfragt, dem der Trainingsplan gehört, auf dessen Trainingseinheit der Nutzer versucht zuzugreifen. Bei dieser Query handelt es sich um eine parametrisierte Query, die verwendet wird, um vor SQL-Injection-Attacken zu schützen [vgl. 12]. Sollte es in dieser Query zu einem Fehler kommen, so wird der Status-Code 500 (Internal Server Error, [vgl. 10]) mit der Fehlermeldung zurückgegeben. Hat jedoch alles geklappt, so wird abgefragt, ob der Nutzer, dem der Trainingsplan gehört, ein anderer Nutzer ist, als derjenige, der versucht auf eine Trainingseinheit des Trainingsplans zuzugreifen. Ist dies der Fall, wird der Status-Code 403 (Forbidden, [vgl. 10]) zurückgegeben. Gehört dem anfragenden Nutzer aber der Trainingsplan, so wird mit next() die nächste Funktion aufgerufen, die die Anfrage dann tatsächlich beantwortet.

2.3 Einbindung eines YouTube-Videos

Der Nutzer soll die Möglichkeit haben, für seine Übungen YouTube-Videos einzubinden, um beispielsweise ein Tutorial abzuspeichern, das ihm die richtige Ausführung einer Übung präsentiert. In diesem Abschnitt wird auf die Implementierung dieser Funktionalität eingegangen.

Zunächst wird betrachtet, wie eine Übung des Nutzers abgespeichert wird. Dies wird im Backend in der Funktion „postExercise“ durchgeführt. Diese ist in Abbildung 9 zu sehen. Hier wird unter anderem ein eingegebener YouTube-Link des Nutzers überprüft und abgespeichert.

```
export const postExercise = (req, res) => {
  const q = "INSERT INTO exercises (`name`, `numberOfSets`, `repsPerSet`,
    `weight`, `ytLink`, `description`,
    `workoutId`) VALUES (?)";

  const ytLink = req.body.ytLink;
  const ytRegex = /(?:(:?https?:)?\\\/)?(?:www\\.)?
    (?:youtube\\.com|youtu\\.be)\\\/(?:watch?v=|embed\\/)?([a-zA-Z0-9_-]{11})/;
  const match = ytLink.match(ytRegex);
  let ytEmbeddedLink = "";

  if (match) {
    ytEmbeddedLink = `https://www.youtube.com/embed/${match[1]}`;
  } else {
    if (ytLink !== "") {
      return res.status(400).json("Kein gültiger YouTube-Link!");
    }
  }

  const values = [req.body.name, req.body.numberOfSets, req.body.repsPerSet,
    req.body.weight, ytEmbeddedLink, req.body.description,
    req.params.workoutId];

  db.query(q, [values], (err, data) => {
    if (err) return res.status(500).json(err);
    return res.status(200).json("Übung erfolgreich angelegt!");
  })
}
```

Abbildung 9: postExercise-Funktion in „backend/controllers/exercise.js“

Zunächst wird der YouTube-Link des Nutzers aus dem Request-Body ausgelesen und in der Variable „ytLink“ gespeichert. Daraufhin wird ein Regex definiert, der überprüfen soll, ob es sich bei dem Link tatsächlich um ein YouTube-Link handelt. Um die Überprüfung durchzuführen, wird die JavaScript-Funktion match() verwendet, die eine

Reguläre Ausdrucks-Übereinstimmung mit dem YouTube-Link des Nutzers durchführt. Handelt es sich um einen gültigen Link, liefert match ein Array zurück. An Stelle 0 des Arrays befindet sich der gesamte YouTube-Link, an Stelle 1 befindet sich die YouTube-Video-ID [vgl. 13]. Dieses Array wird in der Variable „match“ gespeichert. Handelt es sich um keinen gültigen YouTube-Link, so ist die Variable „match“ leer und es wird der HTTP-Code 400 (Bad Request, [vgl. 10]) mit der Fehlermeldung „Kein gültiger YouTube-Link!“ zurückgeliefert. Ist der Link jedoch gültig, so wird in der Variable „ytEmbeddedLink“ ein YouTube-Embed-Link gespeichert, der später im Frontend benötigt wird, um das Video anzuzeigen. Nun kann eine Query mit allen Daten des Nutzers aufgebaut werden. Auch hier handelt es sich wie in Abbildung 8 um eine parametrisierte Query, die vor SQL-Injection-Angriffen geschützt ist. Sollte es in dieser Query zu einem Fehler kommen, wird der Status-Code 500 (Internal Server Error, [vgl. 10]) mit der Fehlermeldung zurückgeliefert. Andernfalls wird die Übung mit dem YouTube-Link, falls angegeben, in der Datenbank gespeichert und es wird der Status-Code 200 (OK, [vgl. 10]) mit der Nachricht „Übung erfolgreich angelegt!“ zurückgeliefert.

Nun wird das Frontend näher betrachtet. Die Komponente „components/Accordion/Accordion.jsx“ erhält alle Daten einer Übung von der Seite „pages/Exercises/Exercises.jsx“ über die Prop „data“ und stellt diese Daten für den Nutzer dar. Ein Ausschnitt der Accordion-Komponente ist in Abbildung 10 zu sehen.

```
{data.ytLink &&
  <div className={styles.ytVideo}>
    <p className={styles.toggleContainerHeadline}>💡Hilfsvideo</p>
    <YTEmbedded ytLink={data.ytLink} />
  </div>
}
```

Abbildung 10: Hilfsvideo-Abschnitt in „components/Accordion/Accordion.jsx“

In diesem Abschnitt wird überprüft, ob in der erhaltenen Übung ein YouTube-Link abgespeichert wurde. Ist das der Fall, so wird ein div gerendert, in dem sich ein p-Element befindet, das die Überschrift „Hilfsvideo“ beinhaltet sowie die Komponente „YTEmbedded“, die in Abbildung 11 zu sehen ist. Wurde kein YouTube-Video abgespeichert, so wird diese div nicht gerendert.

```

import React, { useState, useEffect } from 'react';
import LazyLoad from 'react-lazy-load';

function YTEmbedded({ ytLink }) {
  const [screenWidth, setScreenWidth] = useState(window.innerWidth);

  useEffect(() => {
    const handleResize = () => setScreenWidth(window.innerWidth);
    window.addEventListener("resize", handleResize);
    return () => window.removeEventListener("resize", handleResize);
  }, []);

  const width = screenWidth > 1024 ? 700 : "90%";

  return (
    <div>
      <LazyLoad height={315}>
        <iframe
          width={width}
          height="315"
          src={ytLink}
          title="Embedded YouTube-Video"
          allow="fullscreen"
          style={{zIndex: 0, border: "none", borderRadius: "10px" }}
        />
      </LazyLoad>
    </div>
  )
}

export default YTEmbedded;

