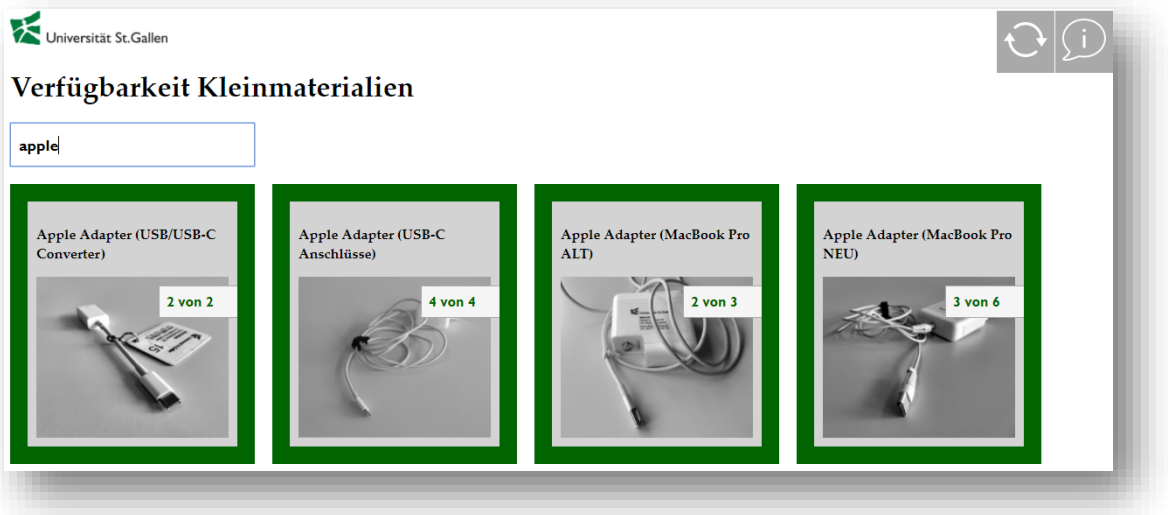
DOKUMENTATION ABSCHLUSSPROJEKT
INTERNETPROGRAMMIERUNG

WEBSERVICE VERFÜGBARKEITS-ANZEIGE KLEINMATERIALIEN

KATHRIN HEIM
BIBINFO16
KAHE2888@TH-WILDAU.DE



INHALT

VERFÜGBARKEITSANZEIGE KLEINMATERIALIEN

DOKUMENTATION WEBSERVICE „AVAILABILITYSERVICE“

PROJEKTZIEL

Studierende und Mitarbeitende der Bibliothek der Universität St.Gallen (HSG-Bibliothek) sollen über eine Webseite eine einfache und rasche Möglichkeit haben, sich selbständig und ortsunabhängig über unser Angebot an Kleinmaterialien (Ladekabel, Adapter, Taschenrechner, etc.) zu informieren und deren aktuelle Verfügbarkeit zu prüfen.
Der hier vorgestellte Webservice gruppiert alle Kleinmaterialien nach Artikel:

- versehen mit einer aussagekräftigen Beschreibung
- illustriert mit einem Bild dazu, auf Klick werden weitere Informationen angezeigt
- aktuelle Verfügbarkeit dieses Artikels (farblich abgehoben, wenn nichts verfügbar)

Die Artikel sollen durchsuchbar sein, damit man nicht durch die ganze Webseite scrollen muss.
Die Webseite soll mobilgerätetauglich sein, damit die Verfügbarkeit auch auf dem Handy rasch geprüft werden kann.

AUSGANGSSITUATION

Die HSG-Bibliothek ist für viele Studierende ein wichtiger Ort zum Lernen und Arbeiten. Damit der «Lernort Bibliothek» möglichst angenehm ist, leihen wir unseren Studierenden auf Tagesbasis diverse Kleinmaterialien aus, die sie im Lern-Alltag brauchen: Ladekabel, Taschenrechner, USB-Sticks, Decken, etc. (über 30 unterschiedliche Artikel). Diese Kleinmaterialien werden von den Mitarbeitenden verwaltet und herausgegeben. Die Kleinmaterialien werden über das Bibliothekssystem ALEPH ausgeliehen. Dabei hat man einen praktischen Ansatz verfolgt: es gibt einen Datensatz «Kleinmaterialien», an dem zurzeit über 150 Exemplare hängen. Die Exemplare sind über eine Kennung und Signatur mit Nummerierung (ein- oder zweistellige Zahl) identifizierbar.

