SWI Dokumentation

Dokumentation der SWI Einzelarbeit zur Erstellung eines Werkzeugverleihs

Auftrag 1: Grundgerüst

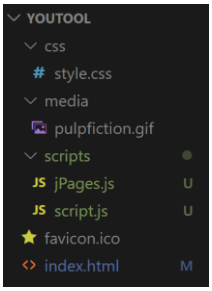

Abbildung 1: Datenstruktur

Während dem ersten Auftrag habe ich, wie in Abbildung 1 ersichtlich, die verschiedenen Dateien mit Visual Studio Code erstellt und auf die Struktur geachtet. Dabei initial die drei Dateien index.html, scripts.js, und style.css. Die Dateien wurden später für eine saubere Darstellung des Verzeichnisses in eigene Ordner verschoben.

Responsive Design

Um eine responsive Seite zu erstellen habe ich das Bootstrap Framework benutzt. Das ermöglicht eine einfache Umsetzung des Responsive Designs. Zudem habe ich (vgl. Abbildung 2) beim Einspeisen der Cards diese mit den Bootstrap col-Klassen versehen,.

```
var card='<div class="card col-9 col-xl-3 col-lg-3 col-md-4 col-sm-6 col-xs-6
```

Abbildung 2: Responsive Cards.

Desktop Variante

Zuerst habe ich die Seite in zwei verschiedene Container aufgeteilt. Ein Container (Abbildung 3 in rot), welcher Warenkorb und Reservationsform enthält und der andere (Abbildung 3 in blau), um die Toolliste zu gruppieren. Die Toolliste wird angezeigt mithilfe der Bootstrap-Klasse justify-content-

