Dokumentation Webprogrammierung

Anmerkungen

Der zugrunde liegende Code wurde in einem privaten GitHubRepository veröffentlicht.

Dem Dozent Jonas Heuer (alias jonHeu643) wurde Zugriff gewährt.

Code oder Zugriff auf Nachfrage: tim.kauer@outlook.com

Matrikelnummer: 5578340

Bewertungskriterien

Aufgabe ist eine Webseite mit React zu entwickeln und den Code auf GitHub zur Verfügung zu stellen. Bestandteil der Aufgabe ist eine Dokumentation (40%) und Code (60%). Dieses Dokument beinhaltet die Dokumentation.

Doku (je 20% zur Doku-Note):

- Requirements:
  - In diesem Dokument werden die Anforderungen des Projekts beschrieben. Sie stellen die Bewertungsgrundlage der Arbeit dar.
- Paper Prototype:
  - Hier soll das Design beschrieben werden. Dies geschieht in Form einer Skizze des Front Ends.
- Klassendiagramme:
  - Hier die Ordnerstruktur dargestellt und erklärt werden. Zudem werden die Komponenten beschrieben.
- Konzept:
  - Ausformulierung und Begründung des Aufbaus (Components) der App.
- Abschluss:
  - Hier wird beschrieben, ob alles funktioniert hat und welche Schwierigkeiten aufgetreten sind.

Entwicklung (je 10% zur Entwicklungs-Note):

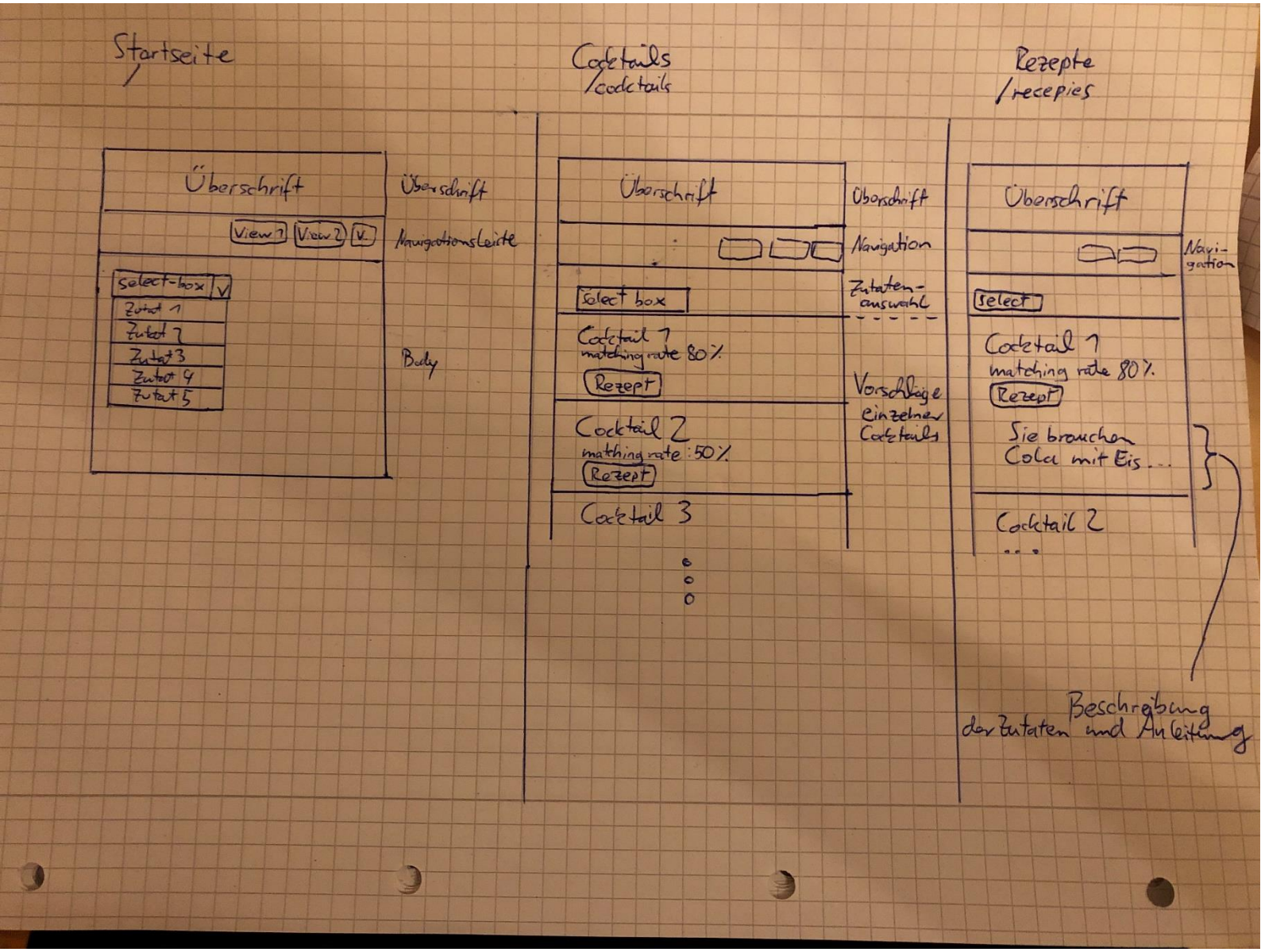
- Struktur
- MaterialUI
- useState
- state/setState
- class based oder function
- Hilfsmittel (props)
- Naming
- Codequalität/Comments
- Funktion
- Router