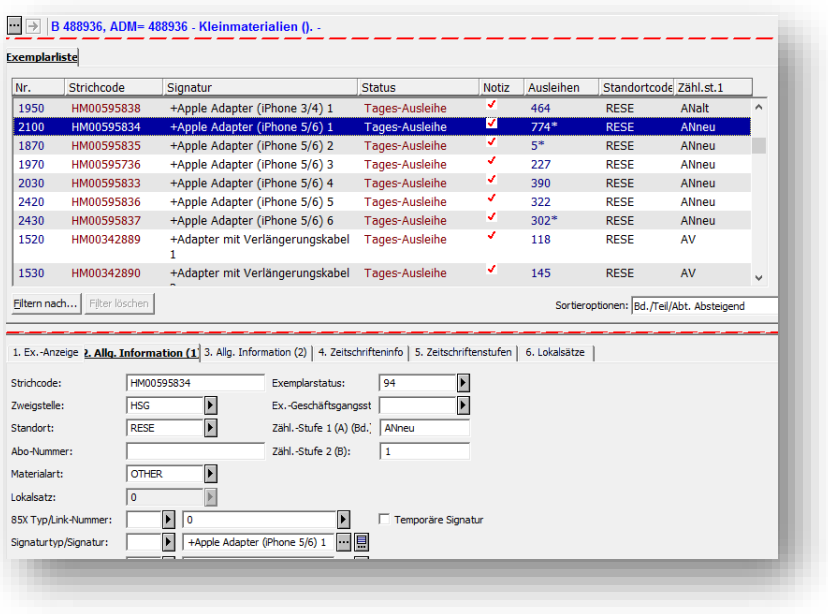


Abbildung 1: Ansicht im Bibliothekssystem ALEPH (GUI)

PROBLEM

ALEPH ist sehr praktisch für die Verwaltung und die Verbuchung an der Ausleihe sowie ein automatisches Mahnwesen, wenn ein Gegenstand nicht rechtzeitig zurückgebracht wird. Allerdings ist es nicht dafür geeignet, das Angebot sowie die Verfügbarkeit der Kleinmaterialien benutzerfreundlich darzustellen. Der Webkatalog wurde für Bücher mit wenigen Exemplaren konzipiert, aber nicht einen Datensatz mit 150 Exemplaren von 30 unterschiedlichen Artikeln. Daher wird die Webansicht nicht benutzt und es ergibt sich an der Ausleihtheke regelmäßig die folgende Situation:
Student: «Haben Sie noch ein Mac-Ladekabel?»
Mitarbeiterin: «Welches brauchen Sie denn?»
Student: «Äh… das fürs Macbook Pro!»
Mitarbeitern: «Das Neue mit USB-C Anschluss?»
Student: «Äh… ???»
Mitarbeitern: «Ich hole Ihnen mal vom Regal, was wir haben und zeig es Ihnen…»
… (geht ans Regal und schaut, was da ist) …
Mitarbeiterin: «Tut mir leid, im Moment sind alle Mac-Ladekabel weg, fragen Sie später nochmals nach.»

Um zu sehen, was überhaupt aktuell verfügbar ist, müssen die Studierenden also zur Ausleihtheke gehen und dort beschreiben können, was sie genau benötigen. Mitarbeitende müssen die Verfügbarkeit in einem Regal und diversen Schubladen überprüfen sowie das Angebot der theoretisch verfügbaren Artikel kennen.
Da es auch immer wieder neue Artikel im Angebot hat, bleibt der Überblick für alle Beteiligten schwierig. Zudem ist diese äußerst beliebte Dienstleistung im Web bisher gar nicht sichtbar (nur Mund-zu-Mund-Werbung).

Abbildung 2: Regal mit diversen Kleinmaterialien (Ausschnitt)

Die generelle Verfügbarkeit von Medien sowie auch der Kleinmaterialien ist in der ALEPH-Datenbank gespeichert und kann über eine Schnittstelle im JSON- oder XML-Format abgefragt werden. Dieser Real Time Availability Checker (RTAC) wird für unser Discovery-System benötigt (EBSCO Information Services, 2016) und kann auch für den Kleinmaterialien-Verfügbarkeitsanzeige genutzt werden. Der Lösungsansatz für das Projekt «Verfügbarkeitsanzeige Kleinmaterialien» basiert auf dieser Schnittstelle.

Die RTAC-Schnittstelle liefert die Daten wahlweise im JSON- oder XML-Format. Da mir aus den anderen Übungen und Projekten JSON (im Zusammenhang mit Ajax/PHP) vertrauter ist, habe ich mich dazu entschieden, die Daten im JSON-Format zu holen.
Leider ist die Performanz der Schnittstelle nicht berauschend, da sie nicht darauf ausgelegt ist, 150 Exemplare auf einmal zu liefern. Für die Performanz der Abfrage spielt das Datenformat keine Rolle.

Die Schnittstelle ist über folgenden Link zugänglich[1]:
http://aleph.unisg.ch/cgi-bin/getRTAC_Info.pl?docnr=488936&lng=fre&format=json
Mit dem GET-Parameter format=xml könnten die Daten auch im XML angefordert werden.

Die Schnittstelle liefert für jedes Exemplar detaillierte Verfügbarkeitsdaten:


```
key:            "000488936002100"
admlib:         "HSD50"
sublib:         "HSG"
sublibLong:     "HSG"
collcode:       "RESE"
collection:     "Ausleihschalter"
level:          2
status_aleph:   "94"
status:         "am Ausleihschalter bestellen"
callno:         "+Apple Adapter (iPhone 5/6) 1"
volume:         "ANneu"
material:       "OTHER"
note:           "Handy-Ladegerät ab iPhone 5"
holdflag:       false
copyflag:       true
available:      false
holdcnt:        0
on_shelf:       false
groupid:        7
due:            20180416
transfer:       false
```

```
key:            "000488936002100"
admlib:         "HSD50"
sublib:         "HSG"
sublibLong:     "HSG"
collcode:       "RESE"
collection:     "Ausleihschalter"
level:          2
status_aleph:   "94"
status:         "am Ausleihschalter bestellen"
callno:         "+Apple Adapter (iPhone 5/6) 1"
volume:         "ANneu"
material:       "OTHER"
note:           "Handy-Ladegerät ab iPhone 5"
holdflag:       false
copyflag:       true
available:      true
holdcnt:        0
on_shelf:       false
groupid:        7
```


