Entwurf Frontend

In diesem Abschnitt wird der Entwurf des Frontend-Systems beschrieben. Hier soll ein detaillierter Überblick über die Frontend-Technologieauswahl gegeben werden. Anschließend soll die Frontend-Architektur für die Entwicklung des Webshops aufgezeigt werden. Dazu gehören Übergangsdiagramme der Webseite, Wireframes der unterschiedlichen Seiten, und unterschiedliche Prototypen des visuellen Designs und Designschemas.

Frontend-Framework

Die Anzahl an verfügbaren Web Frameworks sowohl für die Entwicklung von Frontend als auch Backend Applikationen wächst stetig an. Die folgende Abbildung zeigt einen Ausschnitt aus einer StackOverflow-Studie, welche die am meisten genutzten Frontend-Frameworks vergleicht.

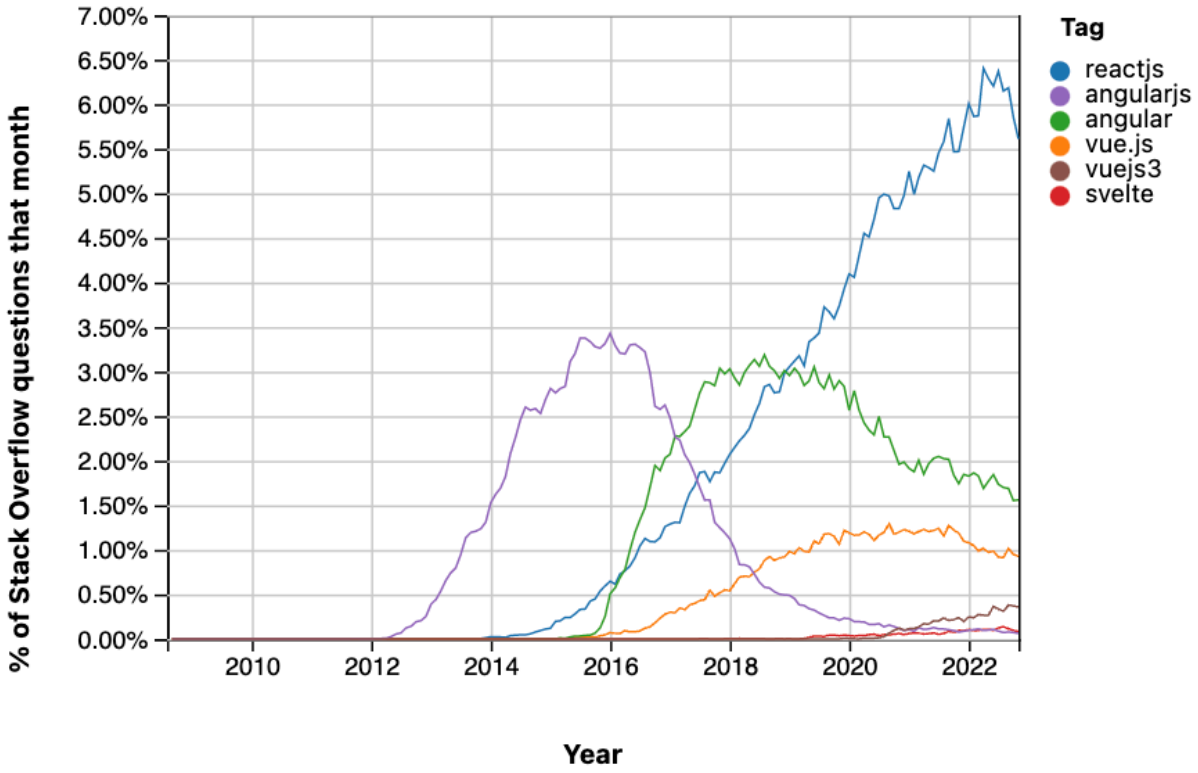

Figure 1: Frontend-Frameworks mit den meisten Fragen auf Stack Overflow. Quelle: Github | Tanguy Krotoff

Im folgenden Abschnitt werden die populärsten Frameworks kurz beleuchtet und einige Vor- und Nachteile bezüglich der Anwendung in einem Webshop-Projekt evaluiert.