```

Abbildung 11: YTEmbedded-Komponente in „components/YTEmbedded/YTEmbedded.jsx“

Obige Komponente ist verantwortlich für das Einbinden eines YouTube-Videos. Zunächst wird sich darum gekümmert, dass das YouTube-Video in einer ordentlichen Breite dargestellt wird. Dafür wird der State „screenWidth“ verwendet, der die aktuelle Breite des Fensters als Anfangswert erhält. Innerhalb eines useEffect-Hooks, wird ein Event-Listener für das „resize“-Event des Fensters erstellt, der die Funktion „handleResize“ aufruft, sobald sich die Fenstergröße ändert. Die Funktion „handleResize“ sorgt dann dafür, dass der State „screenWidth“ auf die neue Breite des Fensters gesetzt wird. Der eben erwähnte Listener wird in der „return“-Anweisung des useEffect-Hooks entfernt. Das verhindert, dass der Event-Listener weiter ausgeführt wird, wenn die Komponente nicht mehr im DOM existiert. In der Variable „width“ wird die Breite des iFrames berechnet, welches das YouTube Video einbindet. Ist die Breite des Fensters größer als 1024px, so wird die Breite auf 700px gesetzt, ansonsten wird sie auf 90% der verfügbaren Breite gesetzt.

Nun folgt die tatsächliche Einbindung des YouTube-Videos. Innerhalb einer div wird die LazyLoad-Komponente der React-Bibliothek „react-lazy-load“ gerendert, die für ein Lazy Loading der eingebundenen YouTube-Videos sorgt. Diese Bibliothek kann aber auch beispielsweise für das Lazy Loaden von Bildern verwendet werden [vgl. 14]. Lazy Loading sorgt dafür, dass Daten erst dann geladen werden, wenn sie im sichtbaren Bereich des Nutzers erscheinen. Dies verbessert die Performance der Website [vgl. 15]. Innerhalb der LazyLoad-Komponente befindet sich ein iframe-Tag. Das iframe-Tag ist verantwortlich für das Einbinden des YouTube-Videos und erhält die oben beschriebene Breite und eine feste Höhe, sowie ein src-Attribut mit dem von der Accordion-Komponente aus Abbildung 10 erhaltenen YouTube-Link. Das Attribut „title“ ist wichtig für Personen, die z.B. einen Screen Reader verwenden. Es unterstützt Personen mit Sehbehinderungen dabei, sich auf der Website zurechtzufinden. Das Attribut ‘allow=“fullscreen“‘ sorgt dafür, dass der Vollbildmodus des YouTube-Videos aktiviert werden kann und das Style-Attribut ändert das Aussehen des iframes ein wenig [vgl. 16].

3 Walkthrough durch die GymJourney-Website

Im Folgenden wird durch die GymJourney-Website geleitet, um die Bedienung sowie die Möglichkeiten der Website zu dokumentieren.

Zunächst landet der Nutzer auf der in Abbildung 12 dargestellten Login-Seite, zumindest wenn er nicht angemeldet ist. Grund dafür ist, dass der Nutzer für jede Funktionalität, die diese Website bietet, angemeldet sein muss. Sollte der Nutzer noch kein Account besitzen, so kann er auf den Link „Jetzt registrieren“ klicken und landet auf der Register-Seite, um sich einen neuen Account zu erstellen. Diese ist in Abbildung 13 abgebildet.