Requirements Cocktail WebApp

Die Webseite soll Rezepte für Cocktails darstellen. Sie soll aus 3 Views bestehen, die über Router verbunden sind. Zutaten können in einer Eingabe hinzugefügt werden. Daneben gibt es eine Rezeptübersicht und das Rezept. Der Nutzer soll Zutaten die vorhanden sind auswählen können. Ob ein zusätzlicher Einkauf notwendig ist, soll auf der Seite ausgegeben werden. In einer Rezeptübersicht werden alle Rezepte die matchen dargestellt. Dazu soll eine prozentuale Matchingrate ausgegeben werden, die zeigt wie viel Prozent der Zutaten, die für den Cocktail benötigt werden, bereits vorhanden sind. Auf der Rezept-Seite sollen die benötigten Zutaten und Zubereitungsschritte ausgegeben werden.

Paper Prototype

- Hier soll das Design beschrieben werden. Dies geschieht in Form einer Skizze des Front Ends.



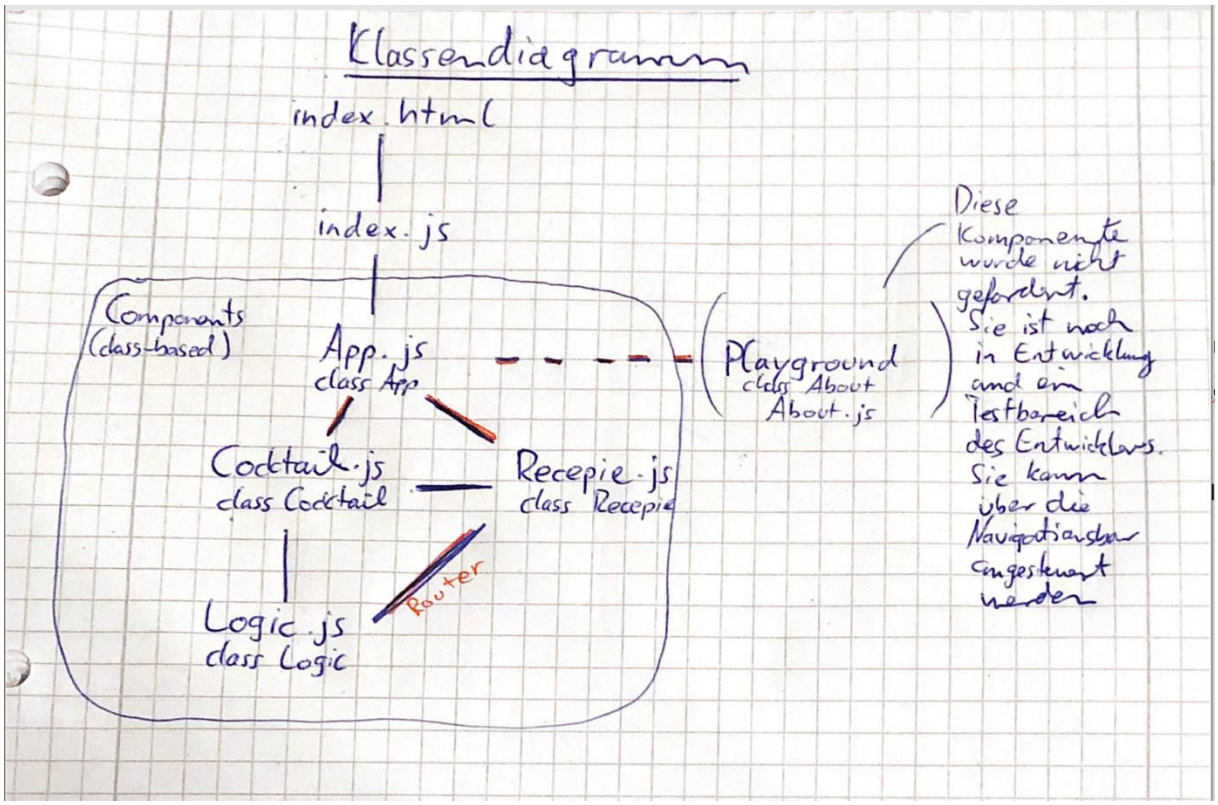
Anmerkung:

Auf der Startseite wird eine zusätzliche Überschrift angezeigt.

Die Navigationsleiste enthält neben den Links/Routes auch einen „Reset“-Button um die Einträge zu leeren.

Klassendiagramme

Hier die Ordnerstruktur dargestellt und erklärt werden. Zudem werden die Komponenten beschrieben.



