

Burren Noah

Helsana Versicherung AG

26. Juni 2024

React.js

GymPlaner

Inhalt

[Dokumentbeschreibung 4](#_Toc170292895)

[Informationen 5](#_Toc170292896)

[Vorgaben 5](#_Toc170292897)

[Ideensammlung 5](#_Toc170292898)

[Risiken 5](#_Toc170292899)

[Schlussfolgerung 5](#_Toc170292900)

[Änderungstabelle 6](#_Toc170292901)

[Zeitplan 8](#_Toc170292902)

[Tagesreflexionen 9](#_Toc170292903)

[Woche 1 9](#_Toc170292904)

[Montag, 3. Juni 9](#_Toc170292905)

[Dienstag, 4. Juni 9](#_Toc170292906)

[Mittwoch, 5. Juni 9](#_Toc170292907)

[Woche 2 10](#_Toc170292908)

[Montag, 10. Juni 10](#_Toc170292909)

[Dienstag, 11. Juni 10](#_Toc170292910)

[Mittwoch, 12. Juni 10](#_Toc170292911)

[Woche 3 11](#_Toc170292912)

[Montag, 17. Juni 11](#_Toc170292913)

[Dienstag, 18. Juni 11](#_Toc170292914)

[Mittwoch, 19. Juni 11](#_Toc170292915)

[Woche 4 12](#_Toc170292916)

[Montag, 24. Juni 12](#_Toc170292917)

[Dienstag, 25. Juni 12](#_Toc170292918)

[Mittwoch, 26. Juni 12](#_Toc170292919)

[Meilenstein 1 Login und Registration 13](#_Toc170292920)

[1.1 User Datenbank erstellen 13](#_Toc170292921)

[1.2 Login Seite 14](#_Toc170292922)

[1.3 Signup Seite 15](#_Toc170292923)

[1.4 JavaScript Signup 16](#_Toc170292924)

[1.5 JavaScript Login 17](#_Toc170292925)

[1.6 Sicherheit 18](#_Toc170292926)

[1.7 Validierung 19](#_Toc170292927)

[1.8 Passwort hashing 20](#_Toc170292928)

[Was ist Bycript 20](#_Toc170292929)

[Meilenstein 2 Gym-Übungsdatenbank und Benutzeroberfläche 21](#_Toc170292930)

[2.1 Erstellung der Übungsdatenbank 21](#_Toc170292931)

[2.2 Entwicklung der Benutzeroberfläche zur Übungsauswahl 22](#_Toc170292932)

[2.3 Speichern des Gym-Plans in der Datenbank 23](#_Toc170292933)

[2.4 Speicherung und Abruf des Gymplans 24](#_Toc170292934)

[Meilenstein 3 Wochenkalender und Planerstellung 26](#_Toc170292935)

[3.1 Entwicklung der Kalenderansicht 26](#_Toc170292936)

[3.2 Einfügen der Trainingspläne in den Kalender 27](#_Toc170292937)

[3.3 Speicherung der Kalenderdaten 27](#_Toc170292938)

[3 28](#_Toc170292939)

[.4 Abruf und Anzeige der Trainingspläne 29](#_Toc170292940)

[Meilenstein 4 Administratives 30](#_Toc170292941)

[4.1 Dokumentation fertigstellen 30](#_Toc170292942)

[4.2 Präsentation 30](#_Toc170292943)

[4.3 React.js 30](#_Toc170292944)

[4.4 Homeoffice Tag 30](#_Toc170292945)

[Persönliches Fazit 31](#_Toc170292946)

[Erfahrungen und Erkenntnisse: 31](#_Toc170292947)

[Technische Herausforderungen und Lösungen: 31](#_Toc170292948)

[Dokumentation und Präsentation: 31](#_Toc170292949)

[Zeitmanagement: 31](#_Toc170292950)

[Lernprozess: 31](#_Toc170292951)

[Quellenverzeichnis 32](#_Toc170292952)

[React Tutorials: 32](#_Toc170292953)

[Full-Stack Authentication mit React, Node, MySQL: 32](#_Toc170292954)

[Passwort Hashing mit React und Node.js: 32](#_Toc170292955)

[Andere Links 32](#_Toc170292956)

[Glossar 33](#_Toc170292957)

[Abbildungsverzeichnis 35](#_Toc170292958)

[Ausgefüllte Checkliste 36](#_Toc170292959)

# Dokumentbeschreibung

Diese Anwendung, die ich entwickelt habe, bietet Benutzern eine intuitive Plattform zur Verwaltung ihrer Übungspläne und deren Anzeige in einem Wochenkalender. Sie basiert auf React für die Frontend-Oberfläche und nutzt Axios für HTTP-Anfragen an unseren Backend-Server, der nahtlos mit einer SQL-Datenbank integriert ist.

**Login und Registrierung:**

Neue Benutzer können sich einfach registrieren, indem sie ihren Namen, ihre E-Mail und ein Passwort eingeben. Die Registrierung führt Validierungen durch, um die Eingaben zu überprüfen, und speichert die Daten sicher in unserer SQL-Datenbank. Bestehende Benutzer können sich mit ihrer E-Mail und ihrem Passwort anmelden, wobei alle Eingaben gründlich überprüft werden, um Sicherheit und Datenintegrität zu gewährleisten.

**Wochenkalender:**

Der Wochenkalender zeigt die geplanten Übungen für jeden Wochentag an. Die Daten werden dynamisch vom Backend abgerufen und benutzerfreundlich dargestellt. Diese Funktion ermöglicht es mir, meine Trainingsroutine effektiv zu organisieren und den Überblick über meine Aktivitäten zu behalten.

**Übungsverwaltung:**

Die Verwaltung der Übungen erlaubt es mir, neue Übungen hinzuzufügen und diesen bestimmte Wochentage zuzuordnen. Nach der Eingabe werden die Informationen über HTTP-POST-Anfragen an unseren Server gesendet und sicher in unserer SQL-Datenbank gespeichert. Diese zentrale Verwaltung erleichtert es mir, meine Trainingspläne jederzeit zu aktualisieren und anzupassen.

**404-Fehlerseite:**

Für nicht gefundene Seiten haben wir eine spezielle 404-Fehlerseite implementiert, die es mir ermöglicht, einfach zur Login-Seite zurückzukehren, falls eine angeforderte Seite nicht verfügbar ist.

**Validierung und Sicherheit:**

Die Anwendung integriert umfassende Validierungslogik für Benutzereingaben, einschliesslich der Überprüfung von E-Mail-Formaten und Passwortstärke. Dadurch stellen wir sicher, dass die Daten korrekt und sicher gespeichert werden. Unsere SQL-Datenbank bietet eine robuste Lösung für die Speicherung von Benutzerkonten und Übungsplänen, was eine skalierbare und zuverlässige Plattform gewährleistet.

# Informationen

## Vorgaben

Während der Zeit des Abschlussprojekts vom 30. Mai bis 28. Juni (je nach Schule) kannst du ein selbst gewähltes Thema bearbeiten und als Projekt umsetzen. Das Projekt gliedert sich in verschiedene Phasen während 12 Tagen. Für einen reibungslosen Start solltest du dich vorgängig mit der Themenwahl auseinandersetzen.

Wähle ein Thema aus, das du schon immer einmal in Angriff nehmen wolltest.

## Ideensammlung

Wie beim Sportferienprojekt wollte ich von Anfang an etwas mit Sport machen. Am besten eine Website mit Login und Datenbanken. Mein Plan war es einen Manager zu machen, wo man seinen eigenen Gymplan erstellen kann. Man sollte Übungen aussuchen und einem Wochentag zuteilen können. Diese Übungen sollte man auch anschauen können. Dies wollte ich mit HTML, CSS, JavaScript und SQL machen.