React

React, entwickelt von Facebook in 2013, ist das beliebteste Frontend-Framework. Es ist weit verbreitet und auch in vielen Stellenausschreibungen für Frontend-Entwickler vertreten. Ausschlaggebend ist für viele Entwickler dabei die große React-Community, die durch die große Popularität des Frameworks Tutorials und Bibliotheken für zahlreiche Anwendungsfälle ermöglicht. React ist im Kern ein sehr minimales Framework. Das bedeutet, dass sehr viele Aufgaben wie Routing von Open-Source Third-Party Bibliotheken übernommen werden. React ermöglicht es, dynamische und komplexe Nutzer-Interfaces zu erstellen, und ist dabei durch sein Virtual DOM schneller darin Webanwendungen zu erstellen.

Angular
Angular, entwickelt und verwendet von Google, ist ebenfalls ein sehr populäres Frontend-Framework. Es verwendet das reguläre DOM-Modell, TypeScript als Programmiersprache und Testing über das Jasmine Testing Framework. Zusätzlich bietet Angular auch offizielle Bibliotheken für Routing, Animationen und Server Side Rendering. Allerdings hat Angular auch eine hohe Lernkurve, da es relativ fest vorschreibt wie ein Projekt zu organisieren und strukturieren ist. Dadurch sinkt die Flexibilität eines Angular Projektes. Durch seine feine Strukturierung ist Angular sehr gut geeignet für größere Projekte mit größeren Teams, während für kleinere Projekte mit 1-2 Entwicklern die komplexe Struktur eher ungeeignet ist. Nach React ist Angular das am zweitmeiste heruntergeladene Framework auf npm.

Vue.js
Vue.js ist ein von Evan You unabhängig entwickelt und gewartetes Framework. Es hat Ähnlichkeiten zu Angular, ist aber zugänglicher für einzelne Entwickler und kleinere Projekte. Vue besitzt offizielle Packages für Routing und State Management und stützt sich in weiteren Funktionen auf ein großes Ökosystem an Third-Party Kontributionen. Seine Popularität ist auf dem gleichen Niveau wie Angular, mit einer änhlichen Anzahl an npm-Downloads pro Monat. Vue sticht auch durch sein Nutzerfreundliches CLI hervor, welches den Entwickler über eine Browser-GUI durch die Erstellung eines Projektes leitet.

Svelte
Svelte, entwickelt von Rich Haris, ist ein aufstrebendes Frontend-Framework, das sich durch eine individuelle Herangehensweise an die Webentwicklung von anderen Frameworks absetzt. Im Gegensatz zu React, Angular und Vue, die zur Laufzeit im Browser arbeiten, kompiliert Svelte den Code zur Built-Zeit in imperativen Code, der die DOM direkt aktualisiert. Dies führt zu schnelleren Ladezeiten und besserer Performance. Durch den integrierten Hot-Module-Replacement-Entwicklungs-Server können trotzdem Live-Änderungen im Browser aktualisiert werden. Svelte bietet keine offiziellen Bibliotheken für Funktionen wie Routing oder State Management, diese werden von Community-Lösungen übernommen. Svelte hat aber eine deutlich kleinere Community und somit auch weniger Third-Party Bibliotheken als React, Angular und Vue. Svelte hat auch eine relativ flexible Projektstruktur und eine flachere Lernkurve im Vergleich zu den größeren und umfangsreicheren Frameworks. Svelte kam auch in einer StackOverflow-Umfrage zu den Vorlieben bei der Entwicklung mit unterschiedlichen Frameworks auf den ersten Platz, wobei über 70% aller Entwickler zugegeben haben gerne mit Svelte zu arbeiten.

