Duale Hochschule Baden-Württemberg Ravensburg
Campus Friedrichshafen

**Web Engineering**

# Shopler

**Dokumentation**

Nico Dahlhaus
Dominik Klisa
Thibault Rey

14. März 2022

# Inhaltsverzeichnis

# 1 Themenfindung

Da wir alle regelmäßig einkaufen war die Idee an sich nicht fernab. In der heutigen Zeit ist sogut wie alles digital vorzufinden. Keiner läuft mehr mit Stift und Block durch die Gegend, so auch wir nicht. Oft genug werden beim Einkaufen wichtige Artikel vergessen die einem noch Tage und auch nur Stunden zuvor noch durch den Kopf gingen. Doch nur einen Schritt in den Laden getan und schon ist der Kopf leer. Brauch ich Nudeln? Hab ich noch Zwiebeln? Um das zu verhindern soll Shopler helfen. Shopler ist von überall auf jedem Gerät abrufbar. Artikel müssen nicht mühselig eingetippt werden wie in einer Notiz App, sondern können mit wenigen Klicks aus dem Katalog hinzugefügt werden. Das nächste Mal Einkaufen gestaltet sich so um vieles einfacher.

**Hat der Block ausgedient?** Für uns ja. Stift und Block sind beides Dinge die schnell vergessen und liegen gelassen werden können. Eine Einkaufsliste im Internet kann nicht vergessen werden. Außerdem wird Umweltschutz immer wichtiger, somit können wir hierdurch einen Beitrag zum erhalten der Wälder leisten, da sowieso so gut wie jeder mittlerweile ein Handy besitzt.

**Warum reicht die Notizen App des Handys nicht?** Der Aufbau einer Notiz App ist simpel. Nur soll eine Notiz App nicht nur als Einkaufsliste dienen. Das führt dazu, dass das Führen einer Einkaufsliste in einer Notiz App Disziplin benötigt. Artikel müssen jedes Mal aufs neue komplett eingetippt werden. Ein einfacher schneller Klick ist da nicht möglich.

**Gibt es nicht schon genug App die das auch können?** Ja und nein. Keiner von uns Entwicklern ist nur in einem Hersteller Universum unterwegs. Kaum eine App wird auf wirklich jedem Gerät unterstützt. Ein Browser besitzt aber heutzutage jedes relevante Gerät. Für das muss nicht einmal extra eine App heruntergeladen werden sondern kann auch an fremden Gerät notfalls genutzt werden.

# 2 Planung

Zu Beginn wird in einer kleinen Liste kurz definiert was wir als Entwickler an Eigenschaften und Funktionen erwarten.

## 2.1 Anforderungen

- Eine Einkaufsliste
  - Einzelne Einträge die mit einem Klick abgehakt werden können
- Einen Shop bzw. Katalog mit Artikelauswahl
  - An jedem Artikel einen Knopf der den jeweiligen Artikel zur Einkaufsliste hinzufügt
- Eigene Artikel die nicht im Katalog vorhanden sind hinzufügen
  - Formular mit Artikelnamen, Kategorien, Beschreibung

## 2.2 Aufbau

Die Website besteht aus einer festen Navigation, einer Willkommensseite die den User zu Beginn begrüßt und kurz erklärt wie Shopler funktioniert, einer Katalog Seite in der alle Artikel vorzufinden sind, einer Einkaufslisten Seite die die Liste des eingeloggten Users zeigt, einer Seite zum hinzufügen neuer Artikel und einer Login/Registrierungsseite zum anmelden und registrieren eines Users.[4]
Elemente:

- Navigation
- Willkommensseite
- Katalog/Shop
- Neuer Artikel
- Einkaufsliste
- Login/Registrierung

## 2.3 Skizzen

### 2.3.1 Richtlinien

Für dieses Projekt soll sich für das Design an den von Google entwickelten „Material Design“ Standard orientiert und angelehnt werden. Material ist ein Design System um Teams dabei zu unterstützen, hochwertige digitale Erlebnisse für das Web zu erstellen. Dabei ist Material Design von der physischen Welt und ihren Texturen inspiriert, einschließlich der Art und Weise, wie diese Licht reflektiert und Schatten wirft.
Google pflegt das Material Design und führt eine umfangreiche Dokumentation zu seiner Verwendung und Implementierung. Diese Art von Unterstützung und Dokumentation kann bei vielen modernen Konstruktions Systemen fehlen und ist daher der Grund warum sich in diesem Projekt an diesen Standard angelehnt werden soll.[1]