Als ich das Projekt bestätigen wollte, kam Reto, mein Coach auf die Idee, dass ich mit React.js arbeiten soll. Ich ging darauf ein, ohne Vorwissen auf React. Somit kam ein neuer Issue zu stande, und zwar React.js zu lernen.

## Risiken

Die grösste Angst die ich hatte war, dass ich mich überschätze und noch lange nicht auf dem Niveau des Plans bin.

Das benutzen von React.js war auch ein Risiko, denn ich habe so noch nie gearbeitet.

## Schlussfolgerung

Dieses Projekt wurde daraufhin von Reto Froelicher bestätigt und kann auf Github gefunden werden:

# Änderungstabelle

|  |  |
| --- | --- |
| Datum | Aktion |
| 3. Juni 2024 | Umbenennung des Projekts, Erster Commit |
| 4. Juni 2024 | Anpassung der Ordnerstruktur, Starten des Servers, Verbindung zur SQL-Datenbank in server.js |
|  | Validierung des Namens, Implementierung der Funktion zur Handhabung der Eingabe bei der Registrierung |
|  | Validierung des Namens während der Registrierung, Anzeige einer Fehlermeldung bei ungültigen Eingaben |
|  | Validierung von E-Mail und Passwort, Handhabung der Eingabe, Implementierung der Validierungsdatei und const values useState |
|  | Hinzufügen eines h2-Titels, Link zur Anmeldeseite, Import der Anmeldeseite |
|  | App-Routen für die Anmeldung, Zentrieren des E-Mail- und Passwortfelds |
| 5. Juni 2024 | Funktionierende Implementierung der Login- und Registrierungsfunktion mit SQL |
|  | Funktionsfähiges Login, Fehlerbehebung beim Anzeigen der Meldung "Kein Account gefunden" beim Login |
|  | Funktionierende Implementierung der Registrierungsfunktion, Wiederholung der Arbeit an allen Elementen |
|  | Implementierung von Login.js, Neustart aufgrund von Fehlern am Ende |
| 12. Juni 2024 | Strukturierung der Ordner und Arbeit am SQL-Teil des Gym-Projekts |
|  | Versuch und Implementierung geschützter Routen |
| 17. Juni 2024 | Auswahl der Muskelgruppe auf der Startseite, Anzeigen von Übungen, Hinzufügen des Übungspfads |
|  | Verbindung zur SQL-Übungsdatenbank |
| 18. Juni 2024 | Speichern der Benutzer-ID, Hinzufügen einer Zurück-zu-Startseite-Schaltfläche |
|  | Entfernen eines Trainings aus der Datenbank, Auswahl von Übungen und Hinzufügen zur Datenbank |
| 19. Juni 2024 | Korrekturen von Schreibfehlern, Hinzufügen von Kommentaren und Fehlermeldungen |
|  | Implementierung der Passwortwiederholung, Wiederherstellung der Registrierungsfunktion |
|  | Verbesserung der responsiven Darstellung, Aktualisierung von app.js und Import Links |
|  | Behandlung von Sonderzeichen, Änderung von Textelementen, Implementierung des NotFound-Komponenten |
|  | Übersetzung von Texten, Löschung von Kalenderdaten, Arbeit an der Datenbank |
|  | Hinzufügen der wöchentlichen Kalenderfunktion und Dateiverwaltung |
|  | Anpassung von CSS-Abständen, Implementierung der sichtbaren Anzeige von Übungen erst nach Klick |
|  | Anpassung des CSS wie im Login-Bereich, Implementierung von geschützten Routen |
|  | Implementierung der Logout-Funktion, Anzeige von Elementen, Änderung der Darstellung von Buttons statt Aufzählungen |
| 24. Juni 2024 | Implementierung von bycript |
|  | Versuch Passwort zu hashen |
|  | Dokumentation schreiben |
| 25. Juni 2024 | Dokumentation schreiben |
| 26. Juni 2024 | Dokumentation schreiben |
|  | Präsentation fertigstellen |
|  | Abgabe Projekt |

# Zeitplan

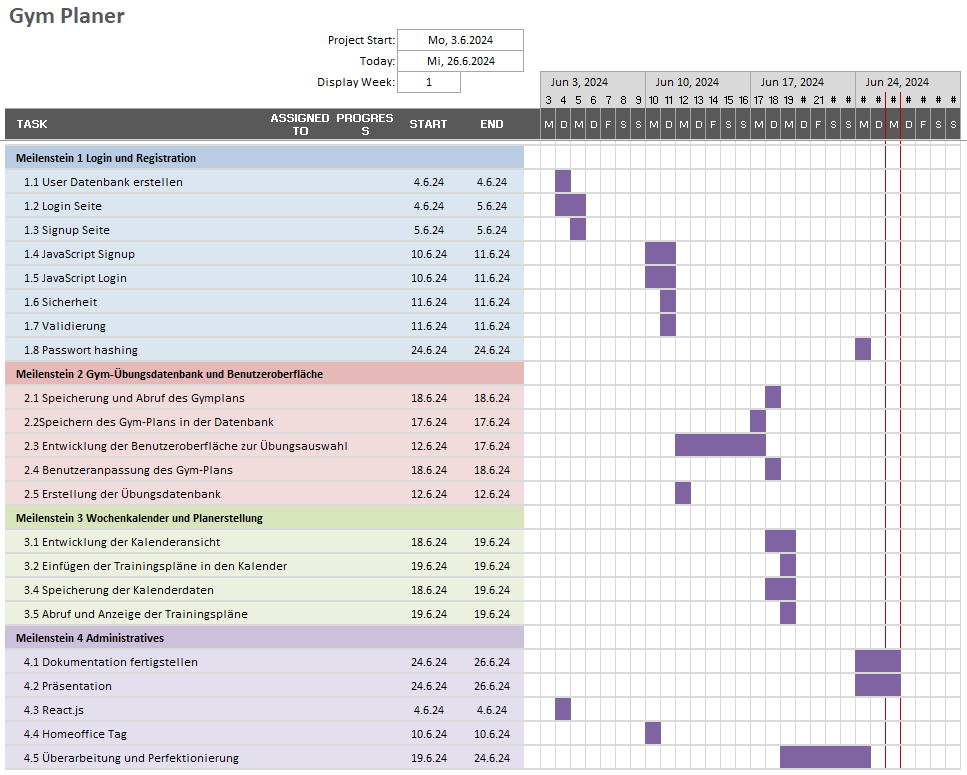


Abbildung 1 Zeitplan

# Tagesreflexionen

*Was ist mir gut gelungen? Wo hatte ich Probleme? Was habe ich heute gelernt? Was würde ich anders machen? Was würde ich wieder so machen?*

## Woche 1

### Montag, 3. Juni

Gelungen ist mir die Ideenfindung und die Planung in Github Projects. Ich hatte eine inhaltsreiche und durchführbare Projektidee, welche auch bestätigt wurde. Es fing schon sehr gut an, denn ich hatte bei der Suche keine Probleme. Ich habe gelernt wie man mit Github Projekte erstellt und planen kann. Nächstes mal würde ich mir mehr Zeit für die Planung geben. Github Projects ist sehr gut, deshalb werde ich es weiterhin benutzen.

### Dienstag, 4. Juni

Das erstellen der User Datenbank in mySQL verlief ohne Probleme. Das Login und dir Registrationsseite konnte ich dank einem YouTube Video meistern Dieses Video ist im Quellenverzeichnis verlinkt. Probleme hatte ich ganz viele im Login und der Registration. Lange funktionierte nichts und es gab unzähliche Errors. Auch wenn es noch nicht funktioniert, lernte ich sehr vieles über das Programmieren eines Logins mit React.js. Anders würde ich definitiv das schauen eines Tutorials, damit nicht immer unnötige Flüchtigkeitsfehler passieren. Ich kann dieses Video nur weiterempfehlen.

