PROJEKTDOKUMENTATION

DYNAMISCHE WEBSITE

Unter Betreuung von Herrn Tatar und Herrn Seitz.

erstellt von

Mattis Willy Karg - 7390050

Simon Sigl - 2290813

Thorsten Licht - 2460385

Jonas Herreiner - 2258110

Inhalt

1. Rückblick

Im ersten Semester entwickelten wir anfangs unsere Vision sowie die Geschäftsidee, welche unser fiktives Unternehmen Delivery-Breakfast verfolgt. Um einen imposanten Internetauftritt zu designen, erstellten wir eine statische Homepage. Bei dieser stand für uns das grafische Design im Vordergrund, da bei unseren Frühstücksprodukten ja bekanntlich auch „das Auge mitisst".

Während der Erstellung dieser Website mit den einfachen Mitteln wie CSS- und HTML-Code sammelten wir auch schon einige deutlich kompliziertere Komponenten zur Weiterentwicklung. Diese ganzen Ideen hielten wir in einem Backlog fest, um sie in diesem Semester zu realisieren. Exemplarische Ideen sind der Warenkorb oder die Verwendung eines Frameworks.

Durch die Erfahrungen aus dem ersten Semester wussten wir bereits beim Kickoff der dynamischen Homepage, wo die Stärken unserer Teammitglieder liegen. Somit konnten wir bereits von Beginn an die Arbeit produktiver bewältigen.

2. Corporate Identity

Durch „delivery of breakfast" kannst Du unbesorgt in deinen Tag starten. Wir sorgen dafür, dass Dein Frühstück zusammen mit deiner Zeitung zu Dir oder zu Deinem Arbeitsplatz geliefert wird. Unser Angebot richtet sich speziell an Personen allen Alters, welche auf dem Land leben und keinen Bäcker in der Nähe haben.

Wir leben Nachhaltigkeit. Diese beginnt bei unseren wiederverwendbaren Behältern und geht bis zur emissionsarmen Lieferung. Durch die Kooperation mit der „Fake-Zeitung" wird die bereits bestehende Infrastruktur genutzt, somit werden keine weiteren Abgase verursacht. Auch bei der Herkunft unserer Lebensmittel achten wir auf kurze Lieferwege und unterstützen die heimischen Lebensmittel-produzenten.

3. Hilfsmittel

3.1. SCRUM

Nach langer Analyse der verschiedenen Vorgehensmodelle haben wir uns schließlich für SCRUM entschieden, da dieses viele Vorteile bietet und all unsere Anforderungen erfüllt.

Die Rollenaufteilung lautete wie folgt: Product-Owner waren Herr Tatar und Herr Seitz, welche uns die beiden Anforderungen „Programmierung einer dynamischen Webseite" und „Benutzung eines Frameworks" stellten. Simon Sigl übernahm die Rolle des SCRUM-Masters und bildete sich sogar speziell für diese Rolle fort. Das Team bestand aus Thorsten Licht, Mattis Willy Karg und Jonas Herreiner. Durch diese Kombination haben wir uns im Laufe des Projektes ein gutes T-Shape Wissen aneignen können.

Wir teilten das Projekt in insgesamt drei verschiedene Sprints auf, alle mit einer gleichen Dauer. Im ersten beschäftigten wir uns mir folgenden Themen: Einführung in Frameworks, damit verbunden der Übertrag der bestehenden Inhalte ins neue Framework sowie mit dem Brainstorming und der Gewichtung der neuen Features. Im zweiten Sprint kümmerten wir uns um die Implementierung der neuen Features und um die Fehlerbehebung. Zuletzt widmeten wir uns im dritten Sprint den optionalen Features sowie finalen Designthemen. Zudem fokussierten wir uns auch auf den Pitch und die Finalisierung der Dokumentation.

Jeder der drei verschiedenen Sprints begann mit der Sprintplanung. Hier verschoben wir gemeinsam die Karten vom Produkt- in den Sprint Backlog, welche wir in diesem Sprint umsetzen wollten. Da wir dieses Projekt neben dem Studienalltag umsetzten, einigten wir uns auf Stand-UP Meetings zweimal in der Woche, nicht wie üblich täglich. Am Ende jedes Sprints trafen wir uns für einen Kombitermin, welcher einerseits das Sprintreview, anderseits auch die Sprintretrospektive beinhaltete. Alles mit dem Ziel, dass der kommende Sprint noch produktiver und erfolgreicher ablaufen wird.

Wir benutzen zudem Trello - ein digitales Aufgabenverwaltungstool - und damit verbunden das Kanban Prinzip, um die verschiedenen Aufgaben zu monitoren. Der Durchlauf der verschiedenen Karten beginnt links im Produkt Backlog, aus dem wir dann bei der Sprintplanung jeweils die relevanten Themen in den Sprint Backlog verschoben. Die Tasks wurden während der Bearbeitung in der „Doing"-Spalte plaziert und wurden nach Beendigung in den Testkatalog verschoben. Erst nachdem jedes Teammitglied die Themen an verschiedensten Endgeräten getestet hatte, wurde die Karte nach „Done" verschoben. Durch das synchrone Board hatte jeder aus dem Team stets Einsicht in den Fortschritt der anderen und konnte bei Bedarf durch die Kommentarfunktion direkt Feedback geben. Nachfolgend ist ein exemplarischer Auszug unseres Kanban-Boards dargestellt.