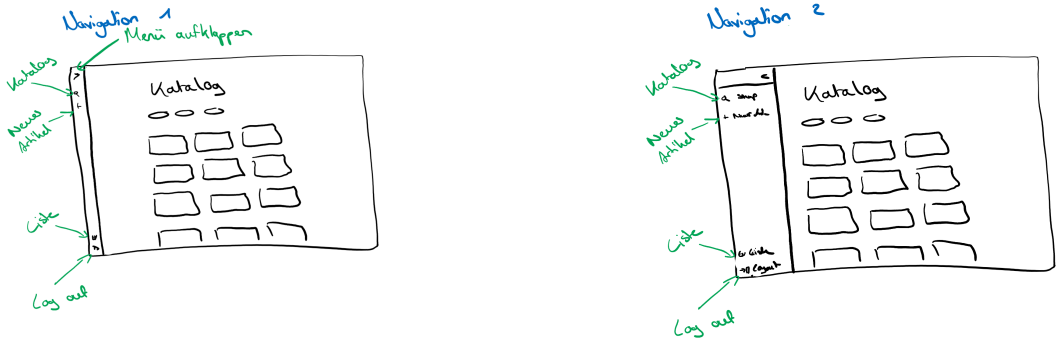
### 2.3.2 Navigation

Die eigentliche Navigation soll immer auf der linken Seite der Website dargestellt und erreichbar sein. Dabei soll es eine schmale und aufgeklappt Variante geben um Einsteigern durch die aufgeklappte Variante das Navigieren selbsterklärend zu gestalten und gleichzeitig durch die schmale Variante erfahrenen Nutzer wenig Platz wegnehmen und trotzdem ein direktes erreichen der einzelnen Pages einfach zu ermöglichen.

[1] Weitere Details: https://material.io/design

### Desktop

Jede Unterseite ist mit einem einzelnen Klick erreichbar. Sollten die Navigationspunkte unklar sein kann dieses wie bereits erwähnt aufgeklappt werden. Neben den Icons die immer dargestellt sind werden dann noch kurze Begriffe dargestellt.

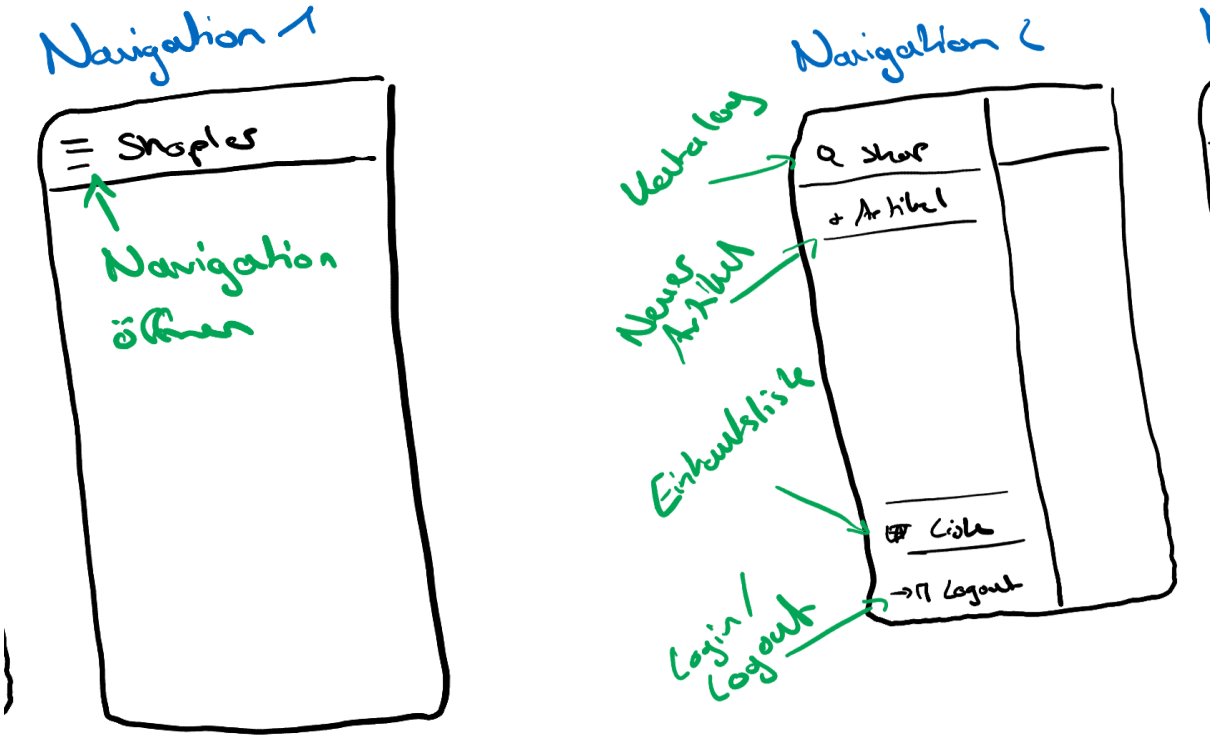



(a) Eingeklappt (b) Ausgeklappt

Desktop Navigation

### Mobil

Da kleinere Bildschirme auch in ihrer Darstellung begrenzter sind und das Bild nicht zu überladen sein soll verschwindet die Navigation komplett kann aber jederzeit durch eine Top Bar von links aufgeklappt werden.