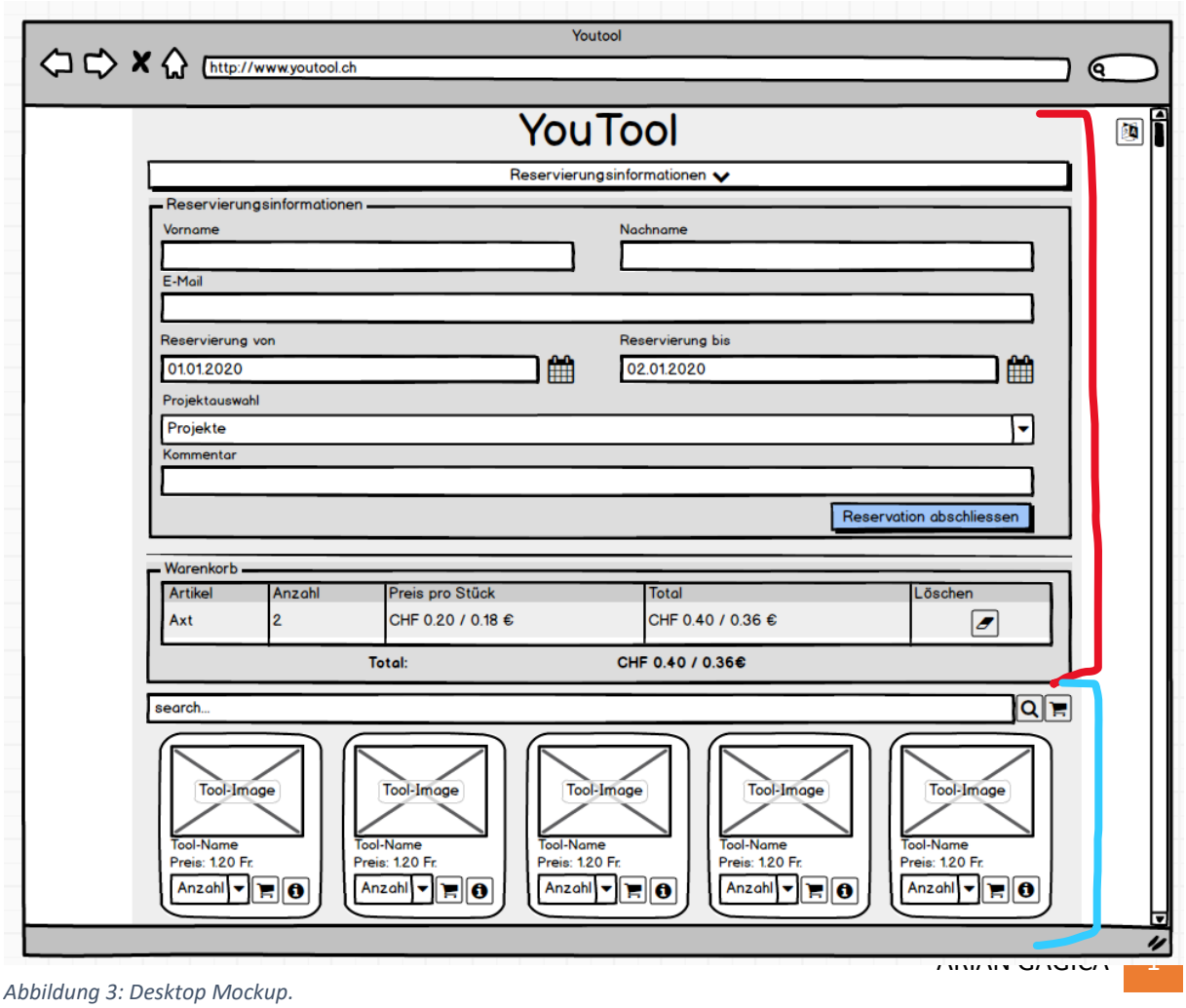

Abbildung 3: Desktop Mockup.

Smartphone Variante

Bei der Smartphone Ansicht (vgl. Abbildung 4) wird auf ein Tool pro Zeile reduziert, um so noch Lesbarkeit beizubehalten und den Bildschirm nicht zu überladen.

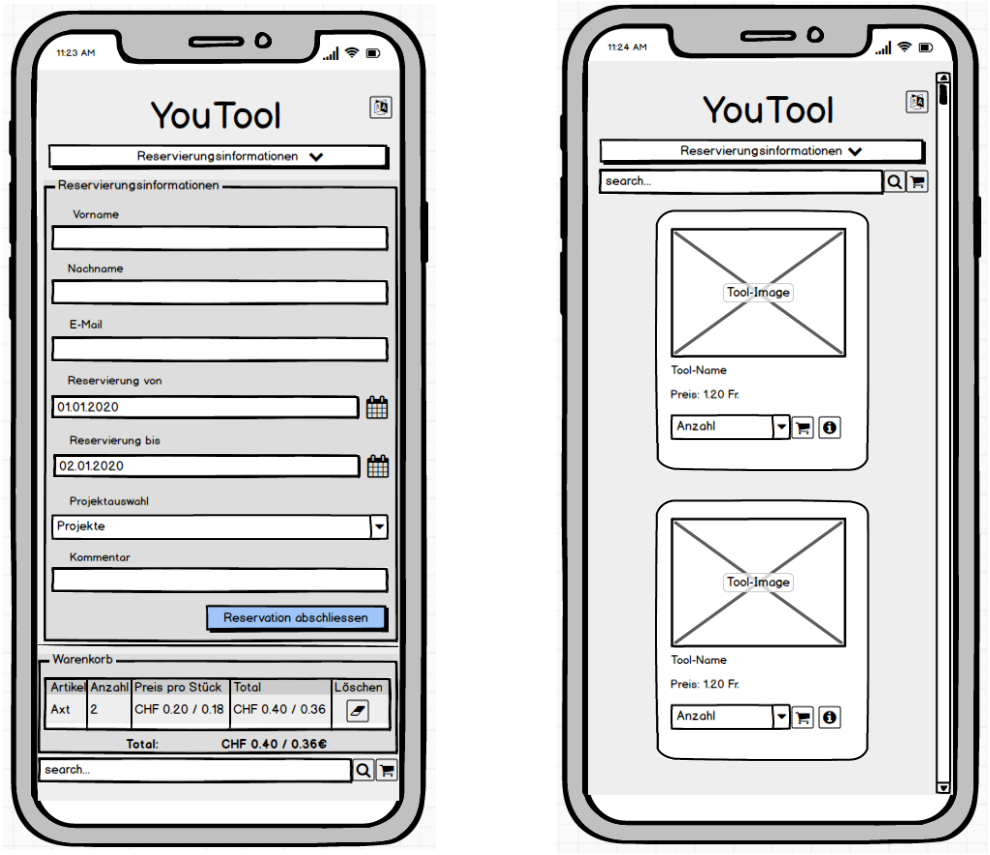

Abbildung 4: Smartphone Mockup.

