Technische Hochschule
Brandenburg
University of
Applied Sciences

Vergleich moderner Frontend-Technologien zur Erstellung von Single-Page-Webanwendungen

Bachelorarbeit
zur Erlangung des Grades Bachelor of Science
des Fachbereichs Informatik und Medien der
Technischen Hochschule Brandenburg

vorgelegt von:
Alexander Diel

Betreuer: Prof. Dr.-Ing. habil. Michael Syrjakow
Zweitgutachter: Prof. Dr.-Ing. Thomas Preuß

Brandenburg an der Havel, 10. Oktober 2022

Kurzfassung

In der heutigen Zeit finden Single-Page-Webanwendung immer mehr Verwendung. Entsprechend dieser Entwicklung wurden Technologien, zur Erstellung von Single-Page-Webanwendungen, angefertigt. Um ihre Unterschiede und individuellen Anwendungsmöglichkeiten zu ermitteln, wurde mit den drei populärsten Technologien eine Webanwendung gebaut und eine Evaluation anhand einer Reihe von Kriterien durchgeführt.

Schlüsselwörter

Single-Page-Webanwendung
React
Vue.js
Angular

Gender-Hinweis

Die in dieser Bachelorarbeit verwendeten Personenbezeichnungen beziehen sich immer gleichermaßen auf weibliche und männliche Personen. Auf eine Doppelnennung und gegenderte Bezeichnungen wird zugunsten einer besseren Lesbarkeit verzichtet.

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Die Bedeutung von Single-Page-Webanwendungen nimmt in der heutigen Zeit immer mehr zu. Anstatt möglichst viele Aufgaben auf dem Server zu erledigen, werden diese immer mehr auf die Seite des Clients verlagert (Steyer, 2021, S. 15). Basierend auf diesem Prinzip wurden Technologien, wie Bibliotheken oder Frameworks erstellt, welche die Implementierung solcher Single-Page-Webanwendungen erleichtert. Die drei meistverwendeten Vertreter sind hierbei die Bibliothek „React“, sowie die Frameworks „Angular“ und „Vue.js“ (Stack Overflow, 2022). Doch worin unterscheiden die sich genau? Um dies zu bestimmen, werden die drei genannten Technologien anhand Kriterien, wie deren Leistung, Beliebtheit oder der Lernkurve verglichen. Zusätzlich wird jeweils eine Webanwendung mit jeder dieser Technologien erstellt. Hierbei lässt sich sehen, inwiefern die konkreten Implementierungen sich unterscheiden. Ziel der Arbeit ist es, Vor- und Nachteile jeder dieser Technologien zu erkennen und zu entscheiden, wann welche Technologie am besten eingesetzt werden sollte.

1.2 Wichtige Begriffe

1.2.1 Client-Server-Modell

Das Client-Server-Modell beschreibt eine Methode mit den Aufgaben in einem verteilten System aufgeteilt werden können. Das System wird dabei in zwei Arten von Akteuren, Client und Server, aufgeteilt. Der Client sendet eine Anfrage an einen Server, der Server verarbeitet die Anfrage und sendet eine Antwort zurück. (syedmodassirali, 2019)

1.2.2 Single-Page-Webanwendung

Im Gegensatz zu klassischen Webanwendungen, welche aus mehreren verlinkten HTML-Dokumenten bestehen und bei einer Interaktion mit dem Nutzer einen fertigen HTML-Code zurückgeben, werden in Single-Page-Webanwendungen die Interaktionen auf Seite des Clients bearbeitet. (Skolski, 2016) Single-Page-Webanwendungen bestehen nur aus einem einzelnen HTML-Dokument, welches unter Verwendung von JavaScript dynamisch veränderbar ist. Hierbei gibt der Server kein fertigen HTML-Code zurück, sondern die Benutzeroberfläche wird von JavaScript-Code erstellt und aktualisiert. (Hartmann & Zeigermann, 2019, S. 3) Dadurch ist die Serverbelastung niedriger, da mehr Aufgaben auf der Seite des Clients erledigt werden. (Schulte, 2021)

1.2.3 Programmbibliothek

Eine Programmbibliothek, oder auch nur Bibliothek, ist eine Ansammlung von Funktionen. Zusammen bilden sie ein Hilfsmodul, das einem Entwickler erlaubt, zusammenhängende Aufgaben zu lösen. (Programmbibliothek, o.D) Über eine Schnittstelle ruft der Entwickler die Funktionen selbst auf. (Domin, 2018)

1.2.4 Framework

Ein Framework kann als Sammlung von Bibliotheken angesehen werden. (InterviewBit, 2021) Der Unterschied liegt darin, dass in einer klassischen Bibliothek der Entwickler selbst bestimmt, wann und wo eine Funktion aufgerufen wird. (Domin, 2018) Im Gegensatz dazu bietet ein Framework eine Basis, auf der die Anwendung gebaut wird. Der Entwickler baut zusätzlich seinen eigenen Programmcode ein, wobei die Funktionen durch das Framework aufgerufen werden. (Domin, 2018)

1.2.5 Node Package Manager

„Node.js“ ist eine Laufzeitumgebung für JavaScript, die es ermöglicht, JavaScript-Code außerhalb eines Webbrowsers auszuführen. (Nodejs, o. D.) Bei der Installation von „Node.js“, wird auch der „Node Package Manager“ installiert, ein Werkzeug, welches dafür zuständig ist, JavaScript-Bibliotheken und Frameworks über die Kommandozeile zu installieren. (npmjs, o. D.)

2 Kriterien

In diesem Kapitel werden die Kriterien für den Vergleich der Frontend-Technologien beschrieben.

2.1 Implementierung

Als Erstes soll die konkrete Implementierung verglichen werden. Dabei wird untersucht, wie typische Aufgaben in der Webentwicklung mit der jeweiligen Technologie umgesetzt werden können, beziehungsweise welche Aufgaben nicht mit der Technologie allein umgesetzt werden können und dadurch auf weitere Bibliotheken gesetzt werden muss.

2.2 Leistung

Um die Leistung der Webanwendungen zu vergleichen, wird Lighthouse benutzt. Dabei handelt es sich um ein automatisiertes Entwicklerwerkzeug, welches auf jeder Webseite verwendet werden kann und als Bewertung eine Punktzahl zwischen 0 und 100 zurückgibt. (web.dev1, o. D.) Die Punktzahl wird von sechs Faktoren bestimmt, die unterschiedlich gewichtet sind. (web.dev2, o. D.) Folgende Faktoren gehen in die Bewertung ein:

- Die benötigte Zeit, bis der erste Text oder das erste Bild gerendert wird.
- Die benötigte Zeit, bis die Webseite vollständig interaktiv ist.
- Wie schnell der Inhalt während des Ladens dargestellt wird.
- Die Zeitdifferenz zwischen dem ersten gerenderten Element und dem Zeitpunkt, an dem die Seite vollständig interaktiv ist.
- Die benötigte Zeit, bis das größte Element gerendert wurde
- Wie sehr sich Elemente unerwartet bewegen.

(web.dev2, o. D.)

2.3 Beliebtheit

Für eine Technologie, die weit verbreitet ist, und viele Anwender hat, sind meistens mehr Ressourcen vorhanden. Dadurch ist es einfacher, bei Problemen, Lösungen und Erklärungen zu diesen zu finden. Zusätzlich deutet die Bevorzugung einer Technologie gegenüber den anderen darauf hin, dass Entwickeln in dieser Technologie angenehmer ist.

2.4 Lernkurve

Technologien sind unterschiedlich schwer zu lernen. Um die Lernkurve zu bestimmen, werden die Erfahrungen verschiedener Entwickler einbezogen. Ein leichterer Einstieg, führt zu einem besseren Ergebnis in dieser Kategorie, da eine flachere Lernkurve zu mehr Anwendern und einem größeren Komfort führen kann.

3 Beispielanwendung

Um die JavaScript-Technologien an einem praktischen Beispiel vergleichen zu können, wird die gleiche Webanwendung mit den drei Technologien gebaut. Daran soll sich erkennen lassen, wie konkrete, typische Anwendungsfälle jeweils umgesetzt werden können und wie der Aufbau der Projekte sich unterscheidet.

3.1 Anforderung

Anforderung an die Webanwendung ist eine dynamische Seite zu erstellen, in der die Inhalte sich nach Anfragen des Nutzers verändern können und gerendert werden, verglichen mit einer statischen Webseite, in der die Inhalte unverändert bleiben. Die Anwendung soll dabei Anfragen mittels HTTP an einen Endpunkt senden und die erhaltene Antwort auf der Seite rendern. Zusätzlich soll der Nutzer interagieren können und gewisse Elemente sollen nur unter bestimmten Bedingungen gerendert werden, während andere Elemente mehrfach hintereinander gerendert werden sollen. Über Routing soll zwischen verschiedenen Seiten gewechselt werden.
Als Beispielanwendung wird eine Rezeptseite erstellt. Über ein Suchfeld kann nach einem Rezept gesucht werden. Bei erfolgreicher Suche werden alle Ergebnisse angezeigt. Diese können ausgewählt werden, um eine Detailansicht des Rezepts zu erhalten. In der Detailansicht werden neben Daten, wie benötigte Kochzeit, die Portionsmenge, alle Zutaten und eine Anleitung für die Zubereitung zu sehen sein.

3.2 Architektur der Anwendung

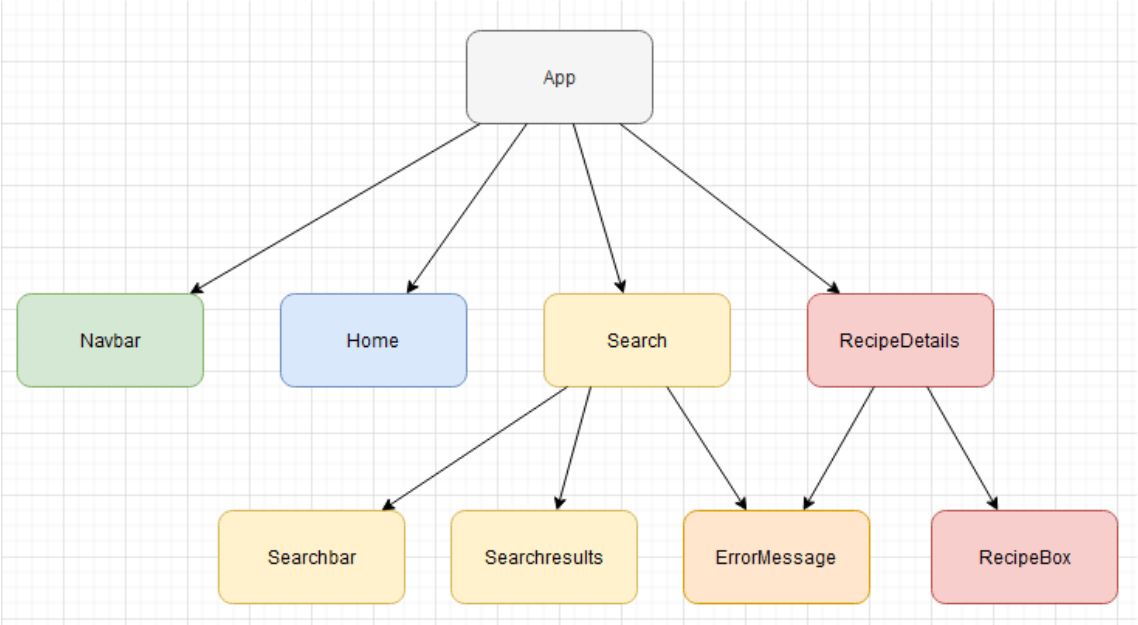

Abbildung 1: Architektur der Anwendung

Die Webanwendung ist in neun Komponenten aufgeteilt. Jede Komponente repräsentiert hierbei einen unabhängigen und wiederverwendbaren Teil der Benutzeroberfläche. Eine Komponente enthält die Benutzeroberfläche, die sie darstellen soll und gegebenenfalls Logik. (Hartmann & Zeigermann, 2019, S. 23) Die Hauptkomponente ist die „App"-Komponente. Diese enthält alle weiteren Komponenten. (Hartmann & Zeigermann, 2019, S. 85)

Die „Navbar"-Komponente ist eine Leiste oberhalb der Webseite, diese enthält Verlinkungen zu den anderen Komponenten.