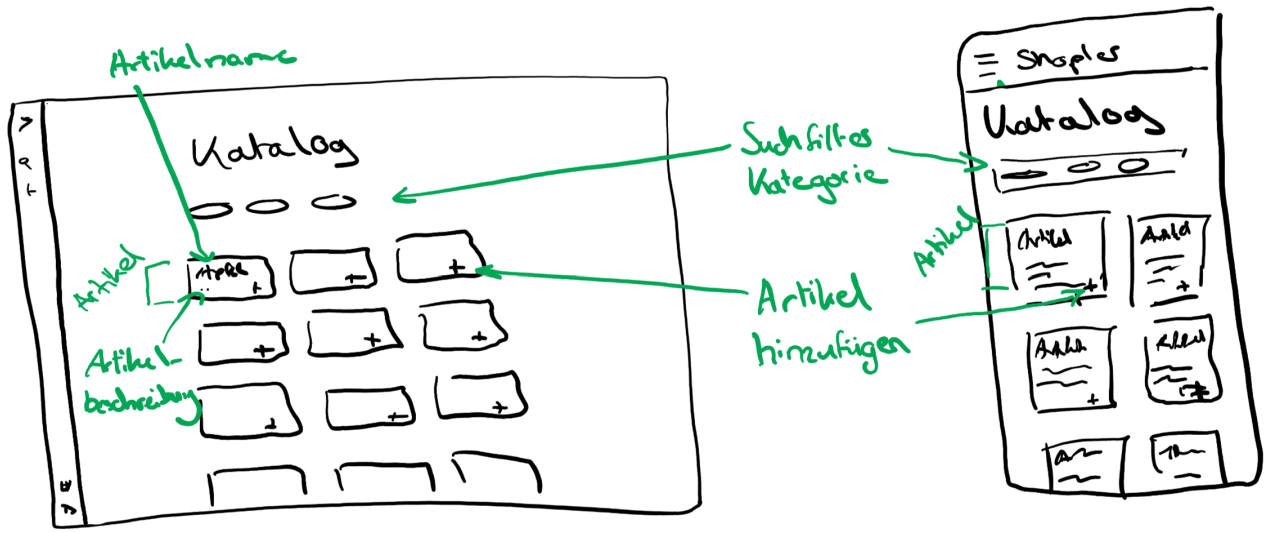



(a) Eingeklappt (b) Ausgeklappt

Mobile Navigation

### 2.3.3 Katalog

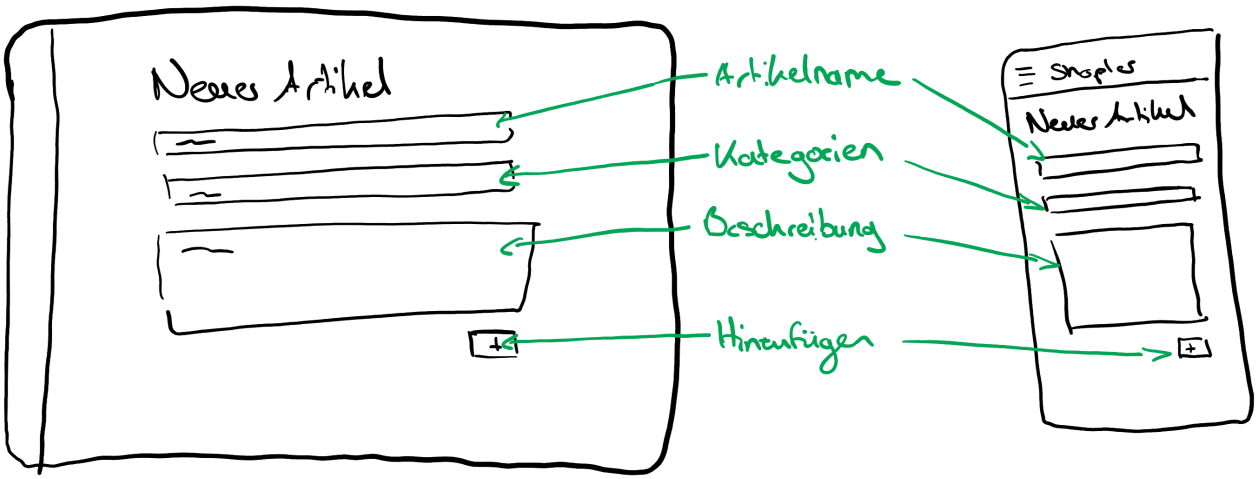
Im Katalog werden alle Artikel aufgelistet die zu einer Einkaufsliste hinzugefügt werden können. Zu jedem Artikel steht dafür der Artikelname, dessen Kategorien falls vorhanden und eine kleine Beschreibung des Artikels. Hinzugefügt werden können diese Artikel durch einen klick auf einen Button am Artikel.