Ordnerstruktur

- public
  - index.html
  - …
- src
  - App.js
  - Components
    - Playground
      - About.js
      - …
    - Cocktail.js
    - Logic.js
    - Recepie.js
  - CSS
    - App.css
  - index.js

- …

Konzept

Ausformulierung und Begründung des Aufbaus (Components) der App.

Ich habe mich für eine klassenbasierten Aufbau meiner Webseite entschieden. Die Nutzung von State ist mir dadurch leichter gefallen. Jede Component wurde in ein eigenes File ausgelagert. Die gesamte App sowie Quellcode sind Englisch.

App

Hier wird die Überschrift, Navigation(-sbar) mit Routing zu den weiteren Komponenten und der State festgelegt. In den State wird ein Array übergeben, das die vorgegebenen Zutaten beinhaltet. Dieser wird an die Children übergeben und kann verändert werden. Beim ersten Aufruf wird hier eine Welcome-Page gerendert, die neben allen Funktionen auch eine kurze Beschreibung und Aufforderung für den User bereithält. Die Multiselectbox ist eine externe Komponente, die zusätzlich importiert wurde. Über das Navigationsmenu können die anderen Komponenten angesteuert werden. Darüber hinaus wurde ein Reset-Button implementiert, der den State zurücksetzt. Alle ausgewählten Cocktails werden auf den Initialwert zurückgesetzt

Cocktail

Ist eine Child-Component von App. Darin sind die verschiedenen Rezepte gespeichert. Cocktail ruft die Child-Componente „Logic“ auf. Der Output (Cocktails mit Name und MatchingRate) wird in Form von Boxen unter der Select Box dargestellt. Zur besseren Übersicht im Code wurde die Logik ausgelagert. Der Endnutzer bekommt davon jedoch nichts mit.

Logic

Der Parent von Logic ist Cocktail. Via props kommunizieren die Komponenten miteinander. In Logic wird die gesamte Matching-Rate berechnet. Diese beschreibt wie viele Zutaten prozentual für die hinterlegten Rezepte vorhanden sind. Nach der Berechnung rendert diese Komponente jeden einzelnen Cocktail, gibt die Matching-Rate, sowie eine Einkaufempfehlung aus. Darunter befindet sich ein Button mit Material UI. Dieser enthält einen Link zur Rezept-Komponente. Die einzelnen Cocktails werden nach ihrer Matching-Rate sortiert. Diese Komponente enthält viel JavaScript. Wählt der User zunächst eine Zutat aus, entmarkiert sie danach wieder, sodass keine Zutat ausgewählt wurde, dann wird eine Aufforderung ausgegeben. (Zuvor wurde ein Error ausgegeben.)

Recepie

Via dem Navigationsmenu oder dem Button kommt man zu dieser Komponente. Betätigt man den „RECEPIE“-Button seines gewünschten Cocktails, wird diese Komponente unter dem Button individuell für genau diesen Cocktail angezeigt. Technisch wird mit dem Link aus Logik auch die in Cocktail definierte Rezeptanleitung als via prop übergeben. Diese Anleitung wird in Recepie gerendert und entsprechend angezeigt.

Playground

Hierbei handelt es sich um eine Komponente die kein Teil der Aufgabenstellung ist. Sie beweist die neu erlernten Kenntnisse des Entwicklers in Form von Routing und Material UI, sowie Nutzung des localStorage. Die Komponente sollte eine Möglichkeit sein, Zutaten manuell anzulegen. Diese Funktion ist noch in Bearbeitung und konnte bis zum

Abgabeschluss nicht mehr implementiert werden. Da sie über den Scope der Anforderungen hinaus geht, sollte die (noch) nicht Funktionalität dieser Komponente keine negativen Auswirkungen auf die Bewertung.

Abschluss

Hier wird beschrieben, ob alles funktioniert hat und welche Schwierigkeiten aufgetreten sind.

Das Ergebnis ist funktionstüchtig, damit sind die Anforderungen erfüllt. Es wurde getestet und dabei noch einige Fehler gecatcht. Das sollte dem Nutzer eine reibungslose Nutzung der Webseite gewährleisten.

Es war durchaus anspruchsvoll sich in kurzer Zeit in React einzuarbeiten und das Konzept zu verstehen. Bei der Entwicklung sind häufig unerwartete Fehler aufgetreten. Mit props zu arbeiten und diese korrekt aufzurufen und zu übergeben war aufwändig zu debuggen. Daneben war insbesondere die Logic eine Fehlerquelle. Nun ist alles getestet und funktionsfähig.

Ausbaufähig ist noch die oben beschriebene Funktionalität von Playground.

Alles in allem konnte ich durch das Projekt viel über React und JavaScript lernen. Im Vergleich zu HTML ist das eine Umstellung. Zeitlich war es mir leider nicht möglich früher anzufangen, trotzdem ist alles rechtzeitig fertig geworden. Nächstes mal würde ich versuchen nicht so knapp fertig zu werden. Das Projekt war durchaus zeitintensiv, mit dem Ergebnis bin ich dafür sehr zufrieden, was das Design und die Funktionalität betrifft.

Oftersheim, 01.03.20

Tim Kauer