### Mittwoch, 5. Juni

Das Registrieren in die Datenbank und das Autentifizieren im Login hat nach vielen Rückschlägen endlich funktioniert. Probleme waren, dass trotz korrekten Anmeldedaten kein Account gefunden wurde. Das Problem war ein falscher Funktionenname. Ich lernte, wie man effizient nach Fehlern suchen kann, ohne dass ein spezifischer Error angezeigt wird. Nächstes mal werde ich genauer schreiben, dass keine Schreibfehler passieren. Ich komme mit React.js schon gut klar und habe spass daran es weiterhun zu nutzen.

## Woche 2

### Montag, 10. Juni

Ich konnte das Login und die home.js seite während des Home Office Tags Überarbeiten. Probleme gab es bei dem Versuch, Protected Routes einzufügen, damit man nicht über /home ohne Login reinkommt. Hat schlussendlich funktioniert, leider kam ich aber gar nicht mehr rein. Trotzdem lernte ich vieles über Seitenverbindungen in App.js und wie Protected Routes funktionieren. Nächstes mal würde ich von anfang an die User ID nach Login speichern, damit ich das Problem mit den Protected Routes nicht habe. Auch wenn ich das Problem noch nicht lösen konnte, bin ich strolz auf mich, dass ich den Felher gefunden habe.

### Dienstag, 11. Juni

Heute habe ich mir vorgenommen, an der Dokumentation zu arbeiten. Ich kam sehr gut voran. Es war gar nicht mein Tag, denn ich hatte mühe mit der Konzentration, am schreiben dran zu bleiben. Erstaunlicherweise lernte ich etwas über Word. Es gibt so viele ansichten, von denen ich nichts wusste.

### Mittwoch, 12. Juni

Das erstellen der SQL Übungsdatenbank ging ganz gut. Ich denke mit JSON wäre es einfacher gewesen, denn für so viele Übungen jedes mal einen INSERT command zu schreiben, kann mühsam werden. Da ich diese Datenbank noch anders brauchen werde, nahm ich mySQL. Probleme hatte ich keine. In Zukunft werde ich meine SLQ Querys speichern, man weiss nie wann man sie brauchen kann. Heute kam ich sehr gut voran, ich kann mir vorstellen dass ich mit gleichem Tempo gut voran kommen werde.

## Woche 3

### Montag, 17. Juni

Sehr stolz bin ich darauf, dass ich es geschafft habe, die Übungen von nur der ausgesuchten Kategorie anzuzeigen lassen. Probleme hatte ich beim verbinden der richtigen SQL Tabelle, was ich aber schnell lösen konnte. Ich lernte sehr vieles über Backendverbindungen in server.js. Weiterhin möchte ich Backend Programme speichern, da es mich einfach fasziniert. Ich hätte bei den Errors die ich hatte Screenshots machen sollen, damit ich diese besser Dokumentieren kann. Ich war jedoch so fokussiert, das ich scheinbar wichtiges vergessen habe.

### Dienstag, 18. Juni

Gelungen ist mir das speichern der User id, was mich sehr erleichtert hat, da ich morgen die Protected Route fertigstellen kann. Zusätzlich kann die Übungen jetzt per Häckchen auswählen und speichern lassen. Diese werden zusammen mit der Kategorie und der User id als Übungstag gespeichert. Diese kann man natürlich auch wieder entfernen und mit einem Button kommt man zurück zu /home. Probleme hatte ich schon wieder im Backend, dass es die User id nicht gefunden hat. Konnte ich aber auch lösen. Das speichern des angemeldeten Benutzers ist sehr wichtig und werde ich bei weiteren Programmen umsetzen.

### Mittwoch, 19. Juni

Heute konnte ich erstaunlich viele dinge erledigen. Das wichtigste war das Implementieren eines Kalenders, was mein 3 Meilenstein beendet. Der Benutzer kann eine Kategorie aussuchen und einem Wochentag zuteilen. Dank des speicherns der User id, funktioniert jetzt auch die Protected Route. Da ich dies geschafft habe, konnte ich zusätzlich eine 404 Page Not Found Seite erstellen. Da ich noch Zeit hatte, machte ich mithilfe von CSS meine Website Responsive und farblich einheitlich. Probleme hatte ich beim Kalender, da es die Kategoie nie angezeigt hat. Im Prinzip bin ich mit dem Projekt fertig, ich muss nur noch die feinarbeiten erledigen.

## Woche 4

### Montag, 24. Juni

Ich habe mein Projekt Reto gezeigt, und er meinte, ich soll unbedingt die Passwörter verschlüsseln. Also Implementierte ich bycript und versuchte meine Passwörter zu hashen. Dies hat jedoch nicht ganz so funktioniert wie ich möchte. Ich konnte sie hashen, aber nicht mehr entschlüsseln. Wegen Zeitdruck lasse ich diesen Schritt aus und fokussiere mich auf die Dokumentation.

### Dienstag, 25. Juni

### Mittwoch, 26. Juni

# Meilenstein 1 Login und Registration

## User Datenbank erstellen

Mit dem Programm XAMPP Control Panel habe ich den SQL server gestartet, womit ich dann auf <http://localhost/phpmyadmin> zugreifen konnte.

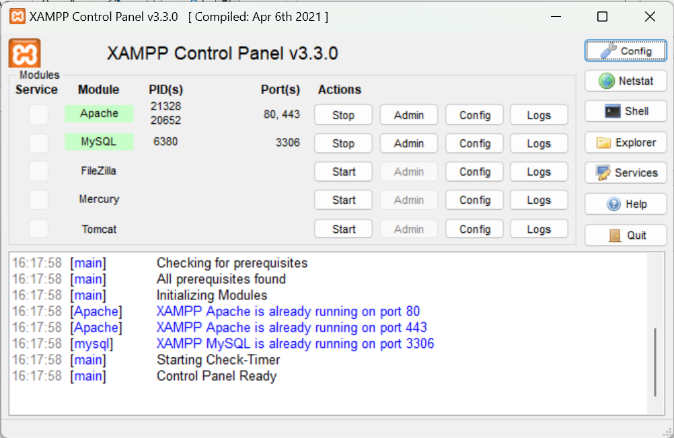


Abbildung 2 XAMPP

Mit phpmyadmin kann man auf einer benutzerfreundlichen Oberfläche MySQL und MariaDB Datenbanken verwalten. Ich habe ohne gross einen Query schreiben zu müssen, manuell die Datenbank und die dazugehörigen Tables erstellt.

Der Table login hat folgende 4 Spalten:

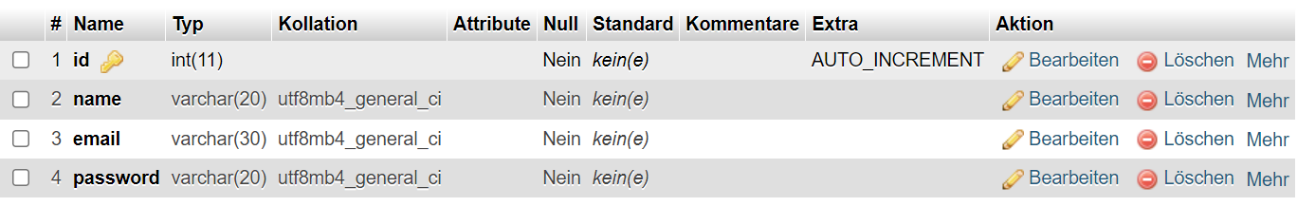


Abbildung 3 phpmyadmin Login Table

Die Spalte id ist ein integer, welche nicht null sein soll und sich automatisch erhöht. Die Spalte name, email und passwort sind je varchars die auch immer einen Datensatz brauchen.

## Login Seite

Ich habe Für Die Login Seite React.js benutzt. Zu React steht mehr im Abschnitt 4.3. Der CSS script ist als className direkt in den <div> eingebunden.