Katalog Desktop und Mobil

### 2.3.4 Neuer Artikel

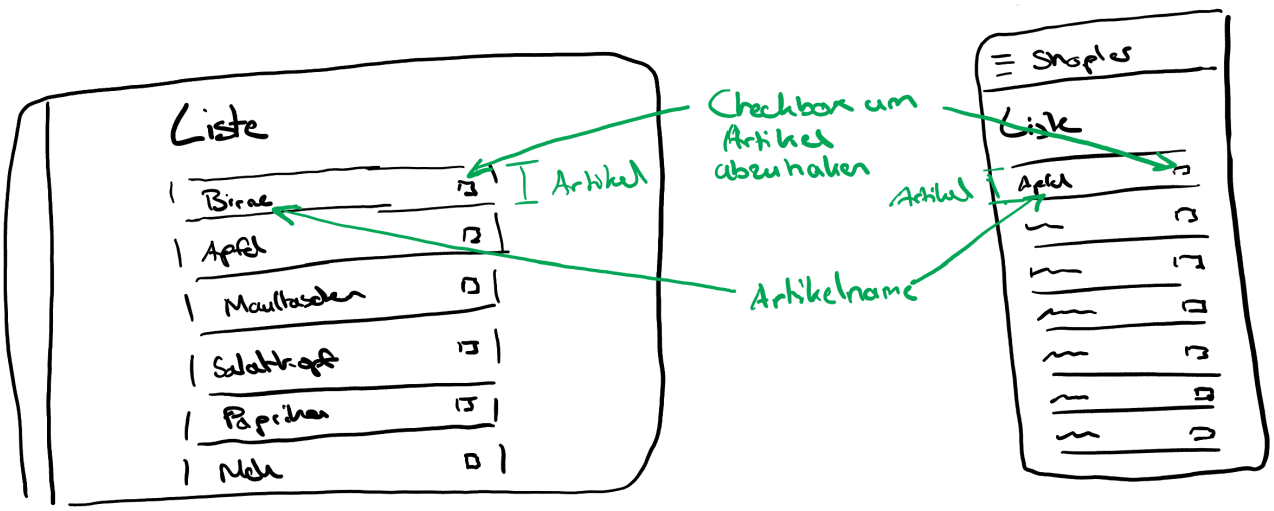
Bei „Neuer Artikel“ können neue eigene Artikel hinzugefügt werden die noch nicht im Artikel vorhanden sind. Dafür muss der Artikelname, eine kleine Beschreibung und Kategorien für den Artikel angegeben werden. Jeder neue Artikel taucht bei allen anderen Artikeln mit im Katalog für jeden mit auf.




Artikel hinzufügen Desktop und Mobil

### 2.3.5 Einkaufsliste

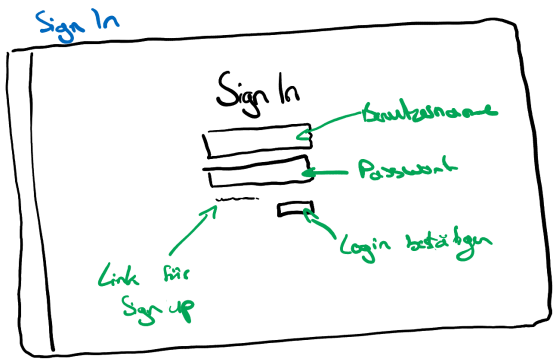
In der Einkaufsliste befinden sich alle vom User hinzugefügten Artikel aufgelistet in einer einfachen Liste untereinander. Jeder Artikel kann jeweils durch eine ein Häkchen an dessen rechten Seite abgehakt und entfernt werden.



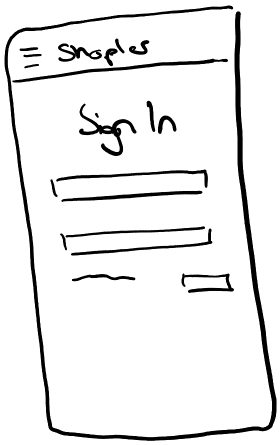

Einkaufsliste Desktop und Mobil

### 2.3.6 Einloggen / Ausloggen

In Login bzw. Ausloggen hat der User die Möglichkeit sich anzumelden, falls bereits ein Account besteht, oder einen neuen Account anzulegen. Dafür benötigt der User nur einen Benutzernamen und ein Passwort.

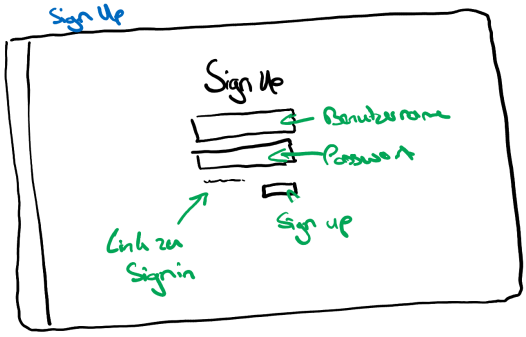



(a) Desktop Login




(b) Ausgeklappt

Login




(a) Desktop Registrierung


(b) Mobile Registrierung

Registrieren

# 3 Entwurf

## 3.1 Verwendete Komponenten/Bibliotheken/Frameworks

Zusätzlich neben Node.js kommen weitere Bibliotheken zum Einsatz. Dazu gehören React.js, React-Router-Dom, MUI und Express.js. Als Datenbank finden MariaDB Einsatz.

### 3.1.1 React.js

Ist ein Webframework und bietet die Möglichkeit von Interface-Komponenten. Diese Komponenten können wie HTML-Elemente verwendet werden. React dient für das Single-Page Erlebnis der Webanwendung. Dabei werden die Pages die von Node.js dem User zur Verfügung gestellt werden Clientseitig gerendert. Ein vorrendern ist auch möglich aber nicht für dieses Projekt relevant. Das Clientseitige Rendern bietet daher die Möglichkeit, den Inhalt der Seite dynamisch zu verändern ohne dass die komplette Seite neu geladen werden muss. Das Navigieren, Bestätigen und Einloggen fühlt sich daher für den Anwender viel mehr wie eine Anwendung an.