Abbildung 3: Ausschnitt aus RTAC-Schnittstelle: iPhone-Ladekabel, nicht verfügbar (links), verfügbar (rechts)

Für die Kleinmaterialien-Verfügbarkeitsanzeige sind folgende Attribute relevant:

- **callno**: normalerweise für Signatur eines Dokumentes verwendet, hier Bezeichnung des Gegenstandes
- **volume**: eindeutige Kennung pro Artikelgruppe (ein oder mehrere Buchstaben)
- **note:** detailliertere Beschreibung des Artikels
- **available:** true (verfügbar) oder false (nicht verfügbar)
- **groupid:** eindeutige Nummer pro Artikelgruppe

Diese Daten können nun gruppiert und für eine Visualisierung aufbereitet werden.

[1] Der GET-Parameter „lng=fre“ ist korrekt. Normalerweise werden bei Dokumenten mit collcode RESE (= Spezialausleihe) nur die verfügbaren Exemplare angezeigt. Damit das Projekt wie gewünscht realisiert werden konnte, hat der Informatiker der HSG-Bibliothek kurzerhand eine neue Sprachversion des RTAC-Service angelegt, welche für diesen collcode auch die nicht verfügbaren Exemplare anzeigt.

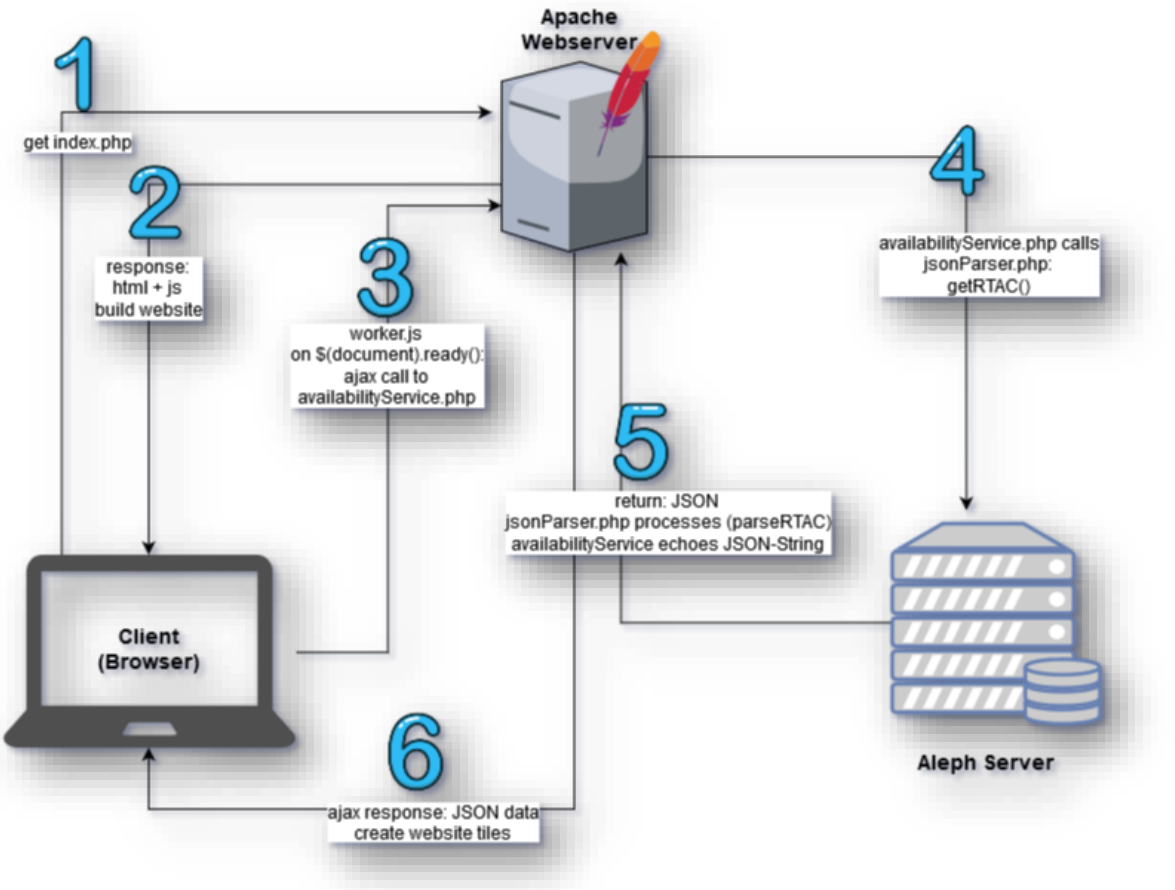


Abbildung 4: Übersicht über die Architektur des Webservice

Die Verfügbarkeitsanzeige für Kleinmaterialien basiert auf Apache, PHP und jQuery (Ajax).

