Universität Bielefeld
Fakultät für Linguistik und Literaturwissenschaft

Projektausarbeitung
23-TXT-BaCL4

Die digitale Cocktailbar

ein Projekt zur Informationsstrukturierung

vorgelegt von

Fabian Wohlgemuth

Begutachtet von: Prof. Dr. Jens Michaelis

Bielefeld, März 2020

Inhaltsverzeichnis

1 Einleitung

1.1 Grundlage

Grundlage des vorliegenden Projektes sind das Modul `23-TXT-BaCL4` im Nebenfach Texttechnologie und Computerlinguistik und das dazugehörige Skript von Prof. Dr. Marcus Kracht [Kracht, 2018].

Im Rahmen der zwei Seminare *Informationsstrukturierung* & *Informationsstrukturierung 2* wurde gelernt, wie eine Datenbank mithilfe von `XML` angelegt und unter Zunahme eines `XML Schemas` strukturiert werden kann. Über eine `XSL`, eine sogenannte Stylesheet-Datei, kann aus der Datenbank eine `HTML` Datei erzeugt werden. Dieser Prozess ist bekannt unter dem Namen `XSLT`; der sogenannten *Transformation*. Diese kann über die `HTML` Datei noch eine Vielzahl anderer Dateiformate erzeugen, denen wir uns jedoch in dieser Ausarbeitung nicht widmen.

1.2 Projekteinführung

Bei dem vorliegenden Projekt handelt es sich um die *digitale Cocktailbar*; ein Rezepteverzeichnis für Cocktails. Eine Sammlung an Cocktailrezepten wird in einer Datenbank gespeichert, aus der mithilfe der oben genannten Transformations-Anwendung eine Website erzeugt wird, die im Web-Browser angeschaut werden kann.

Nach dieser kurzen Einleitung folgt in Kapitel 2 eine detaillierte Beschreibung der im Projekt vorhandenen Dateien und deren Inhalten und Aufgaben.

2 Dateien

Die folgenden Dateien befinden sich in dem vorliegenden Projekt. In diesem Kapitel werden die einzelnen Bestandteile und ihre Funktionen innerhalb des Projektes detailliert besprochen.

Anmerkung: Die Dateien folgen nicht durchgehend den üblichen Einrückungs-Standards. Dies ist absichtlich so und hat den Grund, dass bei dem Anzeigen der Codeblöcke innerhalb der Dokumentations-Datei ansonsten Leerzeichen am Zeilenanfang (`leading whitespace`) mit angezeigt würden.

```
/
├── datenbank.xml
├── schema.xsd
├── stylesheet.xsl
├── css/
│   └── main.css
├── js/
│   └── script.js
├── img/
│   ├── glass-beach.svg
│   ├── glass-cocktail.svg
│   ├── glass-daiquiri.svg
│   ├── glass-long.svg
│   ├── glass-manhattan.svg
│   ├── glass-martini.svg
│   └── glass-whiskey.svg
├── index.html
├── dokumentation.pdf
└── docs/
    └── ...
```

2.1 Die Datenbank - `datenbank.xml`

Die Datenbank ist das Herzstück des Projektes. Sie enthält alle Daten, die später weiterverarbeitet werden sollen. Zunächst werden im Kopfbereich der Datenbank-Datei die `XML` Version und das Text Encoding angegeben. Anschließend folgt die Angabe, unter welchem Dateipfad sich die Stylesheet `XSL` Datei für die spätere Transformation befindet.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <?xml-stylesheet type="text/xsl" href="stylesheet.xsl"?>
```

Codeblock 2.1: Datenbank - Header

Nach dem Header beginnt der Aufbau der eigentlichen Datenbank. Zunächst öffnen wir die Datenbank mit dem `<cocktailbar>` Tag (siehe Codeblock 2.2). Im Anschluss findet sich in der Datenbank eine Liste der Cocktails, eingebettet in das `<cocktails>` Tag.

Jeder einzelne Cocktail ist entsprechend in ein `<cocktail>` Tag eingefasst. Hier sehen wir die erste komplexere Einheit der Datenbank. Ein `cocktail` besteht aus drei Attributen, die sich im Start-Tag angeben lassen und einer Anzahl von Zutaten, die sich im `zutaten` Tag befinden.

Die Attribute sind die `id`, die jedem Cocktail einen eindeutigen Identifizierer zuordnet, der `name`, der die Normalschreibweise des Cocktail-Namens angibt und das `glass`, welches aus einer Liste möglicher Glas-Formen auswählt, in der der Cocktail im Normalfall zubereitet wird.

Jede Zutat hat mindestens zwei Attribute. Wie auch der Cocktail zuerst eine `id`. Dann eine Mengenangabe im `menge` Attribut.

