Easy-Cocktail-Mixing

Tino Kederer
Industrie-4.0-Informatik
OTH Amberg-Weiden
Amberg, Deutschland
t.kederer@oth-aw.de

Michael Eichenseer
Industrie-4.0-Informatik
OTH Amberg-Weiden
Amberg, Deutschland
m.eichenseer1@oth-aw.de

Matthias Schießl
Industrie-4.0-Informatik
OTH Amberg-Weiden
Amberg, Deutschland
m.schiessl1@oth-aw.de

Jakob Götz
Industrie-4.0-Informatik
OTH Amberg-Weiden
Amberg, Deutschland
j.goetz2@oth-aw.de

Stefan Ries
Industrie-4.0-Informatik
OTH Amberg-Weiden
Amberg, Deutschland
s.ries@oth-aw.de

Keywords— web, javascript, mongodb, cocktailsm, react

I. Einleitung

Im Rahmen der Vorlesung „Web-Anwendungsentwicklung“ haben sich verschiedene Teams zusammengefunden, um Projektarbeiten zu realisieren. Unser „Team Blau“ besteht aus 5 Studierenden des Studiengangs „Industrie-4.0-Informatik“, die diese Vorlesung als Wahlfach besuchen. Da keiner aus dem Team tiefergehende Erfahrung in der Web-Entwicklung hatte und diese Inhalte auch sonst im Studium nicht näher beleuchtet wurden, war es allen Teilnehmern sehr wichtig dies auf diesem Weg nachzuholen.

In dem hier geschilderten Projekt soll es um die Entwicklung einer einfach zu bedienenden und übersichtlichen Web-Anwendung gehen, die dem Nutzer hilft gute Cocktails zu mischen. Das Ganze soll als Schritt-für-Schritt abzuarbeitende Anleitung realisiert werden, bei der der Nutzer erst einen Arbeitsschritt ausgeführt haben muss und dies mittels Interaktion bestätigt, bevor er den nächsten Schritt aufrufen kann.

Die ganze Anwendung soll durch eine Datenbank gestützt und unterstützt werden, so dass eine Nutzerverwaltung und verschiedene spezifische Nutzerfunktionalitäten möglich sind. Ebenfalls soll die Anwendung durch eine ansprechende Oberfläche und kleinere Animationen ein gutes Nutzererlebnis bieten.

II. Rahmenbedingungen

A. Festgelegtes Aufgabenfeld

Bereits im Voraus wurde uns in Rahmen der Studienarbeit Vorbereitung mitgeteilt, dass es sich um eine Web-Applikation handeln muss. Herr Prof. Neumann stellte auch bereits eine Reihe von Themen vor, die wir als Inspiration nutzen konnten. Hieraus übernahmen wir auch die Idee, eine Website für Cocktailmixing zu erstellen.

B. Wöchentliche Sprints

Bei den wöchentlichen Sprints konnte jedes Teammitglied seine Arbeitsergebnisse der vergangenen Woche präsentieren. Hier wurden dann Sprintziele für die kommende Woche festgelegt und auf aufgetretene Probleme hingewiesen.

C. Abgabetermin

Da am 02.07.21 der Großteil der Entwicklung abgeschlossen sein sollte, musste man sich entscheiden, welche Features noch implementiert werden sollten und welche außenvor bleiben.

III. Qualitätsziele

Benutzbarkeit	Der Nutzer hat direkt alle Features auf einen Blick vor sich und kann diese ohne Hindernisse einfach Bedienen.
Funktionale Eignung	Die Website macht genau das, was der Nutzer von ihr erwartet. Sie zeigt verschiedene Cocktails an, wie man sie einfach mixen kann und beinhaltet eine Suchfunktion.
Zuverlässigkeit	Durch ausgiebiges manuelles und automatisiertes Testen kann der Nutzer davon ausgehen, dass es zu keinen Abstürzen der Website kommt.
Portabilität	Dadurch, dass es eine Web-Applikation ist, die mit Bootstrap designt wurde, ist sichergestellt, dass die Website höchste Portabilität besitzt.