### 3.1.2 React-Router-Dom

Mit Hilfe von React-Router-Dom kann das Navigieren auf der Website noch dynamischer gestaltet werden. React-Router-Dom bietet die einfache Möglichkeit zwischen verschiedenen React Komponenten zu wechseln. Dazu wird die URL angepasst und ebenso bei manueller Eingabe der URL beachtet.

### 3.1.3 MUI

Material UI liefert von sich aus eine große Auswahl an verschiedensten React Komponenten die auf den Material Design Standard von Google basieren. Das Entwickeln von React Anwendungen gestaltet sich mit MUI strukturiert und einheitlich.[4]

### 3.1.4 Express.js

Express ist ein Webframework und unterstützt Node.js serverseitig. Express erweitert die Basisfunktionalität von Node.js mit weiteren Werkzeugen die das Entwickeln von modernen Webanwendungen einfacher gestalten. Bei Shopler wird Express verwendet um einen Datenaustausch zwischen Client und der Backend Datenbank zu ermöglichen.

### 3.1.5 MariaDB

MariaDB ist ein freies, relationales Datenbankmanagementsystem. Kommuniziert wird mit Hilfe von SQL. Bei Shopler wird MariaDB verwendet um Artikel, User und Einkaufslisten zu verwalten und abspeichern.

## 3.2 Architektur

### 3.2.1 Aufbau

Im Groben besteht die Anwendung aus drei Komponenten. Zum einen der Frontend Server, der mit dem React Framework implementiert wurde. Dann das Backend, was mit Express.js realisiert wurde. Und zur Haltung der Daten eine MariaDB Datenbank.
Hierbei kommunizieren alle Komponenten untereinander mithilfe von HTTP. Zur Übertragung der Daten wurde eine REST API entwickelt, die im Backend die Daten bei Anfrage von der Datenbank abruft und an das Frontend übermittelt. Außerdem können über die API neue Daten angelegt werden, bzw. vorhandene manipuliert oder gelöscht werden.


Komponentendiagramm der Architektur

Durch die strikte Trennung des Front-, Backends und der Datenbank ist es möglich die drei Komponenten auf physikalisch verschiedenen Servern zu hosten. Dadurch kann das Frontend auch als native Applikation auf Android, iOS, Windows, Linux und macOS bereit gestellt werden.

### 3.2.2 Sicherheit

#### Passwort Speicherung

Zur sicheren Speicherung von Passwörtern wurde der Hash Algorithmus mit dem gleichnamigen npm Modul Argon2 verwendet. Dieser war 2015 der Gewinner der Password Hashing Competition.[1] Das Modul erzeugt Automatisch einen Hash und Salt für das Password und speichert ihn in einem String mit anderen Metainformationen, wie bspw. der Version von Argon2. In unserem Fall wird das Passwort an das Backend übertragen und dort gehashed.

#### Benutzerauthentifizierung

Zur Speicherung von Sitzungen und zur Authentifizierung von API Anfragen wurden zwei JSON Web Token (JWT) verwendet. Ein JWT wird für alle API Anfragen verwendet und im folgenden einfach „Token“ genannt. Da mit diesem Token alle Daten abgerufen werden können, ist dieser mit Vorsicht zu behandeln. Daher hat er eine kurze Lebensdauer von nur 15 min und wird auf Frontend Seite ausschließlich in einer Variable gespeichert. Dies erschwert es Angreifern den Token auszulesen und durch die kurze Lebensdauer kann er nur kurz verwendet werden. Um jedoch eine gute Benutzererfahrung bieten zu können, wird ein zweiter JWT (im Folgenden „Refresh Token“ genannt) verwendet. Dieser kann nur dafür verwendet werden, um sich anzumelden und einen neuen Token zu erhalten.[2] Da mithilfe des Refresh Tokens keine Daten abgefragt werden können, kann beim Speichern des Tokens eine einfachere Methode gewählt werden. In unserem Fall wird der Refresh Token im Local Storage des Browsers gespeichert. Das hat den Vorteil, dass auf Cookies verzichtet werden können, da die Anforderung an Cookies durch die Browser in den letzten Jahren immer Strenger geworden ist. Cookies würden zwar eine leicht erhöhte Sicherheit bieten, da jedoch zwei Token verwendet werden, spielt dies nur eine kleine Rolle. Damit sich der Nutzer mit verschiedenen Endgeräten angemeldet bleiben kann, wird für jedes Gerät ein Refresh Token generiert und in der Datenbank gespeichert. Beim ausloggen wird dieser Refresh Token wieder gelöscht und man muss sich erneut anmelden. Dieses verwendete System bietet Vor- und Nachteile. Ein Authentifizierungsanbieter wie Auth0 würde die Authentifizierung stark vereinfachen, da er bereits vorgefertigte Komponenten und Strategien liefert. Jedoch würde zum einen die Benutzerdaten an den Anbieter gesendet werden und zum anderen wird ab einer gewissen Benutzeranzahl Geld verlangt.
Eine weitere einfache Möglichkeit wäre das Nutzen von nur einem JWT. Dies würde die Entwicklung ebenfalls stark vereinfachen und man könnte fast ausschließlich schon vorhandene Module verwenden. Jedoch wäre hierdurch entweder die Sicherheit, oder die Benutzerfreundlichkeit stark beeinträchtigt.