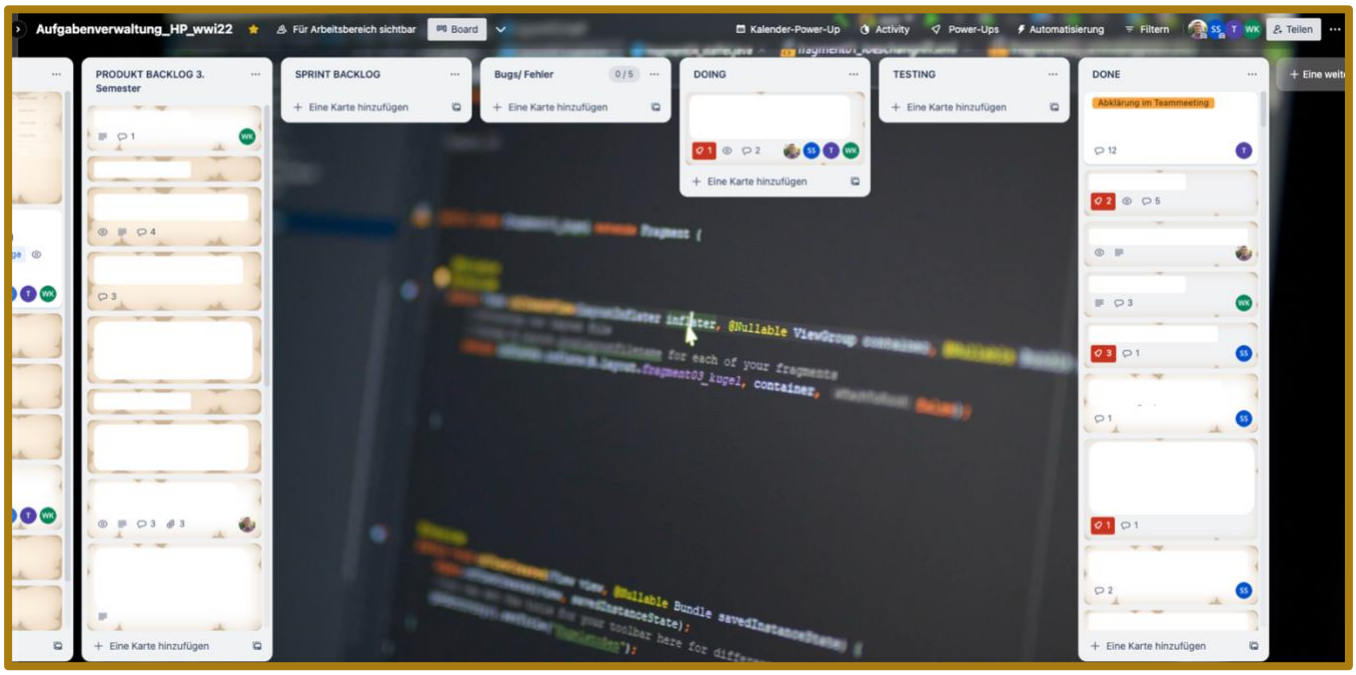

Abbildung 1: Auszug Trelloboard

3.2. Recherche

Durch künstliche Intelligenzen wie ChatGPT und GPT-4, welche bekanntlich zurzeit in aller Munde sind, konnten wir unser Wissen zu TypeScript, dem Framework React sowie zu den anderen Themen rund um die dynamische Homepageprogrammierung erweitern und somit auch die größten Herausforderungen meistern. Zudem half auch YouTube, die offizielle Redux- und die React-Router Website bei der Implementierung und Problemlösung.

3.3. Quellcodeverwaltung

Für die Synchronisierung unseres Codes verwenden wir die Software Git, sodass jedes Teammitglied stets die aktuelle Version bearbeiten kann. Zudem führt Git eine Historie mit allen Änderungen, mithilfe derer man nachvollziehen konnte, welche Änderungen die anderen vorgenommen hatten.

Zur Bearbeitung unseres Quellcodes verwenden wir den Quelltext-Editor Visual Studio Code von Microsoft, somit können die Änderungen am Code direkt aus dem gleichen Programm synchronisiert werden.

ESLint, ein Werkzeug zur statischen Quellcodeanalyse, hat nur sehr wenige Formatierungsfunktionen für Code, weswegen wir uns dazu entschieden haben, Prettier als Code-Formatter zu verwenden. Außerdem benutzen wir das Plugin TabNine AI. Dies ist eine künstliche Intelligenz, welche den Code intelligent ergänzt.

3.4. Geschwindigkeitstest

Während des Baus analysierten wir mit Google PageSpeed Insights durchgehend die Leistung der einzelnen Unterseiten und fixierten uns auf die Verbesserung der Geschwindigkeit sowie der Barrierefreiheit.