IV. Verwendete Technologien

Die Anwendung ist auf dem Prinzip des MERN-Stacks aufgebaut. Dieses Prinzip bezieht sich bei der Datenbank auf eine MongoDB (in unserem Fall Atlas). In der Backendprogrammierung auf NodeJS und ExpressJS. Die Oberfläche wird mit dem Frontendframework ReactJS umgesetzt. Im folgenden wird noch einmal auf die einzelnen Komponenten des Stacks eingegangen.

A. MongoDB

In der MongoDB werden bestimmte Benutzerdaten, wie der Benutzername, der Passwort Hash und die für den Benutzer spezifisch favorisierten Cocktails in einer MongoDB Atlas Datenbank gespeichert. Des Weiteren enthält diese Datenbank alle Rezept mit dazugehörigen Zutaten mit Menge sowie in Base64 kodierte Bilder.

B. ExpressJS

ExpressJS ist ein Framework, das auf NodeJS aufsetzt. Es dient dazu die Funktionalitäten, welche NodeJS bereitstellt, einfacher und schneller zu benutzen. Wir benutzten ExpressJS im Backend, um Routen festzulegen und um mit der Datenbank zu kommunizieren.

C. ReactJS

Die ReactJS Library wurde verwendet, um die im Backend bereitgestellten Daten im Frontend darzustellen. Es wurde ein komponentenbasierter Aufbau gewählt, sodass die Webapplikation in verschiedene Container aufgeteilt ist. Im Wesentlichen besteht die Applikation auf drei großen Containern (TopContainer, MainContainer, BottomContainer). Je nachdem in welcher Route man sich aktuell befindet ändert sich der Inhalt der verschiedenen Container.

V. Frontend Containeraufbau

A. TopContainer

Der TopContainer befindet sich auf der oberen Seite der Webapplikation. Er ist bei jeder Route vorhanden. Er besteht aus folgenden drei Komponenten:

1. Banner
   Das Banner, welches sich auf der linken Seite des TopContainers befindet, wechselt durch ein KlickEvent die Route auf die Gesamtübersicht aller Cocktails (Cocktailoverview).

2. NavigationBar
   Die Navigation Bar enthält verschiedene Buttons (Favoriten, Letzte Cocktail, Meine Cocktails, Cocktail erstellen, Login/Logout), welche jeweils den Benutzer pro Klick auf eine eigene Route weiterleiten. Alle Buttons (bis auf Login/Logout) sind ohne Funktion, solange der Benutzer nicht eingeloggt ist. (Wegen Zeitgründen haben nur die Buttons Favoriten und Login/Logout eine Funktion). Der Inhalt des Login/Logout Buttons ändert sich je nachdem ob ein Benutzer eingeloggt ist oder nicht.

3. Suchleiste
   Die Suchleiste filtert die Cocktails in der Cocktailoverview nach dem eingegebenen String und zeigt diese an.

B. MainContainer

Im MainContainer werden mithilfe der React Komponenten BrowserRouter und Switch verschiedene Routen erstellt. Jeweils eine Route für die Funktionen der NavigationBar Buttons, sowie jeweils eine Route für die aus dem Backend empfangenen Cocktails. Als Hauptroute wird „/Cocktailoverview“ festgelegt

1. CreateRoutes
   Die Komponente CreateRoutes befindet sich im MainContainer und erstellt für jeden aus dem Backend empfangenen Cocktail eine Route, die den Inhalt eines StepbyStep Komponenten hat.

2. Cocktailliste
   Die Hauptroute der Applikation führt zur Cocktailoverview. Diese Overview besteht aus Cocktailkomponenten.