Der Arbeitsablauf des Webservice ist folgendermaßen:

1. Bei Aufruf der Webseite im Browser wird die **index.php** vom Apache-Webserver angefordert.

2. Der Webserver liefert als Response das HTML sowie Javascript an den Browser zurück. Der Header, Logo, Suchschlitz, Navigation und Footer sind da.

3. Der Browser ruft nun das JavaScript File **worker.js** auf. Mit der Funktion **on $(document).ready()** wird der Ajax-Call aufgerufen.
    a. *beforeSend:* Während die Daten geladen werden, wird ein „Lädt…"-icon angezeigt, welches bei success wieder verschwindet.
    b. *success*: der **availabilityService.php** wird aufgerufen.

4. Der Webserver holt nun mittels des **availabilityService.php** bzw. des **jsonParser.php** die Verfügbarkeitsdaten vom ALEPH-Server (RTAC-Schnittstelle, siehe voriges Kapitel). Dazu dient die Funktion **getRTAC():** zunächst werden die Daten von ALEPH angefordert und als String zurückgegeben.

5. Dieser String wird an die Funktion **parseRTAC()** übergeben. Die decodierten Daten werden mithilfe einer Hashtabelle (Array) nach Artikelgruppen aggregiert. Für jede Artikelgruppe werden:
    - die benötigten Attribute gespeichert,
    - das Gesamttotal pro Artikelgruppe ermittelt,
    - die aktuell verfügbaren Exemplare gezählt.

   Wird ein neuer Artikel gefunden, so werden die *groupid* (key) sowie *callno*, *volume*, *total*, *note* und *img_id* (values) dem Array hinzugefügt. Ist der Artikel bereits im Array vorhanden, wird nur *total* hochgezählt. Ist ein Artikel derzeit verfügbar, wird der Wert *available* hochgezählt.
   Das Array wird vom **availabilityService.php** wieder zu einem JSON-String umgewandelt und an den Browser zurückgesandt.

```php
$itemtotal = $jsondecode->itemtotal;
$cluster = array('callno'=>'', 'img_id'=>'', 'total'=>0, 'available'=>0, 'note'=>'', 'volume'=> '');
$list = array('id'=>$cluster);
//var_dump($list);

for ($i = 0; $i< $itemtotal; $i++) {

   $id = $jsondecode->items[$i]->groupid;
   $volume = $jsondecode->items[$i]->volume;
   $callno = $jsondecode->items[$i]->callno;
   $callno = substr($callno, 1, -2);
   $note = $jsondecode->items[$i]->note;
   $available = $jsondecode->items[$i]->available;
   $img_id = $volume.'.png';

   if (array_key_exists($id, $list)) { //entry already exists in $list

        $list[$id]['total'] += 1;

   } else { //new entry in $list

        $list[$id] = $cluster;
        $list[$id]['total'] = 1;
        $list[$id]['callno'] = $callno;
        $list[$id]['img_id'] = $img_id;
        $list[$id]['note'] = $note;
        $list[$id]['volume'] = $volume;
   }

   if ($available) {
       $list[$id]['available'] +=1;
   }
}
```

Abbildung 5: Code-Auszug aus Funktion parseRTAC() im jsonParser.php

6. Der Browser erhält nun von der **worker.js** die Ajax-Response. Die **success-function()** iteriert nun durch alle Artikelgruppen, holt sich die Attribute und bereitet das HTML vor. Wenn es verfügbare Artikel hat, wird eine grüne Kachel erstellt, wenn alle Exemplare eines Artikels ausgeliehen sind, eine rote Kachel sowie rote Schrift. Der Titel und die Verfügbarkeit werden eingeblendet. Das Bild wird aufgrund der Kennung eingebunden und mit Titel/Beschreibung als Alt-Text versehen. Gleichzeitig wird Titel/Beschreibung einem Array hinzugefügt (*search_item*), welches für die Filter-Funktion verwendet wird (siehe folgendes Kapitel). Wenn die Schleife durchlaufen ist, wird das HTML angezeigt.

WEITERE FUNKTIONEN

Filter-Funktion

Die Filter-Funktion liegt in der **worker.js**. Sie wird mittels Tastatureingabe-Event im Suchschlitz aufgerufen.

Zunächst wird der eingegebene Text in einer Variable *input* gespeichert (in Kleinbuchstaben), außerdem werden alle Kacheln vorerst ausgeblendet.
Die Funktion iteriert nun durch das oben angelegte Array *search_item* und vergleicht den Input mit den Array-Values. Dies geschieht mittels der Funktion **inputOf().** Die Methode liefert bei einem Match den Index des Wertes im Array, ansonsten -1. Wenn der zurückgelieferte Wert also nicht -1 ist, wird die entsprechende Kachel angezeigt.

```
var filter = function() {
    var input = $('#filter-search').val().toLowerCase();

    $('.tile').css('display', 'none');

    //loop through array
    for (var index in search_item) {

        if (search_item[index].indexOf(input) >-1) {
            $('.'+index+'').css('display', '');
        }

    }


};

$('#filter-search').on('input', filter);
```

Abbildung 6: Code der Filter-Funktion in worker.js

Es handelt sich hierbei nicht um eine intelligente Suche, sondern um eine reine Buchstabenfolge-Filterfunktion.