Die Geschwindigkeit unserer Website liegt uns sehr am Herzen, da vor allem die ländliche Zielgruppe nicht immer Zugang zu einer schnellen Internetverbindung hat und wir diesen Kunden ebenfalls eine perfekte Benutzerexperience bieten möchten. Besonders die Umformatierung der einzelnen Bilder in das .webp-Format sowie den Einbau von Lazy-Loading (zum Kapitel) führten zu einer enormen Geschwindigkeitssteigerung auf der gesamten Homepage.

Auch bei der Barrierefreiheit erhielten wir einige Verbesserungsvorschläge, wie beispielsweise die Vergrößerung der verschiedenen Icons im Footer auf mindestens 48 x 48px. Durch diese Umsetzung konnten wir ebenfalls die Gesamtqualität unseres Internetauftritts steigern.

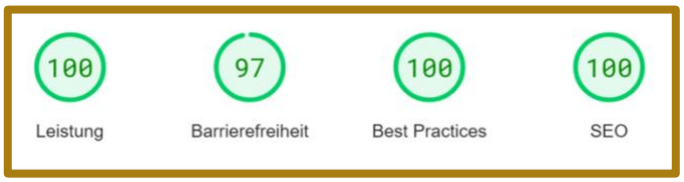

Abbildung 2: Leistungsdaten der Desktopversion

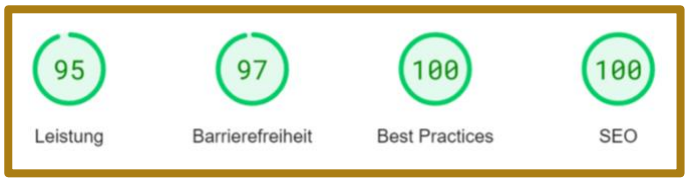

Abbildung 3: Leistungsdaten der mobilen Version

3.5. Bilder

Unsere verwendeten Bilder sind von der KI der Website canva.com generiert worden, danach folgte eine Verkleinerung der Grafiken und teils eine Bearbeitung in Adobe Express. Speziell das Hintergrundbild der Startseite, unser Logo sowie die verschiedenen Icons haben durch die Nachbearbeitung eine deutliche Leistungssteigerung erhalten. Abschließend erfolgte dann die bereits erwähnte Umwandlung ins .webp Format.

4. Von der statischen zur dynamischen Website

4.1. Bestellübersicht

Eine der vielen dynamischen Benutzerinteraktionen ist der Bestellvorgang unserer Produkte, welcher auf der Produktseite beginnt. Durch das Hinzufügen ausgewählter Produkte in den Warenkorb werden diese mit Menge, Stückpreis und dem daraus errechneten Gesamtpreis im Warenkorb aufgelistet. Zudem werden neben dem Icon im Menü, welches die Unterseite des Warenkorbs repräsentiert, die Zahl aller im Warenkorb befindlichen Produkte visualisiert.

4.2. Useranmeldung

Auch der Registrierungsprozess unserer Kunden wurde dynamisch implementiert. Auf dieser Seite werden die Daten des registrierten Benutzers angezeigt. Sobald der Anmeldezustand zutrifft, wird einerseits das Icon angepasst und die Anmeldeseite transformiert zur Übersicht des eigenen Benutzerkontos. Nur nach erfolgreicher Anmeldung kann der Kunde seinen Kauf tätigen und somit die Bestellung abschließen.

4.3. APIs

Wir haben verschiedene APIs in unsere Homepage eingebunden, welche dem Benutzer ein spürbar besseres Benutzererlebnis bei der Interaktion mit der Homepage gibt.

Eine dieser APIs ist „Google Maps Embed API“. Durch die Kontaktseite erreichen wir eine große Transparenz unseres Unternehmens, da den Kunden verschiedene Möglichkeiten aufgezeigt wird, mit uns zu kommunizieren und uns zu finden. Hierzu trägt wesentlich auch die Einbettung der Landkarte bei. Hier hatten wir die Auswahl zwischen der OpenStreetMap und der API von Google Maps, wir entschieden uns für die letztere. Diese zeigt anschaulich den Unternehmenssitz und ermöglich durch einen Direktlink zum Routenplaner, dass unsere Kunden schnell und einfach die beste Wegbeschreibung zu unserem Hauptsitz erreichen.

Eine andere API ist der Chatbot des Unternehmens Chatra, dieser wird in einem separaten Kapitel vorgestellt (zum Kapitel).

5. React

Das Framework React sorgt für ein effizientes Rendern der verschiedenen dynamischen Inhalte unseres Internetauftritts. Deswegen können wir nahtlos diese Art von Content einbauen, welche sich basierend auf den Benutzerinteraktionen automatisch aktualisieren. Durch die Verwendung von Virtual DOM werden nur die geänderten Inhalte der Benutzeroberfläche neu gerendert, was eine bessere Leistung sowie ein reibungsloses Benutzererlebnis zur Folge hat.