3. CocktailKomponent
   Jeder aus der Datenbank gelesene Cocktail wird in eine Cocktailkomponente eingebettet. Diese Komponente besteht aus einem Bootstrap Card Element. Als Überschrift der Cocktailname, als Body des Card Elements das Bild des dazugehörigen Cocktails. Sobald man eingeloggt ist, hat man die Möglichkeit eine Cocktailkomponente zu favorisieren, indem man den Stern auf der Komponente anklickt. Durch erneutes Anklicken des Sternes wird die Favorisierung wieder entfernt. Diese Favorisierung wird für benutzerspezifisch in der Datenbank hinterlegt. Eine weitere Eigenschaft der Cocktailkomponente ist, dass die Zutaten des jeweiligen Cocktails angezeigt werden, sobald man mit der Maus über das Element fährt.

4. Favoritenliste
   Die Favoritenliste wird im MainContainer angezeigt, sobald man in der NavigationBar auf den dazugehörigen Button klickt. Es werden nun alle mit einem Stern markierten Cocktails (benutzerspezifisch) angezeigt. Dazu werden auch die Cocktailkomponenten verwendet.

5. StepbyStep
   Klickt man auf ein Cocktailkomponenten wird man zur dazugehörigen Route weitergeleitet, die die Step by Step Anleitung für den Cocktail enthält. Auf der der linken Seite der Step by Step View ist der Name des Cocktails mit Bild und Zutaten, sowie deren Mengenangaben abgebildet. Auf der rechten Seite befindet sich ein Button, der nach dem Klicken eine sog. Pagination erscheinen lässt. Durch diese Pagination ist es dem Benutzer ermöglicht sich schrittweise durch die Zubereitung des Cocktails zu arbeiten.

6. LoginForm
   Für den Login wird im Frontend eine Eingabemaske zur Verfügung gestellt, die beim Aufruf der Login Seite gerendert wird. Um für ein ansprechendes Design zu sorgen wurde ein eigenes Textfeld-Element entwickelt, das linksbündig ein selbstwählbares Icon anzeigt. Dieses Element wurde

mit CSS so designet, dass sämtliche optischen Attribute weiterhin frei einstellbar sind und das Textfeld ansonsten im Grunde genauso wie ein ganz normales Textfeld funktioniert.
In der Login Maske ist dann ein Feld für den Nutzernamen und ein Feld für das Passwort vorhanden und beide sind mit passenden Icons versehen worden. Beim Drücken der Enter-Taste oder Betätigung des entsprechenden Buttons wird eine POST-Anfrage an die API gesendet und mit der erhaltenen Antwort dann der Status der App neu festgelegt.

7. RegisterForm
   Für das Registrieren eines neuen Nutzers gibt es ebenfalls eine Eingabemaske, die das gleiche Textfeld-Element wie in der Login Maske verwendet. Beim Drücken von Enter oder Betätigung des Buttons wird auch hier wieder ein POST-Request ausgelöst und der Nutzer wird mittels Alert darüber informiert, ob sein eingegebenes Passwort die Sicherheitskriterien erfüllt bzw. die erfolgreiche Registrierung wird rückgemeldet.

C. BottomContainer

Der BottomContainer ist, wie der TopContainer in jeder Route vorhanden. Er enthält ein Impressum- und ein Datenschutzbutton. Diese führen weiter zu Seiten, bei denen das Impressum oder der Datenschutz angezeigt werden soll.
Die Datenschutz Richtlinien und das Impressum wurden über online Generatoren erzeugt. Wir benötigten diese, da unsere Website öffentlich erreichbar sein soll und sie eine Registrierungsfunktion besitzt.

VI. Design

Zum Designen wurde CSS-Framework Bootstrap benutzt. Dieses stellt verschiedene Klassen bereit. Es wurden z. B. im Cocktailkomponent die Klasse „card“ oder im Topcontainer die Klasse „nav-bar“ verwendet. Auch für kleineres Styling wie der Abstand von Komponenten wurde Bootstrap benutzt. Da das Framework jedoch nicht alles Gewünschte bereitstellen kann, mussten teilweise auch selbst „.css“ Dateien erstellt und eingebunden werden.