### 3.2.3 Datenbank

Als Datenbank wurde MariaDB gewählt, da hier bereits Vorerfahrung aus der Datenbankvorlesung vorherrschte. Zudem ist die Datenbank Open Source und hat eine sehr große Community. Für das Projekt wurden dabei fünf Tabellen angelegt, um alle Daten speichern zu können:

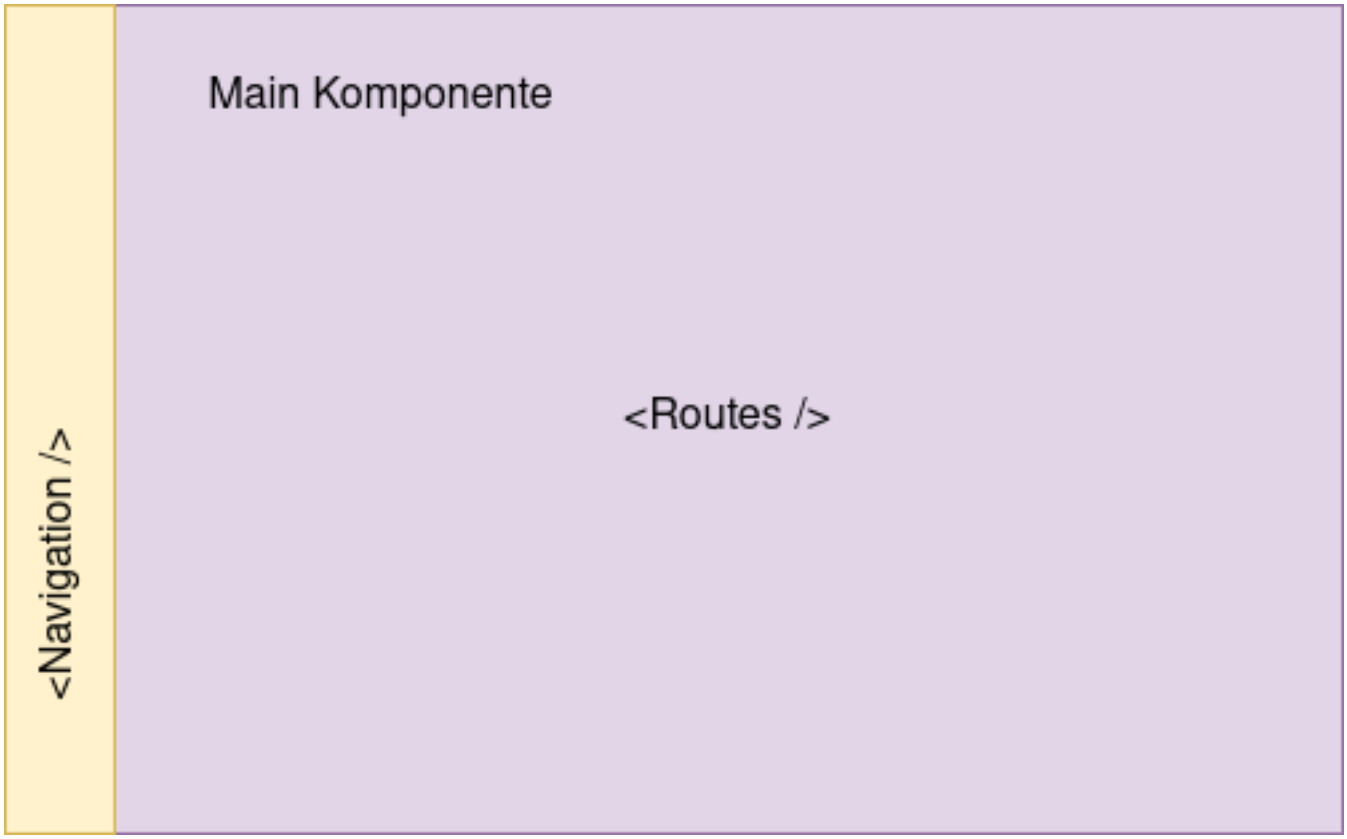
- cart_items: Die Artikel der Einkaufsliste.
- categories: Die Kategorien, die Artikel haben können.
- items: Die Artikel im Katalog.
- item_cat: Zuordnung zwischen Artikel und Kategorien.
- users: Die Benutzer der Anwendung.


ERM-Model der Datenbank

## 3.3 Benutzerinterface

### 3.3.1 Struktur

Dank React und React-Router-Dom fühlt sich das Benutzererlebnis noch besser an. Die Navigationsleiste verhält sich daher als Beispiel wirklich statisch und nur die Subpages werden dabei gewechselt. Folgende Struktur wird dadurch möglich.