```
4 <cocktailbar>
5
6 <cocktails>
7   <cocktail id="negroni" name="Negroni" glass="whiskey">
8   <cocktailZutaten>
9     <cocktailZutat id="london-dry-gin" menge="30" />
10    <cocktailZutat id="vermouth-sweet" menge="30" />
11    <cocktailZutat id="campari" menge="30" />
12  </cocktailZutaten>
13  </cocktail>
```

Codeblock 2.2: Datenbank - Beginn

Standardmäßig wird die `menge` der Zutat in `ml` angegeben. Wenn eine andere Mengeneinheit gewünscht ist, kann diese durch ein entsprechendes drittes Attribut der Zutat hinzugefügt werden. Ein Beispiel ist in Zeile 64 im folgenden Codeblock 2.3 zu sehen.

```
<cocktail id="tequila-sunrise" name="Tequila Sunrise" glass="long
    ">
<cocktailZutaten>
  <cocktailZutat id="tequila-silver" menge="45" />
  <cocktailZutat id="orange-juice" menge="120" />
  <cocktailZutat id="grenadine" menge="1" einheit="Schuss" />
</cocktailZutaten>
</cocktail>
```

Codeblock 2.3: Abweichende Mengeneinheit

Im Anschluss an die Cocktail-Liste befindet sich die Zutaten-Liste. Jede Zutat besitzt hier zwei Attribute. Wie die Cocktails auch jeweils eine `id` und das Attribut `name`, welches die Normalschreibweise der Zutat enthält. Zuletzt wird die Datenbank mit `</cocktailbar>` geschlossen.

```
<zutaten>
  <zutat id="london-dry-gin" name="London Dry Gin" />
  <zutat id="gin" name="Gin" />
  <zutat id="vermouth-sweet" name="Sweet Vermouth" />
  <zutat id="vermouth-dry" name="Dry Vermouth" />
  <zutat id="campari" name="Campari" />
  <zutat id="syrup-gomme" name="Gomme Syrup" />
  <zutat id="syrup-orgeat" name="Orgeat Syrup" />
  <zutat id="peach-schnapps" name="Peach Schnapps" />
  <zutat id="curacao-orange" name="Orange Curacao" />
  <zutat id="grenadine" name="Grenadine" />
  <zutat id="triple-sec" name="Triple Sec" />
  <zutat id="rum-dark" name="Dark Rum" />
  <zutat id="vodka" name="Vodka" />
  <zutat id="cola" name="Cola" />
  <zutat id="tonic-water" name="Tonic Water" />
  <zutat id="angostura-bitters" name="Angostura Bitter" />
  <zutat id="whiskey-rye" name="Rye Whiskey" />
  <zutat id="rum-white" name="White Rum" />
  <zutat id="lime-juice" name="Lime Juice" />
  <zutat id="orange-juice" name="Orange Juice" />
  <zutat id="cranberry-juice" name="Cranberry Juice" />
  <zutat id="lemon-juice" name="Lemon Juice" />
  <zutat id="syrup-simple" name="Simple Syrup" />
  <zutat id="rum-goslings-black-seal" name="Gosling's Black Seal
    Rum" />
  <zutat id="beer-ginger" name="Ginger Beer" />
  <zutat id="tequila-silver" name="Silver Tequila" />
  <zutat id="cointreau" name="Cointreau" />
</zutaten>
```

Codeblock 2.4: Zutatenliste

2.2 Das Schema - `schema.xsd`

Die Schema-Datei stellt sicher, dass die Datenbank in 2.1 dem angegebenen Schema folgt. Gegen die Schema-Datei lässt sich die Datenbank validieren. Es kann also geprüft werden, ob die Datenbank richtig formatiert ist. Auf diese Weise können Fehler in der Datenbank vor der Transformation erkannt werden.

Wie die Datenbank, beginnt auch das Schema mit einem Header (siehe 2.5).