VII. State und Propertys - Frontend

Im ganzen Frontend stellt das „App“- Element alle für den Betrieb der App wesentlichen Informationen bereit und verteilt diese an seine Kindelemente, Da manche Informationen für die korrekte Darstellung der Kindelemente nötig waren, mussten diese von der App als Property an das jeweils nächste Kind im Kompontenbaum weitergeben werden.
So muss zum Beispiel der State der App „UserIsLoggedIn“, der anzeigt, ob der Nutzer der App gerade im Moment eingeloggt ist und der zum Beispiel darüber entscheidet, ob manche Buttons in der NavigationBar gesperrt oder freigegeben sind, erst an diese jeweiligen Buttons weitergegeben werden, damit diese sich selbständig in ihrer Darstellung anpassen können. In diesem Beispiel wird diese State-Variable zuerst an den TopContainer als Property übergeben, dieser übergibt seinerseits dieses Property wieder als Property an die NavigationBar und diese setzt das „disabled“ Attribut der Buttons entsprechend.

Ebenso wie das Herunterreichen von Informationen an die Kind Elemente ist es ebenso nötig, dass diese ihrerseits wiederum den Status der App verändern können, damit zum Beispiel ein neu favorisierter Cocktail angezeigt wird oder die Darstellung der Seite angepasst werden kann, wenn ein Logout durch Drücken des Buttons in der NavigationBar erfolgt.
React gestattet nur das Binden von Informationen eines Elternelements an seine Kindelemente und nicht umgekehrt. Es ist also nur ein One-Way-Binding möglich. Damit ein Kindelement aber den Status seines Elternelements verändern kann, wird im Elternelement eine Funktion erstellt, die einen bestimmten Status verändern kann und eine Referenz auf diese Funktion wird per Property an das Kindeelement übergeben. Somit ist es mit diesem Umweg auch möglich, dass ein Kindelement sein Elternelement beeinflussen kann. Dies war zum Beispiel bei der Suchfunktion, beim Login/Logout und dem Favorisierbutton nötig.

VIII. Aufbau Backend

Im Backend kommt eine Node.js Anwendung zum Einsatz, die für das Frontend eine API zur Verfügung stellt und damit die Verbindung zwischen Frontend und Backend schafft. Für das Routing wurde Express eingesetzt und für die einzelnen Funktionalitäten wurde jeweils eine „*.controller.js“-Datei angelegt, die die entsprechenden API-Funktionalitäten als statische Klassenmethoden realisiert.
Außerdem gehört zu jeder Teilroute noch eine „*.route.js“-Datei, die eine entsprechende POST oder GET-Methode an die jeweiligen Funktionen in der entsprechenden Controller-Klasse bindet.
So wird bei einem entsprechenden POST- oder GET-Aufruf, dann die jeweilige Klassenmethode aufgerufen. Ein Aufruf von „/api/user/register“ führt so zu einer Registrierungsanfrage für einen neuen Nutzer.
Desweiteren wurde für einen einfacheren Zugriff auf die Datenbank eine kleine Wrapper-Klasse entwickelt, die die Handhabung von MongoClient etwas vereinfacht und die wesentlichen Informationen zur Datenbank als Klassenvariablen speichert.

IX. Kommunkation zwischen Frontend und Backend

Die API hat im Wesentlichen die Aufgabe der Frontendanwendung eine Kommunikation mit der Datenbank zu ermöglichen. Da viele Funktionalitäten nur für eingeloggte Nutzer möglich sein sollen, muss sich die Frontenanwendung mittels Session-Token beim Aufruf authentifizieren. Auf die Wesentlichen Funktionalitäten und deren technische Besonderheiten soll im Folgenden kurz eingegangen werden.

A. Login

Für die Anwendung galt als wichtiges Merkmal, dass verschiedene Funktionalitäten wie zum Beispiel eine Favoritenliste zu realisieren sind, die für jeden Nutzer permanent gespeichert werden kann. Dafür war es nötig, dass zunächst eine Userfunktionalität mit Registrierung, Login und Logout vorhanden sein musste. Für den Login wird zunächst geprüft, ob überhaupt ein Datensatz mit den eingegebenem Nutzernamen gefunden wird. Ist dies der Fall, so wird der gespeicherte Passworthash mit dem eingegebenem Passwort verglichen. Ist beides erfolgreich, so erhält der Nutzer eine Antwort als JSON-Format mit dem Feld „success“ als true-Wert und dem Session-Token in einem weiteren Feld. Gibt es irgendwo keine Übereinstimmung so wird „success“ in der Antwort auf false gesetzt und eine entsprechende Fehlermeldung mit in die JSON-Antwort eingefügt.