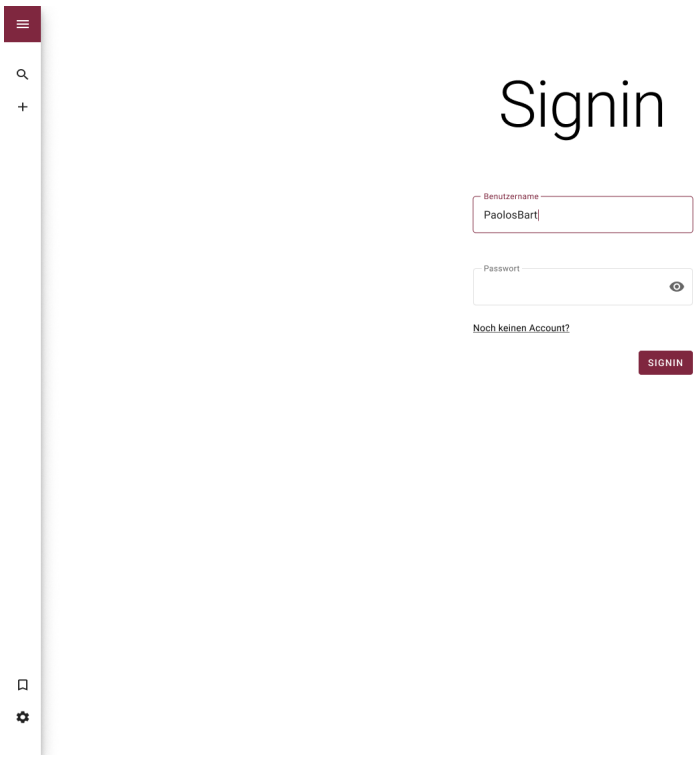



Struktur

Um auf Shopler die Pages zu wechseln werden nur die Routes ausgetauscht. Die Navigation bleibt auf jeder Seite gleich. Die Pages sind dabei die Willkommensseite, der Katalog, die User Einkaufsliste, das Hinzufügen neuer Artikel und das SignIn bzw. SignUp.

### 3.3.2 Design

Das komplette Design wird nochmals aus den Skizzen überführt in Figma bevor es tatsächlich realisiert wird. Figma ist ein Vektor Grafik Editor und Prototyping Tool das primär im Web läuft.[5] [1]

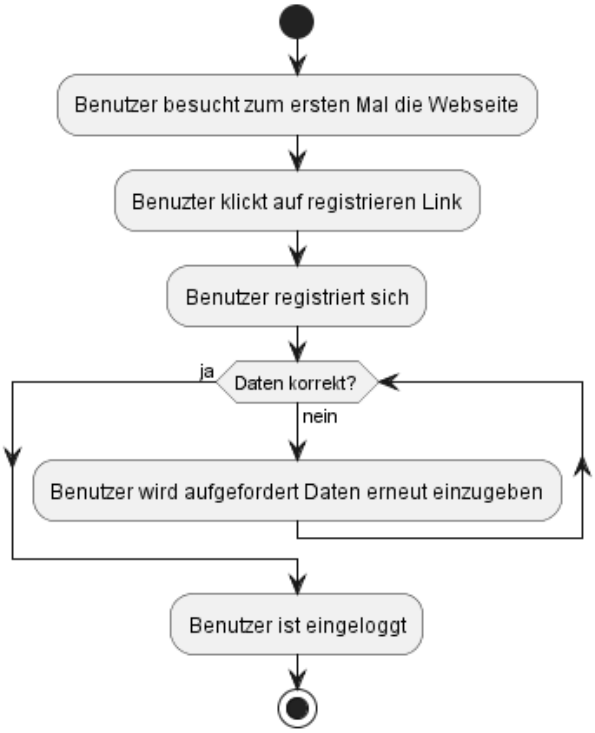



Neuer Artikel

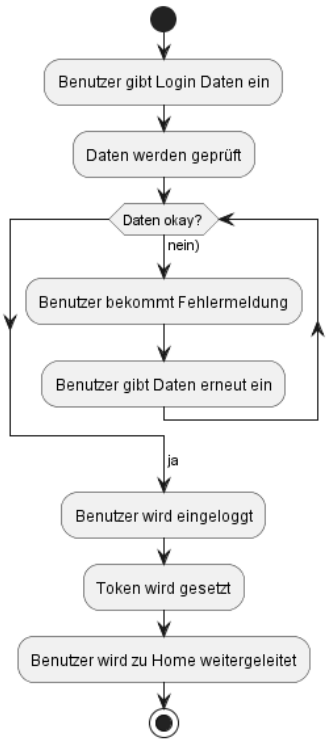
Um die Webseite optisch ansprechender zu gestalten wurden SVG Grafiken genutzt, welche das Projektteam von der Webseite free svg illustrations [3] erhalten hat.