Detailinfo-Anzeige mit Toast

Um die Webseite übersichtlicher zu gestalten, wurden die ausführlichen Beschreibungen (*note*) in den Bild-Texten (Alt-Attribut) hinterlegt. Sie können mit Klick aufs Bild mittels eines Pop-ups (Toast) sichtbar gemacht werden. Der Toast wurde bewusst mittig (mid-center) platziert, so dass er in der mobilen Ansicht gut lesbar ist.
Dazu wurde ein bestehendes jQuery-Plugin (Kamran, 2016) benutzt und angepasst. Der Toast ist in der toast.js definiert. Zusätzlich wurde das dazugehörige CSS-File jquery.toast.min.css eingebunden. Die meisten Attribute sind direkt in der toast.js anpassbar, einige Attribute wie Text- und Toastgröße wurden im availabilityStyle.css jedoch überschrieben und auch an die Media Query angepasst.

Abbildung 7: Beispiel Toast mit Alt-Attribut

Hilfe-Funktion: info.php

Einige Hinweise zur Nutzung des Angebots sowie die Möglichkeit, neue Artikel vorzuschlagen, wurde auf der Seite availabilityService/info.php aufgenommen. Die HSG-Bibliothek bietet den Studierenden eine „Ideenwand“[2], eine Art Open-Innovation-Plattform, wo sie Anregungen und Kritik anonym und ohne Login anbringen können. Zudem sind E-Mailadresse, Telefonnummer und die Öffnungszeiten erwähnt, falls jemand die Bibliothek direkt kontaktieren möchte. Auf ein Kontaktformular wurde daher verzichtet.

LAYOUT / RESPONSIVE DESIGN

Layout

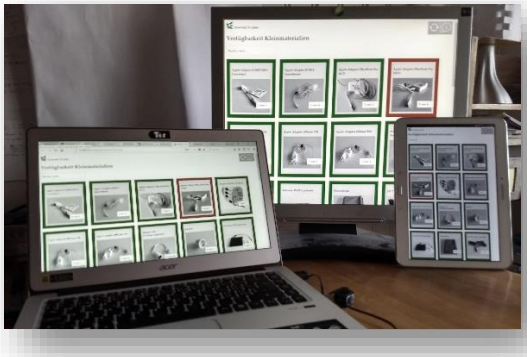
Das Webdesign für die Verfügbarkeitsanzeige orientiert sich lose am Internet Styleguide der Universität St. Gallen[3]: Farben, Logo und Schriftarten wurden vom HSG-Webdesign übernommen. Dies dient v.a. dem Wiedererkennungswert und bietet die Möglichkeit, die Webseite ohne größere Einwände der entsprechenden Gremien auf den Webseiten der Bibliothek zu integrieren.
Das Design ist schlicht gehalten, es gibt – außer den Artikelbeschreibungen – kaum Text. Die Navigation läuft über zwei Icons. Die Icons stammen von icons8.com und laufen unter einer Creative-Commons-Lizenz.[4] Das Hauptaugenmerk der Seite liegt auf den Kacheln mit den Fotos der Artikel. Alle Fotos stammen von der Autorin, damit keine Urheberrechte verletzt werden. Die Bilder wurden so aufgenommen, dass der Artikel gut zu erkennen ist. Als Bildformat wurde 600 x 500px gewählt, die Bilder wurden alle entsättigt, damit die Webseite einheitlich aussieht.

Für jede Artikelgruppe gibt es eine Kachel mit Titel, Bild, Verfügbarkeit sowie einer detaillierten Beschreibung im Bild-Text (Alt-Attribut). Die Beschreibung kann durch Klick auf das Bild in Form eines Toasts angezeigt werden. Der Toast verschwindet automatisch nach einigen Sekunden, kann aber auch manuell geschlossen werden. Die Nichtverfügbarkeit eines Artikels wird durch einen roten Rahmen hervorgehoben. Der Suchschlitz wurde per CSS vergrößert, damit er gut sicht- und nutzbar ist. Es wurde bewusst kein Suchbutton angelegt, da es sich nur um eine Filter-Funktion handelt. Es ist somit für den Nutzer bereits beim Tippen sichtbar, was die Funktion tut.

Responsive Design

Die Webseite wurde so gestaltet, dass sich die Anzahl der Kacheln je nach Bildschirmgröße, bzw. Bildschirmausrichtung automatisch anordnen. Die Größe der Kachel ist mit fixen Werten festgelegt, welche für die meisten Laptop-, Desktop-, sowie ältere Smartphone- und Tablet-Bildschirme geeignet ist:

[2] https://de.padlet.com/HSGBibliothek/Ideenwand
[3] https://www.unisg.ch/de/universitaet/bibliothek
[4] https://icons8.com/license



```
.green_tile {
    border: 20px solid darkgreen;
}

.red_tile {
    border: 20px solid darkred;
}

.tile {
    background-color: lightgrey;
    padding: 10px;
    margin: 10px;
    float:left;
    width: 220px;
    max-height: 260px;
    overflow: hidden;
}
```