B. IsLoggedIn

Für viele Elemente im Frontend ist es für die Bedienung und Darstellung wesentlich, dass die Frontanwendung abfragen kann, ob der Nutzer gerade eingeloggt ist. Dafür wurde eine Schnittstelle geschaffen, die abprüft ob das gesendete Session-Token vorhanden ist und dem Frontend dann ein true signalisert. Ist das Session-Token nicht vorhanden oder sendet die Anwendung ein falsches Session-Token, so wird false zurückgesendet.

C. RegisterNewUser

Für eine einfache Mehrbenutzerbedienung ist ebenfalls eine Registrierungsfunktion nötig, damit auch neue Nutzer die entsprechenden Funktionen der Anwendung nutzen können. Die entsprechende Schnittstelle prüft zunächst, ob das eingegebene Passwort und die Passwort-Bestätigung übereinstimmen. Ist dies der Fall, so werden anschließend die Sicherheitsrichtlinien bezüglich der Mindestlänge von 8 Zeichen und dem Vorhandensein von jeweils mindestens einer Ziffer und mindestens einem Sonderzeichen geprüft. Für die entsprechenden Prüfungen wurde ein kleiner Regex geschrieben. Erfüllt das eingegebene Passwort alle Kriterien, so wird als nächstes geprüft, ob der eingegebene Nutzername bereits in der Datenbank vorhanden ist. Sollte der Nutzer bereits existieren oder das Passwort die Sicherheitskriterien nicht erfüllen, so wird eine Antwort mit entsprechendem Fehlertext zurückgesendet. Wurde eine gültige Anfrage gestellt, so wird ein neuer Datensatz mit den eingegebenen Daten angelegt.

D. Favoriten

Für die avorisierten Cocktails eines jeden Nutzers ist im jeweiligen Datensatz des Nutzers ein Array hinterlegt, welches die entsprechenden Objekt-IDs der Cocktails beinhaltet.
Beim Abfragen der Cocktails muss die Anwendung wieder das Session-Token des jeweiligen Nutzers senden und die entsprechende API-Schnittstelle prüft zunächst ob das Session-Token gültig ist. Anschließend werden für diesen Nutzer mittels der Objekt-IDs aus seinem Datensatz die Daten zu den entsprechenden Cocktails abgefragt. Da jedes Abfragen eines Cocktails ein asynchroner Aufruf ist und die komplette Ausführung aller Abfragen für das Sammeln der entsprechenden Daten nötig war, musste hier ein „Promise.all“-Aufruf erfolgen, damit die Methode an dieser Stelle so lange wartet, bis alle asynchronen Aufrufe beendet sind. Anschließend werden die entsprechenden Cocktails als JSON zurückgegeben.
Für das Favorisieren/Entfavorisieren eines Cocktails wurde ebenfalls eine Schnittstelle geschaffen, die den Cocktail in das entsprechende Feld des Nutzerdatensatzes einträgt, falls dieser noch nicht vorhanden ist und ihn entfernt, falls er bereits vorhanden ist.

E. Cocktailliste

Um die Cocktailliste im Frontend darzustellen, muss diese vom Backend ins Frontend übertragen werden. Dazu wird die vom Backend bereitgestellt API genutzt. Die Abfrage wird mit einen http-get-request durchgeführt. Im Frontend wird in der App.js Datei die Datenbank mithilfe der API abgefragt. Für die Gesamtliste aller Cocktails existiert im Backend ein Cocktails-DataObject welches an dieser Stelle die Verbindung mit der Datenbank herstellt. Beim Starten der Backend-Anwendung sammelt dieses einmalig alle Cocktails aus der Datenbank und stellt eine entsprechende API-Schnittstelle für das Frontend bereit.