Auftrag 2: Dynamische Elemente

Daten dynamisch abrufen

Die dynamische Abfrage der Tools habe ich mit der Funktion jQuery getJSON gelöst. Diese Funktion habe rufe ich mit getTools() auf. Diese Funktion habe ich absichtlich erstellt, da ich bei einem Suchbefehl auch diese Funktion aufrufe und die URL anpasse mit dem zusätzlichen Suchterm. Die Projekte werden auch mithilfe des jQuery-Befehls getJSON durchgeführt.

Kostenrechner

Alle Items, die im Warenkorb sind, befinden sich im localStorage. Ich habe im localStorage ein Key-Value Paar erstellt, welches den key «warenkorb» hat. In diesem Key speichere ich immer alle Artikel im JSON Format ab und kann das JSON Objekt immer über localstorage.getItem() abrufen. So habe ich mit dieser Methode einen einfachen Weg die Summe aller Preise der Werkzeuge und kann sie mit den Anzahl Tagen multiplizieren (vgl. Abbildung 5).

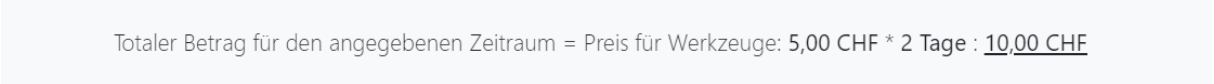

Abbildung 5: Kostenanzeige nach vollständiger Reservation.

Auftrag 2: Wahlaufgaben

Grafische Wahl von Zeitraum

Für die Grafische Auswahl habe ich mithilfe einer Google Suche die Library «AirDate-Picker» gefunden und diese für mich eingebunden. Ich habe noch ein paar Features eingesetzt (vgl. Abbildung 6), damit man keine Daten in der Vergangenheit setzen kann, und sobald das erste Datum gesetzt ist, wird das zweite Eingabefeld enablet und kann nicht vor dem ersten Datum sein.

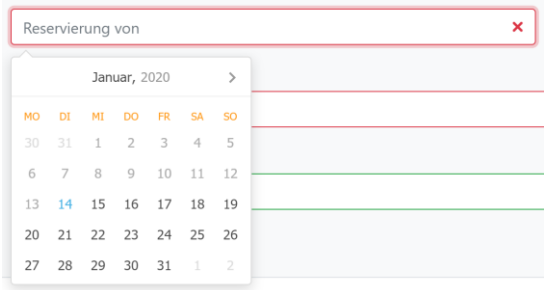

Abbildung 6: Datepicker.

Sprachwahl

Die Sprachwahl (vgl. Abbildung 7) konnte ich auch im Internet fündig, werden. Diese habe ich oben rechts auf der Seite platziert und habe die Artikel Beschriftungen mit einem «lang» Tag versehen. So kann ich 2 verschiedene Funktionen, die diese lang tags auf hidden oder visible setzen.

Abbildung 7: Languagepicker.

Textfilter für Werkzeuge

Die Suchbar hatte ich anfangs komplett im Internet gefunden, doch dann habe ich gesehen, dass man mit separaten GET Befehlen die Werkzeuge aufrufen sollte. Daher habe ich meine Seite auch so umgestellt, dass die Funktion getTools() vor dem getJSON Befehl noch prüft ob es eine Suchanfrage ist. Falls es eine Suchanfrage ist, wird zuerst die Sprache geprüft, danach wird die Service-URL des getJSON Befehl so modifiziert, dass es die richtige Sprache der URL übernimmt und es den Suchterm noch am Schluss anhängt.