<div className="bg-white p-3 rounded w-25">

Mithilfe eines returns wird das HTML angezeigt. Die Login Seite ist die Standard Seite und wird bei / verlinkt.

Ohne Authentifikation über diese Seite kommt man nicht weiter, denn abgesehen noch von der Registration ist alles über Protected Route geschützt.

Ein Bild, das Text, Screenshot, Software, Multimedia-Software enthält.

Automatisch generierte Beschreibung

Abbildung 4 Sign in Seite

## Signup Seite

Das Prinzip dieser HTML Seite ist genau das Gleiche wie vom Login. Wie man in der Abbildung sieht, gibt es nur minimale unterschiede.

Ein Bild, das Text, Screenshot, Software, Computersymbol enthält.

Automatisch generierte Beschreibung

Abbildung 5 Sign up Seite

Ich habe eine Funktion eingebunden, welche Vorgaben für unter anderem dem Passwort setzt. Zum Beispiel dass das Passwort mindestens 8 Zeichen enthalten muss. Ansonsten wird dies mit einer Errormeldung angezeigt. So eine Ausgabe sieht im Code folgendermassen aus:

{errors.email && (

<span className="text-danger"> {errors.email}</span>

)}

Genauer wird im Punkt 1.7 Validierung beschrieben.

Ein Bild, das Text, Screenshot, Schrift, Reihe enthält.

Automatisch generierte Beschreibung

Abbildung 6 Error mind 1 Zahl

So würde das Aussehen, wenn das Passwort keine Zahl enthalten würde.

## JavaScript Signup

1. Initialisieren des Zustands:

function Signup() {

  const [values, setValues] = useState({

    name: "",

    email: "",

    password: "",

    confirmPassword: "",

  });

Der useState-Hook wird verwendet, um ein Zustandsobjekt namens values zu erstellen. Dieses Objekt enthält die initialen Werte für die Formularfelder: name, email, password und confirmPassword, die alle als leere Strings initialisiert werden.

1. Initialisieren der Fehlerzustände:

const [errors, setErrors] = useState({});

Ein weiterer useState-Hook wird verwendet, um ein leeres Objekt namens errors zu initialisieren, das zur Speicherung von Validierungsfehlern dient.

1. Navigation:

const navigate = useNavigate();

Der useNavigate-Hook wird initialisiert, um später im Code eine Navigation zu einer anderen Seite der Anwendung zu ermöglichen.

1. Verarbeitung der Eingaben:

 const handleInput = (event) => {

    setValues((prev) => ({

      ...prev,

      [event.target.name]: event.target.value,

    }));

  };

Die handleInput-Funktion wird definiert, um die Änderungen der Eingabefelder im Formular zu verarbeiten. Diese Funktion wird bei jeder Eingabeänderung aufgerufen und aktualisiert den entsprechenden Wert im values-Objekt. Sie nutzt das prev-Objekt, um den vorherigen Zustand zu übernehmen und nur das Feld zu aktualisieren, das geändert wurde, basierend auf dem Namen des Eingabefeldes (event.target.name) und dessen neuen Wert (event.target.value).

## JavaScript Login

1. Initialisieren des Zustands für Formularwerte:

function Login() {

  const [values, setValues] = useState({

    email: "",

    password: "",

  });

Der useState-Hook wird verwendet, um ein Zustandsobjekt namens values zu erstellen. Dieses Objekt enthält die initialen Werte für die Formularfelder email und password, die beide als leere Strings initialisiert werden.

1. Navigation:

  const navigate = useNavigate();

Der useNavigate-Hook wird initialisiert, um später im Code eine Navigation zu einer anderen Seite der Anwendung zu ermöglichen.

1. Initialisieren der Fehlerzustände:

  const [errors, setErrors] = useState({});

Ein weiterer useState-Hook wird verwendet, um ein leeres Objekt namens errors zu initialisieren, das zur Speicherung von Validierungsfehlern dient.

1. Verwendung des Authentifizierungskontexts:

  const { login } = useContext(AuthContext);

Der useContext-Hook wird verwendet, um den AuthContext abzurufen, der die Authentifizierungslogik der Anwendung bereitstellt. Die login-Funktion wird aus dem Kontext extrahiert und kann später verwendet werden, um den Anmeldevorgang zu initiieren.

1. Verarbeitung der Eingaben:

const handleInput = (event) => {

    setValues((prev) => ({

      ...prev,

      [event.target.name]: event.target.value,

    }));

  };

Die handleInput-Funktion wird definiert, um die Änderungen der Eingabefelder im Formular zu verarbeiten. Diese Funktion wird bei jeder Eingabeänderung aufgerufen und aktualisiert den entsprechenden Wert im values-Objekt. Sie nutzt das prev-Objekt, um den vorherigen Zustand zu übernehmen und nur das Feld zu aktualisieren, das geändert wurde, basierend auf dem Namen des Eingabefeldes (event.target.name) und dessen neuen Wert (event.target.value).

## Sicherheit

const ProtectedRoutes = () => {

  const { userId } = useContext(AuthContext); // AuthContext verwenden, um userId zu erhalten

  // Überprüfen, ob userId vorhanden ist

  const isAuthenticated = !!userId; // Wandelt userId in ein Boolean um (true, wenn userId vorhanden ist, sonst false)

  return isAuthenticated ? <Outlet /> : <Navigate to="/" />;

};

Damit man nicht über den URL direkt in die Website reinkommt, ohne sich angemeltet zu haben, implementierte ich die ProtectedRoutes Funktion. Diese überprüft, ub eine userId vorhanden ist. In einem anderen Script wird dies bei der anmeldung im Local Storage gespeichert.

// Funktion zum Einloggen eines Benutzers

  const login = (userData) => {

    setUser(userData); // Setzt den Benutzer im Zustand

    localStorage.setItem("user", JSON.stringify(userData)); // Speichert den Benutzer im localStorage

  };

Somit wird wenn man versucht, auf /home zu zugreifen, zurück zum Login, beziehungsweise zu / verlinkt wird.

In App.js habe ich weitere Routes hinzugefügt, und dann die Seiten, welche geschützt werden sollen, von der ProtectedRoute umklammert.

return (

    <AuthProvider>

      <BrowserRouter>

        <Routes>

          <Route path="/" element={<Login />} />

          <Route path="/signup" element={<Signup />} />

          <Route element={<ProtectedRoutes />}>

            <Route path="/home" element={<Home />} />

Wenn man einen URL eingibt, der nicht existiert, kommt man auf eine Not Found Seite. Von dieser kann man per Button klick zurück zum Login gehen.