Die „Home"-Komponente ist die Startseite, der Anwendungen, welche einen Informationstext über die Webseite enthält.

Die „Search"-Komponente ist die Suchseite, sie enthält zusätzlich eine Komponente für die Suchleiste und eine Komponente, um alle Suchergebnisse darzustellen. Sollte etwas bei der Suche nicht funktionieren, wird eine Fehlermeldung angezeigt.

Die „RecipeDetails"-Komponente stellt alle Details eines ausgewählten Rezeptes dar; auch hier wird eine Fehlermeldung dargestellt, sollte ein Fehler passiert sein.

3.3 spoonacular API

Um die Daten für die Rezepte zu erhalten, werden Anfragen an die „spoonacular API“ gesendet. Über diese API kann nach Rezepten über den Namen oder Eigenschaften wie „Vegetarisch“ oder „Vegan“ gesucht werden. (spoonacular, o. D.) Dafür muss eine HTTP-Anfrage an den entsprechenden Endpunkt gesendet werden. Die Antwort wird in Form eines oder mehrerer Rezepte im JSON-Format erhalten. (spoonacular, o. D.)

4 React

4.1 Allgemeines

„React“ ist eine Open Source Bibliothek zur Erstellung von Single-Page-Webanwendungen. (Hartmann & Zeigermann, 2019, S. 3) Entwickelt wurde „React“ von „Meta Platforms, Inc.“ (damals „TheFacebook, Inc.“) und wurde auch zur Erstellung der Benutzeroberfläche für „facebook.com“ und „Instagram“ verwendet, aber auch in der Entwicklung von „Netflix“, „Airbnb“, „Twitter“ und „Wall Street Journal“ eingesetzt. (Hartmann & Zeigermann, 2019, S. 3) Die erste für die Öffentlichkeit zugängliche Version war die Version 0.3.0, welche am 29. Mai 2013 erschienen ist. (javatpoint, o. D.) Die für August 2022 aktuelle Version ist die Version 18.2.0, welche am 14. Juni 2022 erschienen ist. (facebook1, 2022)
Eine „React“-Anwendung ist dabei in einzelne Komponenten eingeteilt. Um eine Komponente darzustellen, bietet „React“ eine Render-Methode, welche die Komponente in das HTML einfügt. (Meta1, o. D.)

4.1.1 Aufsetzen eines Projekts

Um mit „React“ eine Anwendung zu erstellen, müssen einige Vorbereitungen getroffen werden. Es gibt zwei Möglichkeiten, mit der „React“ in einer Anwendung verwendet werden kann. Einerseits am Ende des Bodys der HTML-Datei zwei Skripte einbinden. (Meta2, o. D.)

index.html

```
<body>
…
<script src="https://unpkg.com/react@18/umd/react.development.js"
crossorigin></script>
<script src="https://unpkg.com/react-dom@18/umd/react-
dom.development.js" crossorigin></script>
</body>
```

Quelltext 1.1: Einbinden der Skripte in „React“

Danach kann auf die Methoden von „React“ zugegriffen werden. (Meta2, o. D.)
Der jedoch vom Entwicklerteam empfohlene Weg, um eine bestmögliche Erfahrung für Nutzer und Entwickler zu schaffen ist die „Create React App“ zu verwenden. Dafür muss „Node.js“ und der „Node Package Manager“ installiert sein. (Meta3, o. D.)

Mit dem Befehl „npx create-react-app“ in der Konsole wird automatisch ein neues Projekt mit allen benötigten Bibliotheken erstellt. (Meta3, o. D.)

4.1.2 JSX

Um die Benutzeroberfläche zu erstellen, empfiehlt „React“ die Spracherweiterung JSX, welche an eine Template Sprache erinnern kann. (Meta4, o. D.) Diese Spracherweiterung ermöglicht es, HTML-ähnlichen Code direkt in JavaScript zu schreiben. (Hartmann & Zeigermann, 2019, S. 63) JSX bietet Vorteile, wie beispielsweise das Verwenden von Variablen. (Meta4, o. D.)

```
let technologie = "react";
return(
       <h1>Ich programmiere in {technologie}</h1>
    )
```

Quelltext 1.2: Beispiel für JSX

Innerhalb der geschweiften Klammern können aber auch alle anderen JavaScript Ausdrücke verwendet werden. (Meta4, o. D.)
Es ist jedoch keine Pflicht, die Spracherweiterung zu verwenden. JSX-Elemente sind syntaktischer Zucker für die „createElement“-Methode der „React“-Bibliothek, daher kann der Code auch ohne die Verwendung von JSX geschrieben werden. (Meta5, o.D.)

4.1.3 Babel

Wenn JSX in einem Projekt verwendet wird, wird zusätzlich ein JavaScript-Compiler benötigt, da kein Webbrowser standardmäßig JSX versteht. (Hartmann & Zeigermann, 2019, S. 9) In „React“ wird dafür „Babel“ verwendet. (Meta2, o. D.) „Babel“ konvertiert ECMAScript 2015+ Code in eine kompatible Version, damit dieser auch in älteren Umgebungen lauffähig ist und kann zusätzlich auch JSX konvertieren. (Babel, o. D.)

4.2 Aufbau eines Projekts

Wenn die „Create React App“ verwendet wird, wird automatisch ein Projekt mit grundlegenden Dateien angelegt. Werkzeuge wie „Babel“ werden dabei auch installiert und konfiguriert. (facebook2, o. D.)
Die Projektstruktur mit den wichtigsten Dateien sieht folgendermaßen aus:

```
my-app
├--- node_modules
├--- package.json
├---- public
│   ├---- favicon.ico
│   ├---- index.html
└─── src
   ├---- App.js
   ├---- index.js
```

(facebook2, o. D.)

Die index.html Datei enthält, neben einigen Metadaten, einen Container mit der ID „root“.

In diesem Container werden alle Komponenten von „React“ eingesetzt. (Meta1, o. D.) Das Aufrufen der „render“-Funktion der ReactDOM-Bibliothek geschieht in der index.js.

index.js

```
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
    <App />
);
```

Quelltext 1.3: „render“-Funktion in „React“

Hier wird der Container mit der ID „root“ festgelegt, um dort die „App“-Komponente zu rendern.

App.js enthält die „App“-Komponente. Es ist die Hauptkomponente und enthält alle weiteren Komponenten. (Hartmann & Zeigermann, 2019, S. 85)

4.3 Komponenten erstellen

In „React“ gibt es zwei verschiedene Arten von Komponenten, Klassenkomponenten und Funktionskomponenten. (Meta1, o. D.) Eine Funktionskomponente ist hierbei eine JavaScript-Funktion, welche die Benutzeroberfläche im JSX-Format zurückgibt. Eine Klassenkomponente ist eine ES6-Klasse, welche von „React.Component“ erbt und eine Render-Funktion enthält, die das JSX zurückgibt. (Meta1, o. D.) Im Folgenden wird gezeigt, wie die „Home“-Komponente der Beispielanwendung sowohl als Funktionskomponente als auch Klassenkomponente aussieht.

```
//Funktionskomponente
export default function Home() {
    return(
        <div>
            <p>Discover over 5000 recipes <br/>
            Start your culinary journey now</p>
        </div>
    )
}
```

Quelltext 1.4: Funktionskomponente in „React“

```
//Klassenkomponente
class Home extends React.Component {
    render() {
        return(
            <div>
                <p>Discover over 5000 recipes <br/>
                Start your culinary journey now</p>
            </div>
        )
    }
}
```

Quelltext 1.5: Klassenkomponente in „React“

Vor der Version 16.8 wurden Klassenkomponenten verwendet, wenn eine Komponente einen internen Zustand verwalten muss, da dies Funktionskomponenten zu der Zeit nicht konnten. Mit der „Hooks“-API können Funktionskomponenten nahezu alles, was zuvor nur Klassenkomponenten vorbehalten war. (Hartmann & Zeigermann, 2019, S. 26) Daher sind sie mittlerweile der empfohlene Weg, eine Komponente zu erstellen. (Jöch, 2018)

4.4 Bedingtes Rendern

Manche Komponenten in der Beispielanwendung sollen nur unter bestimmten Bedingungen dargestellt werden, beispielsweise wenn ein bestimmtes Ereignis eingetreten ist. Da JSX eine Spracherweiterung von JavaScript ist, können „if“-Anweisungen verwendet werden, diese können auch mit dem „Und-Operator“ abgekürzt werden. (Meta6, o. D.) In der Beispielanwendung soll die „Search-Komponente“ ein Suchfeld darstellen. Dieses soll jedoch nur gerendert werden, solange ein bestimmter Knopf nicht geklickt wurde. Die Variable „clicked“ repräsentiert, ob der Knopf bereits geklickt wurde.

Search.js

```
import React, { useState } from "react";
…
export default function Search() {
    const [clicked, setClicked] = useState(false);
    …
    return (
            {!clicked && <SearchField />}
            …
    )
}
```

Quelltext 1.6: Bedingtes Rendern in „React“

Die „SearchField-Komponente“ wird nur dargestellt, wenn die Variable auf „falsch“ gesetzt ist.
Eine „if-else“-Anweisung kann als ternärer Operator abgekürzt werden, um einen schlankeren Code zu erhalten. (Meta6, o. D.) Die Komponente, die alle Details des gewünschten Rezeptes anzeigt, soll, wenn ein Rezept „Vegetarisch“ ist, den Text „Yes“ darstellen, ansonsten ein „No“.

RecipeBox.js

```
import React from "react";
…
export default function RecipeBox(props) {
    …
    return (
        <p>Vegetarian: {props.vegetarian ? 'Yes' : 'No'}</p>
    )
}
```

Quelltext 1.7: Ternärer Operator in „React“

Ist die Aussage im linken Teil des ternären Operators „wahr“, wird ein „Yes“ dargestellt, ansonsten ein „No“.

4.5 Wiederholtes Rendern

Wenn bestimmte Komponenten mehrfach hintereinander in einer Schleife gerendert werden sollen, können dafür schon vorhandene JavaScript-Funktionen im JSX verwendet werden. (Meta7, o. D.) Die „map“-Funktion kann auf eine Liste angewendet werden und nimmt eine Funktion als Parameter an. Die Funktion, die

als Parameter übergeben wurde, wird für jedes Element der Liste angewendet. (MDN1, 2022)
In der Beispielanwendung soll nachdem nach einem Rezept gesucht wurde, jedes Suchergebnis in einer Box gerendert werden. Die Suchergebnisse sind Objekte, die zuvor in der Liste „result“ gespeichert wurden.

Search.js

```
import React from "react";
…
export default function Search() {
   …
  return (
   …
  {results.map(result => {
     return <SearchResults image={result.image} title={result.title}/>
   })}
    )
}
```

Quelltext 1.8: Wiederholtes Rendern in „React“

Die „map“-Funktion gibt hier für jedes Element der Liste eine Komponente zurück und übergibt als Eigenschaften das zugehörige Bild und den Titel. Alle Komponenten werden nacheinander gerendert.

4.6 Kommunikation zwischen Komponenten

In „React“ können Komponenten miteinander Daten austauschen, wenn diese in einer „Eltern-Kind-Beziehung“ zueinanderstehen. Die „Eltern“-Komponente enthält eine Komponente, dies ist die „Kind“-Komponente. (Trent, 2019)

4.6.1 Von Eltern-Komponente zur Kind-Komponente

Wenn Daten aus der „Eltern“-Komponente an einer der „Kind“-Komponenten weitergegeben werden sollen, wird das über Eigenschaften gemacht. (Trent, 2019)
Die „Kind“-Komponente kann im Funktionskopf die Eigenschaften als Argument übergeben bekommen und die innerhalb der Funktion verwenden. In der Beispielanwendung haben die „Search“-Komponente und die „SearchResult“-Komponente eine „Eltern-Kind-Beziehung“. Die „SearchResult“-Komponente soll von seiner „Eltern“-Komponente als Eigenschaften den Titel und das Bild des darzustellenden Rezeptes erhalten.