Wir verwenden React zusammen mit der Sprache TypeScript, welche sehr gut miteinander harmonieren. Die nachfolgende Übersicht stellt den Anteil der verschiedenen, von uns verwendeten Sprachen dar:

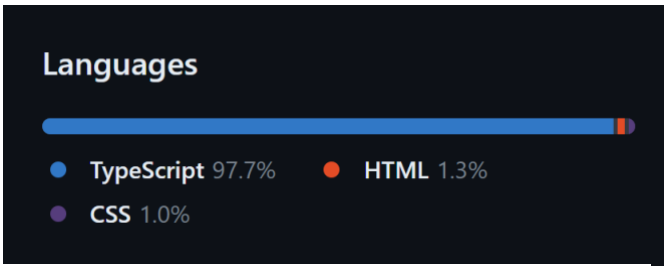

Abbildung 4: Anteil der einzelnen Sprachen

5.1. Styled Components

Die Styled Components Bibliothek von React ermöglicht es, dass wir unseren JS-Code, welcher für das jeweilige Aussehen der Komponenten verantwortlich ist, zu den betreffenden Seiten speichern können. Diese Methode vermeidet somit eine große, unübersichtliche CSS-Datei, welche den Code für die gesamte Homepage implementiert. Somit konnten wir zudem auch anfängliche Namenskonflikte lösen. Ein weiterer Vorteil ist der begrenzte Wirkungsbereich des Codes, da nur die direkte Komponente gestylt wird.

5.2. React Router

Eine weitere verwendete JavaScript Bibliothek ist der React-Router. Diese ermöglicht ein URL-spezifisches Rendern der Homepage. Somit werden nur geänderte Komponenten neu geladen und gleichbleibende wie beispielsweise der Footer werden nicht aktualisiert.

Das nachfolgende Beispiel zeigt das Hauptmenü, wobei <Layout/> nur einmal zu Beginn gerendert wird. Die übrigen Komponenten aktualisieren sich abhängig von der URL.

```
<BrowserRouter>
  <Routes>
    <Route path="/" element={<Layout />}>
      <Route index element={<Home />} />
      <Route path="Produkte" element={<Produkte />} />
      <Route path="Bestellung" element={<Bestellung />} />
      <Route path="Unsere Geschichte" element={<UnsereGeschichte />} />
      <Route path="Kontakt" element={<Kontakt />} />
      <Route path="LogIn" element={<LoginForm />} />
      <Route path="SignUp" element={<SignUp />} />
      <Route path="LoggedIn" element={<DeinKonto />} />
```

Abbildung 5: Codebeispiel Menü

Dadurch, dass nur die notwendigen Elemente aktualisiert werden, schafft diese Art des Renderns einen großen Performancevorteil, welcher für den Benutzer deutlich spürbar ist, zudem verringert sich auch die Netzwerklast. Zur Umsetzung dieses Konzepts benutzt der React Router den Virtual Dom.

5.3. Material UI

Material UI ist ein von React verwendbares Designframework, welches vorgefertigte Komponenten zur Gestaltung von einzelnen Seiten bereitstellt. Diese Komponenten sind bereits automatisch responsiv. Trotz der ganzen voreingestellten Werte können Details im Nachhinein selbst festgelegt werden, dies ist im nebenstehenden Code ersichtlich.

Dieses Framework kam vor allem bei den neuen Seiten, welche in der statischen Version der Website noch nicht existierten, zum Einsatz.

```
<TextField
  margin="normal"
  fullWidth
  id="email"
  label="Email Address"
  name="email"
  autoComplete="email"
  autoFocus
  required
  inputProps={{
    maxLength: 50,
  }}
  InputLabelProps={{
    sx: {
      backgroundColor: "white",
    },
  }}
/>
```

Abbildung 6: angepasste Material UI Komponente

5.4. Redux

Zur Verwaltung der dynamischen Elemente auf unserer Homepage greifen wir auf die JavaScript-Bibliothek Redux zurück. Diese speichert die Variablen in sogenannten Stores und ermöglicht somit einen späteren Zugriff. Die Änderung der Inhalte kann nur mittels eines Reducers sowie einer Action erfolgen. Der nachfolgende Codeausschnitt erklärt den Reducer und die Action beim Anlegen neuer Benutzer.

```
export const addNewUser = (LogInData: LogInData) => {
  return {
    // Angabe es soll den Case ADD_NEW_USER ausführen
    type: ActionTypesUser.ADD_NEW_USER,
    // payload der benutzt werden soll Funktionsparameter
    payload: LogInData,
  };
};
```

Abbildung 7: Action

```
const userReducer = (
  //der Default State sind hier leere Parameter
  state: UserDataState | undefined = {
    LogInData: { firstName: "", lastName: "", email: "", password: "" },
  },
  //Die Action müssen diese return Werte haben
  action: { type: ActionTypesUser; payload: LogInData }
) => {
  switch (action.type) {
    // Fall neuen nutzer anlegen
    case ActionTypesUser.ADD_NEW_USER:
      return {
        //der neue State wird übergeben
        ...state,
        // LogInData state wird gleich dem action.payload
        LogInData: action.payload,
      };
    default:
      return state;
  }
};
```