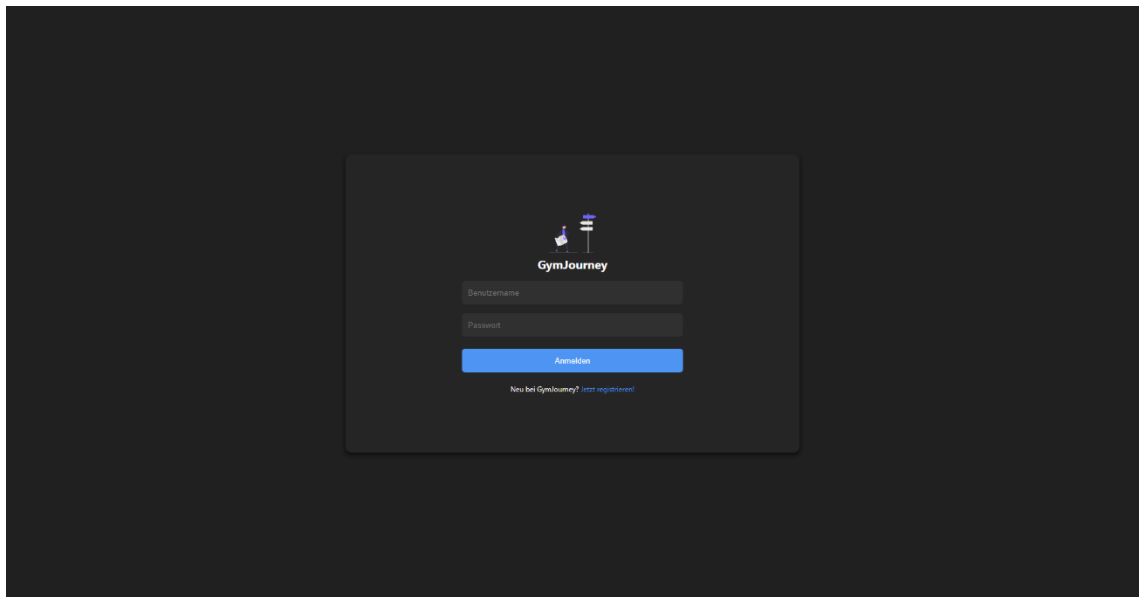


Abbildung 12: Login-Seite

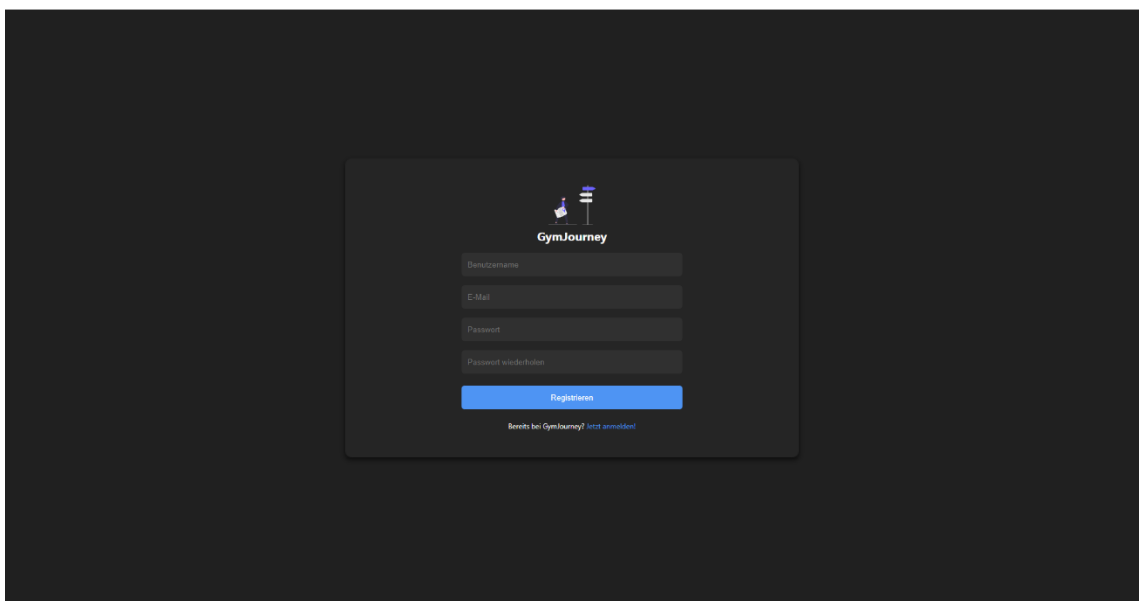


Abbildung 13: Register-Seite

Sowohl nach einem erfolgreichen Login als auch nach einer erfolgreichen Registrierung landet man auf der in Abbildung 14 dargestellten Home-Seite. Am oberen Bildschirmrand ist eine Navigationsleiste zu sehen, die das Logo der GymJourney-Website darstellt, mit dem man per Klick wieder auf die Home-Seite zurückkehren kann, sowie den Namen des angemeldeten Nutzers und einen Button, mit dem sich der Nutzer abmelden kann. Auf der Home-Seite werden die Trainingspläne abgebildet. Da der Nutzer aber noch keine Trainingspläne erstellt hat, werden ihm auch keine angezeigt. Um einen neuen Trainingsplan zu erstellen, klickt man auf den „+“ Button. Daraufhin öffnet sich ein

Fenster, in dem man den Namen für den Trainingsplan vergeben kann, wie Abbildung 15 zeigt.

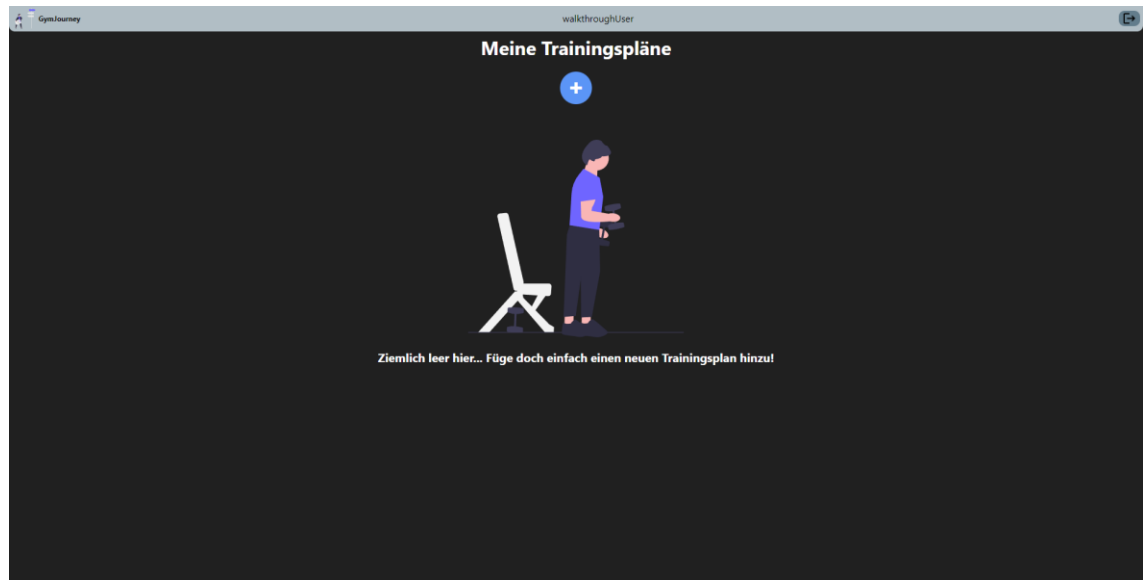


Abbildung 14: Home-Seite

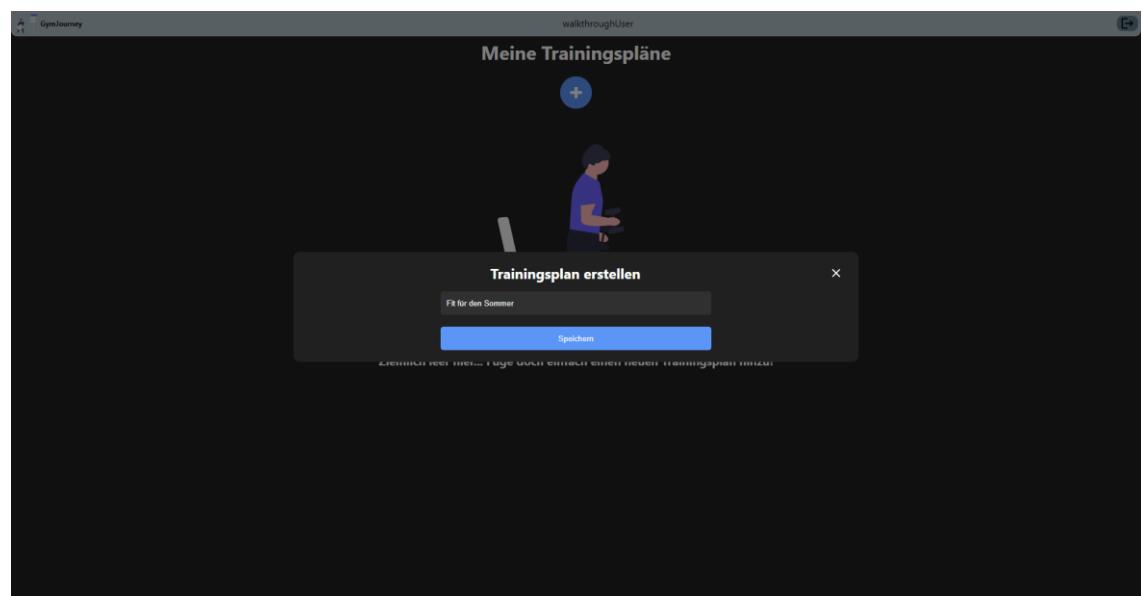


Abbildung 15: Trainingsplan erstellen

Klickt man auf den „Speichern“ Button des Fensters, so wird ein neuer Trainingsplan erstellt. Dieser Trainingsplan wird dann direkt auf der Home-Seite angezeigt zusammen mit einer Erfolgsmeldung über die Erstellung des Trainingsplans. Beides kann in Abbildung 16 gesehen werden.