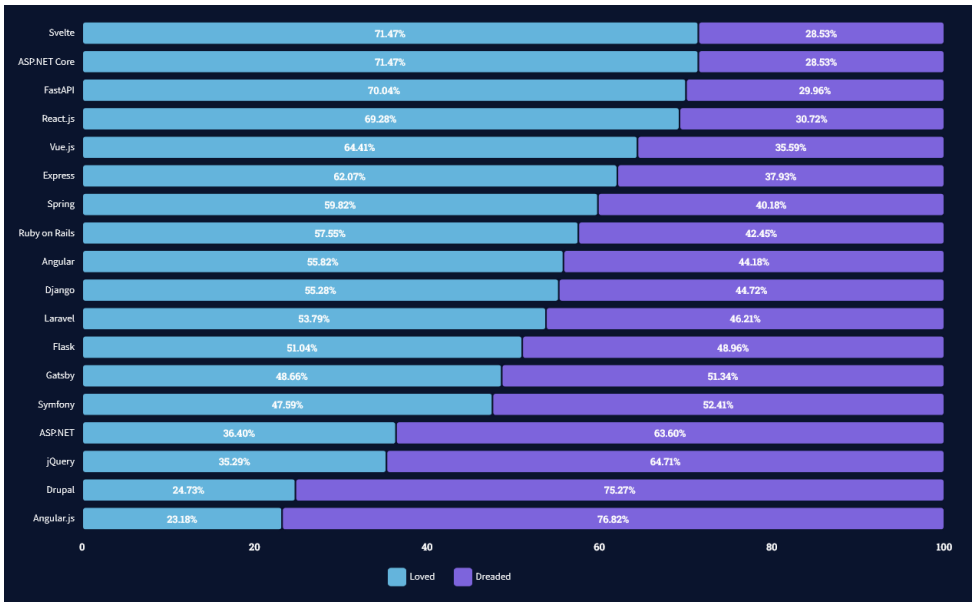

Figure 2: Ergebnisse einer Stack Overflow-Umfrage zum Thema, mit welchem Framework am liebsten gearbeitet wird. Quelle: Stack Overflow Annual Developer Survey 2021

Vergleich der Frameworks anhand fester Kriterien

Um ein Framework auszuwählen wurden folgende Faktoren in die Entscheidung einbezogen:

- Entwicklungsaufwand und Lernkurve
- Developer Experience
- Community-Support
- Eignung für kleinere bis mittlere Projektgrößen

Framework	Support	Lernkurve	Eignung
React	Große Anzahl an Community-Lösungen und Support verfügbar	Relativ steile Lernkurve für Neulinge, guter Enstieg für Entwickler mit Web-Dev-Erfahrung	Für jede Projektgröße geeignet
Angular	Eingebaute Lösungen für viele gängige Anforderungen	Sehr steile Lernkurve mit strikter Projektstruktur.	Eher für größere Projekte und Teams geeignet
Vue.js	Offizielle Packages für gängige Anforderungen und angemessenes Ökosystem an Third-Party Bibliotheken und Support	Relativ flache Lernkurve	Für einzelne Entwickler gut geeignet

Framework	Support	Lernkurve	Eignung
Svelte	Geringe Anzahl and Ressourcen	Relativ flache Lernkurve	Für einzelne Entwickler gut geeignet

Quelle: [1]

Auswahl
Für die Entwicklung des Webshops wird das **React-Framework** gewählt, da es sich für jede Projektgröße eignet und somit auch für ein kleines Team von 1-2 Entwicklern geeignet ist. React sticht vor allem durch die große und aktive Community heraus, was bedeutet dass problemlos Tutorials, Forenbeiträge, Bibliotheken und Tools zu jedem Problem gefunden werden können, was den Entwicklungsaufwand und die Developer Experience maßgeblich verbessert.

CSS-Framework

Für das Projekt wird ein CSS-Framework und kein blankes CSS verwendet, da es für ein React-Projekt zahlreiche Vorteile bietet. Ein CSS-Framework wie z.B. **Bootstrap** [2] oder **Tailwind CSS** [3] stellt eine Sammlung vorgefertigter Stile und Komponenten bereit, die die Entwicklungszeit erheblich verkürzen. Außerdem ermöglicht CSS Framework das Einhalten eines einheitlichen Stils, was zu einem konsistenten “Look and Feel” der Webseite über verschiedene Seiten hinweg führt. So wird Nutzern das Navigieren erleichtert. Zusätzlich haben CSS-Frameworks oft eingebaute Methoden um Responsive Design zu ermöglichen, also die korrekte Darstellung der Komponenten auf unterschiedlichen Bildschirmgrößen, ohne das zusätzlicher Entwicklungsaufwand entsteht. [4]

Tailwind CSS
Tailwind CSS hebt sich von anderen CSS-Frameworks insofern ab, dass es keine vordefinierten Komponenten bereitstellt, welche eine gewisse Richtung vorgeben würden. Stattdessen bietet Tailwind CSS eine Sammlung von niedrigstufigen Utility-Klassen, die dem Entwickler die Möglichkeit geben selbst eigene Designs zu entwerfen ohne sich dabei groß mit dem hinterliegenden CSS auseinandersetzen zu müssen.