Abbildung 8: Reducer

Insgesamt existieren drei Redux-Stores, welche die Adressdaten, die Benutzerdaten sowie die Bestellinformationen speichern, damit diese Informationen auf anderen Seiten ebenfalls eingeblendet werden können. Eine dieser Seiten ist der „Warenkorb“. Hier werden alle Bestellinformationen an einer zentralen Stelle angezeigt und der Kunde kann diese nochmals vor dem finalen Absenden der Bestellung auf Korrektheit prüfen. Der nebenstehende Ausschnitt des Codes zeigt die Abbildung des Bestellstatus.

```
<BestellungsWrapper>
  {cartItems.map((item, index) => (
    <Warenkorb
      key={index}
      image={item.logo}
      price={item.preis}
      onRemove={() => handleRemoveItem(item)}
      productName={item.produktname}
      count={item.anzahl}
    />
  ))}
</BestellungsWrapper>
```

Abbildung 9: Speicherung Bestellstatus

5.5. QR Code Generator

Sobald eine Bestellung erfolgreich durch den Kunden abgeschlossen wurde, erscheint ein Popup. In diesem bedanken wir uns für die Bestellung und bieten dem Kunden die Möglichkeit, dass er den Lieferweg seines Frühstücks tracken kann. Um die Komplexität des Nachverfolgens zu minimieren und das Benutzererlebnis zu steigern, verwendeten wir eine React Komponente, welche den langen Trackinglink dynamisch in einen QR-Code umwandelt. Dieser kann nun ganz einfach abgescannt werden, ohne dass irgendwelche langen Links im Browser eingetippt werden müssen. Dieser „Frühstückstracker“ verlinkt dann auf eine OpenStreetMap, auf der man die aktuelle Position seines Frühstücks erkennen kann. Somit kann ich mein Frühstück bereits auf dem Weg in die Arbeit abholen und muss nicht warten, bis das Paket an der Lieferadresse angekommen ist.

Sollte der Kunde jedoch keinerlei Möglichkeiten haben, den QR-Code zu scannen, oder präferiert nachvollziehbare Links, kann er diesen ebenfalls im Popup aufrufen.

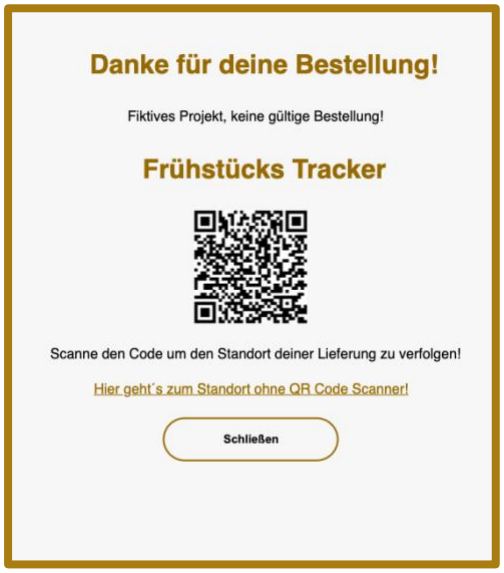

Abbildung 10: QR-Code Generierung

5.6. React Date Picker

Um bei dem Bestellvorgang den Tag der Anlieferung benutzerfreundlich auswählen zu können, benutzen wir die React Komponente namens React Date Picker. Dieser hat verschiedene Konfigurationsmöglichkeiten, mithilfe deren wir dieses Modul perfekt auf das Corporate Design sowie auf die gewünschten Funktionen unserer Website anpassen konnten. Exemplarische Anpassungen sind die Fristen beim Bestellvorgang, so können die Kunden frühestens in zwei Tagen bestellen und maximal sechs Monate im Voraus.

```
const DatePicker = lazy(() => import("react-datepicker"));
const LazyDatePicker = (
  props: JSX.IntrinsicAttributes &
    ReactDatePickerProps<string, boolean | undefined> &
    React.RefAttributes<ReactDatePicker<string, boolean | undefined>>
) => (
  <Suspense fallback={<div>Lädt...</div>}>
    <DatePicker {...props} />
  </Suspense>
);

export const StyledDatePicker = styled(LazyDatePicker)`
  font-family: "Oswald", sans-serif;
  font-size: 14px;
  color: ${colors.companycolor};
  background-color: ${colors.white};
  width: 200px;
  height: 25px;
  padding: 15px 0;
  text-align: center;
  border-radius: 5px;
  z-index: 1;
`;
```

Abbildung 11: Anpassungen Style Date Picker Teil 1

```
function CalendarComponent(): JSX.Element {
  return (
    <div>
      {load && (
        <StyledDatePicker
          selected={selectedDate}
          onChange={(date: Date) => setSelectedDate(date)}
          minDate={minDate}
          maxDate={maxDate}
          dateFormat={"dd.MM.yyyy"}
          locale={de}
          popperPlacement={"top"}
          onFocus={() => {}}
          calendarContainer={Calendar}
          popperContainer={Popper}
          customInput={<DatepickerInput />}
        />
      )}
    </div>
  );
}
```

Abbildung 12: Anpassungen Style Date Picker Teil 2

5.7. Icons

Wir haben Icons aus den Bibliotheken Material-UI, Phosphor-React und React-Icons in unserer Homepage eingebaut. Diese haben den Vorteil, dass sie perfekt mit dem Framework harmonisieren und somit eine anschauliche, einfache und problemlose Implementierung sichergestellt ist. Die rechte Grafik ist eine exemplarische Implementierung der verschiedenen Zahlungsmöglichkeiten.