SearchResults.js

```
import React from "react";
…
export default function SearchResults(props) {
    return (
            <div className="result-box">
                <img src={props.image} alt="Bild des Essens"></img>
                <p>{props.title}</p>
            </div>
    )
}
```

Quelltext 1.9: Verwenden von Eigenschaften in „React“

Die „SearchResult“-Komponente erhält als Eigenschaften ein Objekt, welches ein Bild und einen Titel enthält. Diese werden für die dynamische Darstellung der Komponente verwendet.
Die „Eltern“-Komponente übergibt dem „Kind“ die Eigenschaften.

Search.js

```
import React, { useState } from "react";
…

export default function Search() {
    const [results, setResults] = useState([]);
    …
    return (
        <SearchResults image={result.image} title={result.title} />
        …
    )
}
```

Quelltext 1.10: Übergeben von Eigenschaften an die „Kind“-Komponente in „React“

4.6.2 Von Kind-Komponente zur Eltern-Komponente

Für den umgekehrten Fall, dass die „Kind“-Komponente mit seiner „Eltern“-Komponente kommunizieren soll, muss die „Eltern“-Komponente eine Rückruffunktion der „Kind“-Komponente übergeben. (Trent, 2019) In der Beispielanwendung enthält die „SearchField“-Komponente ein Eingabefeld. Der Wert, der in dem Eingabefeld steht, soll an die „Eltern“-Komponente weitergegeben werden, damit diese den Wert weiterverarbeiten kann.

Search.js

```
import React, { useState } from "react";
…

export default function Search() {
    const [search, setSearch] = useState();

    function handleChange(value) {
        setSearch(value);
    }

    return (
        {!clicked && <SearchField onChange={handleChange} />}
        …
    )
}
```

Quelltext 1.11: Übergeben einer Rückruffunktion in „React“

Die „Search“-Komponente übergibt seiner „Kind“-Komponente die „handleChange“-Funktion, diese wird beim „onChange“-Event aufgerufen.
Die „SearchField“-Komponente, welche in diesem Fall die „Kind“-Komponente ist, kann in seinen Eigenschaften auf das „onChange“-Event zugreifen und aufrufen.

SearchField.js

```
import React from "react";
…
export default function SearchField(props) {

    function handleChange(event) {
    //hier wird die übergebene onChange-Funktion mit dem Inhalt des
    // Suchfeldes aufgerufen
        props.onChange(event.target.value);
    }

    return (
            …
            <input type="text" onChange={handleChange}></input>
    )
}
```

Quelltext 1.12: „Kind“-Komponente benachrichtigt „Eltern“-Komponente in „React“

Jedes Mal, wenn das „onChange“-Event im Input-Feld geschehen ist, wird die „handleChange“-Funktion der „Kind“-Komponente aufgerufen. Die gibt den Wert des Input-Feldes weiter an die „Eltern“-Komponente.

4.7 Routing

4.7.1 Installation und Implementierung

In der Beispielanwendung soll zwischen verschiedenen Seiten gewechselt werden können. In der Navigationsleiste sollen verschiedene Links anklickbar sein, mit denen auf die zugehörige Seite gewechselt wird. Um das umzusetzen, kann Routing verwendet werden. (Remix1, o. D.) „React“ ist eine Bibliothek zum Erstellen von Benutzeroberflächen und bietet keine Funktionalitäten an, um Routing umzusetzen. Es existieren jedoch Routing-Bibliotheken, welche von „React“-Nutzern erstellt wurden. Die Meistverwendete ist „React Router“, welche mit dem „Node Package Manager“ und dem Befehl „npm install -D react-router-dom“ installiert werden kann. (w3schools, o. D.)

Um eigene Routen zu definieren und diese an Komponenten zu binden, muss zuerst in der index.js die „BrowserRouter“-Komponente importiert werden und als „Eltern“-Komponente der „App“-Komponente eingesetzt werden. (Remix1, o. D.)

index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App';
import { BrowserRouter } from "react-router-dom"

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
    <BrowserRouter>
        <App />
    </BrowserRouter>
);
```

Quelltext 1.13: Verwenden des „BrowserRouter“ in „React“

Nun können in der „App“-Komponente mit dem „Routes“-Element und den Unterelementen „Route“ die Routen festgelegt werden. Zu jeder Route wird eine Komponente zugewiesen, welche geladen wird, wenn die URL übereinstimmt. (Remix1, o. D.)

App.js

```
import React from "react";
import { Route, Routes } from "react-router-dom"
import Home from "./sites/Home";
import Search from "./sites/Search";

function App() {
  return (
    <>
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/search" element={<Search />} />
      </Routes>
    </>
  );
}
```

Quelltext 1.14: Definieren von Routen in „React“

Zum Schluss können in der Navigationsleiste die Links zu anderen Seiten hinzugefügt werden. Dafür wird das „Link“-Element aus der „React Router“-Bibliothek verwendet. (Remix1, o. D.)

Navbar.js

```
import React from "react";
import { Link } from "react-router-dom";

export default function Navbar() {

    return(
        <>
            <header>
                <Link to="/" id="logo">
                    <h1>Dinely</h1>
                </Link>
                <nav>
                    <Link to="/search">Search</Link>
                </nav>
            </header>
        </>
    )
}
```

Quelltext 1.15: Hinzufügen von Verlinkungen in „React“

Beim Anklicken eines Links wird die URL verändert und die zugehörige Komponente wird gerendert.

4.7.2 Routing mit Parametern

Auf der „RecipeDetail“-Seite befindet sich eine „RecipeBox“-Komponente. Die „RecipeBox“-Komponente soll dynamisch ein Rezept übergeben bekommen, welches mit allen Details, wie Titel, Zutaten, Portionen und mehr dargestellt werden soll. Welches Rezept genau gerade dargestellt wird, soll über die URL bestimmt werden. Die URL enthält entweder die ID eines Rezeptes oder eine Kategorie wie „vegetarian“. Um das umsetzen zu können, wird Routing mit Parametern verwendet. In den Routen in der „App“-Komponente kann im Pfad angegeben werden, dass zusätzlich ein Parameter erwartet wird. (Remix2, o. D.)

App.js

```
<Route path="/recipeDetails/:id" element={<RecipeDetails />} />
```

Quelltext 1.16: Angeben eines Pfads mit Parametern in „React“

Auf den Parameter kann in der zugehörigen Komponente zugegriffen werden, indem ein „Hook“ der „React Router“-Bibliothek verwendet wird. Der „useParams-Hook“ kann dekonstruiert werden, um auf die Variable „id“ des Hooks zuzugreifen. (Remix3, o. D.)

RecipeDetails.js

```
import React from "react";
import { useParams } from "react-router-dom";
…
export default function RecipeDetails() {
    const  { id } = useParams();
…
}
```

Quelltext 1.17: Zugreifen auf die Parameter in „React“

Die Variable speichert als Wert die aktuellen Parameter des Pfades und aktualisiert sich, sobald diese sich verändern. Die Parameter können in der „RecipeDetail“-Komponente verwendet werden, um festzulegen, welches Rezept dargestellt werden soll.

4.8 HTTP-Anfragen senden

Zum Schluss müssen noch die Rezepte von der „spoonacular API“ abgefragt werden. dafür muss mit dem entsprechenden Endpunkt kommuniziert werden. Die Anfrage erfolgt über ein HTTP „GET-Request“. „React“ bietet für das Versenden von http-Anfragen keine Funktionen an. Daher muss zusätzlich eine weitere Bibliothek installiert werden. Eine beliebte Option ist „axios“ welche ebenfalls mit dem „Node Package Manager“ installiert werden kann. (Barger, 2021) Nach der Installation kann das „axios“-Objekt in die Dateien importiert werden, in denen es benötigt wird. Mit dem „axios“-Objekt können Methoden, wie „get“ oder „delete“ für HTTP-Anfragen verwendet werden. (Barger, 2021) Um Rezepte nach Namen zu suchen, muss eine Anfrage, an einen bestimmten Endpunkt gesendet werden.
Die erhaltene Antwort kann als Zustand gespeichert werden und an die Komponenten weitergegeben werden. Im Falle der Beispielanwendung wird die Anfrage in einer Methode namens „searching“ durchgeführt. Diese Methode wird ausgeführt, wenn in der „Kind“-Komponente das „onClick“-Event passiert ist.

Search.js

```
import React, { useState } from "react";
import axios from "axios";
import SearchField from "../components/SearchField";
…

export default function Search() {
    const [search, setSearch] = useState();
    const [results, setResults] = useState([]);
…

    function searching() {
        axios.get(`https://api.spoonacular.com/recipes/complexSearch?qu
ery=${search}`)
        .then(res => {
            setResults(res.data.results);
        })
    }

    return (
        <>
           <SearchField onClick={searching}/>
           …
      <>
    )
}
```

Quelltext 1.18: Senden von HTTP-Anfragen in „React“

Damit ist die Suche implementiert.
Die zweite Komponente, die Anfragen an die „spoonacular API“ senden soll ist die „RecipeDetails“-Komponente, welche anhand der Parameter der URL eine andere Anfrage senden soll. Die Anfrage soll gesendet werden, sobald die Komponente erstellt wurde. Dafür kann ein Hook verwendet werden. Der „useEffect-Hook“ wird ausgeführt, nachdem eine Komponente gerendert wurde und ist ein sogenannter Seiteneffekt. (Meta8, o. D.) Im „useEffect-Hook“ wird auf die Parameter der URL zugegriffen und eine Anfrage gesendet. Die Antwort wird als Zustand gespeichert und kann an die „RecipeBox“-Komponente weitergegeben werden.

RecipeDetails.js

```
import React, { useEffect, useState } from "react";
import { useParams } from "react-router-dom";
import RecipeBox from "../components/RecipeBox";
import axios from "axios";

export default function RecipeDetails() {
    const [recipe, setRecipe] = useState();
    const  { id } = useParams();

    useEffect(() => {
     axios.get(`https://api.spoonacular.com/recipes/${id}/information`)
        .then(res => {
             const recipeData = res.data;
             setRecipe(recipeData)
         })
    }, [id])

    return(
        <>
            <RecipeBox recipe={recipe} />
        </>
    )
}
```

Quelltext 1.19: Verwenden des „useEffect-Hooks“ in „React“

5 Vue.js

5.1 Allgemeines

„Vue.js“ ist ein JavaScript-Framework, welches für die Erstellung von Webanwendungen verwendet wird. Ähnlich wie in „React“ werden Anwendungen komponentenorientiert entwickelt. Besonders geworben wird mit der Anpassungsfähigkeit und dem übersichtlichen Code. (Hecht et al., 2019, S. 6) „Vue.js“ wurde anders als „React“ oder „Angular“ anfangs nur von einer Person entwickelt. (Hecht et al., 2019, S. 8) Der Entwickler, Evan Yu, hatte zuvor im Entwicklungsteam von „Angular“ mitgearbeitet und wollte dabei den Teil, der ihn besonders interessierte, das Binden von Daten, in einem eigenen Framework umsetzen. Im Februar 2014 wurde die erste Version veröffentlicht. (Hecht et al., 2019, S. 9)

5.1.1 Aufsetzen eines Projekts

Ähnlich wie in „React“ existieren zwei Wege „Vue.js“ in einem Projekt zu verwenden. Die erste Methode ist im Body des HTML-Dokuments ein Skript einzubinden. Das ermöglicht, das globale Vue-Objekt im Projekt zu verwenden. (Hecht et al., 2019, S. 10) Das Vue-Objekt enthält die benötigten Funktionen, um Komponenten zu erstellen. (Hecht et al., 2019, S. 11)
Auf diesen Weg kann „Vue.js“ in bereits existierenden Projekten, welche nicht auf „Vue.js“ basieren, erweitert werden. Hierbei ist keine Installation nötig. (vue1, o. D.) Für den zweiten Weg wird „Vue.js“ mit dem „Node Package Manager“ installiert. Mit dem Befehl „npm init vue@latest“ in der Kommandozeile wird der Installationsvorgang gestartet. (vue1, o. D.) Anders als in „React“, welches eine Bibliothek ist, werden bei „Vue.js“ einige Fragen gestellt, um alles zu konfigurieren. Diese Fragen beinhalten unter anderem das Einrichten von TypeScript und das Hinzufügen von Routing.

5.1.2 Template-Sprache

„Vue.js“ verwendet, um die Benutzeroberfläche zu erstellen, eine Template-Sprache, welche auf HTML basiert. Das Template wird in JavaScript übersetzt. Im Template kann durch das Verwenden von zwei geschweiften Klammern JavaScript-Ausdrücke verwendet werden und auf Variablen zugegriffen werden. (vue2, o. D.)