```
1 <?xml version="1.0"?>
2 <xs:schema attributeFormDefault="unqualified" elementFormDefault=
    "qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

Codeblock 2.5: Schema - Header

Im Anschluss definieren wir im Schema die gesamte `cocktailbar` als Sequenz, die zum einen die `cocktails` und zum anderen die Liste der `zutaten` enthält.

```
4 <xs:element name="cocktailbar">
5   <xs:complexType>
6     <xs:sequence>
7       <xs:element ref="cocktails"/>
8       <xs:element ref="zutaten"/>
9     </xs:sequence>
10  </xs:complexType>
11 </xs:element>
```

Codeblock 2.6: Schema - Cocktailbar

Sowohl die `cocktails` als auch die `zutaten` bestehen aus einem einzigen Element. Dies ist der einzelne `cocktail` und die einzelne `zutat`. Beide Elemente können beliebig oft vorkommen `maxOccurs`, müssen jedoch mindestens einmal vorhanden sein `minOccurs`.

```
13 <xs:element name="cocktails">
14   <xs:complexType>
15     <xs:sequence>
16       <xs:element ref="cocktail" maxOccurs="unbounded" minOccurs=
     "1"/>
17     </xs:sequence>
18   </xs:complexType>
19 </xs:element>
20
21 <xs:element name="zutaten">
22   <xs:complexType>
23     <xs:sequence>
24       <xs:element ref="zutat" maxOccurs="unbounded" minOccurs="1"
     />
25     </xs:sequence>
```

```
  </xs:complexType>
</xs:element>
```

Codeblock 2.7: Schema - Cocktails & Zutaten

Anschließend definieren wir, was einen einzelnen `cocktail` ausmacht. Er besteht aus einer Reihe an Zutaten, die wir über `cocktailZutaten` definieren und drei Attribute. Die Attribute sind `id`, `name` und `glass`. Alle drei Attribute sind notwendig (`required`).

Die `id` wird zusätzlich über den `xs:unique`-Eintrag spezifiziert, sodass jede `id` nur einmalig vergeben werden kann. Dies ist besonders wichtig, da wir im späteren Verlauf auf einzelne Cocktails verweisen möchten. Wäre die Möglichkeit gegeben, dass mehrere Cocktails dieselbe `id` besäßen, könnten wir nicht eindeutig auf einen spezifischen Cocktail referieren.

```
<xs:element name="cocktail">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="cocktailZutaten"/>
    </xs:sequence>
    <xs:attribute type="xs:string" name="id" use="required"/>
    <xs:attribute type="xs:string" name="name" use="required"/>
    <xs:attribute type="xs:string" name="glass" use="required"/>
  </xs:complexType>
  <xs:unique name="cocktail-id">
    <xs:selector xpath="cocktail"/>
    <xs:field xpath="@id"/>
  </xs:unique>
</xs:element>
```

Codeblock 2.8: Schema - Cocktail

Selbige Definition nehmen wir nun für die einzelne `zutat` vor. Einziger Unterschied ist der, dass die `zutat` ein Attribut weniger hat. Es fehlt das `glass`-Attribut, welches beim `cocktail` dafür zuständig ist, festzulegen, welches Glas für den Cocktail genutzt werden soll.

```
<xs:element name="zutat">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute type="xs:string" name="id" use="required"/>
        <xs:attribute type="xs:string" name="name" use="required"
   />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
```

```
53   <xs:unique name="zutat-id">
54     <xs:selector xpath="zutat"/>
55     <xs:field xpath="@id"/>
56   </xs:unique>
57 </xs:element>
```

Codeblock 2.9: Schema - Zutat

Auf der zweitkleinsten Ebene widmen wir uns nun den Zutaten, die in einem Cocktail zufinden sind (`cocktailZutaten`). Die Cocktail-Zutatenliste besteht aus beliebig vielen jedoch mindestens einer `cocktailZutat`.

```
59 <xs:element name="cocktailZutaten">
60   <xs:complexType>
61     <xs:sequence>
62       <xs:element ref="cocktailZutat" maxOccurs="unbounded"
     minOccurs="1"/>
63     </xs:sequence>
64   </xs:complexType>
65 </xs:element>
```

Codeblock 2.10: Schema - Cocktailzutaten

Die kleineste Ebene ist die einzelne `cocktailZutat`. Diese besteht aus maximal drei Attributen. Diese sind wieder eine `id` und zusätzlich eine Mengenangabe und eine Angabe zur Mengeneinheit. `id` und `menge` sind Pflichtangaben. Die Angabe der `einheit` ist `optional`. Warum dies so ist, klärt sich in Kapitel 2.3.