In der State of CSS Studie 2023 sticht Tailwind CSS als der Vorreiter in der Developer Experience heraus, mit einer deutlichen Mehrheit in Entwicklern die mit Tailwind CSS arbeiten und es weiterhin nutzen würden

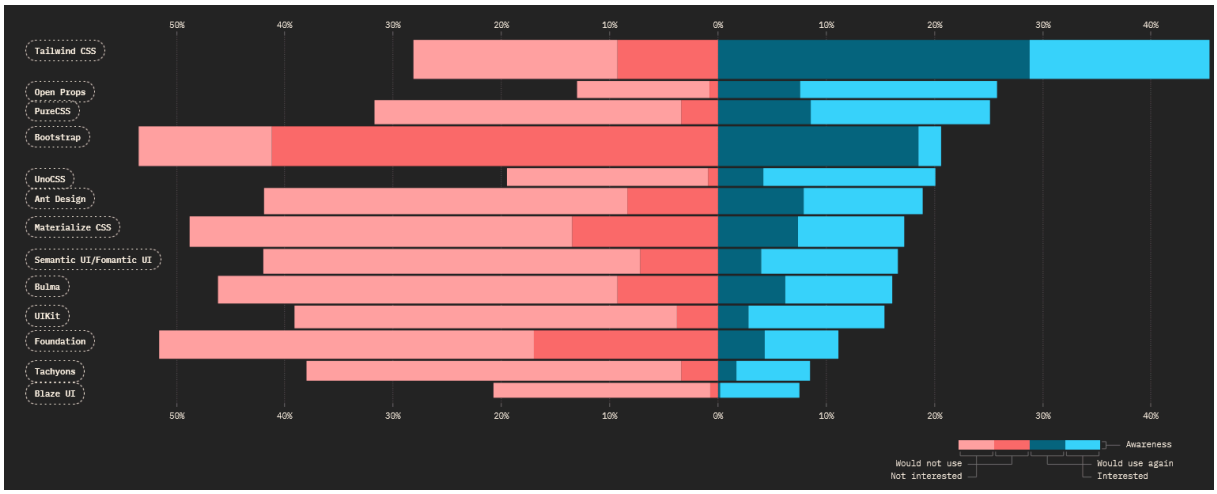

Figure 3: Ergebnisse einer Studie zum Thema, mit welchem CSS-Framework am liebsten gearbeitet wird. Quelle: State of CSS 2023

Tailwind CSS wird verwendet, indem für jeden HTML-Komponenten ein Attribut “className” verwendet wird. Im className-Attribut werden dann die Stilisierungs-Optionen von Tailwind in beliebiger Reihenfolge gelistet. Diese können angepasst werden um das Desing des Komponenten im Browser zu verändern.

```
<footer className="bg-white w-full py-4 shadow-md mt-4">
```

Listing 1: Stilisierung eines Footer-Komponenten durch Tailwind CSS, Quelle: Eigene Darstellung

Building Tool: Vite

In der modernen Webentwicklung spielen Build-Tools eine entscheidende Rolle bei der Optimierung und Automatisierung des Entwicklungsprozesses. Diese Werkzeuge helfen Entwicklern dabei, den Code zu bündeln, zu komprimieren und zu transformieren.

Vite ist ein Frontend Building Tool, “das ein schnelleres und schlankeres Entwicklungserlebnis für moderne Webprojekte bieten soll” [5]. Vite besteht aus zwei Komponenten:

- Einem Entwicklerserver, der unterschiedliche Features über native ES-Module bietet, wie z.B. schnelles Hot Module Replacement
- Ein Build-Befehl, der den JavaScript Modulbundler rollup.js verwendet, um hochoptimierte statische Assets für die Produktion zu produzieren

Vite wurde ursprünglich für Vue.js entwickelt, unterstützt aber inzwischen eine Vielzahl von Frontend-Frameworks, einschließlich React. Dazu muss lediglich das Projekt mit dem Vite-Command erstellt werden:

```
bun create vite webshop-app --template react
```

Listing 2: Erstellen eines Webshop-Projektes mit Vite und React über Bun, Quelle: Eigene Darstellung

Frontend Designentscheidungen

Für das Frontend mussten unterschiedliche Designentscheidungen getroffen werden.