```
<p>{{ 1 + 1 }}</p>
<p>{{ message.toUpperCase() }}</p>
```

Quelltext 2.1: Template-Sprache von „Vue.js“

Neben dem Verwenden von JavaScript innerhalb des HTML-Codes, bietet die Template-Sprache Erweiterungen der Syntax, um Daten zu binden. (vue2, o. D.)

5.2 Aufbau eines Projekts

Nachdem ein Projekt mit dem „Node Package Manager“ erstellt wurde, ist folgende Ordnerstruktur mit den wichtigsten Dateien vorzufinden:

```
my-app
├--- node_modules
├--- package.json
├---- public
│   ├---- favicon.ico
│   ├---- index.html
└─── src
    ├---- App.vue
    ├---- main.js
```

Die index.html-Datei ist ein fertig generiertes HTML-Dokumenten, welches im Body einen Container mit der ID „app“ enthält. In diesen Container werden die Komponenten von „Vue.js“ eingesetzt. Dies geschieht über die main.js. In der main.js wird der Container ausgewählt, an dem die „App“-Komponente gebunden wird.

main.js

```
import { createApp } from 'vue';
import App from './App.vue';

createApp(App).mount('#app');
```

Quelltext 2.2: Binden der „App“-Komponente in „Vue.js“

Die „App“-Komponente wird in einer eigenen Datei implementiert und kann weitere Komponente enthalten.

Die Beziehungen zusammengefasst:

index.html enthält einen Container mit der ID „app“ → main.js bindet die „App“-Komponente an den Container → App.vue ist die Implementierung der „App“-Komponente und kann weitere Komponenten enthalten.

5.3 Komponente erstellen

Der empfohlene Weg, Komponenten zu erstellen, sind Single-File Komponenten. In Single-File Komponenten wird jede Komponente in einer eigenen Datei beschrieben. Die Dateiendung ist „.vue“. Die Datei ist in drei Abschnitten aufgeteilt: Template, Style und Script. (vue3, o. D.) Im Template wird die Struktur der Benutzeroberfläche beschrieben. Im Style-Abschnitt kann lokales CSS für das Template definiert werden. Zum Schluss wird die Programmierlogik im Script implementiert. (vue3, o. D.)

HomeView.vue

```
<template>
    <div class="content-text">
        <p>Discover over 5000 recipes <br/>
        Start your culinary journey now</p>
    </div>
</template>

<script>
export default {
  name: 'HomeView'
}
</script>

<style scoped>
  @import '../styles/HomeView.css';
</style>
```

Quelltext 2.3: „HomeView“-Komponente in „Vue.js“

5.4 Bedingtes Rendern

„Vue.js“ stellt für den Fall, dass ein Element nur unter bestimmten Bedingungen dargestellt werden soll, die „v-if“ und die „v-else“ Direktiven bereit. (vue4, o. D.) Elemente, die eine „v-if“ Direktive beinhalten, werden nur dargestellt, wenn der Ausdruck innerhalb der „v-if“ Direktive als „wahr“ ausgewertet wird. Wenn der Ausdruck als „falsch“ ausgewertet wird, aber unmittelbar darunter eine „v-else“ Direktive folgt, wird dieses Element stattdessen dargestellt. (vue4, o. D.)
In der Beispielanwendung soll in der Box, welche die Details des Rezeptes darstellt, angezeigt werden, ob ein Rezept „Vegetarisch“ ist oder nicht. Wenn das Attribut „vegetarian“ den Wert „wahr“ hat, soll der Text „Yes“ gerendert werden, ansonten „No“.

RecipeBox.vue

```
…
<p v-if="recipe.vegetarian">Vegetarian: Yes</p>
<p v-else>Vegetarian: No</p>
…
```

Quelltext 2.4: Bedingtes Rendern in „Vue.js“

5.5 Wiederholtes Rendern

Auch für das wiederholte Rendern einer Komponente bietet „Vue.js“ eine Direktive an. Die „v-for“ Anweisung funktioniert wie eine Schleife in JavaScript. Der Schleife wird eine Liste übergeben und für jedes Element der Liste wird das Element gerendert. (vue5, o. D.)
Erneut wird implementiert, dass in der Beispielanwendung nach einer erfolgreichen Suche eines Rezeptes alle Ergebnisse hintereinander gerendert werden sollen. Die Rezepte werden zuvor in einer Liste namens „recipes“ gespeichert.

SearchView.vue

```
…
<template>
…
    <SearchResult v-for="recipe in recipes"
        :title="recipe.title" :image="recipe.image" :id="recipe.id"/>
</template>
```

Quelltext 2.5: Wiederholtes Rendern in „Vue.js“

5.6 Kommunikation zwischen Komponenten

5.6.1 Von Eltern-Komponente zur Kind-Komponente

Das Weitergeben von Daten der „Eltern“-Komponente an die „Kind“-Komponente funktioniert ähnlich wie in „React“. Die „Eltern“-Komponente übergibt die Daten als Eigenschaften an die „Kind“-Komponente. Die „Kind“-Komponente muss explizit alle Eigenschaften, die sie annehmen kann, in der „props“ Option der Komponente deklarieren. (vue6, o. D.) Auch hier erhält die „SearchResult“-Komponente von seiner „Eltern“-Komponente den Titel und das Bild des darzustellenden Rezepts. Danach können die Eigenschaften im Template verwendet werden. (vue6, o. D.)

SearchResult.vue

```
<script>
    export default {
        name: "SearchResult",
        props: {
            title: String,
            image: String,
        }
    }
</script>
<template>
    <div class="result-box">
        <img :src=image alt="Bild des Essens" />
        <p>{{title}}</p>
    </div>
</template>
```

Quelltext 2.6: Verwenden von Eigenschaften in „Vue.js“

Die „Eltern“-Komponente kann im Template der Komponente die Eigenschaften übergeben, die sie benötigt. Für das dynamische Binden von Eigenschaften, also wenn sich die Werte potenziell verändern können, kann die „v-bind“ Direktive verwendet werden. Damit wird die „Kind“-Komponente neu geladen, wenn sich der Wert von einer der Eigenschaften verändert. (vue6, o. D.)

SearchView.vue

```
…
<SearchResult v-for="recipe in recipes"
:title="recipe.title" :image="recipe.image" />
…
```

Quelltext 2.7: Übergeben von Eigenschaften in „Vue.js“

5.6.2 Von Kind-Komponente zur Eltern-Komponente

Der umgekehrte Fall, dass die „Kind“-Komponente mit der „Eltern“-Komponente kommunizieren soll, wird wie in „React“ mit Events umgesetzt. „Vue.js“ hat eine eingebaute Methode namens „emit“, diese kann im Template bei einem Event ausgeführt werden. (vue7, o. D.) In der Beispielanwendung sendet die „SearchField“-Komponente den Wert des Eingabefelds an die „Eltern“-Komponente. Die „Kind“-Komponente hat als Attribut eine Liste an Events, welche es auslösen kann. Im Falle der „SearchField“-Komponente heißt das Event „searching“. Wird der Such-Knopf gedrückt, wird eine Methode, namens „handleEvent“ aufgerufen, welche die „emit“-Methode aufruft und somit das „searching“-Event auslöst.

SearchField.vue

```
<template>
    <div class="search-wrapper">
        <p>Find the perfect recipe</p>
        <input type="text" v-model="inputText" />
        <button @click="handleEvent">Search</button>
    </div>
</template>
<script>
    export default {
        name: 'SearchField',
        data() {
            return {
                inputText: null
            }
        },
        emits: ['searching'],
        methods: {
            handleEvent() {
                this.$emit('searching', this.inputText);
            }
        }
    }
</script>
```

Quelltext 2.8: „Kind“-Komponente benachrichtigt „Eltern“-Komponente in „Vue.js“

Die „Eltern“-Komponente kann mit der „v-on“ Direktive auf das Event hören und sobald es auslöst, wird, wird eine eigene Methode aufgerufen. (vue7, o. D.)

SearchView.vue

```
<template>
    <SearchField @searching="searching"/>
</template>
<script>
    import SearchField from '@/components/SearchField.vue';
    export default {
        name: 'SearchView',
        components: {
            SearchField,
        },
        methods: {
            searching(inputText) {
                //Verarbeiten der Daten
            }
        }
    }
</script>
```

Quelltext 2.9: „Eltern“-Komponente hört auf das Event der „Kind“-Komponente in „Vue.js“

5.7 Routing

5.7.1 Installation und Implementierung

Anders als in „React“ bietet „Vue.js“ eine offizielle Routing-Bibliothek an, welche Komponenten entsprechender der URL rendert. (vue8, o. D.) Wenn ein neues Vue-Projekt mit dem „Node Package Manager“ erstellt wird, kann eingestellt werden, dass die Routing-Bibliothek ebenfalls installiert und eingebunden wird. Ansonsten kann auch zu einem späteren Zeitpunkt die Bibliothek mit dem „Node Package Manager“ und dem Befehl „npm install vue-router@4“ installiert werden. (vue9, o. D.)

Nachdem die Bibliothek installiert wurde, muss in der main.js beim Erstellen der Anwendung angegeben werden, dass der Router verwendet wird. (vue10, o. D.)

main.js

```
import { createApp } from 'vue';
import App from './App.vue';
import router from './router';

createApp(App)
.use(router)
.mount('#app');
```

Quelltext 2.10: Verwenden des Routers in „Vue.js“

Dies ermöglicht den Zugriff auf die Komponenten, der Router-Bibliothek.

Durch die Installation der Bibliothek, wird ein neuer Ordner namens „router“ im Projekt generiert. Dieser enthält eine JavaScript-Datei, die den Namen „index.js“ trägt. In dieser Datei werden alle Routen mit der zugehörigen Komponente festgelegt. (vue10, o. D.)

index.js

```
…
const routes = [
  {
    path: '/',
    name: 'home',
    component: HomeView
  },
  {
    path: '/search',
    name: 'search',
    component: SearchView
  }
]
```

Quelltext 2.11: Definieren von Routen in „Vue.js“

In der „App“-Komponente wird eine Komponente namens „router-view“ ergänzt, diese Komponente stellt die in der index.js festgelegt Komponenten dar, wenn die URL übereinstimmt. So wird beispielsweise im Pfad „/home“ die Home-Komponente gezeigt. (vue10, o. D.)

App.vue

```
<template>
  <NavBar />
  <router-view/>
</template>
```

Quelltext 2.12: Einsetzen des „Router-Views“ in „Vue.js“

Die Navigation erfolgt in der Navigationsleiste mit dem „router-link“-Element der Router-Bibliothek kann ein Pfad angegeben werden, zu dem das Element führt. Die URL verändert sich beim Anklicken und die entsprechende Komponente wird dargestellt. (vue10, o. D.)

Navbar.vue

```
<template>
    <header>
        <router-link to="/">
            <h1>Dinely</h1>
        </router-link>
        <nav>
            <router-link to="/search">Search</router-link>
        </nav>
    </header>
</template>
```

Quelltext 2.13: Einsetzen der Verlinkungen in „Vue.js“

5.7.2 Routing mit Parametern

Um Routing mit Parametern zu implementieren, wird in den Routen, die zusätzlich Parameter annehmen, eine ID hinzugefügt. (vue11, o. D.)

index.js