Bis zu diesem Punkt waren allen Attribute vom Typ `xs:string`; also simple Zeichenketten. Nun haben wir eine Abweichung, da die `menge` vom Typ `xs:int` ist.

```
67 <xs:element name="cocktailZutat">
68   <xs:complexType>
69     <xs:simpleContent>
70       <xs:extension base="xs:string">
71         <xs:attribute type="xs:string" name="id" use="required"/>
72         <xs:attribute type="xs:int" name="menge" use="required"/>
73         <xs:attribute type="xs:string" name="einheit" use="
     optional"/>
74       </xs:extension>
75     </xs:simpleContent>
76   </xs:complexType>
77 </xs:element>
```

Codeblock 2.11: Schema - Cocktailzutat

Nachdem alle in der Datenbank genutzten Elemente im Schema definiert wurden, wird die Schema-Datei mit `</xs:schema>` geschlossen. Im nächsten Abschnitt widmen wir uns dem Stylesheet.

2.3 Das Stylesheet - `stylesheet.xsl`

Bei dem Stylesheet handelt es sich um die Datei, die aus der Datenbank mithilfe von XSL Transformation eine neue Datei erzeugen kann; in unserem Fall eine `HTML`-Datei. Sie beinhaltet `XSL`-Code und `HTML`-Blöcke.

Würden wir nur `HTML` nutzen, so müssten wir unsere Datenbank direkt in die Ausgabe-Datei schreiben (*hard-coden*). Durch Nutzung der XSL Transformation können wir die Semantik (Stylesheet) unserer Datei unabhängig vom Inhalt (Datenbank - Abschnitt 2.1) gestalten und so zu einem späteren Zeitpunkt die Datenbank um Einträge erweitern, ohne die Stylesheet-Datei bearbeiten zu müssen.

Erneut starten wir mit dem Kopfbereich. Hier werden die `XML`-Version und ein Link zum `XML Namespace` angegeben. Darauf folgt die Angabe, dass der `output` unserer Transformation in `HTML` geschehen soll. Zuletzt wird über `xsl:template match` festgelegt, welches Element der Datenbank als Start-Element für spätere Angaben genutzt werden soll. In diesem Fall nutzen wir das Wurzelelement `/`. Würden wir mit einer umfangreicheren Datenbank und einer komplexeren Ausgabe-Datei arbeiten wollen, könnte es hier sinnvoll sein, einen kleineren Bereich der Datenbank anzusprechen, um eine bessere Übersicht zu garantieren.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
     version="1.0">
3 <xsl:output method="html" />
4 <xsl:template match="/">
```

Codeblock 2.12: Stylesheet - Header

Im nächsten Codeblock beginnen wir, `HTML`-Code in unser Stylesheet einzubauen. Zunächst setzen wir mit `<!DOCTYPE html>` fest, dass wir im weiteren Verlauf HTML 5 nutzen möchten. Anschließend öffnen wir die `<html>`- und `<head>`-Tags.

Zwischen `<head>...</head>` stehen Meta-Informationen, die später nicht auf der Website selber angezeigt werden sollen. Dazu gehört die Angabe zum Encoding `charset=„UTF-8"`, sodass wir unter anderem keine Probelem mit Umlauten bekommen, und der Seiten-Titel <title>, der später im Web-Browser im Tab als Titel angezeigt wird.

Außerdem werden hier zusätziche Dateien verlinkt, die wir zur grafischen Anpassung unserer Website benötigen. Zum einen ist dies `main.css`, die `CSS`-Datei, die die Gestaltung festlegt und zum anderen ist dies die Datei `script.js`, in der Javascript-Skripte gespeichert werden können, um die Funktionalität etwaiger Animationen zu gewährleisten. (Anmerkung: Letztere Datei wird zur Zeit nicht genutzt - befindet sich für den Fall der Projekt-Erweiterung aber weiterhin im

Projektordner).

Die zwei `meta`-Einträge dienen der Responsivität der Website-Darstellung. Damit wird gewährleistet, dass unsere Ausgabedatei nicht auf ein fixes Format festgelegt ist sondern auf allen Bildschirmen grafisch ansprechend angezeigt werden kann. Wie dies genau passiert, wird im Abschnitt 2.4 ausgeführt.