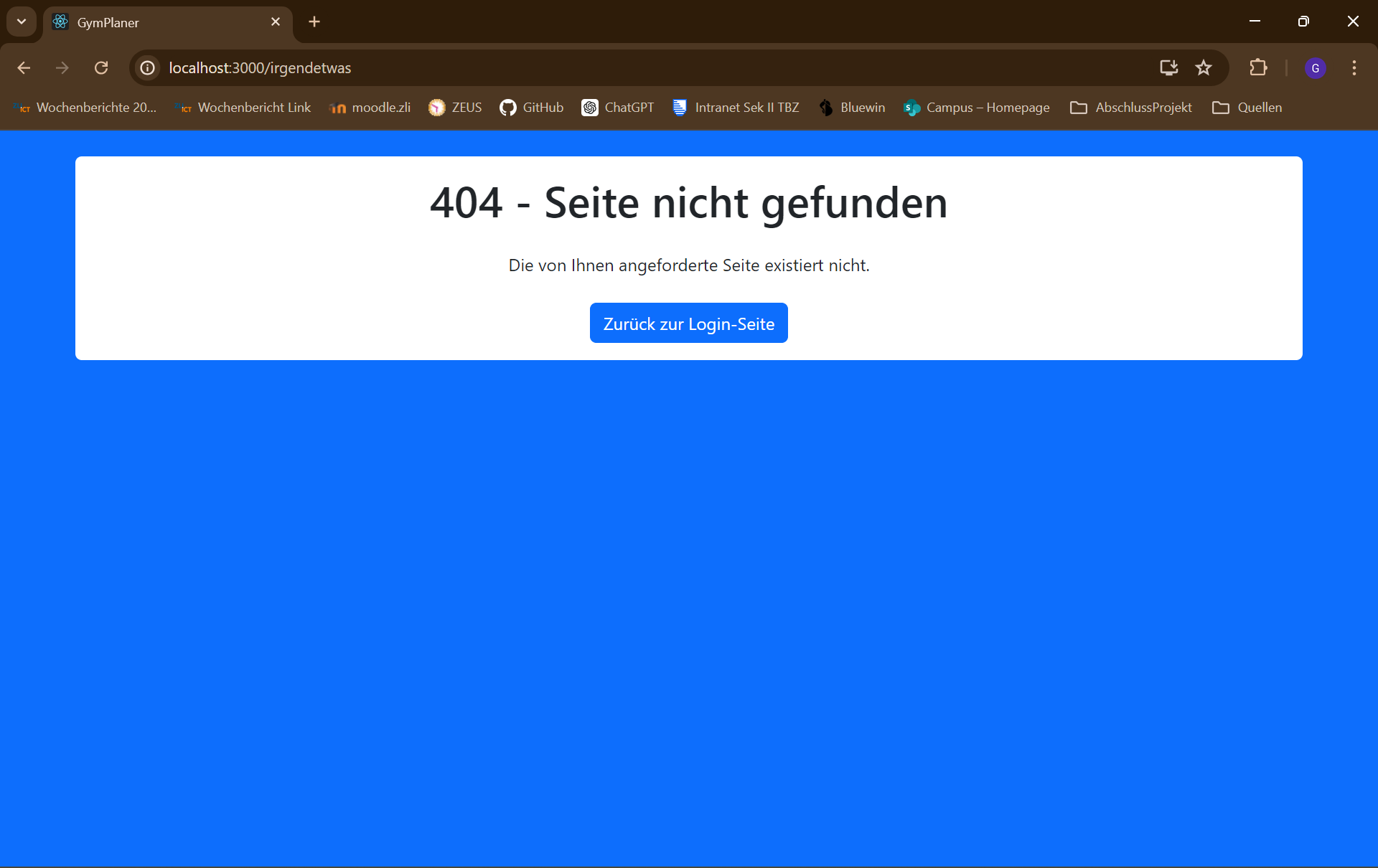


Abbildung 7 404 Not Found

</Route>

          <Route path="\*" element={<NotFound />} />

        </Routes>

Dies habe ich mithilfe des \* path gelöst. Der stern bedeutet so viel wie alles andere.

## Validierung

Für das Login und das Signup habe ich je eine JavaScript Datei erstellt, welche Vorgaben zu den Benutzereingaben definiert.

// Definiert Regular Expressions für E-Mail und Passwort

  const email\_pattern = /^[^\s@]+@[^\s@]+\.[^\s]+$/;

  const password\_pattern = /^(?=.\*\d)(?=.\*[a-z])(?=.\*[A-Z])[a-zA-Z0-9]{8,}$/;

Dieser komplexe Zeichenfolge definiert beim Passwort, dass es

* …mindestens acht Zeichen enthalten muss.
* …mindestens einen Gross- und Kleinbuchstaben enthalten muss
* …mindestens eine Zahl enthalten muss

Mithilfe von Regular Expressions, welche ich unter anderen in PHP kennengelernt habe, ging mir das recht einfach.

Falls diese Vorgaben nicht erfüllt sind, wird per Errormeldung unter dem Eingabefeld angezeigt, was das Problem ist. Jedoch nur beim Signup. Im Login kommt man entweder rein oder auch nicht.

// Validierung des Passworts

  if (values.password === "") {

    errors.password = "Passwort darf nicht leer sein";

  } else if (values.password.length < 8) {

    errors.password = "Passwort muss mindestens 8 Zeichen lang sein";

  } else if (!/\d/.test(values.password)) {

    errors.password = "Passwort muss mindestens eine Zahl enthalten";

  } else if (!/[A-Z]/.test(values.password)) {

    errors.password =

      "Passwort muss mindestens einen Grossbuchstaben enthalten";

  } else if (!password\_pattern.test(values.password)) {

    errors.password = "Ungültiges Passwort";

  }

## Passwort hashing

Ich habe mit bycript versucht, das Passwort zu hashen. Dies hat zum grössten Teil funktioniert. Ich konnte einen Account erstellen und dessen Passwort wurde in der Datenbank verschlüsselt angezeigt.

Jedoch habe ich es nicht geschafft, das Passwort beim Versuch sich einzuloggen, wieder zu entschlüsseln.

### Was ist Bycript

crypt ist ein Passwort-Hashing-Algorithmus, der entwickelt wurde, um Passwörter sicher zu speichern. Er bietet Schutz gegen Brute-Force-Angriffe und ist resistent gegen Hardware-Beschleunigungstechniken, die bei vielen modernen Hashing-Algorithmen Schwachstellen darstellen. Hier sind einige der wichtigsten Merkmale von bcrypt:

1. **Salting**: bcrypt fügt jedem Passwort einen zufälligen Salt hinzu, bevor es gehasht wird. Das bedeutet, dass selbst identische Passwörter unterschiedliche Hashes haben, was Rainbow-Table-Angriffe erschwert.
2. **Adaptivität**: bcrypt ist anpassbar, was bedeutet, dass die Anzahl der Iterationen (Kostenfaktor) erhöht werden kann, um die Zeit zu erhöhen, die zum Berechnen des Hashes benötigt wird. Dies macht bcrypt zukunftssicher, da es an die zunehmende Rechenleistung angepasst werden kann.
3. **Key Stretching**: bcrypt verwendet eine teure Schlüsselableitungsfunktion, die sicherstellt, dass der Hashing-Prozess auch bei einfachen Passwörtern lange dauert.
4. **Speicherbedarf**: bcrypt benötigt mehr Speicherplatz als einige andere Hashing-Algorithmen, was Angriffe weiter erschwert, da Angreifer mehr Speicherressourcen benötigen.

# Meilenstein 2 Gym-Übungsdatenbank und Benutzeroberfläche

## 2.1 Erstellung der Übungsdatenbank

Die Erstellung der Übungsdatenbank bestand aus mehreren Schritten:

Löschen der Datenbank, falls vorhanden:

DROP DATABASE IF EXISTS Uebungsdatenbank;

Erstellung der neuen Datenbank:

CREATE DATABASE Uebungsdatenbank;

Auswahl der neu erstellten Datenbank:

USE Uebungsdatenbank;

Erstellung der Tabelle für Übungen:

CREATE TABLE IF NOT EXISTS Uebungen (

    ID INTEGER PRIMARY KEY AUTO\_INCREMENT,

    NAME TEXT NOT NULL,

    Kategorie TEXT NOT NULL

);

Erstellung der Tabelle für die Zuordnung von Benutzern zu Übungen:

CREATE TABLE IF NOT EXISTS user\_exercises (

    id INTEGER PRIMARY KEY AUTO\_INCREMENT,

    user\_id INTEGER NOT NULL,

    category TEXT NOT NULL,

    exercise\_id INTEGER NOT NULL,

    FOREIGN KEY (exercise\_id) REFERENCES Uebungen(ID)

);

Einfügen von Übungen in die Übungstabelle:

INSERT INTO Uebungen (NAME, Kategorie) VALUES ('Leg Extensions', 'Beine');

INSERT INTO Uebungen (NAME, Kategorie) VALUES ('Hamstring Curls', 'Beine');

Diese Übungen sind nur nicht alle, denn es sind knapp 100 einträge.