```
const routes = [
…
  {
    path: '/recipeDetails/:id',
    name: 'recipeDetails',
    component: RecipeView
  }
]
```

Quelltext 2.14: Definieren einer Route mit Parametern in „Vue.js“

Die Parameter können danach in den Komponenten mit dem Router ausgelesen werden und in einer Variable gespeichert werden. (vue11, o. D.)

RecipeView.vue

```
<script>
    export default {
        name: "RecipeView",
        …
        data() {
            return{
                …
                path: this.$route.params.id,
            }
        }
   …
</script>
```

Quelltext 2.15: Zugreifen auf die Parameter in „Vue.js“

Die Variable wird automatisch aktualisiert, sobald sich die Parameter der URL verändern sollte. (vue11, o. D.) Die Parameter werden verwendet, um in der „RecipeView“-Komponente zu entscheiden, welches Rezept dargestellt werden soll.

5.8 HTTP-Anfragen senden

Ähnlich wie in „React“ bietet Vue.js auch keine Methoden zum Senden von HTTP-Anfragen an, daher muss eine weitere Bibliothek installiert werden. Da „axios“ die meistverwendete Option hierfür ist, wird sie auch in der „Vue.js“ Implementierung verwendet. (vue12, o. D.)

Ähnlich wie in „React“, wird im Suchfeld das gewünschte Rezept eingegeben. An den Such-Knopf, wird ein Event gebunden, welches den gespeicherten Wert des Suchfeldes als Parameter annimmt und mit „axios“ die entsprechende HTTP-Anfrage versendet. Die Antwort des Servers wird gespeichert und jedes Ergebnis wird jeweils an die „SearchResult“-Komponente weitergegeben, die die Suchergebnisse darstellen. Die Methoden für die Anfragen an die API werden für eine bessere Struktur in einer separaten Datei namens „Api.js“ implementiert.

SearchView.vue

```
<template>
    <SearchField @searching="searching"/>

    <div class="result-wrapper">
        <SearchResult
         v-for="recipe in recipes"
         :title="recipe.title" :image="recipe.image" :id="recipe.id"
:key="recipe" />
    </div>
</template>
<script>
    import SearchField from '@/components/SearchField.vue';
    import SearchResult from '@/components/SearchResult.vue';
    import { getRecipesByName } from '@/services/Api.js'
    …
    export default {
        name: 'SearchView',
        components: {
            SearchField,
            SearchResult,
            …
        },
        data() {
            return {
                recipes: null,
                …
            }
        },
        methods: {
            searching(inputText) {
                getRecipesByName(inputText).then(res => {
                    this.recipes = res;
            }
           …
        }
    }
</script>
```

Quelltext 2.16: Senden einer HTTP-Anfrage in „Vue.js“

Die „RecipeDetail“-Komponente soll eine Anfrage anhand der Parameter der URL senden, sobald die Komponente geladen wurde. In Vue.js kann ebenfalls in den Lebenszyklus der Komponente zugegriffen werden. Die „created“-Methode einer Komponente wird ausgeführt, wenn diese gerendert wurde. Dort wird die Methode zum Abfragen der Daten aufgerufen. (vue13, o. D.)

RecipeView.vue

```
<template>
    <RecipeBox v-if="recipe" :recipe=recipe />
</template>

<script>
    import RecipeBox from '@/components/RecipeBox.vue';
    …
    export default {
        name: "RecipeView",
        components: {
            RecipeBox,
            …
        },
        data() {
            return{
                recipe: {},
                …
            }
        },
        methods: {
            setRecipe() {
                …
            }
        },

        created() {
            this.setRecipe();
        }
    }
</script>
```

Quelltext 2.17: Verwenden der created-Methode in „Vue.js“

6 Angular

6.1 Allgemeines

„Angular“ ist ein im September 2016 veröffentlichtes TypeScript-Framework. Entwickelt wurde es von Google und es wird weltweit von großen Unternehmen verwendet. (Malcher et al. 2019) Verwechselungsgefahr besteht zu „AngularJS“. „AngularJS“ ist ein im Jahr 2010 erschienenes JavaScript-Framework, welches ebenfalls von Google entwickelt wurde. (Gavigan, 2018) Im Jahr 2016 wurde eine komplett umgeschriebene Version des Frameworks, welches auf TypeScript basiert, veröffentlicht. Die neue Version heißt nur „Angular“ ohne das „JS“. Die aktuelle Version ist 14.2.2, welche am 28. September 2022 erschienen ist. (Angular, 2022)

6.1.1 Angular CLI

„Angular“ bietet eine eigene Kommandozeilenschnittstelle namens „Angular CLI“ an. Mit der „Angular CLI“ können neue Projekte, Module, Komponenten, Services und weiteres erstellt werden, zusätzlich kann darüber die Anwendung gestartet werden. Die Kommandozeilenschnittstelle kann mit dem „Node Package Manager“ installiert werden. (Google1, o. D.)

6.1.2 Aufsetzen eines Projekts

Um ein Projekt zu erstellen, kann die zuvor installierte „Angular CLI“ verwendet werden. (Google1, o. D.) Ähnlich wie bei „Vue.js“ werden bei der Installation einige Fragen gestellt, um das Projekt zu konfigurieren. Hierbei kann ausgewählt werden, ob Routing verwendet werden soll und auch der bevorzugte CSS-Präprozessor kann direkt ausgewählt werden.

6.1.3 TypeScript

TypeScript ist eine von Microsoft entwickelte Programmiersprache, die auf JavaScript aufbaut. (Steyer, 2021, S. 37) JavaScript ist eine dynamisch typisierte Programmiersprache, das bedeutet, der Datentyp einer Variabel wird erst zur Laufzeit bestimmt. (MDN2, 2022) Dies kann dafür sorgen, dass der Programmcode fehleranfälliger ist und manche Fehler erst zur Laufzeit entdeckt werden. (Hurd, 2021) TypeScript ist es eine statisch typisierte Erweiterung von JavaScript. Es bietet

Erweiterungen, wie das Definieren von Datentypen oder das Erstellen von Interfaces. (Steyer, 2021, S. 53) Da TypeScript eine Obermenge von JavaScript ist, ist jeder valider JavaScript-Code auch valider TypeScript-Code, aber nicht umgekehrt. (Steyer, 2021, S. 37) Im Folgenden sind Beispiele zu sehen, die die Unterschiede beim Programmieren mit JavaScript verglichen zu TypeScript verdeutlichen.

`//JavaScript`
`let x = 3;`
`x = "Hallo Welt";`
`console.log(x);`
	`//TypeScript`
	`let x: Number = 3;`
	`x = "Hallo Welt"; //Fehler`
	`console.log(x);`

Quelltext 3.1: Prüfen des Datentyps in TypeScript

Sowohl der JavaScript-Code als auch der TypeScript-Code führen die gleichen Aktionen aus. Zuerst wird der Variable „x“ ein Wert des Datentyps „number“ zugewiesen. Im nächsten Schritt wird ein neuer Wert vom Typ „string“ zugewiesen. Der Datentyp wurde gewechselt. In der JavaScript-Version wird kein Fehler angezeigt, in TypeScript erhalten wir jedoch eine Fehlernachricht, dass der neue Wert nicht zuweisbar sei.

`//JavaScript`
`let person = {`
`  firstName: "Alex",`
`  age: 21`
`}`
 	
`console.log(person.lastname);`
	`//TypeScript`
	`let person = {`
	`  firstName: "Alex",`
	`  age: 21`
	`}`
	
	`console.log(person.lastname);//Fehler`

Quelltext 3.2: Prüfen der Objekteigenschaften in TypeScript

In diesem Beispiel, wird ein Objekt mit den Eigenschaften „firstname“ und „age“ erstellt. Versuchen wir nun auf eine Eigenschaft zuzugreifen, welche für das Objekt nicht existiert, erhalten wir in TypeScript bereits vor der Laufzeit eine Fehlermeldung, welche uns darauf hinweist, dass diese Eigenschaft für das Objekt nicht definiert wurde. In JavaScript erscheint der Fehler erst zur Laufzeit.
Da einige Fehler so schneller entdeckt werden können, wird TypeScript von manchen Entwicklern bevorzugt. (Steyer, 2021, S. 38)

6.2 Aufbau eines Projekts

Nach dem Erstellen eines Projekts mit der „Angular CLI“ sind folgende wichtige Dateien in der Projektstruktur vorzufinden:

```
my-app
├--- node_modules
├--- .angular
├--- package.json
└── src
   ├----index.html
   ├----main.ts
   └── app
      ├----app.component.css
      ├----app.component.html
      ├----app.component.ts
```

Genau wie in „React“ und „Vue.js“ wird eine index.html erstellt. Dies ist die HTML-Datei, in der die anderen Komponenten generiert werden. Neben der index.html wurden mehrere Dateien für die „App“-Komponente erstellt. Anders als in den anderen Front-End-Technologien besteht in „Angular“ jede Komponente standardmäßig aus drei Dateien.

- Eine HTML-Datei in der die Benutzeroberfläche beschrieben wird, „Angular“ verwendet wie „Vue.js“ eine eigene Templatesprache, die hier verwendet werden kann.
- Eine CSS-Datei beziehungsweise eine CSS-Präprozessor-Datei in der das Styling der Benutzeroberfläche beschrieben werden kann.
- Eine TypeScript-Datei. In der TypeScript-Datei wird die Programmierlogik mit allen Variablen und Methoden implementiert.

(Google2, o. D.)

6.3 Komponente erstellen

Der einfachste Weg, eine neue Komponente in „Angular“ zu erstellen, ist die „Angular CLI“ zu verwenden. Mit dem Befehl „ng generate component home“ wird ein neuer Ordner namens „home“ erstellt. (Google2, o. D.) Dieser enthält die HTML-, Styling- und TypeScript-Datei automatisch.

home.component.ts

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.css']
})
export class HomeComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {
  }

}
```

Quelltext 3.3: „Home“-Komponente in „Angular“

In „Angular“ enthält jede Komponente eine Klasse, welche mit dem „@Component“ Dekorateur versehen ist. Im Dekorateur wird eine Referenz zum entsprechenden Template, also der HTML-Datei, und der Styling-Datei angegeben. (Google2, o. D.) Zusätzlich wird im „selector“ angegeben, unter welchem Namen andere Komponenten diese Komponente in ihrem Template einsetzen können. (Google2. o. D.)
Die Klasse enthält standardmäßig einen leeren Konstruktor und eine „ngOnInit“-Methode. Im Konstruktor können Attribute angegeben werden, wie beispielsweise andere Klassen, von denen die Komponente abhängig ist. Diese werden dort initialisiert. (Google3, o. D.) Mit der „ngOnInit“-Methode kann auf den Lebenszyklus der Komponente eingegriffen werden. Die Methode wird ausgeführt, wann immer die Komponente erstellt wurde. (Google4, o. D.)