Abbildung 8: Ansicht auf Laptop, Desktop, Tablet

Bei älteren Smartphones mit niedriger Bildschirmauflösung wird daher automatisch (ohne Media Query) in der Breite nur eine Kachel angezeigt und alle Kacheln untereinander angeordnet. Neuere Smartphones haben allerdings eine so hohe Auflösung (Retina/HD), dass die Seite auch auf dem kleinen Bildschirm mit 3 Kacheln angezeigt wird, wodurch die Lesbarkeit ungenügend und das Handling mühsam wird (z.B. Navigation, Suche, Toasts).
Die meisten Lehrbücher/Tutorials machen die dazu benötigte Media Query mit min-width, was heutzutage eher nutzlos ist für hochauflösende Smartphones. Daher habe ich nach Media Query für HD und Retina Displays recherchiert und bin nach den empfohlenen Werten (King, 2016) gegangen, wobei ich nur mit der min-resolution gearbeitet habe. Die Auflösung von 200dpi gilt für die meisten neueren Smartphones (z.B. iPhone 4 und höher).
Im CSS wurden in der Media Query die Kacheln und Bilder auf die ganze Bildschirmbreite gesetzt. Des Weiteren wurden die Schriften, der Toast und die Navigations-Icons vergrößert.


```
@media
only screen and ( min-resolution: 200dpi)
{
        /* Resolution-specific CSS for HD & Retina

        header {
            font-size: 3em;
        }

        .tile {
            width: 85%;
            max-height:fit-content;
            font-size: 2.5em;

        }

        .tile img {
            width: 100%;

        }
```

Abbildung 9: Ansicht auf Smartphone mit HD-Auflösung, Ausschnitt Media Query im CSS

Die Webseite wurde mit folgenden Geräten getestet:

- Alter Desktop-Bildschirm (96 dpi), Auflösung: 1280 x 1024
- 3-jähriges Android-Tablet (96 dpi), Auflösung: 1280 x 800
- 1-jähriger Laptop (144 dpi), Auflösung: 1920 x 1080
- 1-jähriges Android-Smartphone (288dpi), Auflösung: 1920 x 1080.

Zum Herausfinden der device pixel density (dpi) des aktuell genutzten Bildschirms gibt es ein sehr praktisches Webtool ("Device pixel density tests," 2018)
Um mit echten Geräten testen zu können statt nur Browser-Emulatoren (welche die Bildschirmauflösung der neueren Geräte nicht korrekt darstellen), wurde in der Entwicklungsumgebung die httpd-xampp.conf für die Dauer der Tests Zugriff via IP gewährt.

TESTS

Da die Verfügbarkeitsanzeige ein Real-Time-Service ist, kann die Webseite jederzeit aufgerufen und aktualisiert werden. Allerdings ist die „Fluktuation" der Verfügbarkeit nur während den Öffnungszeiten[5] der Bibliothek sichtbar. Außerdem wird die Dienstleistung während des Semesters häufig genutzt, während in den Semesterferien kaum etwas entliehen wird. Um das Testen zu vereinfachen, gibt es im Projekt einen Ordner functions/data, in welchem einige Momentaufnahmen aus dem Frühlingssemester 2018 als Testdateien im JSON-Format abgelegt sind. In allen Testdateien ist immer mindestens 1 Artikel nicht verfügbar (roten Kachel).
Die Testdateien können in der Methode getRTAC() in der jsonParser.php ein-/auskommentiert werden. Dazu wird die Variable $file unter „Realtime-Service" auskommentiert und eine beliebige JSON-Datei aus den Test-Files einkommentiert.

```php
function getRTAC () {

    //Realtime-Service:
    //For testing, comment out line $file = "http://aleph.unisg.ch..."

    $format = 'json';
    $docnr = '000488936';
    $file = "http://aleph.unisg.ch/cgi-bin/getRTAC_Info.pl?docnr=" . $docnr . "&lng=fre&format=" . $format;

    //Test files: uncomment the desired test file:
    //Simulate status as of 21 March 2018
    //$file = "data/test_21032018.json";
    //Simulate status as of  as of 4 April 2018
    //$file = "data/test_04042018.json";
    //Simulate status as of 13 April 2018
    //$file = "data/test_13042018.json";
```

Abbildung 10: Testdateien im JSON-Format

[5] https://www.unisg.ch/de/universitaet/bibliothek/ueberuns/oeffnungszeiten

Live-Version der Verfügbarkeitsanzeige für Kleinmaterialien

Ziel dieser Arbeit war, eine brauchbare Webseite abzuliefern. Als ein Prototyp der Webseite fertig war, wurden daher die Mitarbeitenden der Ausleihe, welche regelmäßig mit den Kleinmaterialien arbeiten, konsultiert und um Feedback gebeten. Aufgrund dieses Feedbacks wurden v.a. am Layout noch einige kleinere Änderungen vorgenommen. Beispielsweise verdeckte die Verfügbarkeitsanzeige oft genau den wichtigen Stecker auf dem Bild, daher wurde das Layout angepasst. Einige Fotos werden nochmals neu (mit einer professionelleren Kamera) gemacht oder besser zugeschnitten, da die Unterschiede, z.B. bei den Anschlüssen der Macbook-Ladekabel, zu wenig sichtbar sind.
Die Rückmeldungen waren jedoch durchwegs positiv und so wurde nach einer Lösung gesucht, die Seite online zu stellen. Da die Bibliothek auf dem CMS ihrer Webseite nicht so viele Rechte hat, um auch Skripte zu integrieren, wurde die Seite kurzerhand auf dem ALEPH-Webserver publiziert, auf dem auch die Schnittstelle läuft. Der Bibliotheksinformatiker der HSG-Bibliothek hat die Webseite bereits online gestellt.
Link zur Live-Version: http://aleph.unisg.ch/php/kleinmaterial/kleinmaterial.php