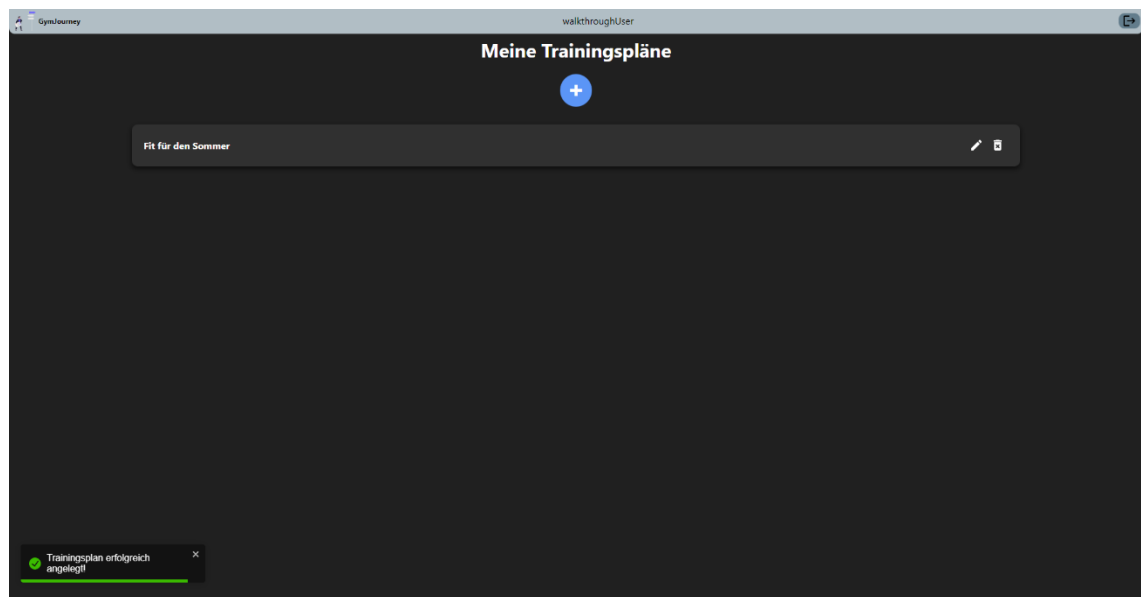


Abbildung 16: Der erstellte Trainingsplan

Klickt man auf das Stift-Icon im Container des erstellten Trainingsplans, so öffnet sich ein Fenster, bei dem man den Namen des Trainingsplans ändern kann (siehe Abbildung 17).

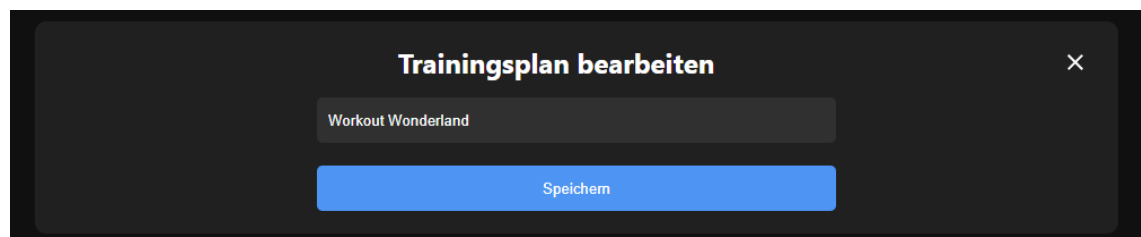


Abbildung 17: Trainingsplan bearbeiten

Betätigt man in diesem Fenster die Speichern Taste, so wird der neue Name des Trainingsplans übernommen und zusammen mit einer Erfolgsmeldung auf der Home-Seite dargestellt (siehe Abbildung 18).

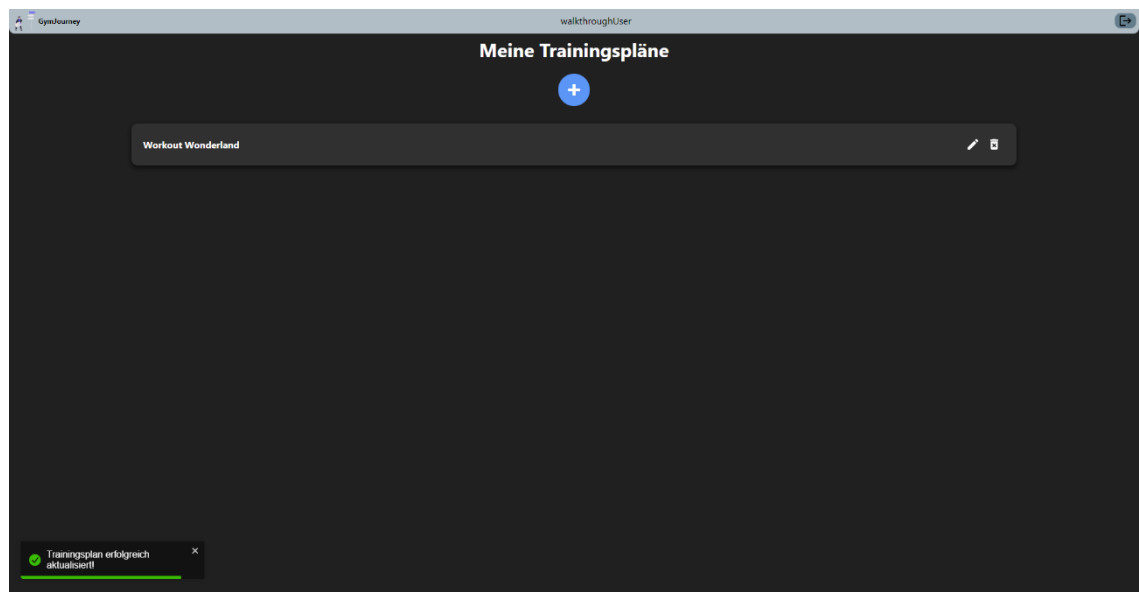


Abbildung 18: Der bearbeitete Trainingsplan

Klickt man nun auf das Mülleimer-Icon im Trainingsplan-Container, so kann der Trainingsplan gelöscht werden. Auch hierfür öffnet sich zunächst ein Fenster, das auf Abbildung 19 dargestellt wird.