6.4 Bedingtes Rendern

Ähnlich wie „Vue.js“ können in „Angular“-Template Direktiven verwendet werden. Wenn ein Element nur unter bestimmten Bedingungen gerendert werden soll, kann die „*ngIf“-Direktive verwendet werden. Ist der Ausdruck innerhalb der Direktive „wahr“, wird das Element hinzugefügt im gegensätzlichen Fall nicht. (Google5, o. D.) In der Beispielanwendung soll erneut „Searchbar“-Komponente nur gerendert

werden, wenn die Variable „clicked“ den Wert „falsch“ gespeichert hat. Die Variable überprüft, dass ein bestimmter Knopf bereits angeklickt wurde.

search.component.html

```
<app-searchbar *ngIf="!clicked"></app-searchbar>
```

Quelltext 3.4: Bedingtes Rendern in „Angular“

Auch eine „if-else“-Aussage, wie sie aus verschiedenen Programmiersprachen bekannt ist, kann in den Templates mit der „*ng-if“-Direktiven verwendet werden. (Google5, o. D.) Nach dem Ausdruck, welcher entweder als „wahr“ oder „falsch“ ausgewertet wird, kann nach einem Semikolon die ID eines anderen Elements angegeben werden. Dieses Element wird im Falle, dass der Ausdruck „falsch“ ist, eingefügt. (Google5, o. D.)
In der Beispielanwendung wird in der Box, die die Details eines Rezeptes anzeigt, als zusätzlich Information angegeben, ob ein Rezept beispielsweise „Vegetarisch“ ist. Wenn die Variable „vegetarian“ den Wert „wahr“ gespeichert hat, wird ein „Yes“ gerendert, ansonsten ein „No“.

recipebox.component.html

```
<p *ngIf="recipe.vegetarian; else notVegetarian> Vegetarian: Yes</p>
<ng-template #notVegetarian><p> Vegetarian: No</p></ng-template>
```

Quelltext 3.5: Bedingtes Rendern in „Angular“, mit alternativem Block

6.5 Wiederholtes Rendern

In „Angular“ existiert ebenfalls eine Direktive für das wiederholte Rendern eines Elements. In der „*ngFor“-Direktive kann eine Liste angegeben werden, welche iteriert werden und für jedes Element der Liste wird das darzustellende Element einmal gerendert. (Google5, o. D.)
Hier wird erneut in der Beispielanwendung die Suchergebnisse gerendert. Nach einer erfolgreichen Suche nach einem Rezept werden Boxen mit Titeln und Bildern der Rezepte hintereinander gerendert. Die Rezepte wurden zuvor in einer Liste namens „recipes“ gespeichert.

search.component.html

```
<app-searchresult *ngFor="let result of results"
 [title]="result.title" [image]="result.image"></app-searchresult>
```

Quelltext 3.6: Wiederholtes Rendern in „Angular“

6.6 Kommunikation zwischen Komponenten

6.6.1 Von Eltern-Komponente zur Kind-Komponente

Wenn eine „Kind“-Komponente Daten von seiner „Eltern“-Komponente entgegennehmen soll, werden diese zuerst als Variablen in der „Kinder“-Komponente definiert. Jede Variable, welche ihren Wert von einer „Eltern“-Komponente erhält, wird mit dem „@Input-Dekorateur“ gekennzeichnet. (Google6, o. D.)

searchresult.component.ts

```
…
export class SearchresultComponent implements OnInit {

  @Input() title = "";
  @Input() image = "";
…
}
```

Quelltext 3.7: Verwenden des „@Input-Dekorateur“ in „Angular“

Die Variablen können im Template eingesetzt werden.

searchresult.component.html

```
<div class="result-box">
    <img [src]="image" alt="Bild des Essens" />
    <p>{{title}}</p>
</div>
```

Quelltext 3.8: Verwenden der erhaltenen Variablen im Template

Die „Eltern“-Komponente kann die Variablen an die „Kind“-Komponente im Template binden. „Angular“ hat eine besondere Syntax für das Binden von Daten. Wenn ein Attribut von außerhalb des Elements dynamisch gebunden werden soll, das bedeutet das Element wird aktualisiert, wenn sich der Wert des Attributs verändert, wird der Name des Attributs in eckigen Klammern geschrieben. (Google6, o. D.) Die „Search“-Komponente ist die „Eltern“-Komponente der „SearchResult“-Komponente und übergibt ihr alle Titel und Bilder.

search.component.html

```
<app-searchresult *ngFor="let result of results" [title]="result.title"
[image]="result.image"></app-searchresult>
```

Quelltext 3.9: Übergeben der Variablen an die „Kind“-Komponente

6.6.2 Von Kind-Komponente zur Eltern-Komponente

Auch für den umgekehrten Fall, dass die „Kind“-Komponente Daten an seine „Eltern“-Komponente senden soll, existiert ein Dekorateur, der „Output“-Dekorateur. (Google6, o. D.) Ähnlich wie in „React“ und „Vue.js“, hört die „Eltern“-Komponente auf ein Event der „Kind“-Komponente.
Die „Kind“-Komponente enthält ein Objekt der „EventEmitter“-Klasse. Die „EventEmitter“-Klasse ist eine Klasse, die „Angular“ bereitstellt. Sie verfügt über eine Methode namens „emit“, welche die „Eltern“-Komponente benachrichtigt, die auf das Event hört. (Google6, o. D)

searchbar.component.ts

```
…
export class SearchbarComponent implements OnInit {

  @Output() searchEvent = new EventEmitter();

  constructor() { }

  ngOnInit(): void {
  }

  searching(inputText: string) {
    this.searchEvent.emit(inputText);
  }

}
```

Quelltext 3.10: Verwenden eines EventEmitters in „Angular“

Im Template der „Kind“-Komponente wird ein Element mit einem Event versehen, welches die „searching“-Funktion aufruft, sobald das Event eintrifft. (Google6, o. D.)

searchbar.component.html

```
<input [(ngModel)]="inputText" type="text" />
<button (click)="searching(inputText)">Search</button>
```

Quelltext 3.11: Binden von Methoden an ein Event in „Angular“

Die „Eltern“-Komponente bindet das Event auch im eigenen Template. Wann immer in der „Kind“-Komponente das Event eintrifft, wird die Funktion der „Eltern“-Komponente ausgeführt. (Google6, o. D.)

search.component.html

```
<app-searchbar (searchEvent)="searching($event)"></app-searchbar>
```

Quelltext 3.12: „Eltern“-Komponente hört auf „Kind“-Komponente

6.7 Routing

6.7.1 Installation und Implementierung

Genau wie „Vue.js“ bietet das „Angular“-Framework eigene Routing Funktionen an. Bei der Erstellung eines neuen Projekts mit der „Angular CLI“ wird bereits die Option gegeben, Routing in das Projekt hinzuzufügen. Dann werden bereits zu Erstellung des Projekts die nötigen Dateien generiert, aber dies kann auch im Nachhinein in ein bestehendes Projekt hinzugefügt werden, indem ein neues Modul mit der „Angular CLI“ und dem Befehl „ng generate module app-routing“ erstellt wird. (Google7, o. D.) In dem Modul wird das Router Module, welches „Angular“ bereitstellt importiert und in einer Liste werden alle Routen mit den zugehörigen Komponenten definiert. Jedes Modul, welches Routing verwenden soll, muss das app-routing-Modul importieren. (Google7, o. D.)

app-routing.module.ts

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { HomeComponent } from './home/home.component';
import { SearchComponent } from
'./searchview/components/search/search.component';

const routes: Routes = [
  {path: '', component: HomeComponent},
  {path: 'search', component: SearchComponent},
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Quelltext 3.13: Definieren der Routen in „Angular“

Ähnlich wie in „React“ und „Vue.js“ kann in der „App“-Komponente eine Komponente verwendet werden, welche je nach Pfad in der URL, die entsprechende Komponente rendert. Diese heißt in „Angular“ „router-outlet“. (Google7, o. D.)

app.component.html

```
<app-navbar></app-navbar>
<router-outlet></router-outlet>
```

Quelltext 3.14: Einsetzen des „router-outlets“

Um die Navigation zu implementieren, werden in der Navigationsleiste die HTML Anchor-Tags verwendet, diesen wird jedoch ein „routerLink“-Attribut übergeben, welches auf den Pfad verweist, zu denen sie führen. (Google7, o. D.)

navbar.component.html

```
<header>
    <a routerLink="/">
        <h1>Dinely</h1>
    </a>
    <nav>
        <a routerLink="/search">Search</a>

    </nav>
</header>
```

Quelltext 3.15: Einsetzen der Verlinkungen in „Angular“

6.7.2 Routing mit Parametern

Um anzugeben, dass eine Route zusätzlich Parameter entgegennimmt, muss ähnlich wie in „React“ und „Vue.js“, dass in der Liste der Routen angegeben werden. (Google7, o. D.)

app-routing.module.ts

```
…
const routes: Routes = [
  …
  {path: 'recipeDetails/:id', component: RecipedetailsComponent}
];
…
```

Quelltext 3.16: Definieren einer Route mit Parametern in „Angular“

Durch das „Angular-Routing Module“ kann innerhalb einer Komponente auf den aktuellen Pfad zugegriffen werden. (Google7, o. D) Im Konstruktor wird ein Objekt der Klasse „Activated Route“ hinzugefügt. Das Objekt enthält Methoden und Variablen, mit denen auf die Route der URL zugegriffen werden kann, diese kann in einer Variable gespeichert und weiterverarbeitet werden. (Google7, o. D.)

recipedetails.component.ts

```
export class RecipedetailsComponent implements OnInit {

  path: string | null = this.route.snapshot.paramMap.get('id');

  constructor(private route: ActivatedRoute) { }

  ngOnInit(): void { }

}
```

Quelltext 3.17: Zugreifen auf die Parameter in „Angular“

6.8 HTTP-Anfrage senden

Zum Schluss müssen nur noch HTTP-Anfragen an die „spoonacularAPI“ gesendet werden. Anders als „React“ und „Vue.js“ bietet „Angular“ einen HTTP-Client an und es müssen keine externen Bibliotheken zusätzlich installiert werden.
Um auf den HTTP-Client zuzugreifen, muss dieser nur im App-Modul importiert werden. (Google8, o. D.) Über den Konstruktor kann der Client in den benötigten Klassen verwendet werden. In „Angular“ ist es üblich für das Versenden von HTTP-Anfragen einen Service zu schreiben. (Google8, o. D.) Eine Komponente soll nach Konvention idealerweise nur Eigenschaften und Methoden verwalten, die für das Binden von Daten an die Benutzeroberfläche nötig sind. Businesslogik oder das Abfragen von Daten eines Servers sollen in Service-Klassen implementiert werden, auf die die Komponenten zugreifen können. (Google10, o. D.)
Ein Service kann mit der „Angular CLI“ und dem Befehl „ng generate service“ gefolgt vom Namen des Services erstellt werden. (Google10, o. D.)
In dem Service, welcher die Anfragen an die „spoonacular API“ senden, wird zuerst der HTTP-Client eingesetzt und dann Methoden implementiert, welche auf die „get“-Funktion zugreifen. Beim Methodenaufruf kann ein generischer Typ angegeben werden, welcher vorgibt wie die Daten aufgebaut sind, die angefragt werden. (Google8, o. D.)

recipeByName.ts

```
export interface IRecipeByName {
    results: [
        {
            id: number,
            title: string,
            image: string,
            imageType: string
        }
    ],
    totalResults: number
}
```

Quelltext 3.18: Rezept-Model

api.service.ts

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { IRecipeByName } from '../models/recipeByName'

@Injectable({
  providedIn: 'root'
})
export class ApiService {

  baseUrl = "https://api.spoonacular.com/recipes/";

  constructor(private http: HttpClient) { }

  getRecipesByName(name: string | null): Observable<IRecipeByName> {
    const search = `complexSearch?query=${name}`;
    const request = this.baseUrl + search;

    let response = this.http.get<IRecipeByName>(request)
    return response;
  }
}
```

Quelltext 3.19: Senden von HTTP-Anfragen in „Angular“

Der fertige Service kann danach in Komponenten verwendet werden, welche HTTP-Anfragen versenden sollen. (Google8, o. D.) In der Beispielanwendung sind das die „Search“-Komponenten und die „RecipeDetail“-Komponente. In der Suchleiste kann wie in „React“ und „Vue.js“ eine Funktion an ein Event gebunden werden, welche jedes Mal ausgeführt wird, wenn das Event eintrifft. Im Falle der Suchleiste wird

eine Funktion gebunden, welche die Suchfunktion des Services aufruft, sobald der Such-Knopf gedrückt wurde.

search.component.html

```
<app-searchbar (searchEvent)="searching($event)"></app-searchbar>
…
```

Quelltext 3.20: Zugreifen auf die Parameter in „Angular“

search.component.ts

```
import { Component, OnInit } from '@angular/core';
import { ApiService } from '../../../shared/services/api.service';