Allerdings stellte sich heraus, dass die Skripte so nicht funktionieren, da wir auf dem ALEPH-Webserver eine alte PHP-Version (5.1.6) einsetzen müssen, welche die Funktion decode_json nicht kennt. Daher hat unser Bibliotheksinformatiker Felix Leu den Code umgeschrieben und so vereinfacht, dass gar kein PHP benötigt wird. Seine Version ist in der kleinmaterial.js:
https://aleph.unisg.ch/php/kleinmaterial/js/kleinmaterial.js

Die Verfügbarkeitsanzeige ist auf der Homepage der HSG-Bibliothek bereits verlinkt und kann seit 12.4.2018 genutzt werden. Allerdings ist dies nicht die Originalversion von mir, sondern die umgeschriebene Version von Herr Leu, daher sollte zur Bewertung des Projekts die gezippten Dateien (Moodle-Upload) verwendet werden.

Neuen Artikel einfügen

Die Verfügbarkeitsanzeige ist so konzipiert, dass (fast) alle Änderungen direkt in ALEPH erfasst werden können. Somit können die Kolleginnen an der Ausleihe mit dem ihnen vertrauten System arbeiten wie bisher. Kommt ein zusätzliches Exemplar eines bereits bestehenden Artikels hinzu (z.B. ein weiteres iPhone-Ladekabel), wird in Aleph ein bestehendes Exemplar dupliziert und nur die Nummerierung in der Signatur sowie der Strichcode angepasst. Der Webservice ist so aufgebaut, dass es keine Rolle spielt, wie viele Exemplare von einem Artikel existieren.
Kommt ein gänzlich neuer Artikel hinzu, so muss in ALEPH eine neue Kennung vergeben und ein neuer Exemplarsatz mit dieser Kennung angelegt werden. Dies war eigentlich bereits bisher so, der einzige Unterschied ist nun, dass die Kennung eine Rolle für die Sortierung auf der Webansicht spielt und daher sinnvoll gewählt werden muss. Außerdem sollen die Detail-Beschreibungen auch bei neuen Artikeln in ALEPH gepflegt werden. Dazu wurde eine Anleitung fürs hausinterne Wiki erstellt.
Der einzige Punkt, den die Kolleginnen nicht vollständig selber erledigen können, ist das Hinzufügen eines Fotos des neuen Artikels. Grundsätzlich funktioniert der Webservice auch ohne Bild, in der Kachel wird dann einfach der Alt-Text angezeigt. Da die Kachel dann aber nicht mehr gleich groß ist, wird je nach Browser die Seite nicht mehr schön einheitlich dargestellt. Daher sollte baldmöglichst ein entsprechendes Bild hochgeladen werden.
Das neue Bild (png-Format) im korrekten Format muss mit der Kennung benannt und im Ordner „availabilityServic/img“ abgelegt werden. Für den Live-Service hat Herr Leu dazu ein Upload-Formular angepasst, welches auch für andere Datei-Hochladevorgänge auf den ALEPH-Server genutzt wird. So können einige berechtigte Personen ohne Umweg über die IT selbständig das neue Bild hochladen.
Für die Pflege der Texte, Layout, etc. muss die IT-Abteilung kontaktiert werden.

Ausbau- / Verbesserungsmöglichkeiten

- Es wäre wünschenswert, die Seite auch auf **Englisch** anbieten zu können. Dies kann sicher noch realisiert werden, wurde aber für dieses Projekt nicht umgesetzt, da ALEPH nicht zweisprachig verfügbar ist und die Text der Artikel somit nur in einer Sprache abgelegt werden können. Für einen zweisprachigen Dienst müsste man sich also überlegen, wie und wo die Übersetzungen der Artikel sowie Beschreibungen abgelegt werden, damit sie trotzdem weiterhin von Nicht-IT-Personal gepflegt werden können.

- Der Wunsch nach einem „**Wunschformular**" für neue Artikel wurde bereits geäußert, es wird vorerst abgewartet, wie populär die Seite wird, bevor ein solches Formular erstellt wird. Vorerst ist für Neuanschaffungs-Wünsche die Ideenwand[6] und Kontakt per E-Mail verlinkt.

- Um eine bessere **Suche** zu ermöglichen, wurde der Datensatz in der ALEPH-Datenbank ebenfalls mit Stichworten angereichert, so dass auch über die Bibliothekssuche nach „Ladekabel", etc. gesucht werden kann. Herr Leu, der Informatiker, hat es so eingerichtet, dass bei diesem Datensatz nicht die normale Verfügbarkeitsanzeige für Bücher, sondern direkt die Verfügbarkeitsanzeige für Kleinmaterialien angezeigt wird. Diese Suche funktioniert erst ab ca. Mai 2018, da der Datensatz noch nicht vom Discovery-System neu indexiert wurde (dauert i.d.R. eine Woche).

- Die **info.php** könnte theoretisch aufgehoben und in die index.php integriert werden (z.B. als , welches nur auf Klick eingeblendet wird). Dann hätte man einen echten „One-Pager", und es wäre die elegantere Lösung. Dies kann bei einer Überarbeitung der Webseite sicher noch umgesetzt werden.