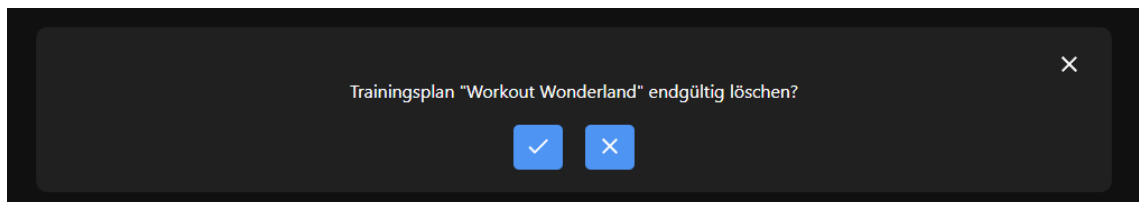


Abbildung 19: Trainingsplan löschen

In diesem Fenster hat man nun die Möglichkeit die Aktion durch Klick auf den rechten Button abubrechen, während der Trainingsplan durch klick auf den linken Button endgültig gelöscht wird und auch nicht wieder hergestellt werden kann. Wird ein Trainingsplan gelöscht, so werden auch alle zum Trainingsplan gehörigen Workouts und Übungen gelöscht. Führt man das Löschen aus, so erhält man auch hier wieder eine Erfolgsmeldung und sieht den gelöschten Trainingsplan nicht mehr auf der Home-Seite, wie Abbildung 20 zeigt.

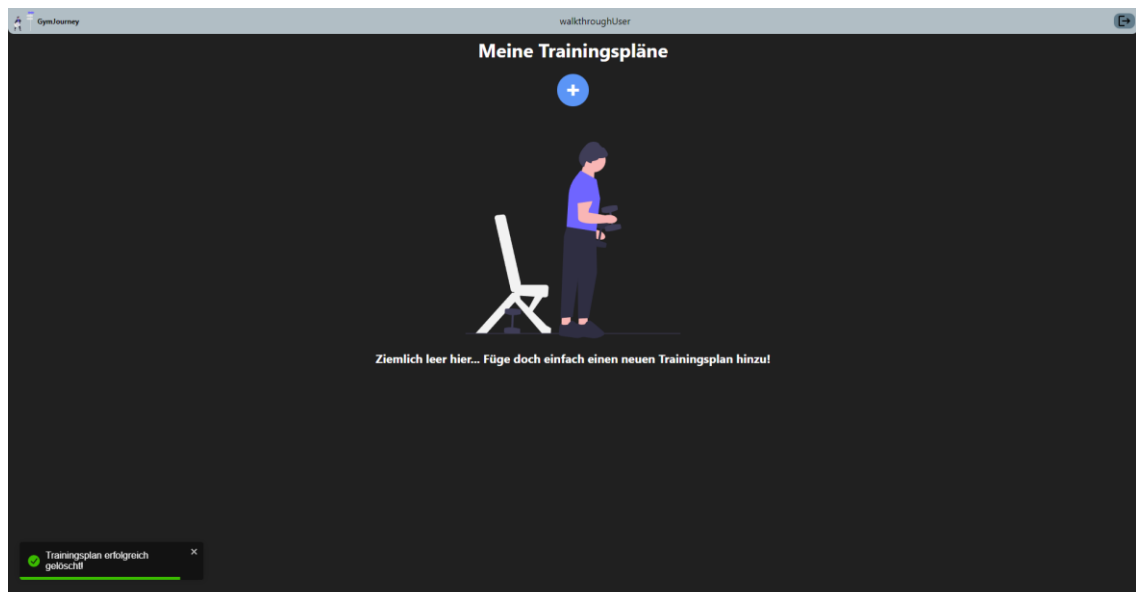


Abbildung 20: Erfolgreiche Löschung eines Trainingsplans

Klickt man auf einen Trainingsplan aus der Trainingsplan-Seite wie beispielsweise „Workout Wonderland“ aus Abbildung 18, so landet man auf der in Abbildung 21 dargestellten Workouts-Seite des jeweiligen Trainingsplans, bei der aufgelistet wird, welches Workout an welchem Wochentag geplant ist. Der aktuelle Tag wird hierbei farblich hervorgehoben. Da aktuell noch keine Workouts erstellt wurden, ist an keinem Tag ein Workout geplant.

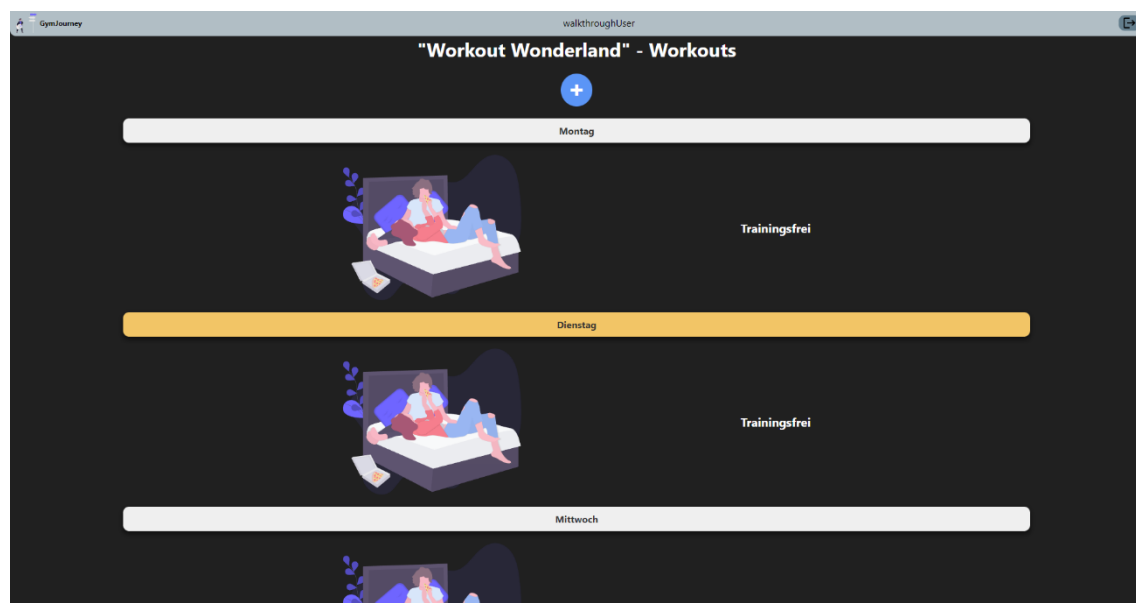


Abbildung 21: Workouts-Seite

Hier kann durch einen Klick auf den „+“-Button ein neues Workout erstellt werden. Da sich hier ein ähnliches Fenster öffnet wie in Abbildung 15, wird hier auf eine Abbildung