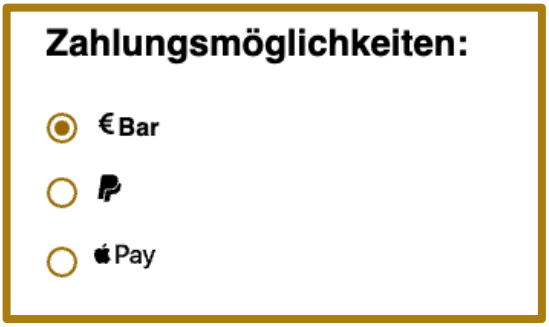

Abbildung 13: Einbau von React Icons

5.8. React-Toastify

Mittels dieser Komponente implementierten wir die Push Up Nachrichten, welche vor allem auf der Produktseite zu finden sind. Diese geben dem Benutzer Feedback auf seine Interaktionen. Eine Auswahl der verschiedenen Nachrichten kann im kommenden Screenshot eingesehen werden.

Der nachfolgende Code zeigt beispielshaft die Gestaltung der CustomToast.success Nachricht:

Abbildung 14: Toastify Nachrichten

```
success: (message: string) =>
  toast.success(message, {
    autoClose: 2000,
    closeButton: false,
    hideProgressBar: false,
    closeOnClick: true,
    pauseOnHover: true,
    draggable: true,
    draggablePercent: 60,
    position: "top-left",
    transition: Slide,
    progressClassName: "my-toast-progress-bar",
    progressStyle: { backgroundColor: colors.companycolor },
    style: { backgroundColor: colors.white, color: colors.black },
  }),
```

Abbildung 15: Codeausschnitt Toastify

6. Corporate Design

Im ersten Semester befassten wir uns mit der Auswahl einer geeigneten Farbpalette, die einerseits eine gute Lesbarkeit durch starke Kontraste, andererseits unsere Geschäftsidee passend verkörpert. Diese Idee der grafischen Gestaltung unserer Benutzeroberfläche perfektionierten wir im dritten Semester.

Unsere finale Entscheidung galt einem dunklen Orange mit dem Hexadezimalen Farbcode #aa7d03. Diese Farbe assoziiert der Kunde mit einem goldigen Sonnenaufgang und frischem, noch lauwarmen Gebäck. Dieses Ideal eines Starts in den Tag möchten wir aufgreifen und somit dem Kunden aufzeigen, dass unsere Produkte ihm den Morgen versüßen.

Die Schriftfarbe variiert zwischen Schwarz, Weiß und dem dunklen Orange, je nach Hintergrund.

Auffallend ist das Hintergrundbild auf der Startseite. Dieses wurde dunkel eingefärbt, wo hingegen die Hintergrundbilder der anderen Unterseiten eine weiße Einfärbung erhalten haben. Wie der natürliche Helligkeitsverlauf am Morgen erhellt sich auch unsere Internetseite mit dem Fortschreiten durch die verschiedenen Unterseiten. Während es anfangs noch dämmert, strahlt dem Besucher auf der darauffolgenden Seite schon die Helligkeit des Vormittages entgegen. Zudem charakterisiert sich die Startseite durch ihren dunklen Auftritt sehr edel und lädt zum Erkunden ein, wobei die nachfolgenden Unterseiten den Fokus auf den Inhalt und dessen Lesbarkeit legen.

Abbildung 16: Landingpage

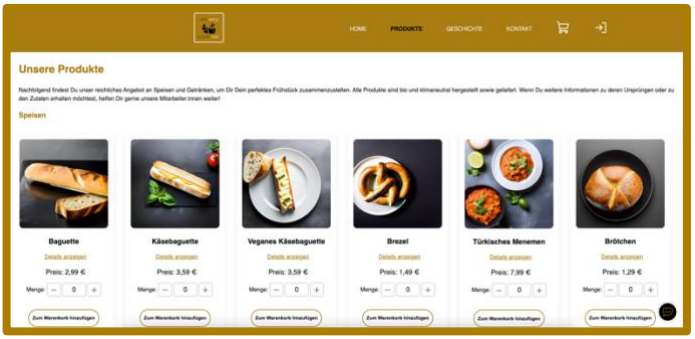

Abbildung 17: exemplarische Subpage

7. Identifikationsmerkmale

7.1. Chatbot

Direkt auf der Startseite können unsere Besucher mittels eines Chatbots unser Unternehmen sowie unsere Produkte kennenlernen. Auch auf allen anderen Unterseiten ist das Icon zuverlässig an der gleichen Stelle zu finden und ist somit eine zentrale Anlaufstelle für Fragen, Anregungen oder für sonstige Themen unserer Kunden. Durch die sehr einfache Bedienung sowie die maßgeschneiderten Informationen erhöhen wir unseren Kundenservice sowie deren Zufriedenheit und können auch bei besonderen Anliegen persönlich mit den Kunden kommunizieren. Dieser Chatbot, eine externe JavaScript API, stammt von chatra.com und wurde von uns nahtlos in unser Erscheinungsbild eingebunden.

