3.1.1: Struktur der HTML-Seiten:

Im Allgemeinen besteht die Website aus 21 HTML-Seiten:

• author.html

• author-detail.html

• author-add.html

• books.html

• books-detail.html

• books-add.html

• leihen.html

• leihen-detail.html

• Reserve.html

• reservations-detail.html

• Reserve-Add.html

• location.html

• location-detail.html

• location-add.html

• datasecurity.html

• imprint.html

• dashboard.html

• index.html

• profile.html

• login.html

• logout.html

Um durch die vielen Seiten der Website zu navigieren, haben wir uns für eine Navigationsbar entschieden, die sich im Header-Bereich der Website befindet und den unteren Teil des Headers verwendet, der aus einer Liste von Elementen besteht, die die Links enthalten, welche den Benutzer zur gewünschten Seite führen.

Die Seiten, die wir in der Kopfzeile aufgelistet haben, sind Seiten, die Listen der Objekte enthalten, die auf der Website am relevantesten sind.

zum Beispiel:

Eine Liste aller Autoren.

Eine Liste aller Bücher.

Eine Liste aller Standorte.

Auf diesen Seiten kann der Benutzer dann alle auf der Website vorhandenen Objekte sehen und alle wichtigen Informationen zu diesem bestimmten Objekt direkt anzeigen lassen.

Dies wurde durch das Entwerfen der Seiten mit Listen erreicht, da Listen die Möglichkeit bieten, eine große Menge an Informationen gleichzeitig anzuzeigen und dabei eine übersichtliche Ordnung beizubehalten.

Auf jede Liste folgte eine Schaltfläche, mit der der Benutzer der Liste ein Element hinzufügen kann. Durch Drücken der Taste wird eine Formularseite aufgerufen. Diese hat viele Eingabefelder, die dem Benutzer helfen, alle erforderlichen Daten einzugeben, um ein neues Objekt zu erstellen.

Für den Fall, dass die in der Liste angezeigten Informationen nicht ausreichen und der Benutzer auf mehr Details zugreifen will, verfügt jede Liste über einen Link "Details", der zu einer Detailseite führt, die alle Informationen zum gewünschten Objekt enthält.

Die Detailseiten wurden größtenteils mit zwei Abschnitten erstellt: Ein oberer Abschnitt, der ein Foto des Objekts und seiner Beschreibung enthält, und ein unterer Abschnitt, der weitere Informationen zu dem Objekt enthält, die sich von einer Seite zur anderen unterscheiden.

3.1.2: Layout

Um ein einheitliches Layout auf der gesamten Website zu erzielen, haben wir uns entschlossen, zunächst die Kopf-, Seiten- und Fußzeile zu entwerfen und sie dann mit Ejs auf jede Seite der Website anzuwenden. Auf diese Weise haben wir ein einheitliches Design erstellt, das dem Benutzer hilft, sich mit der Struktur der vielen Seiten auf der Website vertraut zu machen.

Zum Beispiel: Unabhängig davon, welche Seite der Benutzer gerade durchsucht, kann er immer die wichtigen Nachrichten und Informationen sehen, die auf der Webseite angezeigt werden.

Nachdem wir das Grundlayout der Seiten fertiggestellt hatten, mussten wir die Mitte jeder Seite mit den relevanten Informationen für diese Seite füllen.

Dafür haben wir auf allen Seiten der Website ein Flex-Design verwendet, da ein Flex-Design sehr dynamisch ist.

Durch die kombinierte Verwendung des Flex-Designs und der Sicherstellung, dass immer Prozentsätze anstelle einer bestimmten Anzahl von Pixeln verwendet werden, konnten wir ein sehr dynamisches Design erzielen, da sich die Elemente im Flex-container nun relativ zueinander bewegen konnten. Das heißt, sie beanspruchen Platz, wenn er verfügbar ist, oder verkleinern sich, wenn ein anderes Element hinzugefügt wird.

Nach Abschluss des Flex-Container-Designs war es eine einfache Aufgabe, zu entscheiden, welches Element welchen Platz beansprucht, je nachdem, um welche Art von Element es sich handelt und wie viel Platz es benötigt.

Durch die Aufteilung der Seite in verschiedene Abschnitte in HTML konnten wir mit mehreren Elementen arbeiten, als wären sie ein größeres Element. Dies erleichterte die Arbeit mit CSS erheblich und führte zu einem sehr soliden Design.

3.1.3: Responsivität

Die Verwendung eines Flex-Container-Designs mit CSS erleichterte unsere Arbeit mit Reaktionsfähigkeit erheblich, weil ein Flex-Container bereits eine gewisse Reaktionsfähigkeit zwischen seinen Elementen hinzufügt.

Wenn beispielsweise ein Element im Flex-container keinen Platz mehr in einer Zeile hat und die Anzeigegröße verkleinert wird, erstellt der Container eine neue Zeile und fügt dieses Element hinzu. Dies bietet ein hohes Maß an Reaktionsfähigkeit auf der Seite.

Um jedoch sicherzustellen, dass unsere Website über viele verschiedene Anzeigegrößen hinweg stabil bleibt und ordnungsgemäß funktioniert, haben wir drei break-points unserer Wahl definiert, anstatt zu versuchen, den Standard-Breakpoints zu folgen, die normalerweise sicherstellen, dass die Website auf vielen verschiedenen Geräten funktioniert (Handys, Tablets, Laptops), da unsere Website hauptsächlich für Desktops konzipiert ist, haben wir dies bei der Festlegung die break-points als höchste Priorität beibehalten.

Unser erster break-point wird erst ausgelöst, wenn die Bildschirmbreite kleiner als 1100 Pixel ist.

Im 1100PX break-point wollten wir nicht viel ändern, da so große Bildschirme bereits genügend Platz bieten, um alle Informationen anzuzeigen, die wir auf unserer Website anzeigen möchten. Deshalb haben wir uns dafür entschieden, den Header-Bereich zu verkleinern, wodurch das Gesamtdesign etwas enger an den Bildschirm angepasst wird.

Danach haben wir weiter getestet und festgestellt, dass wir bei Erreichen einer Breite von 900 Pixel auf Sichtbarkeitsprobleme und einige Fehler gestoßen sind, sodass wir dort einen break-point festgelegt haben.

Der 900PX break-point existiert, weil in dieser Breite die Elemente zu klein sein müssten, um nebeneinander angezeigt zu werden. Dies würde zur Unlesbarkeit der Elemente führen. Um sicherzustellen, dass die gesamte Seite sicht- und lesbar ist, haben wir alle Elemente, die nebeneinander platziert waren, verschoben, um sie in einer einzigen Spalte untereinander zu platzieren. Dies haben wir zunächst auf die gesamte Webseite angewandt, bevor wir auf jeder einzelnen Seite sicherstellten, dass eine einspaltige Ansicht der Flex-Elemente umgesetzt werden kann.

Das Element, das von diesem break-point am stärksten beeinflusst wird, ist die Navigationsbar. Da die 900PX-Anzeigegröße die Links nicht mehr funktionsfähig machte, haben wir die gesamte Leiste in ein Burger-Menü umgewandelt, in welchem beim Anklicken desselben eine Liste der verschiedenen Navigationselemente angezeigt wird.

Danach stellten wir fest, dass der obere Teil des Headers bei 600 Pixel auf Sichtbarkeitsprobleme stieß, und das war unserer letzter break-point: Wir stellten sicher, dass dies behoben wurde, indem wir die Elemente des oberen Teils des Headers mit Display: block übereinander stapelten.

Danach reagierte unsere Website auf allen Bildschirmgrößen eines Desktops vollständig und stabil.