verzichtet. Der einzige Unterschied ist lediglich, dass hier andere Angaben gemacht werden müssen. Es muss ein Wochentag aus einem Drop-Down-Menü ausgewählt werden und der Name des Workouts sowie die Dauer dieses Workouts in Minuten müssen angegeben werden. Danach kann auf Speichern geklickt werden und das Workout wird abgebildet. In Abbildung 22 wird gezeigt, wie die Workouts-Seite mit einigen gespeicherten Workouts aussieht.

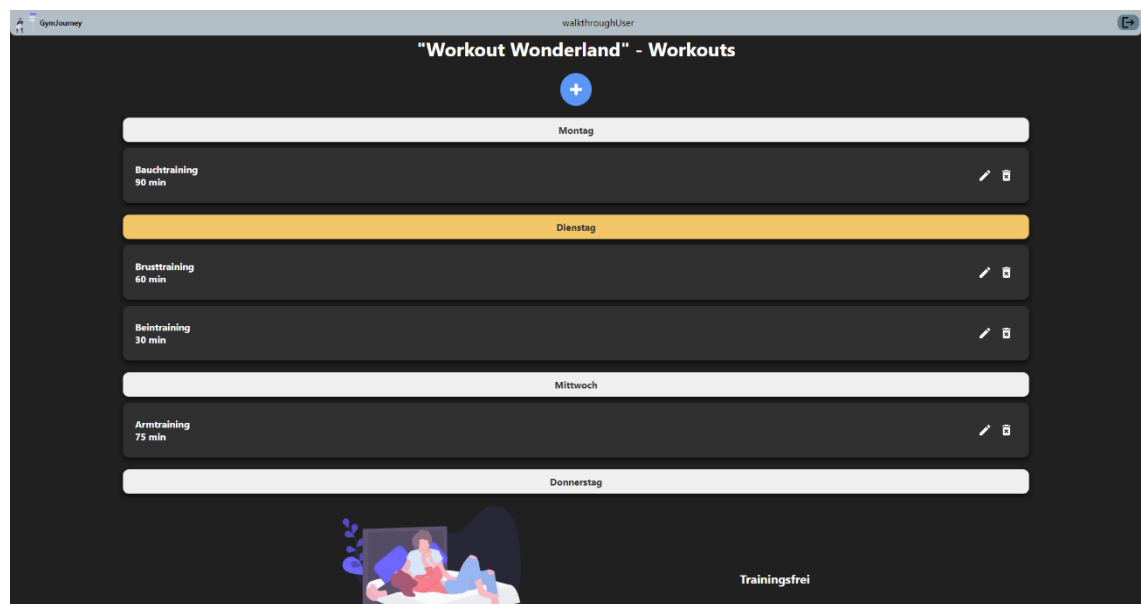


Abbildung 22: Workouts-Seite mit gespeicherten Workouts

Möchte man ein Workout bearbeiten, so klickt man auch hier auf das Stift-Icon und es öffnet sich auch hier ein PopUp wie in Abbildung 17, in dem man nun den Wochentag, den Namen und die Länge des Trainingsplans ändern kann. Für das Löschen eines Workouts betätigt man das Mülleimer-Icon und bestätigt in einem Ähnlichen PopUp wie in Abbildung 19 die Löschung des Workouts. Wird ein Workout gelöscht, so werden auch alle zugehörigen Übungen gelöscht.

Für ein Workout können beliebig viele Übungen erstellt werden. Um Übungen für ein Workout zu erstellen, wählt man einen Workout aus der Liste wie in Abbildung 22 aus und landet auf der in Abbildung 23 dargestellten Übungen-Seite.

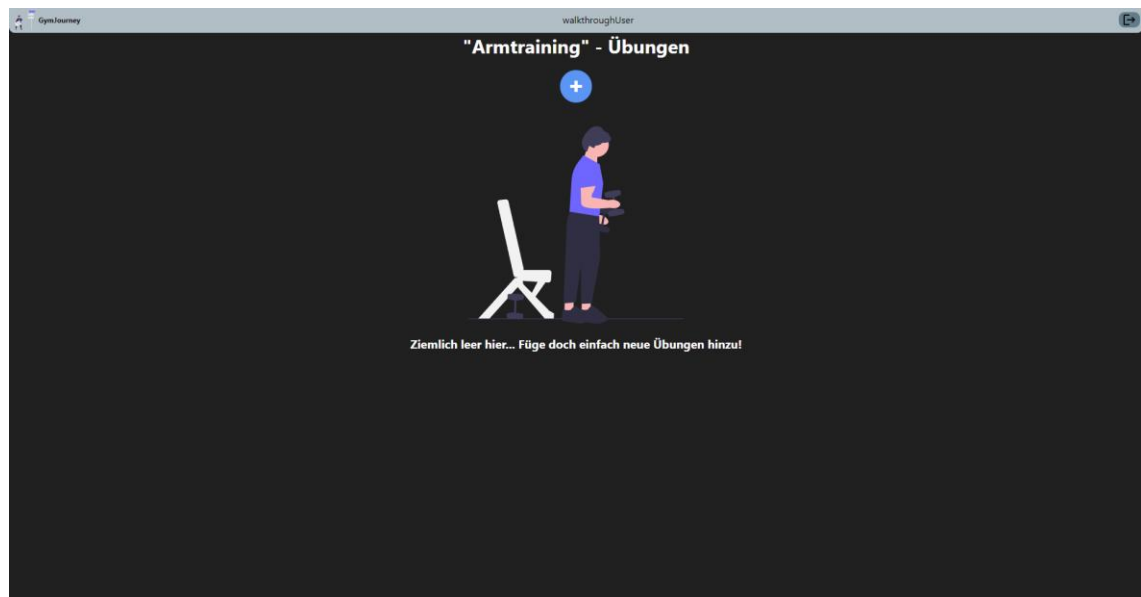


Abbildung 23: Übungen-Seite

Auch diese Seite ist zunächst leer, jedoch können, wie in Abbildung 15, mit Klick auf den „+“-Button durch ein PopUp neue Übungen erstellt werden. Hierfür **müssen** der Name der Übung, die Anzahl der Sätze sowie die Anzahl der Wiederholungen pro Satz angegeben werden, während das verwendete Gewicht für die Übung in kg, ein Link zu einem YouTube-Tutorial sowie die Beschreibung der Übung **freiwillig** sind. Abbildung 24 zeigt die Übungen-Seite nach dem Erstellen einer Übung.