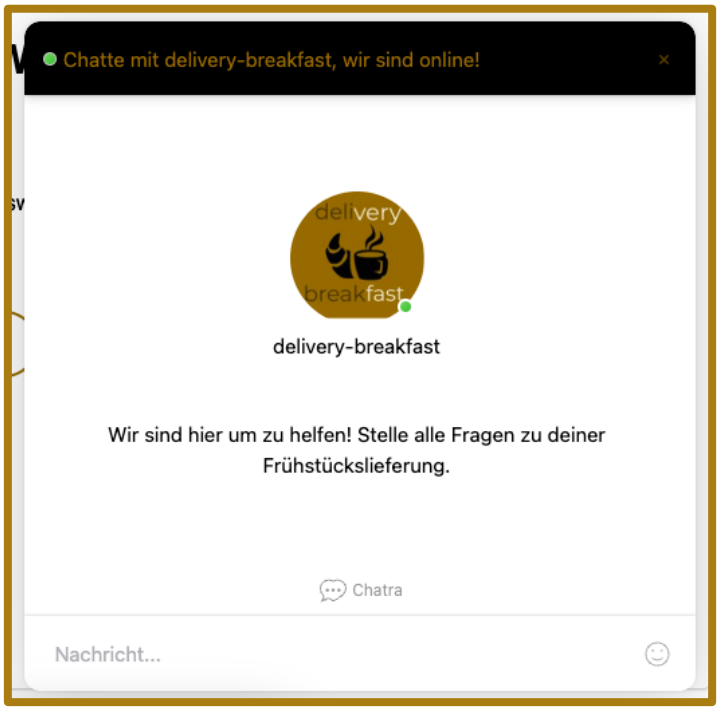

Abbildung 19: Chatbot Popup

Abbildung 18: Chatbot-Icon

7.2. Lazy-Loading

Eines unserer Hauptanforderungen an die finale Homepage ist eine sehr gute Geschwindigkeit. Wie bereits erwähnt soll auch unsere ländliche Zielgruppe eine flüssige Interaktion mit unserer Homepage spüren können.

Nach dem Einbau des Frameworks, der Gestaltung der zahlreichen Unterseiten und des Einbaus verschiedener APIs erfüllte jedoch die Schnelligkeit nicht unser ehrgeiziges Ziel. Daraufhin suchten wir nach verschiedensten Möglichkeiten die Performance zu steigern und stießen auf Lazy-Loading. Durch diese Methode kann man einzelne Teile einer Seite erst nach einer selbst festgesetzten Verzögerung laden lassen.

Dadurch, dass unser Chatbot ein großer Teil der Leistungseinbußen verursachte, wird dieser erst nach dem erstmaligen Scrollen auf der Landingpage geladen und beeinträchtigt somit nicht mehr die Geschwindigkeit bei dem erstmaligen Aufruf unserer Homepage. Durch diese Art und Weise der Programmierung erreichten wir eine Geschwindigkeitssteigerung von circa 20 Punkten bei Google Page Speed Insights.

Auch die React Komponente Date Picker implementierten wir nach diesem Konzept, da dieses Modul nur bei einem Bestellvorgang relevant ist. Besucher der Homepage, welche ohne eine Bestellung die Homepage wieder verlassen, haben dieses Modul nicht geladen, was wiederum eine positive Auswirkung auf die Geschwindigkeit hat.

7.3. Responsives Design

Heutzutage werden sehr viele Bestellungen von mobilen Endgeräten wie Smartphones oder Tablets getätigt. Um auch für diese kleinen Bildschirmgrößen sowie für die Touch-Interaktionen ein perfektes Benutzererlebnis zu bieten, haben wir ein eigenes Menü für diese Geräteklasse bis 1024px Bildschirmbreite entwickelt. Einerseits sind die verschiedenen Icons durch einen größeren Abstand separiert, andererseits befindet sich dieses Menü nun nicht mehr am oberen, sondern gut erreichbar am unteren Bildschirmrand, eine sogenannte Bottom Navigation Bar. Wer dennoch die Desktopvariante des Menüs favorisiert, kann jederzeit durch das Burgermenü, platziert in der oberen, rechten Ecke, die traditionelle Art und Weise des Navigierens verwenden. Dieses schließt sich automatisch bei Beginn des Scrollens oder wenn auf eine andere Unterseite navigiert wird.

Auch bei der Entdeckung unserer großen Produktpalette achteten wir auf eine gute Navigation per Touch-Eingabe, da die Produktkarten simpel durch horizontale Wischgesten durchstöbert werden können. In der Desktopansicht erscheinen Pfeile am linken und rechten Bildschirmrand, welche ebenfalls eine einfache und flüssige Interaktion mit der Homepage ermöglichen. Diese verschwinden, wenn nach drei Sekunden keine Mausinteraktion stattfindet und verhindern somit das Verdecken von wichtigem Content.