- Der Aufbau der Webseite und die Routing-Optionen
- Aufbau der unterschiedlichen Seiten
- Corporate Design des Webshops

Für den Aufbau der Webseite und den Routing-Optionen wird ein Routing-Diagramm erstellt. Für den Aufbau der unterschiedlichen Seiten des Webshops verwenden wir mehrere Wireframes, die die

wichtigen Unterseiten darstellen. Für das Corporate Design wird eine Farbpalette und ein Logo gewählt, mit welcher dann eine Seite beispielhaft als Prototyp entworfen wird.

Routing der Webseite

Die Webseite benötigt für die Erfüllung der geforderten Funktionalitäten folgende Seiten:

- Startseite
- Anmelde- / Registrierungsbereich
- Waren-Browser mit Suchfunktion
- Einzelne Seite für jedes Produkt mit Details
- Warenkorb
- Checkout und Bestellung
- Benutzerprofil mit vergangenen Bestellungen
- Administratorbereich mit Kontrolle über Benutzerkonten und Produktstand

Eine Sitemap dafür könnte wie folgt aussehen:

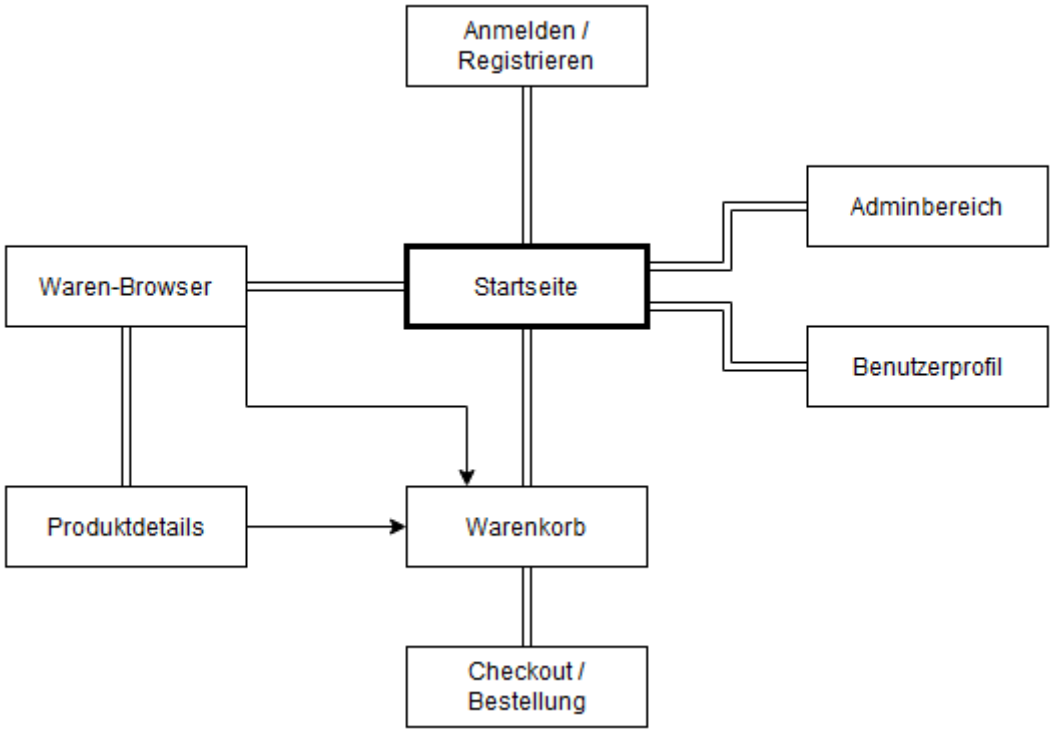

Figure 4: Navigationsdiagramm des Webshops, Quelle: Eigene Darstellung

Einzelne Striche bedeuten hier einseitige Navigation, doppelte Striche stellen Navigation in beide Richtungen dar. Die Startseite fungiert als zentraler Einstiegspunkt und als Wurzel aller Routen. Über eine Navigationsbar sollen direkt von der Startseite aus der Login/Register-Bereich (welcher über das Backend nach Kinde weitergeleitet wird, deshalb also nicht Teil des Frontend-Designs ist), das Benutzerprofil (inkl. Admin-Bereich für Administratoren), der Warenkorb und der Warenbrowser (inkl. Suchfunktion) über eine zentrale Navigationsleiste erreichbar sein. Unterpunkte wie einzelne Produktseiten und der Checkout-Bereich sind dann aus den ihnen zugehörigen Oberbereichen aufrufbar. Das Routing-Diagramm sieht wie folgt aus:

```
/
├── /products
│   └── /product/:id
├── /admin
│   ├── /admin/products
│   └── /admin/users
├── /user
├── /contact
├── /cart
└── /checkout
```

Wireframes der unterschiedlichen Seiten

Als nächster Schritt wurden Wireframes für die unterschiedlichen Seiten der Webseite entworfen. Diese enthalten nur die grobe Struktur der Seite und können sich im Entwicklungsprozess verändern. Es soll lediglich ein Grundriss sein, welche Elemente verfügbar sein sollen.