## 2.2 Entwicklung der Benutzeroberfläche zur Übungsauswahl

Ein Bild, das Text, Screenshot, Schrift, Reihe enthält.

Automatisch generierte Beschreibung

Abbildung 8 Übungsauswahl

Hier kann man ganz einfach aussuchen, welche Übungen man will. Dies Funktioniert mithilfe einem input type checkbox.



Abbildung 9 Übungen bestätigen oder Löschen

Es gibt einen Button, wo man seine Übungen Speichern kann, einen um zurück zu /home zu gelangen und einen, um die Übungen aus der Datenbank zu löschen.

Das Style ist ähnlich wie beim Login und den restlichen Seiten.

        <h1 className="text-center mb-4">Übungen für {category}</h1>

Der Titel ändert sich in Zusammenhang zur ausgewählten Kategorie. Dies habe ich so gemacht, dass ich die Variable anzeigen lasse.

## 2.3 Speichern des Gym-Plans in der Datenbank

Dieser Code ermöglicht es, die Übungen eines Benutzers für eine bestimmte Kategorie zu aktualisieren. Zuerst werden alle bestehenden Einträge für den Benutzer in dieser Kategorie gelöscht. Danach werden die neuen Übungsdaten in die Datenbank eingefügt. Wenn irgendein Schritt fehlschlägt, wird ein entsprechender Fehler gemeldet und eine Fehlermeldung zurückgegeben.

app.post("/user-exercises", (req, res) => {

  const { userId, category, exercises } = req.body;

**Beschreibung:** Dieser Abschnitt definiert einen POST-Endpoint /user-exercises. Beim Aufruf dieses Endpoints erwartet der Server im Request-Body die Parameter userId, category und exercises. Diese Parameter werden verwendet, um die Benutzerübungen in der Datenbank zu aktualisieren.

  const deleteQuery =

    "DELETE FROM user\_exercises WHERE user\_id = ? AND category = ?";

  dbExercises.query(

    deleteQuery,

    [userId, category],

    (deleteErr, deleteResult) => {

      if (deleteErr) {

        console.error("Error deleting exercises:", deleteErr);

        return res.status(500).json("Internal server error");

      }

**Beschreibung:** Hier wird eine SQL-Abfrage definiert, die alle bisherigen Übungen eines Benutzers für eine bestimmte Kategorie löscht. Die Methode dbExercises.query führt diese Abfrage aus. Falls ein Fehler auftritt, wird dieser im Serverprotokoll ausgegeben und der Server antwortet mit einem Statuscode 500 und einer Fehlermeldung.

      const insertQuery =

        "INSERT INTO user\_exercises (user\_id, category, exercise\_id) VALUES ?";

      const values = exercises.map((exercise) => [

        userId,

        category,

        exercise.ID,

      ]);

**Beschreibung:** Wenn das Löschen erfolgreich war, wird eine neue SQL-Abfrage definiert, um die neuen Übungen des Benutzers in die Datenbank einzufügen. Die Variable values ist ein Array, das die neuen Übungsdaten in der Form [userId, category, exercise.ID] enthält, wobei jede Übung in eine entsprechende Zeile in der Tabelle user\_exercises eingefügt wird.

      dbExercises.query(insertQuery, [values], (insertErr, insertResult) => {

        if (insertErr) {

          console.error("Error saving exercises:", insertErr);

          return res.status(500).json("Internal server error");

        }

        return res.json("Exercises saved successfully!");

      });

    }

  );

});

**Beschreibung:** Diese Methode führt die Einfügeabfrage aus. Wenn ein Fehler auftritt, wird dieser im Serverprotokoll ausgegeben und der Server antwortet mit einem Statuscode 500 und einer Fehlermeldung. Wenn die Einfügeoperation erfolgreich ist, antwortet der Server mit einer Bestätigungsmeldung, dass die Übungen erfolgreich gespeichert wurden.

## 2.4 Speicherung und Abruf des Gymplans

Auf der Homepage werden die Unterschiedlichen Übungskategorien in <div> angezeigt.

Diese Balken haben drei Zustände, in denen sie angezeigt werden.

Der erste Zustand ist der, bei dem nichts angezeigt wird. Wenn man auf die Seite kommt, sieht jede Kategorie so aus.

Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte Beschreibung

Abbildung 10 Keine Übung ausgewählt

Sobald man auf den Grünen Button klickt, öffnet sich das Anzeigefeld. Wenn man noch keine Übungen ausgesucht hat, kommt man schon zum zweiten Zustand. Hier steht, dass man noch keine Übungen ausgesucht hat.

Ein Bild, das Text, Screenshot, Schrift, Reihe enthält.

Automatisch generierte Beschreibung

Abbildung 11 Keine Übungen vorhanden

Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte BeschreibungWenn man seine Übungen schon in der Übungsauswahl gespeichert hat, werden diese hier aufgerufen, und als Auflistung angezeigt. Es kann nur immer eine Kategorie geöffnet sein, der Rest bleibt oder kehr in den ersten Zustand zurück.

Abbildung 12 Übungen werden anzezeigt

# Meilenstein 3 Wochenkalender und Planerstellung

## 3.1 Entwicklung der Kalenderansicht

Ähnlich wie bei der Übungsansicht auf der Homepage hat diese Seite mehrere Balken. Um genau zu sein für jeden Wochentag einen. Anders ist nur, dass die Kategorie schon von Anfang an angezeigt wird.

Wenn noch keine Übung ausgewählt wurde, steht dass noch keine geplant sind.

Ein Bild, das Text, Screenshot, Software, Zahl enthält.

Automatisch generierte Beschreibung

Abbildung 13 Wochenkalender

Die Wochentage sind mit einem Array definiert und in der Variablen daysOfWeek gespeichert.

const daysOfWeek = [

    "Montag",

    "Dienstag",

    "Mittwoch",

    "Donnerstag",

    "Freitag",

    "Samstag",

    "Sonntag",

  ];

## 3.2 Einfügen der Trainingspläne in den Kalender

<select

     id="daySelect"

     className="form-select"

     value={selectedDay}

     onChange={(e) => setSelectedDay(e.target.value)}>

     <option value="">Tag auswählen</option>

     {daysOfWeek.map((day) => (

       <option key={day} value={day}>

         {day}

       </option>

     ))}

 </select>

Das <select>-Element in React ermöglicht es Benutzern, eine Option aus einer Dropdown-Liste auszuwählen. Im vorliegenden Code wird es verwendet, um Tage der Woche zur Auswahl anzubieten. Die Dropdown-Liste wird durch das id "daySelect" identifiziert und verwendet die Bootstrap-Klasse "form-select" für das Styling.

Der aktuell ausgewählte Tag wird durch die State-Variable selectedDay gesteuert, die über die value-Prop des <select>-Elements gebunden ist. Durch die onChange-Prop wird eine Funktion aufgerufen, die den Wert von selectedDay aktualisiert, basierend auf der ausgewählten Option. Die Optionen werden dynamisch durch das Mapping über das Array daysOfWeek generiert, wobei jedem Tag ein <option>-Tag mit einem eindeutigen Schlüssel und dem Tag selbst als Wert zugewiesen wird.