…
export class SearchComponent implements OnInit {

 results: {title: string, image: string, imageType: string}[] = [];

  constructor(private _apiService: ApiService) { }

  ngOnInit(): void {
  }

  searching(inputText: string) {
    this._apiService.getRecipesByName(inputText)
    .subscribe(data => {
      for(let i = 0; i < data.results.length; i++) {
        this.results.push(data.results[i]);
      }
    })
  }
```

Quelltext 3.21: Aufrufen der Funktion aus dem Service

In der „RecipeDetail“-Komponente soll eine Anfrage gesendet werden, wenn die Komponente geladen wurde. Dafür kann wie in „React“ und „Vue.js“ auf den Lebenszyklus der Komponente zugegriffen werden. Die „ngOnInit-Methode“ wird ausgeführt, wenn die Komponente gerendert wurde daher wird dort die Methode der Service-Klasse mit dem Pfad der URL aufgerufen. (Google4, o. D.)

7 Evaluation der Kriterien

7.1 Leistung

Um die Leistung der Webtechnologien zu vergleichen, wurde wie in Kapitel 2 beschrieben, Lighthouse verwendet. Die Messungen wurden an den drei von mir erstellten Webanwendungen durchgeführt.

Kategorie	React	Vue.js	Angular
Erster Pixel	0.5s	0.4s	0.8s
Geschwindigkeitsindex	1.1s	2.2s	1.3s
Größtes Element	3.0s	2.3s	2.4s
Zeit bis Interaktiv	2.5s	3.4s	2.2s

Tabelle 1: Messungen der drei Webanwendungen

Neben den eigenen Messungen werden zusätzlich die Ergebnisse von realcodecamp und ihren „A RealWorld Comparison of Front-End Frameworks with Benchmarks“ berücksichtigt. Die Punktzahl wurden ebenfalls mit Lighthouse und deren Metriken bestimmt.
In ihren Messungen der Kategorie Leistungen kamen folgenden Werte raus:

1. „Vue.js“ mit einer Bewertung von 94 von 100 Punkten
2. „Angular“ mit einer Bewertung von 93 von 100 Punkten
3. „React“ und „Redux“ mit einer Bewertung von 80 von 100 Punkten

(freecodecamp, 2019)
Das Fazit von realcodecamp zu dieser Kategorie ist, dass kein großer Unterschied jedoch bemerkbar sein würde, da die meisten verglichenen Technologien eine Punktzahl von 90 oder höher haben. (freecodecamp, 2019)

7.2 Beliebtheit

7.2.1 GitHub-Sterne

Erste Anlaufstelle, um die Beliebtheit einer Technologie zu bestimmen, sind die GitHub-Sterne, welche jede der Webtechnologien erhalten hat.

Die meisten Sterne hat das Repository von „Vue.js“ erhalten, welches bis zum September 2022 insgesamt ungefähr 200.000 Sterne von Nutzern erhalten hat, gefolgt von „React“, welches ungefähr 195.000 Sterne erhalten hat. (vuejs, 2022; facebook1, 2022) Am wenigsten Sterne hat „Angular“ mit ungefähr 85.000 Sternen. (angular, 2022)
Sterne werden meistens von Nutzern vergeben, wenn ihnen ein Projekt besonders gefällt, oder sie über weitere Entwicklungen auf dem Laufenden gehalten werden wollen. (GitHub, 2022)

7.2.2 Stack Overflow Developer Survey

In der Stack Overflow Developer Survey haben in der Kategorie Web Frameworks und Technologien insgesamt 58.743 Personen abgestimmt, darunter waren 45.297 beruflich aktive Entwickler. (StackOverflow, 2022) Die Umfrage ergab, dass „React“ von 42.62% der Teilnehmer verwendet wird, was „React“ auf den zweithöchsten Platz bringt. „React“ wird hierbei sowohl von professionellen Entwicklern als auch von Personen, die Programmieren lernen, verwendet. Nur „Node.js“ wurde von mehr Teilnehmern verwendet. (StackOverflow, 2022) „Angular“ erreichte in der Umfrage 20.39%, was dem fünften Platz entspricht. „Angular“ wird jedoch mehr von beruflichen Entwicklern verwendet als von Anfängern. (StackOverflow, 2022) Platz 6 belegt „Vue.js“ mit 18.82% der Stimmen. „Vue.js“ wurde von den drei Technologien am wenigsten von beruflichen Entwicklern verwendet, wird jedoch von mehr Personen, die Programmieren lernen, verwendet als „Angular“. (StackOverflow, 2022)
Tabelle 2: Ergebnisse der „StackOverflow Developer Survey 2022“

Personengruppe	React	Vue.js	Angular
Alle Teilnehmenden	42.62%	18.82%	20.39%
Berufliche Entwickler	44.31%	23.06%	19.9%
Personen, die Programmieren lernen	42.81%	10%	13.48%

Quelle: StackOverflow, 2022
In einer weiteren Kategorie der Umfrage, wurde gegenübergestellt, wie sehr eine Technologie gemocht beziehungsweise gefürchtet wird. (StackOverflow, 2022) „React“ wird von ungefähr 68% der Teilnehmenden gemocht und von ungefähr 32% gefürchtet. „Vue.js“ wird von ungefähr 63% der Teilnehmenden gemocht und von

den restlichen ungefähr 37% gefürchtet. Im Gegensatz dazu wird „Angular“ von ungefähr von 52% gemocht und von 48% gefürchtet. (StackOverflow, 2022) Das macht „Angular“ in dieser Umfrage von den drei Technologie zu der unbeliebtesten und gefürchtetsten, während „React“ von den dreien am beliebtesten ist. (StackOverflow, 2022)

7.2.3 The State of JavaScript

Die Umfrage „The State of JavaScript“ wird seit 2016 jedes Jahr durchgeführt und soll helfen herauszufinden, welche JavaScript Bibliotheken und Frameworks gerade beliebt sind und was Entwickler lernen wollen. (stateofjs1, 2021)
Die Kategorie Front-End Frameworks wird dabei noch in Zufriedenheit, Interesse und Nutzung eingestuft. (stateofjs2, 2021) Im Punkt Zufriedenheit hat „React“ von den drei Technologie den höchsten Wert mit 84%, was in der Gesamtwertung den dritten Platz belegt. Auf den vierten Platz ist „Vue.js“ mit einer Zufriedenheit von 80% und „Angular“ auf den neunten und vorletzten Platz mit einer Zufriedenheit von 45%. (stateofjs2, 2021)
Im Hinblick auf Interesse, also wie viel Prozent der Befragten, diese Technologie erlernen wollen, erreicht „Vue.js“ von den drei Technologien den ersten Platz mit 50%, gefolgt von „React“ mit 48% und wieder auf dem vorletzten Platz „Angular“ mit 16%. (stateofjs2, 2021) 2016 erreichte „Angular“ in der Intressenseinstufung noch 50% und nahm die folgenden Jahre ab, ähnlich wie „React“, welches 2016 noch 75% erreicht hat. (stateofjs2, 2021)

7.3 Lernkurve

Der Einstieg mit „React“ wird als weniger schwer beschrieben und die Lernkurve solle nicht zu steil sein. (Reis, 2020) Dies lässt sich damit erklären, dass für den Anfang nur Kenntnisse in HTML und JavaScript nötig sind. Die Spracherweiterung JSX ähnelt HTML und erlaubt es, bereits bekannte JavaScript-Funktionen darin zu verwenden. (Meta4, o. D.)
Ebenfalls lässt sich die flache Lernkurve durch die Menge an Ressourcen erklären. Allein auf StackOverflow wurden 416.920 Fragen über „React“ gestellt. (Stack Overflow, o. D.) Dadurch kann bei Problemen schneller eine Lösung mit Erklärung gefunden werden. Zusätzlich ist „React“ kein vollständiges Framework, wodurch der Kern von „React“ leichter und schneller gelernt werden kann. (Daityari, 2022)

Eine ähnliche Lernkurve wird „Vue.js“ zugesagt. Auch hier sind nur Kenntnisse in HTML und JavaScript notwendig. Die Vue-Dateien verwenden zwar teilweise eine eigene Syntax und die Template-Sprache bietet einige Erweiterungen, welche über das Standart-HTML hinaus gehen, diese Grundlagen können jedoch schnell erlernt werden. (vue14, o. D.)
Die steilste Lernkurve ist bei „Angular“ vorzufinden. In „Angular“ werden zusätzlich Kenntnisse in TypeScript vorausgesetzt. Die steile Lernkurve kommt auch Zustande, da „Angular“ ein komplettes Framework ist, welches verschiedene Funktionalitäten, wie Routing oder einen HTTP-Client anbietet und nicht eine Bibliothek wie „React“ ist. (Fleury, 2022) Dadurch müssen mehr Konzepte gelernt werden. (vue14, o. D.)

8 Fazit

Nachdem die drei Webtechnologien vollständig verglichen wurden, kann nun geschlussfolgert werden, was die Vorteile und Nachteile jeweils sind.
Eine Stärke von „React“ ist der leichte Einstieg, welcher lediglich Vorkenntnisse in HTML und JavaScript voraussetzt. Außerdem wird „React“ von vielen Entwicklern verwendet, wodurch Lösungen zu Problemen leichter gefunden werden können. (StackOverflow, 2022) Da „React“ eine Bibliothek ist und kein vollständiges Framework, bietet es Freiheiten in der Strukturierung des Projekts. „React“ gibt hierbei nichts vor. (Meta9, o. D.) Dies kann jedoch auch als Nachteil gesehen werden, da für bestimmten Funktionen, wie das Routing oder das Senden von Anfragen, zusätzliche Bibliotheken installiert werden müssen.
„React“ bietet sich daher besonders für Entwickler an, die erste Erfahrungen in HTML und JavaScript gesammelt haben und dieses Wissen nutzen wollen, um Single-Page-Webanwendungen zu programmieren. Die vielen Ressourcen, durch die große Beliebtheit, machen Anfängern den Einstieg leicht. Damit ist „React“ ein guter Startpunkt. Zusätzlich wird „React“ von den drei Technologien am meisten verwendet, auch unter professionellen Entwicklern. (StackOverlow, 2022) Somit ist es in einem beruflichen Feld zukunftssicher.
Im Kontrast hierzu steht „Angular“. Die Lernkurve von „Angular“ wird steiler als die von „React“ beschrieben, dies lässt sich dadurch begründen, dass zusätzlich Kenntnisse in TypeScript nötig sind, eine Sprache, welche strengere Regeln als JavaScript hat. Außerdem bietet „Angular“ weniger Freiheiten, bei der Struktur der Projekte. Die „Angular CLI“ erstellt für einen fertige Ordnerstrukturen und in der offiziellen Dokumentation wird eine Projektstruktur vorgegeben. (Google9, o. D.) Da „Angular“ ein vergleichsweise großes Framework ist, welches schon Funktionen Aufgaben wie Routing oder Senden von HTTP-Anfragen anbietet, muss sich hier keine weitere Bibliothek rausgesucht werden. Dadurch sind „Angular“-Projekte vom Aufbau oftmals ähnlich und Entwickler, welche Erfahrung in „Angular“ bereits haben, können sich dadurch schneller einfinden als in ein „React“-Projekt, welche theoretisch sehr unterschiedlich vom Aufbau und den verwendeten Bibliotheken sein kann.
„Angular“ bietet sich daher besonders für große Projekte an, an denen mehrere verschiedene Personen arbeiten. Durch TypeScript und die bereits vorgegebene Projektstruktur finden Entwickler, die Angular beherrschen, sich darin besser ein. (Steyer, 2021, S. 38) „Angular“ findet daher eher Verwendung bei professionellen

Entwicklern, welche an großen Projekten arbeiten, die skalieren müssen, als bei Personen, welche Programmieren lernen. (StackOverflow, 2022)
„Vue.js“ liegt dazwischen. Es ist ein leichtgewichtiges Framework. Es verwendet eine eigene Template-Sprache, welche das Binden von Daten ermöglicht, sowie eine offizielle Routing-Bibliothek. Dennoch sind dem Entwickler mehr Freiheiten in Bezug auf die Strukturierung der Projekte möglich als beispielsweise in „Angular“. (Kelly, 2021) Es ist die Neueste der drei Webtechnologien und wurde anders als „React“ und „Angular“ nicht von einem großen Unternehmen, sondern von einer einzelnen Person entwickelt. Dennoch konnte „Vue.js“ in Umfragen große Beliebtheit erlangen und sogar von den drei Technologien die meisten GitHub-Sterne sammeln. (vuejs, 2022)