Das Grundprinzip jeder Seite soll wie folgt aufgebaut sein:

- Eine Navigationsleiste mit wichtigen Icons und einer Suchleiste am oberen Ende der Seite
- Seiteninhalt in der Mitte der Seite
- Ein Fußzeile mit Kontaktdaten und wichtigen Links zu z.B. Social-Media Pages des Webshops

Startseite / Landing Page

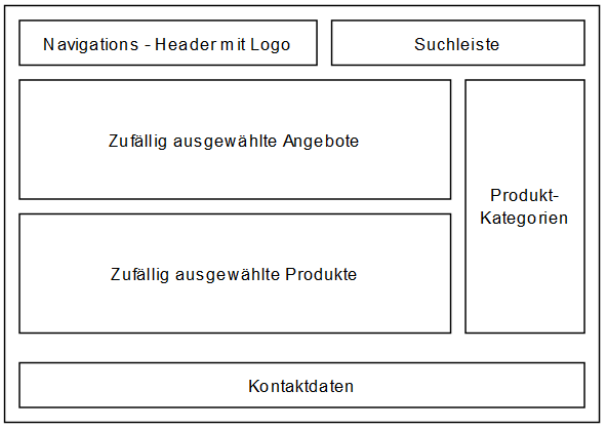

Figure 5: Wireframe der Startseite des Webshops, Quelle: Eigene Darstellung

Die Startseite folgt dem Standard-Design des Webshops. Der Seiteninhalt besteht aus zwei Reihen an Produkten: Eine Reihe mit zufällig ausgewählten Angeboten, und eine Reihe mit zufällig ausgewählten Standardprodukten. Diese sollen den Nutzer bereits beim Betreten der Seite Möglichkeiten geben die Produktauswahl einzuschätzen und dazu verleiten auf eines der Produkte zu klicken. An der rechten Seite soll sich zusätzlich eine Leiste mit Produktkategorien befinden. Damit soll der Produktbrowser geöffnet werden, welcher dann nur Produkte einer bestimmten Kategorie anzeigt.

Diese Anordnung an Komponenten sorgt für zwei Effekte:

- Der Nutzer hat sofort einen Überblick über die verfügbaren Produkte und Produktkategorien
- Der Nutzer wird direkt gereizt, sich einige ausgewählte Produkte näher anzuschauen und diese gegebenenfalls in den Warenkorb zu legen

Somit wird die Kundenbindung und der Umsatz des Webshops möglicherweise bereits durch das Design der Startseite gesteigert.

Waren Browser

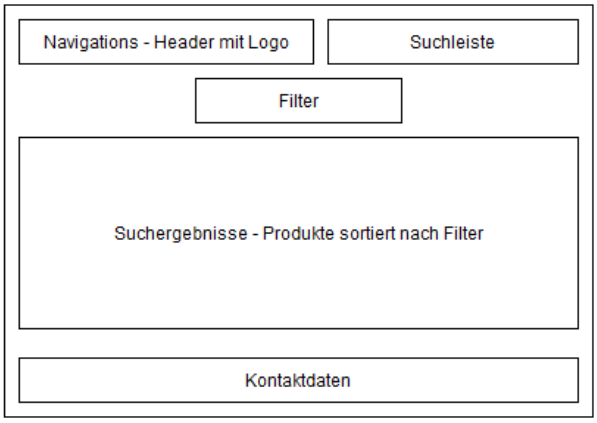

Figure 6: Wireframe des Produkt-Browsers des Webshops, Quelle: Eigene Darstellung

Der Produktbrowser folgt ebenfalls dem Standard-Design. Die Suchleiste der Navigationsleiste wird auch im Warenbrowser verwendet um nach Artikeln zu suchen. Die Suchergebnisse sollen in der Mitte der Seite angezeigt und aufgelistet werden. Jedes Produkt soll dabei eine klickbare Fläche darstellen, auf dem Name, Beschreibung und Preis des Produktes zu sehen sind. So wird der Nutzer direkt mit den wichtigsten Informationen zu jedem Produkt versorgt bevor es überhaupt angeklickt wird. Für den zusätzlichen Nutzerkomfort soll sich über den Produkten eine Filterleiste befinden. Hier kann der Nutzer die Produkte z.B. alphabetisch, nach Preis, oder nach verfügbarer Anzahl sortieren.