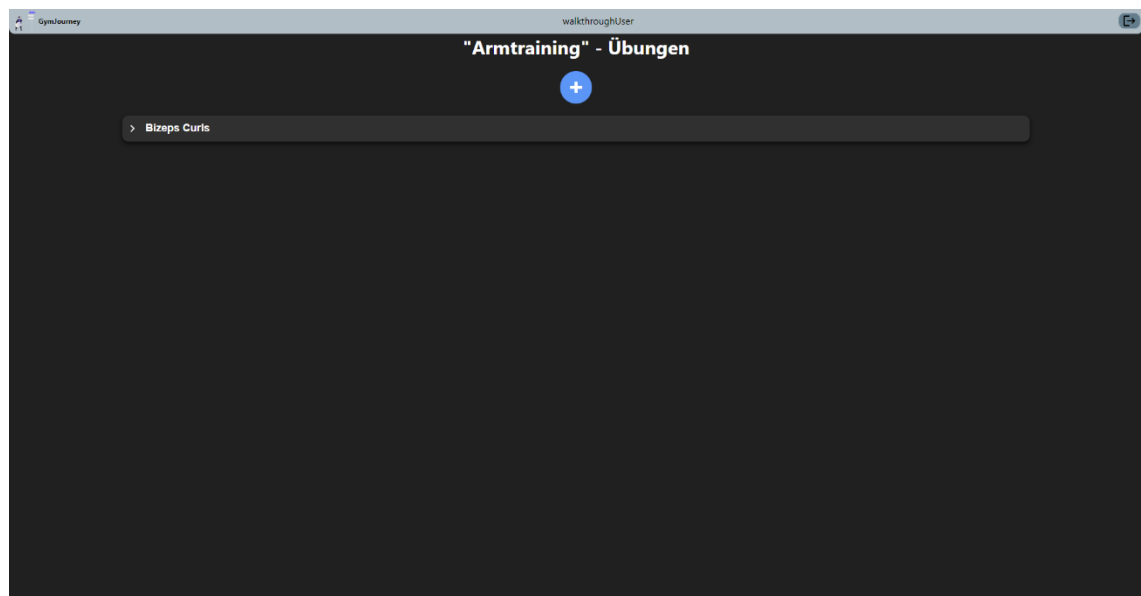


Abbildung 24: Übungen-Seite mit gespeicherter Übung

Um weitere Informationen zu der Übung zu sehen, klickt man lediglich auf die Übung. Das Resultat davon zeigt Abbildung 25.

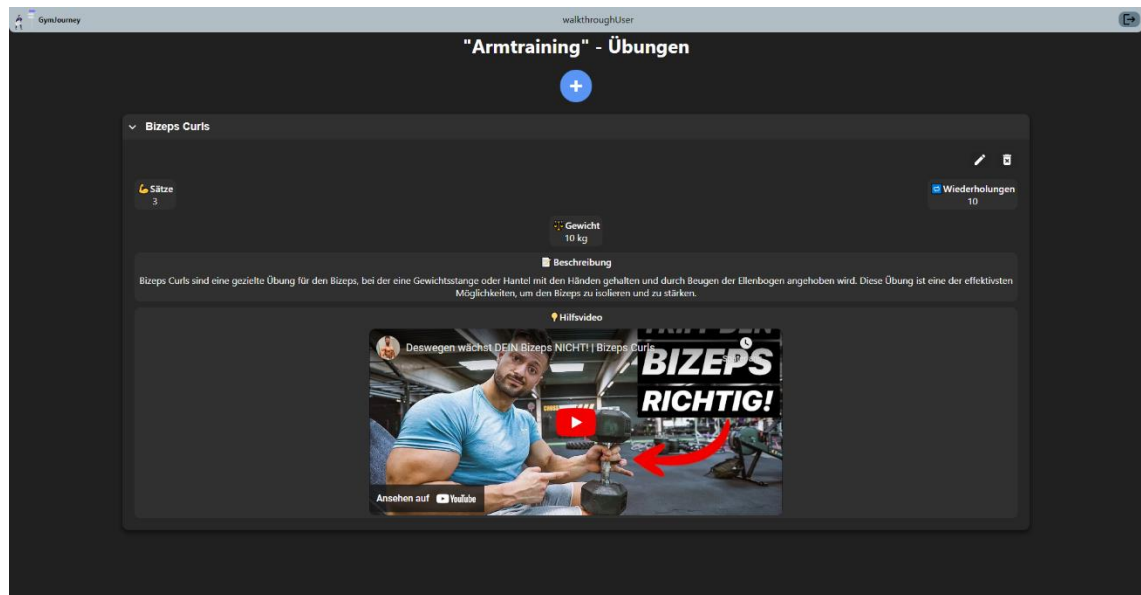


Abbildung 25: Informationen zu einer Übung

Hier erhält man nun auch die Möglichkeit die Übung durch Klick auf das Stift-Icon ähnlich wie in Abbildung 17 zu bearbeiten, in dem man nun jegliche Angabe, die man zur Übung gemacht hat, in einem PopUp verändern kann. Durch einen Klick auf das Mülleimer-Icon ist es auch hier möglich, nach Bestätigung eines PopUp wie in Abbildung 19, die Übung zu löschen.

Gelangt der Nutzer beispielsweise durch Eingabe einer URL auf einer nicht existierenden Seite oder versucht der Nutzer durch Änderung einer ID in der URL auf eine Seite eines anderen Nutzers zuzugreifen, so wird ihm die Fehlerseite aus Abbildung 26 angezeigt.