X. Datenaufbau

Bei einer MongoDB werden einzelne Datensätze gespeichert, welche in sog. Collections gruppiert werden können. Wir haben uns dafür entschieden zwei Collections zuhalten, eine namens „Recipes“, in welcher alle Daten zu den Rezepten gespeichert werden und eine „Users“ Collection, welche userspezifische Daten enthält.
Unabhängig von der Collection, in welcher ein Datensatz gespeichert wird, muss er immer eine eindeutige ID (_id) haben, über welche er identifiziert werden kann. Zusätzlich zur ID wurden dann je nach Typ des Datensatzes weitere Daten gespeichert, so dass alle Datensätze innerhalb einer Collection den gleichen Aufbau besitzen.

XI. Hosting

Ursprünglich war für das Hosting unserer Website AWS angedacht. Da niemand aus unserem Team bisher Erfahrungen in Bereich Hosting von dynamischen Websites hatte, war der erste Gedanke AWS zu nutzen, da dieser Dienst sehr weit verbreitet ist, auch in der freien Wirtschaft. Ein weiterer Grund, der für AWS sprach, war dass wir ein Educated Account mit bis zu 20€ Kreditrahmen zur Verfügung gestellt bekommen haben. Leider stellte sich das Hosting als komplizierter heraus als gedacht und so war innerhalb von kurzer Zeit der Kreditrahmen aufgebraucht. Deshalb entschieden wir uns kurzerhand die Applikation auf Heroku zu hosten, da dieser Dienst kostenlos ist.

XII. Testing

Nach einer kurzen Informationsphase und nachgehender Absprache im Team entschieden wir uns dazu, als

Testframework Jest zu benutzen. Denn Jest war bereits in React integriert, gut dokumentiert und kostenlos.

XIII. Umsetzung UserStorys

Die vor dem Projekt festgelegten UserStorys wurden gut umgesetzt. Alle Must- und Should-UserStorys wurden eingehalten und es wurden sogar einige Could-UserStorys umgesetzt. Lediglich die UserStorys, die sich auf das Erstellen, Verändern und Teilen eigens erstellter Cocktails bezogen konnten nicht mehr realisiert werden.
Mit Blick auf den anfänglichen Wissenstand der Gruppenmitglieder darf dieser Projektstand aber durchaus als großer Erfolg gewertet werden.

XIV. Fazit und Ausblick

A. Fazit

Das Definieren von UserStorys mit den Labeln Must, Should, Could am Anfang des Projekts, gaben dem Projekt einen roten Faden, an dem man sich gut orientieren konnte.
Die Teammitglieder haben in Hinblick auf Web-Entwicklung viel gelernt, also das ursprüngliche Ziel der Gruppenmitglieder sich hier neues Wissen anzueignen wurde definitiv erreicht.
Durch die Verwendung des MERN-Stacks wurde in diesem Projekt die Anwendung von gleich mehreren in der Web-Entwicklung verbreiteten Technologien erlernt. Jeder im Team konnte sich somit für zukünftige Projekte in diesem Bereich eine solide Wissensgrundlage erarbeiten.

B. Ausblick

Da noch ein paar Could-UserStorys unbearbeitet sind, würde es als nächsten Schritt gelten diese Umzusetzen. Außerdem kann man das Design noch weiter verbessern, sodass ein noch besseres Nutzergefühl entsteht. Man könnte z. B. für jede Zutat, in der Step by Step Route ein Bild bzw eine Animation anstatt nur einen Text einfügen. Man könntet die Kalorienanzahl dynamisch halten, sodass der Zähler sich erhöht, sobald man einen weiteren Schritt bei den Steps macht. Des Weiteren wäre es gut, wenn die Datenbank noch mehr Cocktails hätte, was sich zum Teil sicher dadurch lösen würde, dass die verbleibendenden UserStorys noch umgesetzt werden.