„Vue.js“ bietet sich damit sowohl für Anfänger als auch erfahrene Entwickler an. Die Lernkurve wird nicht als steil beschrieben und soll leichter zu lernen sein als „Angular“. (vue14, o. D.) Ebenfalls ist „Vue.js“ für Personen zu empfehlen, die eine neuere Technologie ausprobieren wollen. Es muss jedoch damit gerechnet werden, dass für „React“ und „Angular“ mehr Ressourcen zu Verfügung stehen, da es sie länger gibt und mehr Nutzer dahinterstehen. (stateofjs2, 2021)

Literaturverzeichnis

Hartmann, N. & Zeigermann, O. (2019). React Grundlagen, fortgeschrittene Techniken und Praxistipps – mit TypeScript und Redux (2. Aufl.). dpunkt.verlag

Steyer, M. (2021). Angular Das Praxisbuch zu Grundlagen und Best Practices (3. Aufl.). O´Reilly

Stack Overflow. (2022). Web frameworks and technologies. https://survey.stackoverflow.co/2022/#most-popular-technologies-webframe

Stack Overflow. (o. D.). Questions tagged [reactjs]. Abgerufen am 1. Oktober 2022, von https://stackoverflow.com/questions/tagged/reactjs

Skolski, P. (2016, 1. Dezember). Single-page application vs. multiple-page application. medium. https://medium.com/@NeotericEU/single-page-application-vs-multiple-page-application-2591588efe58

Schulte, J. (2021, 9. Dezember). Single Page Applications (SPA): Erklärung, Vorteile und Beispiele. magnolia. https://www.magnolia-cms.com/de_DE/blog/alles-wissenswerte-ueber-single-page-applications.html

spoonacular. (o. D.). The only food API you'll ever need. Abgerufen am 12. August 2022, von https://spoonacular.com/food-api/docs

syedmodassirali. (2019, 15. November) Client-Server Model. GeeksforGeek. https://www.geeksforgeeks.org/client-server-model/

Programmbibliothek (o. D.). DeWiki. Abgerufen am 8. August 2022, von https://dewiki.de/Lexikon/Programmbibliothek

Domin, A. (2018, 21. April). Library vs. Framework: Das sind die Unterschiede. t3n. https://t3n.de/news/library-vs-framework-unterschiede-1022753/

InterviewBit. (2021, 23. Oktober). Framework vs Library: Full Comparison. https://www.interviewbit.com/blog/framework-vs-library/

Microsoft. (2022, 24. Juni). TypeScript for JavaScript Programmers. https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html

Javatpoint. (o. D.). React Version. Abgerufen am 8. August 2022, von https://www.javatpoint.com/react-version

Nodejs. (o. D.). Introduction To Node.js. Abgerufen am 10. August 2022, von https://nodejs.dev/en/learn/

npmjs. (o. D.). Build amazing things. Abgerufen 10. August 2022, von https://www.npmjs.com/

facebook1. (2022, 14. Juni). react. GitHub. https://github.com/facebook/react

facebook2. (o. D.) Getting Started. Abgerufen am 9. August 2022, von https://create-react-app.dev/docs/getting-started

angular. (2022, 28. September). Angular. GitHub. https://github.com/angular/angular

vuejs. (2022, 23. August). vue. GitHub. https://github.com/vuejs/vue

Meta1. (o. D.). Components and Props. Abgerufen am 9. August 2022, von https://reactjs.org/docs/components-and-props.html

Meta2. (o. D.). Add React to a Website. Abgerufen am 9. August 2022, von https://reactjs.org/docs/add-react-to-a-website.html

Meta3. (o. D.). Create a new React App. Abgerufen am 9. August 2022, von https://reactjs.org/docs/create-a-new-react-app.html

Meta4. (o. D.). Introducing JSX. Abgerufen am 9. August 2022, von https://reactjs.org/docs/introducing-jsx.html

Meta5. (o. D.). React Without JSX. Abgerufen am 9. August 2022, von https://reactjs.org/docs/introducing-jsx.html

Meta6. (o. D.). Conditional Rendering. Abgerufen am 9. August 2022, von https://reactjs.org/docs/conditional-rendering.html

Meta7. (o. D.). Lists and Keys. Abgerufen am 10. August 2022, von https://reactjs.org/docs/lists-and-keys.html

Meta8. (o. D.). Using the Effect Hook. Abgerufen am 18. August 2022, von https://reactjs.org/docs/hooks-effect.html

Meta9. (o. D.). File Structure. Abgerufen am 15. September 2022, von https://reactjs.org/docs/faq-structure.html

Babel. (o. D.). What is Babel? Abgerufen am 9. August 2022, von https://babeljs.io/docs/en/

MDN1. (2022, 14. September). Map. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Map#try_it

MDN2. (2022, 21. September). Dynamic typing. https://developer.mozilla.org/en-US/docs/Glossary/Dynamic_typing

Jöch, D. (2018, 11. Juli). Functional vs Class-Components in React. https://djoech.medium.com/functional-vs-class-components-in-react-231e3fbd7108

freeCodeCamp. (2019, 8. April). A RealWorld Comparison of Front.End Frameworks with Benchmarks (2019 update). https://www.freecodecamp.org/news/a-realworld-comparison-of-front-end-frameworks-with-benchmarks-2019-update-4be0d3c78075/

krausest. (2022, 4. August). Results for js web frameworks benchmark – official run. https://krausest.github.io/js-framework-benchmark/2022/table_chrome_104.0.5112.79.html

web.dev1. (o. D.). Measure page quality. Abgerufen am 15. August 2022, von https://web.dev/measure/?gclid=CjwKCAjwhNWZBhB_EiwAPzlhNq1i2z8Vg2M-GLaA_WhVdWMfkb71sglCK0SYx-EwZWwCxPgd9a1VlRoCR6gQAvD_BwE

web.dev2. (o. D.). First Contentful Pain. Abgerufen am 15. August 2022, von https://web.dev/first-contentful-paint/

Trent, J. (2019, 24. April). Communication Between Components in React. pluralsight. https://www.pluralsight.com/guides/react-communicating-between-components

Remix1. (o. D.). Quick Start. Abgerufen am 17. August 2022, von https://v5.reactrouter.com/web/guides/quick-start

Remix2. (o. D.). URL Parameters. Abgerufen am 17. August 2022, von https://v5.reactrouter.com/web/example/url-params

Remix3. (o. D.). `use Params`. Abgerufen am 17. August 2022, von https://reactrouter.com/en/main/hooks/use-params

w3schools. (o. D.). React Router. Abgerufen am 17. August 2022, von https://www.w3schools.com/react/react_router.asp

Barger, R. (2021, 13. Juli). How To Use Axios With React: The Definitive Guide (2021). freecodecamp. https://www.freecodecamp.org/news/how-to-use-axios-with-react/

Hecht, J., Rauh S. & Schwartz A. (2019). Vue.js für alle: Wissenswertes für Einsteiger und Experten. entwickler.press

vue1. (o. D.). Quick Start. Abgerufen am 19. August 2022, von https://vuejs.org/guide/quick-start.html

vue2. (o. D.). Template Syntax. Abgerufen am 19. August 2022, von https://vuejs.org/guide/essentials/template-syntax.html

vue3. (o. D.). Single-File Components. Abgerufen am 20. August 2022, von https://vuejs.org/guide/scaling-up/sfc.html

vue4. (o. D.). Conditional Rendering. Abgerufen am 21. August 2022, von https://vuejs.org/guide/essentials/conditional.html

vue5. (o. D.). List Rendering. Abgerufen am 21. August 2022, von https://vuejs.org/guide/essentials/list.html

vue6. (o. D.). Props. Abgerufen am 22. August 2022, von https://vuejs.org/guide/components/props.html

vue7. (o. D.). Events. Abgerufen am 22. August 2022, von https://vuejs.org/guide/components/events.html

vue8. (o. D.). Routing. Abgerufen am 23. August 2022, von https://vuejs.org/guide/scaling-up/routing.html

vue9. (o. D.). Installation. Abgerufen am 23. August 2022, von https://router.vuejs.org/installation.html

vue10. (o. D.). Getting Started. Abgerufen am 23. August 2022, von https://router.vuejs.org/guide/

vue11. (o. D.). Dynamic Route Matching with Params. Abgerufen am 24. August 2022, von https://router.vuejs.org/guide/essentials/dynamic-matching.html

vue12. (o. D.) Using Axios to Consume APIs. Abgerufen am 25. August 2022, von https://v2.vuejs.org/v2/cookbook/using-axios-to-consume-apis.html?redirect=true

vue13. (o. D.). Options: Lifecycle. Abgerufen am 25. August 2022, von https://vuejs.org/api/options-lifecycle.html#created

vue14. (o. D.). Comparison with Other Frameworks. Abgerufen am 1. Oktober 2022, von https://v2.vuejs.org/v2/guide/comparison.html?redirect=true

Malcher, F., Hoppe, J. & Koppenhagen, D. (2019). Angular Grundlagen, fortgeschrittene Themen und Best Practices – inclusive NativeScript und NgRx. dpunkt.verlag

Gavigan, D. (2018, 3. April). The History of Angular. medium. https://medium.com/the-startup-lab-blog/the-history-of-angular-3e36f7e828c7

Google1. (o. D.). Setting up the local environment and workspace. Abgerufen am 2. September 2022, von https://angular.io/guide/setup-local

Google2. (o. D.). Angular components overview. Abgerufen am 4. September 2022, von https://angular.io/guide/component-overview

Google3. (o. D.). Understanding dependency injection Abgerufen am 6. September 2022, von https://angular.io/guide/dependency-injection

Google4. (o. D.). OnInit. Abgerufen am 6. September 2022, von https://angular.io/api/core/OnInit

Google5. (o. D.). Built-in directives. Abgerufen am 7. September 2022, von https://angular.io/guide/built-in-directives

Google6. (o. D.). Sharing data between child and parent directives and components. Abgerufen am 7. September 2022, von https://angular.io/guide/inputs-outputs

Google7. (o. D.) Common Routing Tasks. Abgerufen am 8. September 2022, von https://angular.io/guide/router

Google8. (o. D.). Communication with backend services using HTTP. Abgerufen am 9. September 2022, von https://angular.io/guide/http

Google9. (o. D.). Workspace and project file structure. Abgerufen am 18. September 2022, von https://angular.io/guide/file-structure

Google10. (o. D.). Introduction to services and dependency injection. Abgerufen am 9. September 2022, von https://angular.io/guide/architecture-services

Hurd, T. (2021, 3. Juni). Introduction to Data Types: Static, Dynamic, Strong & Weak. sitepoint. https://www.sitepoint.com/typing-versus-dynamic-typing/

GitHub. (2022). Saving repositories with stars. https://docs.github.com/en/get-started/exploring-projects-on-github/saving-repositories-with-stars

stateofjs1. (2021). State of JavaScript. https://stateofjs.com/en-us/

stateofjs2. (2021). Front-End Frameworks. https://2021.stateofjs.com/en-US/libraries/front-end-frameworks

Kelly, D. (2021, 6. Juli). How to Structure a Large Scale Vue.js Application. VueSchool.io. https://vueschool.io/articles/vuejs-tutorials/how-to-structure-a-large-scale-vue-js-application/

Daityari, S. (2022, 17. August). Angular vs React vs Vue: Which Framework to Choose. codeinwp. https://www.codeinwp.com/blog/angular-vs-vue-vs-react/#learning-curve

Reis, J. (2020, 6. Februar). Angular vs React: a comparison of both frameworks. imaginarycloud. https://www.imaginarycloud.com/blog/angular-vs-react/

Fleury, D. (2022). Angular vs. React in 2022: Side-by-Side Comparison. trio.dev. https://www.trio.dev/blog/angular-vs-react

Abbildungsverzeichnis

Tabellenverzeichnis

Quelltextverzeichnis

Ehrenwörtliche Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen oder Hilfsmittel benutzt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt wurde.

Ort, Datum

Unterschrift
Alexander Diel