```
6  <xsl:text disable-output-escaping='yes'>&lt;!DOCTYPE html&gt;</
       xsl:text>
7  <html>
8  <head>
9    <meta charset="UTF-8"/>
10   <meta name="viewport" content="width=device-width,
       initial-scale=1.0"/>
11   <meta http-equiv="X-UA-Compatible" content="ie=edge"/>
12   <link rel="stylesheet" type="text/css" href="css/main.css"/>
13   <script src="js/script.js"/>
14   <title>Digitale Cocktailbar</title>
15 </head>
```

Codeblock 2.13: Stylesheet - Head

Nachdem der `<head>`-Bereich geschlossen wurde, öffnen wir den `<body>`; also den Bereich, der am Ende im Web-Browser dargestellt wird. Der `<body>` unterteilt sich im vorliegenden Projekt in drei Teile. Den `<header>` und zwei `<div>`-Elemente, die jeweils eine „Seite“ repräsentieren. Dazu mehr in den Codeblöcken 2.15 ff.

Der `<header>` beinhaltet die Seitenüberschrift `<h1>`, den Untertitel `<small>` und die Menüleiste `<div class="menu">`. Das Menü verlinkt jeweils zu den Seiten `Dokumentation` und `Cocktails`.

```
18 <header>
19   <h1>Die digitale Cocktailbar</h1>
20   <small>Ein Projekt im Rahmen des Moduls
21     <a href="https://ekvv.uni-bielefeld.de/sinfo/publ/modul
       /26802491" title="Modulbeschreibung -
       Informationsstrukturierung">
22     <strong>23-TXT-BaCl4 - Informationsstrukturierung</strong>
23     </a> an der Universität Bielefeld.</small>
24   <div class="menu">
25     <a href="#site-doc">Dokumentation</a>
26     <a href="#site-cocktails">Cocktails</a>
27   </div>
28 </header>
```

Codeblock 2.14: Stylesheet - Body - Header

Die Seite `Dokumentation` beinhaltet lediglich einen kurzen Einführungstext und den Link zu dieser Dokumentation.

```
30 <div id="site-doc" class="site">
31   <h2>Dokumentation</h2>
32   <p>Dieses Projekt demonstriert die Informationsstrukturierung
      mithilfe  der XSLT Transformation einer XML Datenbank.</p>
33   <p>Die Dokumentation befindet sich unter <code><a href="./
      dokumentation.pdf">.  dokumentation.pdf</a></code>.</p>
34 </div>
```

Codeblock 2.15: Stylesheet - Body - Seite - Dokumentation

Die Seite `Cocktails` enthält nun die gesamte Logik, die aus der Datenbank unsere Ausgabedatei erzeugen kann. Entsprechend werde ich diesen Abschnitt in kleinere Teile aufteilen.

Wir starten mit einem `<div>`-Tag, welches eine `id` enthält, um die Seite über die Menüleiste ansteuern zu können. Nach der Überschrift `<h2>` folgt der Kern der Cocktail-Liste. Bei diesem handelt es sich um eine ungeordnete Liste `<ul>`. Die `id` und `class` werden später beim Styling der Website genutzt. (Siehe Cascading Stylesheet - `css/main.css`)

```
36 <div id="site-cocktails" class="site">
37   <h2>Cocktails</h2>
38   <ul id="cocktail-list" class="cocktail-list">
```

⋮

```
78   </ul>
79 </div>
```

Codeblock 2.16: Stylesheet - Body - Seite - Cocktails - Liste

Nun wird für jeden Cocktail aus der Datenbank (`for-each` ... `cocktail`) ein Listen-Eintrag `li` angelegt.

```
39 <xsl:for-each select="//cocktail">
40   <li class="single-cocktail">
```

⋮

```
76   </li>
77 </xsl:for-each>
```

Codeblock 2.17: Stylesheet - Body - Seite - Cocktails - for-each

Jeder dieser Einträge in der Liste besteht nun aus mehreren Teilen. Zuerst wird der `name` des Cocktails aus der Datenbank entnommen und als Überschrift verwendet. Dies ist zu sehen in Zeile 42. Von jedem Cocktail (siehe vorheriger Codeblock) wird das Attribut `name` gewählt (`select ... @name`). Wir betten den Namen in `<div>`-Tags ein und vergeben die Klasse `cocktail-name`, um später die Gestaltung anpassen zu können.

```
41 <div class="cocktail-info">
42 <div class="cocktail-name"><xsl:value-of select="@name" /></div>
```

Codeblock 2.18: Stylesheet - Body - Seite - Cocktails - Name

Bei dem nächsten Abschnitt handelt es sich wieder um eine `for`-Schleife. Dieses mal iterieren wir über die Zutaten in der Zutatenliste eines jeden Cocktails. Wir befinden uns also in einer Schleife innerhalb einer Schleife. Für jede Zutat innerhalb des aktuellen Cocktails, wird nun in der ungeordneten Liste `ingredient-list` ein Listen-Eintrag `li` erstellt.