Produktdetails

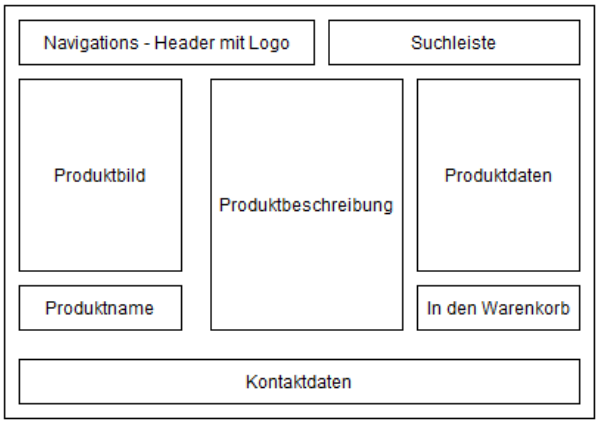

Figure 7: Wireframe einer Produktseite des Webshops, Quelle: Eigene Darstellung

Die Produktdetails werden angezeigt, nachdem der Nutzer auf eines der Produkte im Warenbrowser geklickt hat. Produkte im Webshop haben jeweils folgende Daten:

- Einen Produktnamen
- Ein Produktbild (optional)
- Eine produktbeschreibung (optional)
- Einen Bestand (wie viele Produkte sind verfügbar)

Das Produktbild wird, falls verfügbar, auf dieser Seite gerendert. Über den Button “In den Warenkorb” soll der Artikel direkt in den Warenkorb des Benutzers gelegt werden. Zusätzlich soll hier ein kleines Pop-Up Fenster erscheinen, welches die Aktion bestätigt.

Warenkorb

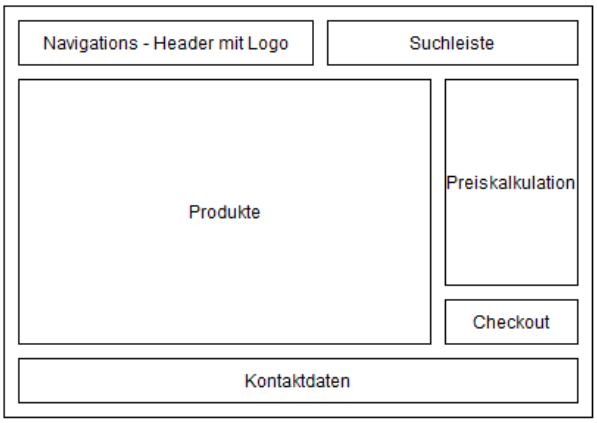

Figure 8: Wireframe des Warenkorbs des Webshops, Quelle: Eigene Darstellung

Der Warenkorb soll dem Nutzer einerseits zeigen, welche Produkte sich gerade im Warenkorb befinden, und gleichzeitig die Möglichkeit geben diese daraus wieder zu entfernen. Zusätzlich soll sich im Warenkorb eine Preiskalkulation befinden. Diese summiert die Preise der ausgewählten Produkte und zeigt dem Nutzer an, welcher Endpreis zu bezahlen ist.

Bibliography

[1] Sebastian Springer, “Svelte vs. Angular vs. React vs. Vue - wer gewinnt?.” Accessed: Jun. 11, 2024. [Online]. Available: https://entwickler.de/javascript/svelte-angular-react-vue

[2] “Bootstrap.” Accessed: Jun. 11, 2024. [Online]. Available: https://getbootstrap.com/

[3] “Tailwind CSS.” Accessed: Jun. 11, 2024. [Online]. Available: https://tailwindcss.com/

[4] John Ayebola, “CSS Frameworks vs Custom CSS – What's the Difference?.” Accessed: Jun. 11, 2024. [Online]. Available: https://www.freecodecamp.org/news/css-frameworks-vs-custom-css/

[5] Evan You and Vite Contributors, “Vite Guide.” Accessed: Jun. 11, 2024. [Online]. Available: https://vitejs.dev/guide/