[1] Weitere Skizzen: https://www.figma.com/file/VMlpILkl2wBpK8pE2Cidb7/shopler?node-id=0%3A10

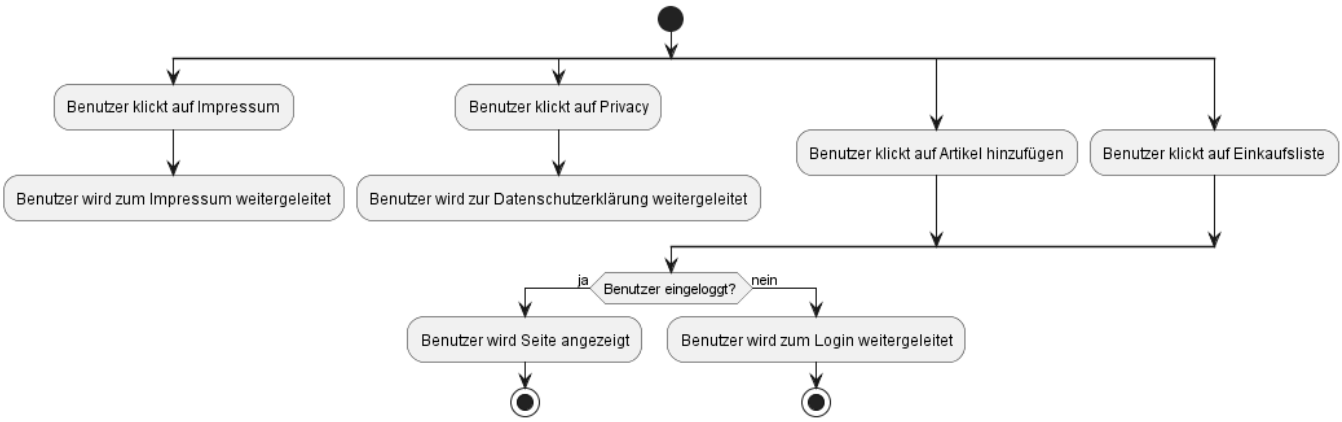
### 3.3.3 Flow-Charts




Erster Besuch




Login


Benutzer klickt auf Impressum
Benutzer klickt auf Privacy
Benutzer klickt auf Artikel hinzufügen
Benutzer klickt auf Einkaufsliste
Benutzer wird zum Impressum weitergeleitet
Benutzer wird zur Datenschutzerklärung weitergeleitet
ja
Benutzer eingeloggt?
nein
Benutzer wird Seite angezeigt
Benutzer wird zum Login weitergeleitet

Navigation

# 4 User Tests

Die User Tests, die Projekt Fremde Kommilitonen der DHBW durchgeführt haben, resultieren in positive wie negative Ergebnisse.

- negativ
  - Was schnell auffiel das jegliche Form eines Impressums oder einer Datenschutzseite viel
  - Den Usern kamen die Seiten zu abgehakt vor. Besser wäre ein klar erkennbares Ende wie eines farblich getrennten Footers.
  - Nicht jedem ist von Anfang an klar wie sich der Ablauf des Nutzen des Tools gestaltet
- positiv
  - Die Navigation ist dank Items und notfalls Text schnell durchschaut. Nach wenigen Umschauen ist die Funktion der Navigationsleiste durchschaut.
  - Die Bedienung des Tools ist intuitiv und so simpel gehalten dass ein schnelles verwenden möglich ist, auch wenn dieser nicht von jedem direkt durchschaut wird reichen nur wenige Anläufe.

Aufgrund der erkannten Schwäche wurde in einem zweiten Entwicklungsschritt ein Footer noch hinzugefügt der auf die Home Seite, das Impressum und den Datenschutz verweist.
Mit Hilfe des Tutorials auf der Home Seite ist die Idee des Projekts jetzt auch für jeden schnell erkannt.

# 5 Installationsanleitung

Benötigt wird:

- node.js (getestet mit v. 17.5.0)
- npm (getestet mit v. 8.3.1)

Ablauf:

1. Das Projekt entpacken.
2. Per Konsole in den „frontend“ Ordner und “npm ci” ausführen
3. In „frontend“, mit „npm start“ den React Server starten
4. Per Konsole in den „server“ Ordner und „npm ci“ ausführen
5. In „server“, mit „npm start“ den Express Server starten

Hinweis:
Falls das Projekt komplett offline ausgeführt werden soll, muss zusätzlich MariaDB installiert werden. Eine Datenbank „shopster“ angelegt werden und ein Benutzer mit allen Rechten auf dieser Datenbank. Die Zugangsdaten müssen anschließend in der „.env“ Datei im Ordner „server“ eingetragen werden. Ebenfalls müssen die Kategorien per Hand in die Datenbank eingetragen werden, da diese global vorhanden sein sollen. Die Erstellung der Tabellen und Relationen passiert automatisch beim Starten des Servers.
Falls das Projekt Zugang zum Internet hat, können die bereits eingetragenen Zugangsdaten verwendet werden. Diese verweisen auf einen MariaDB Server, der auf einem Raspberry Pi von einem der Projektmitglieder gehostet wird.

# 6 Literatur

[1] Jean-Philippe Aumasson. *Password Hashing Competition*. URL: https://www.password-hashing.net/.

[2] Abhishek EH. *Authentication in React using Express, Node, Passport and MongoDB*. URL: https://www.codingdeft.com/posts/react-authentication-mern-node-passport-express-mongo/.

[3] Wizard Infosys. *Free illustrations*. URL: https://freesvgillustration.com/.

[4] *React Templates*. URL: https://mui.com/getting-started/templates/.

[5] Material-UI SAS. *All Components*. URL: https://mui.com/components/.