```
43 <ul class="ingredient-list">
44   <xsl:for-each select="cocktailZutaten/cocktailZutat">
45     <li class="single-ingredient">
```

⋮

```
76     </li>
77   </xsl:for-each>
78 </ul>
```

Codeblock 2.19: Stylesheet - Body - Seite - Cocktails - Zutaten

Im folgenden Codeblock nutzen wir eine Variable `xsl:variable`, um die `id` der aktuellen Zutat zwischenzuspeichern. Diese `id` wird dann innerhalb einer `for`-Schleife, die über alle Zutaten aus der Datenbank iteriert, dafür genutzt, um abzufragen, ob die gespeicherte `id` mit der `id` der abgefragten Zutat übereinstimmt (`if test`). Ist dies der Fall, wird der vollständige Name der Zutat `@name` ausgegeben.

```
46 <xsl:variable name="zutat_id" select="@id" />
47 <xsl:for-each select="//zutat">
48   <xsl:if test="$zutat_id=@id">
49   <xsl:value-of select="@name" />
50   </xsl:if>
51 </xsl:for-each>
```

Codeblock 2.20: Stylesheet - Body - Seite - Cocktails - Zutat

Nun suchen wir die Menge der ausgewählten Zutat. Hier ergibt sich eine weitere Besonderheit. Wie in Codeblock 2.11 angegeben war, handelt es sich bei der Mengeneinheit um ein optionales Attribut. Der folgende Codeblock zeigt ein verschachteltes `choose X when Y; otherwise Z` Element. Ist keine Mengeineinheit über das `menge`-Attribut in der Zutat angegeben, wird auf die festgelegte Einheit `ml` zurückgegriffen. Ansonsten wird über das Attribut `einheit` die entsprechende Einheit aus dem Attribut extrahiert und dargestellt.

Auf diese Weise konnten wir in der Datenbank die Mengeneinheit `ml` auslassen. Da dies die übliche Einheit ist, sparen wir uns somit repetitives Eingeben.

```
52 <xsl:choose>
53   <xsl:when test="@menge">
54   <small class="menge"><xsl:value-of select="@menge" />
55   <xsl:choose>
56   <xsl:when test="@einheit">
57   <xsl:text> </xsl:text>
58   <xsl:value-of select="@einheit"/>
59   </xsl:when>
60   <xsl:otherwise>
61   <xsl:text> ml</xsl:text>
62   </xsl:otherwise>
63   </xsl:choose>
64   </small></xsl:when>
65   <xsl:otherwise></xsl:otherwise>
66 </xsl:choose>
```

Codeblock 2.21: Stylesheet - Body - Seite - Cocktails - Zutat - Menge & Einheit

Zuletzt stellen wir das Cocktail-Glas dar. Hierzu betten wir in `<img>`-Tags das jeweils passende Bild ein, welches sich im Bilder-Ordner befindet. Hierzu nutzen wir das Attribut `glass` des derzeit in der Schleife aktiven Glases.

Bei den Bildern handelt es sich um `.svg`-Dateien, um die Dateigröße möglichst gering zu halten. Außerdem sind `.svg`-Dateien Vektorgrafiken, die den Vorteil haben, sich beliebig skalieren zu lassen. So kann auch eine sehr kleine Datei auf einem sehr großen Bildschirm scharf dargestellt werden.

```
71 <div class="cocktail-glass">
72 <img class="glass-img">
73   <xsl:attribute name="src">img/glass-<xsl:value-of select="
     @glass" />.svg</xsl:attribute>/
74 </img>
75 </div>
```

Codeblock 2.22: Stylesheet - Body - Seite - Cocktails - Glas

2.4 Cascading Stylesheet - `css/main.css`

Die `.css`-Datei, das sogenannte `Cascading StyleSheet`, wird genutzt, um Elemente der `HTML`-Seite grafisch anzupassen. Über diese Datei lassen sich unter anderem Schriftarten, -größen, -farben und weitere Parameter, wie Seitenränder und Zeilenabstände, anpassen.

Im Folgenden werde ich die relevanten Abschnitte der Datei beschreiben.

Zuerst werden externe Schriftarten importiert. Diese sind lizenzfrei zugänglich über `fonts.google.com`.

```
1 @import url('https://fonts.googleapis.com/css?family=Inconsolata|
    Oswald|Raleway');
```

Codeblock 2.23: CSS - Schriftart

Anschließend definieren wir einige Variablen, um Schriftarten und Farben im späteren Verlauf der Datei wiederverwenden zu können.