- Es gehen immer wieder Artikel verloren oder müssen repariert werden. Bis der Artikel wieder verfügbar ist, werden diese Fälle in Aleph umgestellt auf einen **Reparatur- oder Verlust-Status**, welcher jedoch von der Verfügbarkeitsanzeige bisher nicht berücksichtigt wird. Je nach verwendetem Status wird der Artikel doppelt angezeigt, da ALEPH in solchen Fällen eine neue *groupid* erstellt und daher 2 Kacheln entstehen.

Abbildung 11: Fehlerhafte Anzeige bei Reparaturfällen

- Es könnten noch zusätzliche **Media Queries** eingefügt werden, um mehr Geräte abzudecken. Es hat sich z.B. gezeigt, dass die Seite bei älteren Smartphones, welche eine Device-Pixel-Ratio grösser 1, aber eine kleinere Resolution als 200 dpi haben, die Seite „klein gezoomt" wird. Außerdem haben wir einen Anzeigebildschirm vor der Bibliothek

[6] https://de.padlet.com/HSGBibliothek/Ideenwand

für die Bewerbung von Dienstleistungen und News-Meldungen, auf dem die Seite ebenfalls angezeigt wird. Jedoch sollten für diese Anzeige mindestens 4 Kacheln in der Breite gezeigt werden, damit alle Artikel sichtbar sind (der Bildschirm scrollt nicht):

Abbildung 12: Bildschirm vor der Bibliothek

ARBEITSWEISE / LESSONS LEARNED

Ich habe das Projekt in Netbeans und XAMPP erstellt, die Versionierung machte ich mit GitHub. Der Code des Webservice kann somit auch von GitHub geklont werden: https://github.com/kathrin77/availabilityService

Als Quellen für das Projekt habe ich mich hauptsächlich auf die Online-Dokumentation von PHP (Cowburn, 2018), jQuery (jQuery Foundation, 2018) sowie W3schools (w3schools.com, 2018) gestützt. Weiter nutzte ich einige Bücher, die mir empfohlen wurden, vor allem für die Gestaltung der Seite (Duckett, 2014) sowie einige Hintergrundinformationen zu Ajax (Dayley, 2014; Duckett, 2015). Für einige Probleme waren Online-Hilfeforen wie stackoverflow sehr hilfreich, da die Beispiele in den Büchern und Dokumentationen oft sehr allgemein gehalten sind.

Ich habe auf jeden Fall bei diesem Projekt viel gelernt, insbesondere über Ajax und jQuery. Ein erster Versuch, die ganze Seite nur mit JavaScript umzusetzen, scheiterte an meinen (zu geringen) Kenntnissen. Geprägt von den Übungen aus dem Unterricht, realisierte ich zunächst das Parsen und „Clustern" der Artikelgruppen mit PHP, um dann mit dem Ajax-Call nur noch das HTML zu erstellen. Dabei werden die JSON-Daten in meinem Code mehrfach de- bzw. encoded. Es ist mir aufgefallen, dass dies nicht sonderlich elegant ist, eine einfachere Lösung erschloss sich mir aber nicht. Erst als mir Herr Leu seine Variante für den Live-Service präsentierte, erkannte ich, dass es nicht so schwer gewesen wäre.

Für das Layout wurde mir empfohlen, mit Bootstrap o.ä. zu arbeiten, um das Kachel-Design einfacher gestalten zu können. Ich hatte zu dem Zeitpunkt jedoch bereits einen großen Teil des Layouts fertig und kam auf Anhieb nicht klar mit diesem Framework, es hätte mich eine längere Einarbeitungszeit gekostet, die ich nicht hatte. Für ein nächstes Projekt wäre es sicher empfehlenswert, von Anfang an mit einem CSS-Framework zu arbeiten, insbesondere bei größeren Projekten über mehrere Seiten.

Cowburn, P. (2018). PHP-Handbuch - Manual. Retrieved from http://php.net/manual/de/index.php

Dayley, B. (2014). *jQuery and JavaScript phrasebook*. Upper Saddle River, NJ: Addison-Wesley. Retrieved from http://proquest.tech.safaribooksonline.de/9780133410877

Device pixel density tests. (2018). Retrieved from https://bjango.com/articles/min-device-pixel-ratio/

Duckett, J. (2014). *HTML & CSS: Erfolgreich Websites gestalten & programmieren* (1. Aufl.). Weinheim: Wiley/Wrox.

Duckett, J. (2015). *JavaScript & jQuery: Interaktive Websites entwickeln* (1. Aufl.). Weinheim: wrox, a Wiley brand.

EBSCO Information Services. (2016). Real Time Availability Checking (RTAC). Retrieved from https://help.ebsco.com/interfaces/EBSCO_Discovery_Service/EDS_Admin_Guide/Real_Time_Availability_Checking

JQuery Foundation. (2018). jQuery API Documentation. Retrieved from http://api.jquery.com/

Kamran, A. (2016). Jquery Toast Plugin. Retrieved from http://kamranahmed.info/toast

King, M. (2016). HD & Retina Display Media Queries. Retrieved from https://medium.com/@micjamking/hd-retina-display-media-queries-b5562b5430d6

W3schools.com. (2018). JavaScript and HTML DOM Reference. Retrieved from https://www.w3schools.com/jsref/default.asp