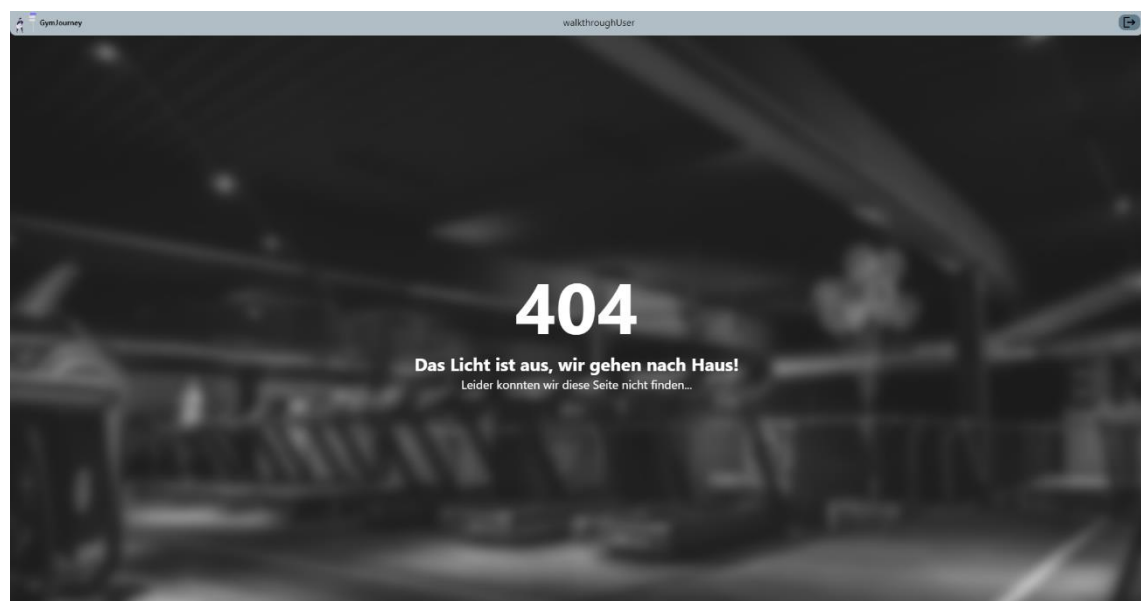


Abbildung 26: Fehlerseite

4 Fazit

Durch diese Studienarbeit konnten einige Erkenntnisse gewonnen werden, gerade was die Implementierung des Backends betrifft. So wurde dargestellt, wie die Authentifizierung des Nutzers mit JSON Web Tokens abwickelt werden kann, wie man den Nutzer daran hindert, auf fremde Daten zuzugreifen und wie man ein YouTube-Video in einer Website einbetten kann. Des Weiteren war es interessant zu sehen, wie man MySQL zur Speicherung von Daten für eine Webanwendung verwendet. Doch auch in der Frontend-Entwicklung konnten wertvolle Erkenntnisse gesammelt werden. So wurden hier interessante Bibliotheken eingesetzt, wie beispielsweise die „reactjs-popup“-Bibliothek, die für die PopUp-Fenster verwendet wurde [vgl. 17], oder die Bibliothek „react-lazy-load“ [vgl. 14], die für das Lazy Loading der eingebetteten YouTube-Videos aus Abschnitt 2.3 verwendet wurde.

5 Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Studienarbeit selbstständig und ohne fremde Hilfe verfasst und keine anderen Hilfsmittel als die angegebenen verwendet habe.

Vor allem versichere ich, dass ich alle wörtlichen und sinngemäßen Übernahmen aus anderen Werken als solche kenntlich gemacht habe.

Ort, Datum

95126 Schwarzenbach a. d. Saale, 06.06.2023

Unterschrift

A. Kunz

Literaturverzeichnis

- [1] Manuel Unger. *Mit Anforderungsanalyse zum erfolgreichen Software-Projekt*. URL: <https://www.adito.de/knowhow/blog/anforderungsanalyse> (besucht am 07.05.2023, veröffentlicht 2021).
- [2] Kinsta. *What Is React.js? A Look at the Popular JavaScript Library*. URL: <https://kinsta.com/knowledgebase/what-is-react-js/> (besucht am 26.04.2023, veröffentlicht 2023).
- [3] MDN Web Docs. *Express/Node introduction*. URL: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction (besucht am 26.04.2023, veröffentlicht 2023).
- [4] Kinsta. *Was ist MySQL? Eine anfängerfreundliche Erklärung*. URL: <https://kinsta.com/de/wissensdatenbank/was-ist-mysql/> (besucht am 26.04.2023, veröffentlicht 2022).
- [5] Iconfinder. *Js, react js, logo icon*. URL: https://www.iconfinder.com/icons/1174949/js_react_js_logo_react_react_native_icon (besucht am 26.04.2023, veröffentlicht 2023).
- [6] Wikimedia Commons. *File:Expressjs.png*. URL: <https://commons.wikimedia.org/wiki/File:Expressjs.png> (besucht am 26.04.2023, veröffentlicht 2017).
- [7] Creazilla, Gil Barbara. *Mysql icon*. URL: <https://creazilla.com/nodes/3254121-mysql-icon> (besucht am 26.04.2023, veröffentlicht 2023).
- [8] Webdesign Journal. *Informationsarchitektur: So strukturierst du die Inhalte deiner Website*. URL: <https://www.webdesign-journal.de/informationsarchitektur/> (besucht am 07.05.2023, veröffentlicht 2023).
- [9] IBM. *Was ist ein Datenbankschema?* URL: <https://www.ibm.com/de-de/topics/database-schema> (besucht am 07.05.2023, veröffentlicht 2023).
- [10] MDN Web Docs. *HTTP response status codes*. URL: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Status> (besucht am 08.05.2023, veröffentlicht 2023).
- [11] auth0. *Node-jsonwebtoken*. URL: <https://github.com/auth0/node-jsonwebtoken/blob/master/README.md> (besucht am 10.05.2023, veröffentlicht 2022).

- [12] SQL-Server-Team. *How and Why to Use Parameterized Queries?* URL: <https://techcommunity.microsoft.com/t5/sql-server-blog/how-and-why-to-use-parameterized-queries/ba-p/383483#> (besucht am 14.05.2023, veröffentlicht 2019).
- [13] MDN Web Docs. *String.prototype.match()*. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/match?retiredLocale=de (besucht am 14.05.2023, veröffentlicht 2023).
- [14] loktar & sergeylaptev. *react-lazy-load*. URL: <https://www.npmjs.com/package/react-lazy-load> (besucht am 15.05.2023, veröffentlicht 2022).
- [15] Ionos. *Lazy Loading: So optimieren Sie die Performance Ihrer Website*. URL: <https://www.ionos.de/digitalguide/websites/webseiten-erstellen/lazy-loading/> (besucht am 15.05.2023, veröffentlicht 2022).
- [16] MDN Web Docs. *<iframe>: The Inline Frame element*. URL: <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/iframe?retiredLocale=de> (besucht am 15.05.2023, veröffentlicht 2023).
- [17] Youssouf EL Azizi. *reactjs-popup*. URL: <https://www.npmjs.com/package/reactjs-popup> (besucht am 16.05.2023, veröffentlicht 2021).