```
3 :root {
4     --font-sec: 'Raleway', sans-serif;
5     --font-prim: 'Oswald', sans-serif;
6     --font-mono: 'Inconsolata', monospace;
7     --c-white: #FEFEFE;
8     --c-gray-l: #EEE;
9     --c-gray: #555;
10    --c-gray-d: #333;
11 }
```

Codeblock 2.24: CSS - Variablen

Die nächsten Zeilen sind entscheidend für die Menüführung der Website. Wenn der Codeblock nicht vorhanden wäre, würde die einzelnen „Seiten“ (Dokumentation, Cocktails) untereinander angezeigt. Durch das Ansteuern der Pseudoklasse `:target`, die die aktuell aktive Seite definiert, können wir alle anderen Seiten aus- und nur die aktive Seite einblenden.

```
13 div.site:not(:target) {
14     display: none;
15 }
16 div.site:target {
17     display: block;
18 }
```

Codeblock 2.25: CSS - Tabs

In einer `CSS`-Datei können auf verschiedene Arten die einzelnen Seiten-Elemente angesteuert werden. Im nächsten Codeblock ist die Variante dargestellt, die ein Element über den Element-Typ ansteuert. Dies geschieht, indem der rein Name des Tags angegeben wird. Hier: `body`.

Über Attribut-Wert-Paare können nun die Eigenschaften der vorher ausgewählten Elemente geändert werden. Die Angaben `var(-XYZ)` verweisen hier auf die in Codeblock 2.24 angelegten Variablen. Auf diese Art kann über das Ändern des Variablen-Werts die gesamte Seite angepasst werden.

```
20 body {
21     font-family: var(--font-prim);
22     padding: 2rem;
23     background-color: var(--c-white);
24     color: var(--c-gray-d);
25     font-size: 16px;
26 }
```

Codeblock 2.26: CSS - Body

Eine andere Art des Ansteuerns ist, die `CSS`-Klassen der Elemente zu nutzen. Wie in Abschnitt 2.3 bereits mehrfach erwähnt, wurden einzelnen Elementen Klassen zugeordnet. Diese können in der `CSS`-Datei über das Voranstellen eines Punktes angesteuert werden. Hier: `.menu`.

Außerdem können genauere Verschachtelungen von Elementen mithilfe der Schreibweise `X > Y` angesteuert werden. Im vorliegenden Beispiel werden alle Verweise (Anchor) `a` grafisch verändert; jedoch nur, wenn sie sich unterhalb eines Elementes mit der Klasse `menu` befinden.

```
44 .menu > a {
45     padding: 0 1rem;
46     background-color: var(--c-gray-l);
47     color: var(--c-gray-d);
48     margin-right: 1rem;
49     text-decoration: none;
50     text-transform: uppercase;
51 }
```

Codeblock 2.27: CSS - Menü-Links

Die folgenden Codeblöcke zeigen stellen die Responsivität der Website sicher. Dabei wird das moderne `CSS-Grid` genutzt. Im ersten Teil werden die Elemente der Klasse `cocktail-list` bearbeitet. Die interessante Einheit befindet sich in Zeile 65. Das Attribut-Wert-Paar `grid-template-columns: repeat(1, 1fr)` gibt an, dass ein Ein-Spalten-Raster genutzt wird, in dem jede Spalte (also eine) den selben Platz einnimmt (`repeat(1, 1fr)`). `fr` steht hier für *Fraction*; zu deutsch *Teil*.

Ab Zeile 75 beginnen nun die sogenannten `media-queries`. Der Web-Browser nutzt die Informationen über die Fenster-Größe aus, um für unterschiedliche Fenster-Größen andere Angaben zu machen.

Im vorliegenden Beispiel nutzen wir drei sogenannte `break-points`, also Größen, an denen sich die Darstellung der Website verändert. Hier handelt es sich um die Größen `600px-1199px`, `1200px-1799px` und $\geq$`1800px`. Die Größe `<600px` ist implizit durch die ersten Zeilen abdegeckt.

Respektive erhöhen wir hier die Zahlen der Spalten in unserem Raster von einer Spalte auf vier Spalten im größten Raster. Auf diese Weise können wir Bildschirme aller Größen unterstützen.