## 3.3 Speicherung der Kalenderdaten

Die Kalenderdaten werden in der MySQL-Datenbank in der Tabelle user\_calendar gespeichert, wie im folgenden SQL-Statement definiert:

CREATE TABLE IF NOT EXISTS user\_calendar (

    id INTEGER PRIMARY KEY AUTO\_INCREMENT,

    user\_id INTEGER NOT NULL,

    DAY VARCHAR(10) NOT NULL,

    category VARCHAR(50) NOT NULL,

    FOREIGN KEY (user\_id) REFERENCES signup.login(id)

);

Hier ist eine Erklärung der Struktur und des Speicherprozesses der Kalenderdaten:

1. **Tabelle user\_calendar**: Diese Tabelle speichert Einträge für Benutzerkalender. Jeder Eintrag hat eine eindeutige ID (id), die automatisch inkrementiert wird. Es gibt auch eine Spalte user\_id, die auf die id in der login Tabelle der signup Datenbank verweist, um die Beziehung zwischen Kalendereinträgen und Benutzern herzustellen.
2. **Spaltenbeschreibungen**:
   * id: Primärschlüssel, automatisch inkrementiert, um jeden Eintrag eindeutig zu identifizieren.
   * user\_id: Fremdschlüssel, verweist auf die id des Benutzers in der login Tabelle.
   * DAY: Speichert den Wochentag als VARCHAR(10). In der Anwendung wird der Tag als String (z.B. "Montag", "Dienstag") gespeichert.
   * category: Speichert die Kategorie der Übung, die für den jeweiligen Tag geplant ist, als VARCHAR(50).
3. **Daten speichern**:
   * Beim Speichern eines neuen Kalendereintrags (POST /user-calendar) wird zuerst geprüft, ob bereits ein Eintrag für den angegebenen Tag (day) und Benutzer (user\_id) existiert. Falls ein Eintrag vorhanden ist, wird dieser gelöscht, um Aktualisierungen zu ermöglichen.
   * Anschliessend wird ein neuer Eintrag in die Tabelle user\_calendar eingefügt, der die user\_id, den day und die category enthält.
4. **Daten abrufen**:
   * Zum Abrufen der Kalenderdaten eines Benutzers (GET /user-calendar) wird die user\_id als Parameter verwendet. Die Anfrage ruft alle Einträge aus der user\_calendar Tabelle ab, die mit der angegebenen user\_id übereinstimmen.
   * Die abgerufenen Daten enthalten die day und category für jeden Kalendereintrag des Benutzers.

In der Anwendung wird die user\_calendar Tabelle verwendet, um die geplanten Übungen für jeden Wochentag eines Benutzers zu speichern und abzurufen, um sie dann im WeeklyCalendar-Component darzustellen.

## 3

## .4 Abruf und Anzeige der Trainingspläne

Die Komponente interagiert mit einem Backend-Server über HTTP-Anfragen, um die Kalenderdaten eines Benutzers abzurufen und diese dann dynamisch darzustellen. Hier ist ein detaillierter Blick darauf, wie sie funktioniert:

**Datenabruf und Verarbeitung**

1. **Zustände und Hooks**:
   * **userId**: Wird über das AuthContext geladen und identifiziert den aktuellen Benutzer.
   * **weeklyCategories**: Ein Zustand, der die Daten für die wöchentlichen Kategorien speichert, die vom Backend abgerufen werden.
2. **Verwendung von useEffect**:
   * Beim ersten Laden der Komponente und bei Änderungen der userId wird eine Funktion fetchWeeklyCategories aufgerufen.
   * Diese Funktion sendet eine GET-Anfrage an den Backend-Server (http://localhost:8081/user-calendar), um die Kalenderdaten für den aktuellen Benutzer abzurufen.
   * Die erhaltenen Daten werden in einem Objekt gespeichert, das die Tage der Woche als Schlüssel und die entsprechenden Kategorien (oder "Keine Übungen geplant") als Werte enthält.

**Darstellung der Kalenderansicht**

Die gerenderte Ansicht der WeeklyCalendar-Komponente ist darauf ausgelegt, dem Benutzer eine klare Übersicht über seine wöchentlichen Übungen zu bieten:

* **Wochentage und Übungsplan**:
  + Die Tage der Woche (Montag bis Sonntag) sind in Karten (Cards) organisiert, um visuell zu gruppieren.
  + Jede Karte zeigt den Wochentag als Überschrift (<h3>), was dem Benutzer hilft, sich zu orientieren.
  + Unterhalb der Überschrift wird dynamisch entweder die geplante Übung oder die Meldung "Keine Übungen geplant" angezeigt, je nachdem, was im weeklyCategories-Zustand für diesen Tag gespeichert ist.

# Meilenstein 4 Administratives

## 4.1 Dokumentation fertigstellen

Meine Dokumentation muss geschrieben und Vollständig abgegeben werden.

## 4.2 Präsentation

Mit Powerpoint muss eine Präsentation von guten Zehn Minuten erstellt werden. Diese Präsentation wird den Coaches und meinem Lehrbetrieb vorgestellt.

## 4.3 React.js

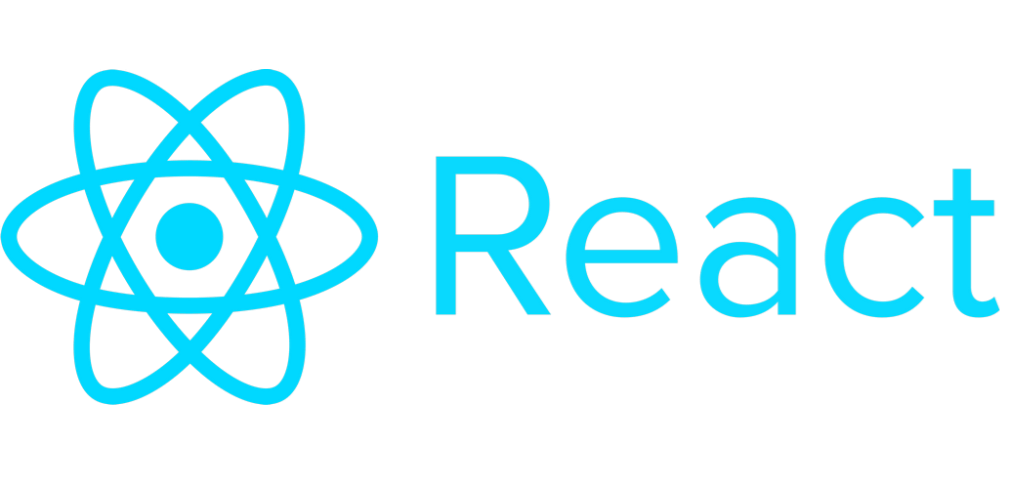


Abbildung 14 React.js

React.js ist eine JavaScript-Bibliothek, die für die Entwicklung von Benutzeroberflächen (UI) verwendet wird. Sie wurde von Facebook entwickelt und wird auch von einer grossen Community unterstützt. Hier sind einige wichtige Merkmale von React.js:

1. **Komponentenbasiert**: React ermöglicht die Erstellung wiederverwendbarer UI-Komponenten. Jede Komponente kann ihren eigenen Zustand verwalten und wird in einer hierarchischen Struktur angeordnet, um komplexe UIs zu erstellen.
2. **Virtual DOM**: React verwendet einen virtuellen DOM (Document Object Model), der Änderungen effizient verarbeitet und nur die notwendigen Teile der Benutzeroberfläche aktualisiert, anstatt das gesamte DOM neu zu rendern. Dies führt oft zu einer besseren Leistung.
3. **Einweg-Datenfluss**: Daten fliessen in React normalerweise in eine Richtung, von der übergeordneten Komponente zu untergeordneten Komponenten. Dies fördert eine klare Datenflussrichtung und erleichtert das Debugging und das Verständnis des Codes.
4. **JSX**: JSX ist eine JavaScript-Syntaxerweiterung, die es ermöglicht, HTML in JavaScript zu schreiben. React-Komponenten können in JSX geschrieben werden, was die Erstellung und Verwaltung von UI-Komponenten erleichtert.
5. **Grosse Community und Ökosystem**: React hat eine aktive Community und ein umfangreiches Ökosystem von Drittanbieter-Bibliotheken und Tools, die die Entwicklung erleichtern.