Auftrag 3: Daten versenden

Formularvalidierung

Beim Formular habe ich mich mit der Vorlage von Bootstrap bedient und habe sie für mich angepasst. Diese ist auch responsive und mit dem Attribut required ist die Formularvalidierung implementiert. Die nicht ausgefüllten Eingabefelder zeigen ein rotes Kreuz und die korrekten ein grünes Kreuz. Beim Versenden von inkorrekten Daten wird eine Meldung angezeigt, welches sagt was nicht korrekt ausgefüllt wurde.

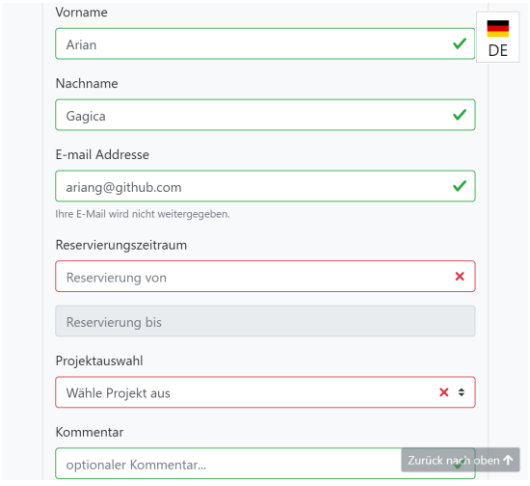

Abbildung 8: Reservationsform.

Absenden des Reservierungsantrags

Das Absenden wird mit einem Post Befehl durchgeführt, welcher beim Klicken des Buttons «Reservierung abschliessen» aufgerufen wird. Die Daten, welche mit einem «name» Tag in der Form versehen wird, werden serialisiert. Zudem wird der Cart eingeholt und auch die ID's und Anzahl der Items werden an den String angefügt. Nach erfolgreichem Absenden der Reservierung wird die Map geladen und noch zusätzlich ein Toast (Abbildung 9) angezeigt, dass die erfolgreiche Reservation

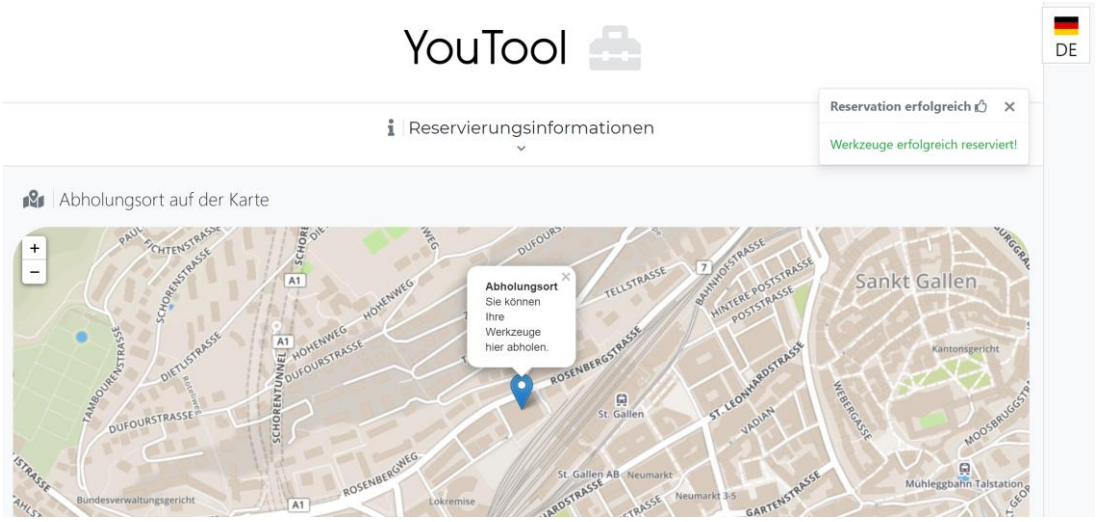

Abbildung 9: Reservationsabschluss.

signalisiert.