Abbildung 21: Bottom Nav Bar und Burgermenü

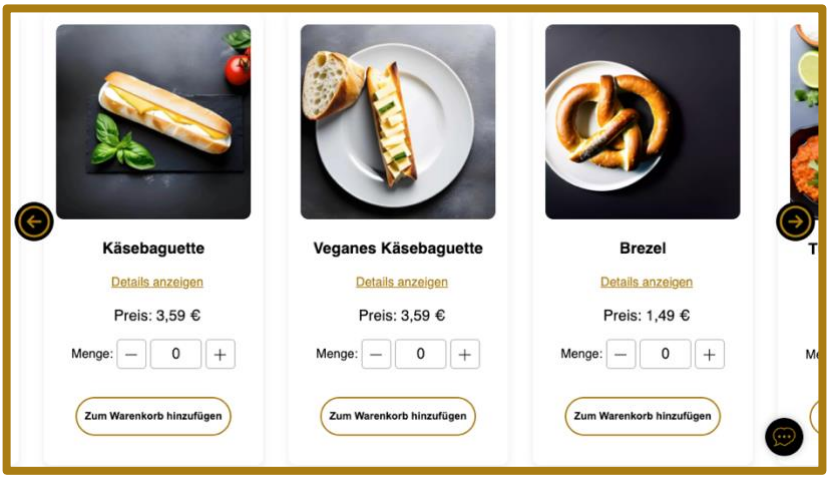

Abbildung 20: Navigation durch Produktkarten

8. Herausforderungen

8.1. Verwendung eines Frameworks

Eine der größten Herausforderungen war die Verwendung eines Frameworks, in unserem Falle React. Aufgrund der Tatsache, dass ausschließlich ein Teammitglied bereits erste Erfahrungen hatte, war dieses Thema für die anderen noch sehr abstrakt, weswegen wir zu Beginn des Projektes viel Geduld bei dem Übertragen der statischen Website in das auserwählte Framework benötigten.

Auch die dafür erforderliche Installation von node.js auf dem Webserver gestaltete sich deutlich schwerer als gedacht. Durch lange Recherche und Teamwork meisterten wir schließlich die Ersteinrichtung und hatten diesbezüglich seitdem keine weiteren Herausforderungen.

Im Laufe des Semesters stiegen wir dann immer tiefer in das Framework ein und lernten die vielen Vorteile, welche ein solches Konzept mit sich bringt, zu schätzen.

8.2. Sicherheit

Sicherheit ist ebenfalls ein hochrelevantes Kriterium unserer Homepage, da beispielsweise bei dem Bestellungsprozess sensible Daten wie Adresse oder Bankverbindungen verarbeitet werden. Aus diesem Grund haben wir unsere Website durch ein SSL-Zertifikat der Non-Profit Organisation Let's Encrypt verschlüsselt.

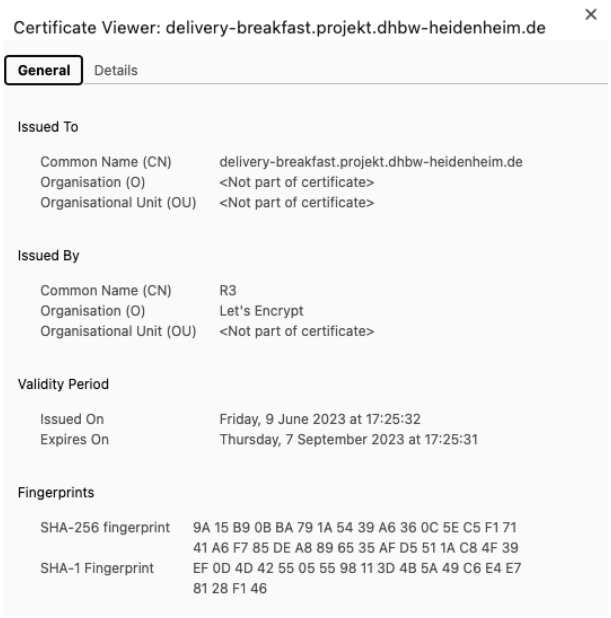

Abbildung 22: SSL-Zertifikatsinformationen

Die eingebauten Links wurden durch den Zusatz von „rel="noopener noreferrer"" erweitert, um das Risiko von Clickjacking-Angriffen zu minimieren.

```
<a
  href="https://www.instagram.com/delivery_breakfast_"
  aria-label="Link zu Instagram"
  target="_blank"
  rel="noopener noreferrer"
  style={{ display: "inline-block", width: "48px", height: "48px" }}
>
  <InstagramLogo size={24} />
</a>
```

Abbildung 23: Implementierung Links

Auch die Content Security Policy Richtlinie erweiterten wir in diesem Semester. Es sind nun nur noch Inhalte vom eigenen Server erlaubt, ausgenommen sind die beiden externen Quellen, welche die APIs von Google und Chatra verwenden.

```
<meta
  http-equiv="Content-Security-Policy"
  content="
default-src 'self';
style-src 'self' 'unsafe-inline';
frame-src 'self' https://www.google.com https://call.chatra.io https://chat.chatra.io;
script-src 'self' https://call.chatra.io/chatra.js;
child-src 'self' https://chat.chatra.io"
/>
```

Abbildung 24: Content Security Policy

9. Abbildungsverzeichnis