## Homeoffice Tag

Ich konnte am 10. Juni meinen Homeoffice Tag einlösen. Ich hatte fünf Checkpoints, wo ich Reto Froelicher eine Nachricht auf Slack schreiben musste. Unteranderem wann ich begann und wo ich im Moment stand.

# Persönliches Fazit

## Erfahrungen und Erkenntnisse:

### Technische Herausforderungen und Lösungen:

### Dokumentation und Präsentation:

### Zeitmanagement:

### Lernprozess:

# Quellenverzeichnis

### React Tutorials:

1. [React JS Explained In 10 Minutes - YouTube](https://www.youtube.com/watch?v=MRIMT0xPXFI)
2. [React in 100 Seconds - YouTube](https://www.youtube.com/watch?v=Ke90Tje7VS0)
3. [React protected routes in 4 minutes - YouTube](https://www.youtube.com/watch?v=ojN8f2TyjH4)
4. [Create A Workout Planner With React | Medium](https://coltonkaiser.medium.com/how-to-create-a-unique-workout-planner-with-reactjs-1bf56491a94a)

### Full-Stack Authentication mit React, Node, MySQL:

1. [Login and Registration Form using React + Node + MySQL | Login and Sign Up Form with Validation - YouTube](https://www.youtube.com/watch?v=ZlE2djesDiw)
2. [Build a Full-Stack Authentication App With React, Node, Express, MySQL | Login, Registration, Logout - YouTube](https://www.youtube.com/watch?v=AU9E6XqXtCw)

### Passwort Hashing mit React und Node.js:

1. [react js password bcrypt - YouTube](https://www.youtube.com/watch?v=MN_1maGvR8c)
2. [react js password hashing - YouTube](https://www.youtube.com/watch?v=0oFq7zDvTsg)
3. [Hashing Passwords in Node and Express using bcrypt - YouTube](https://www.youtube.com/watch?v=O6cmuiTBZVs)
4. [Password Hashing in Node.js with bcrypt - YouTube](https://www.youtube.com/watch?v=vyz47fYjiJM)

### Github:

1. [Repository](https://github.com/noahburren/Abschlussprojekt_2024_API-1_noabur_Board_-GamificationAndroidApplication-)
2. [Backlog](https://github.com/users/noahburren/projects/3/views/1)
3. [Roadmap](https://github.com/users/noahburren/projects/3/views/4)

### Andere Links

1. [Create a Logout Button - React Tutorial 47 - YouTube](https://www.youtube.com/watch?v=PKwu15ldZ7k)
2. [How to Make a Gantt Chart in Excel - YouTube](https://www.youtube.com/watch?v=eayt8QY9ZUQ)
3. [JavaScript – SELFHTML-Wiki](https://wiki.selfhtml.org/wiki/JavaScript)
4. [HTML Tutorial](https://www.w3schools.com/html/)

# Glossar

**A**

**Authentication (Authentifizierung):** Prozess der Überprüfung der Identität eines Benutzers, um sicherzustellen, dass sie berechtigt sind, auf bestimmte Ressourcen zuzugreifen. Im Projekt wurde dies durch die Implementierung von Login und Registration mit React.js und MySQL realisiert.

**AuthProvider:** Eine React-Komponente, die den Authentifizierungskontext bereitstellt, um den Anmeldestatus über die Anwendung hinweg zu verwalten und zu überprüfen.

**B**

**bcrypt:** Ein Passwort-Hashing-Algorithmus, der verwendet wird, um Passwörter sicher zu speichern. Im Projekt wurde bcrypt verwendet, um Passwörter vor dem Speichern in der Datenbank zu hashen.

**C**

**CSS (Cascading Style Sheets):** Eine Sprache zur Gestaltung der Präsentation eines HTML-Dokuments. Im Projekt wurde CSS verwendet, um die Benutzeroberfläche der Anwendung zu stylen und responsive zu gestalten.

**D**

**Database (Datenbank):** Eine strukturierte Sammlung von Daten, die in digitalen Formaten gespeichert sind und leicht abgerufen, verwaltet und aktualisiert werden können. Im Projekt wurden sowohl MySQL als auch JSON für die Speicherung und Verwaltung von Benutzerdaten und Übungsplänen verwendet.

**Debugging (Fehlerbehebung):** Der Prozess der Identifizierung und Behebung von Fehlern oder Problemen in der Software. Im Projekt wurde Debugging verwendet, um Probleme bei der Benutzeranmeldung, der Datenbankverbindung und anderen Funktionalitäten zu lösen.

**G**

**Github Projects:** Ein Tool zur Projektverwaltung, das auf GitHub verfügbar ist und verwendet wird, um Aufgaben, Funktionen und Fortschritte innerhalb eines Projekts zu planen, zu organisieren und zu verfolgen.

**J**

**JavaScript:** Eine weit verbreitete Programmiersprache, die für die Entwicklung von Webanwendungen verwendet wird. Im Projekt wurde JavaScript für die Implementierung von Funktionen wie Formularvalidierung, Benutzerinteraktionen und API-Aufrufen verwendet.

**L**

**Login:** Der Vorgang, bei dem sich ein Benutzer in ein Computersystem oder eine Anwendung einloggt, um auf geschützte Bereiche oder Ressourcen zuzugreifen. Im Projekt wurde ein Login-System mit React.js und MySQL implementiert.

**M**

**MySQL:** Ein relationales Datenbankverwaltungssystem, das in vielen Webanwendungen verwendet wird. Im Projekt wurde MySQL verwendet, um Benutzerdaten, Übungspläne und andere relevante Informationen zu speichern und zu verwalten.

**P**

**Protected Routes (Geschützte Routen):** Routen in einer Webanwendung, die nur zugänglich sind, wenn der Benutzer authentifiziert ist. Im Projekt wurden Protected Routes implementiert, um sicherzustellen, dass nur angemeldete Benutzer auf bestimmte Seiten zugreifen können.

**R**

**React.js:** Eine JavaScript-Bibliothek zur Entwicklung von Benutzeroberflächen, die von Facebook entwickelt wurde. Im Projekt wurde React.js verwendet, um eine interaktive Benutzeroberfläche zu erstellen, einschließlich Komponenten für Login, Registration, Übungsauswahl und Kalender.

**S**

**SQL (Structured Query Language):** Eine spezielle Programmiersprache zur Verwaltung von Daten in relationalen Datenbankmanagementsystemen. Im Projekt wurde SQL verwendet, um Datenbankoperationen wie das Erstellen von Tabellen, das Einfügen von Datensätzen und das Abfragen von Informationen zu ermöglichen.

**useState:** Ein Hook in React, der es Funktionalkomponenten ermöglicht, ihren eigenen lokalen Zustand zu haben. Im Projekt wurde useState verwendet, um Benutzereingaben und Validierungsfehler zu verwalten.

**U**

**useContext:** Ein Hook in React, der es Funktionskomponenten ermöglicht, den Kontext zu abonnieren, der von einem übergeordneten Komponenten bereitgestellt wird. Im Projekt wurde useContext verwendet, um den Authentifizierungskontext für die Verwaltung des Anmeldestatus zu nutzen.

# Abbildungsverzeichnis

Abbildung 1 Zeitplan 8

Abbildung 2 XAMPP 13

Abbildung 3 phpmyadmin Login Table 13

Abbildung 4 Sign in Seite 14

Abbildung 5 Sign up Seite 15

Abbildung 6 Error mind 1 Zahl 15

Abbildung 7 404 Not Found 19

Abbildung 8 Übungsauswahl 22

Abbildung 9 Übungen bestätigen oder Löschen 22

Abbildung 10 Keine Übung ausgewählt 24

Abbildung 11 Keine Übungen vorhanden 24

Abbildung 12 Übungen werden anzezeigt 25

Abbildung 13 Wochenkalender 26

Abbildung 14 React.js 30

# Ausgefüllte Checkliste