```
63 .cocktail-list {
64     display: grid;
65     grid-template-columns: repeat(1, 1fr);
66     grid-gap: 1rem;
67 }
```

⋮

```
75 @media only screen and (min-width: 600px) {
76     .cocktail-list {
77         grid-template-columns: repeat(2, 1fr);
78     }
79 }
80 @media only screen and (min-width: 1200px) {
81     .cocktail-list {
82         grid-template-columns: repeat(3, 1fr);
83     }
84 }
85 @media only screen and (min-width: 1800px) {
86     .cocktail-list {
87         grid-template-columns: repeat(4, 1fr);
88     }
89 }
```

Codeblock 2.28: CSS - Media-Queries

Zum Ende der Datei definieren wir die Bewegungs-Animation der Glas-Bilder, die aktiviert wird, wenn der Mauszeiger über einem Bild verharrt (`:hover`).

```
115 .cocktail-glass:hover {
116   -webkit-animation-name: wiggle;
117   -webkit-animation-duration: 0.9s;
118   -webkit-transform-origin:50% 50%;
119   -webkit-animation-iteration-count: 1;
120     -webkit-animation-timing-function: linear;
121 }
```

Codeblock 2.29: CSS - Menü-Links

2.5 Weitere Dateien

Abzüglich der bisher beschriebenen Dateien bleiben uns noch ein paar Dateien übrig, wie aus dem nachfolgenden reduzierten Datei-Baum erkennbar wird.

```
/
├── js/
│   └── script.js
├── img/
│   ├── glass-beach.svg
│   ├── glass-cocktail.svg
│   ├── glass-daiquiri.svg
│   ├── glass-long.svg
│   ├── glass-manhattan.svg
│   ├── glass-martini.svg
│   └── glass-whiskey.svg
├── index.html
├── dokumentation.pdf
└── docs/
    └── ...
```

Wie in Kapitel 2.2 (Codeblock 2.13) bereits erwähnt, handelt es sich bei `js/script.js` derzeit um eine leere Datei. Sie ist dafür gedacht, um `javascript`-Skripte in die Website einzubinden und so zum Beispiel komplexere Animationen zu ermöglichen.

Im Ordner `/img` befinden sich die `.svg`-Dateien der Cocktail-Gläser, die in Codeblock 2.22 genutzt werden.

`index.html` ist die Website-Datei, die durch die `XSL` Transformation der Datenbank und des Stylesheets generiert wird. Sie ist die Ausgabe-Datei, in der die generierte Cocktail-Liste dargestellt wird.

Bei der Datei `dokumentation.pdf` handelt es sich um diese Dokumentation. Sie wird aus den Dateien in `/docs` generiert.

3 Schluss-Erklärung und Ausblick

Wie in 2.5 beschrieben, wird aus der Datenbank `datenbank.xml` und dem Stylesheet `stylesheet.xsl` mithilfe der `XSL` Transformation die Datei `index.html` generiert.

Die Transformation lässt sich beispielsweise mit dem Programm `xsltproc` (Dokumentation unter xmlsoft.org/XSLT/xsltproc.html) über die Kommandozeile vornehmen.

Befindet man sich mit der Kommandozeile im Wurzelverzeichnis dieses Projektes, ist der Transformations-Befehl wiefolgt:

```
1 xsltproc -o index.html stylesheet.xsl datenbank.xml
```

Codeblock 3.1: `XSL` Transformation

Zum Abschluss dieses Projektes möchte ich einen Ausblick geben über mögliche Erweiterungen dieses Projektes.

Die Erweiterung der Datenbank um weitere Cocktails liegt auf der Hand. Zur Veranschaulichung der eingebauten Funktionalitäten befinden sich momentan zwölf Cocktail-Einträge und die dafür nötigen Zutaten in der Datenbank. Über eine Automatisierung ließen sich entsprechend auch größere Mengen an Cocktail-Rezepten in der Datenbank einpflegen.

Eine Rechnung, die man in einem weiteren Projekt einbauen könnte, wäre ein Alkohol-Messer. Wenn man die Zutaten-Einträge in der Datenbank um den Wert des Alkohol-Gehaltes erweitert, so ließe sich aus der Gesamtmenge der Flüssigkeit innerhalb eines Cocktails und den entsprechenden Alkohol-Werten der Gesamt-Alkohol eines Cocktails errechnen.

Literaturverzeichnis

[Kracht, 2018] Kracht, M. (2018). Skript: Informationsstrukturierung.

Eigenständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Bachelorarbeit eigenständig verfasst, und gelieferte Datensätze, Zeichnungen, Skizzen und graphische Darstellungen eigenständig erstellt habe. Ich habe keine anderen Quellen als die angegebenen benutzt und habe die Stellen der Arbeit, die anderen Werken entnommen sind - einschließlich verwendeter Tabellen und Abbildungen - in jedem einzelnen Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht.

Bielefeld, den 2020-03-25

Fabian Wohlgemuth