Daten lokal abspeichern

Da der Warenkorb mit der gleichen Methodik erstellt wurde, habe ich hier dasselbe Schema angewendet. Bei erfolgreichem Absenden einer Reservation wird der Vorname, Name und die E-Mail über localStorage.setItem('User', 'JSON Objekt') gesetzt. Wenn der Key User mit Daten vorhanden ist, werden die Daten beim Laden der Seite automatisch in die Registrationsform gesetzt. In Abbildung 10 ist ein Beispiel des Localstorages zu sehen, nachdem alle Inhalte ausgefüllt worden sind.

warenkorb	{"28839283":{"id":"28839283","label-de":"Hammer gross","label-en":"Hammer Big","img":"images/Hammer-Big-icon.png","stock...
User	{"firstname":"Arian","lastname":"Gagica","email":"ariang@github.com"}

Abbildung 10: Localstorage mit Inhalten.

Auftrag 3a: Wahlaufgaben

Spezielle Schriftart

Die Fonts lade ich bei der Seite im Head Tag ein, dies ist in Abbildung 11 zu sehen, und setze über das CSS die Schriftart für die nötigen HTML Elemente. Diese drei Fonts werden in der Itemdarstellung, im Footer und in den Titeln benutzt.

```
<!-- Google Font -->
<link href="https://fonts.googleapis.com/css?family=Montserrat&display=swap" rel="stylesheet">
<link href="https://fonts.googleapis.com/css?family=Ubuntu&display=swap" rel="stylesheet">
<link href="https://fonts.googleapis.com/css?family=Sulphur+Point&display=swap" rel="stylesheet">
```

Abbildung 11: Einbindung der Fonts.

Kartendarstellung

Die Karte habe ich mit der leaflet Dokumentation problemlos eingebunden. Diese wollte ich aber erst anzeigen nachdem die Reservation abgeschlossen wird. Ein kleines Problem ist entstanden, da ich die Reihenfolge des Ladens nicht berücksichtigt habe. Ich habe die Map geladen bevor der Container

geladen ist, dann hat die map erst richtig funktioniert nachdem ich die Fenstergrösse verändert habe. Die map hat mit diesem Problem die Tiles grau dargestellt und nur Teile der Map geladen. Nach ein paar Google Suchanfragen habe ich das Problem beseitigen können und die Map wird wunderbar mit den erhaltenen Koordinaten angezeigt. Ich musste mit einer Funktion, welche invalidateSize() die Fenstergrösse neu initialisieren und ich habe versucht das Laden der Map nach dem Container durchzuführen.

CSS-Animation

Ich habe einige Animationen auf der Seite angewendet. Zuerst habe ich auch hier die animate library in den header eingebunden (vgl. Abbildung 11) und danach mit Klassen wie bspw. «animate fadeInUp» die HTML Elemente animiert. Zudem habe ich die Library Fontawesome oft benutzt um die Headers schöner darzustellen. In Abbildung 12 sind zwei Einsätze der Fontawesome-Library zu sehen und die effektive Code Zeile ist in Abbildung 13 zu sehen.

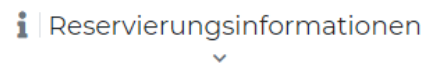

Abbildung 13: Titel mit 2 Fontawesome Icons.

```
<i class="text-muted fas fa-info border-right">  </i> Reservierungsinformationen</br>
<i class="text-muted fas fa-chevron-down pt-1" style="font-size: 8pt;"></i>
```

Abbildung 12: Einsatz von Fontawesome Icons im Code.

Schlusswort

Ich möchte eine kurze Reflexion zu diesem Projekt anhängen. Es hat mir sehr spass bereitet eine Hands-On Experience mit diesem Modul zu erleben. Ich habe viel gelernt in diesem Projekt und habe gesehen, dass mir das Programmieren Freude bereitet. Ich möchte mich bedanken für die Möglichkeit und auch für die spannende Aufgabenstellung, die wir hier lösen durften.

Vielen Dank für die spannende Durchführung des Moduls.

Arian Gagica