Anselm Bunsen, Leonard Follner

Dokumentation - Arabismen

Technische Dokumentation

Systemvorraussetzungen

Clientseitig

Da es sich um eine Webanwendung handelt, sind die Anforderungen clientseitig den üblichen Anforderungen für WebApps entsprechend.

Es wird ein beliebiger moderner Browser benötigt. Auch wenn die Entwicklung für möglichst viele Browser erfolgt ist, kann es vereinzelt zu meist visuellem Fehlverhalten kommen. Daher ist die Verwendung von Chrome sowohl auf Desktops und Laptops alsauch auf mobilen Endgeräten empfohlen, jedoch nicht erforderlich.

Die Anwendung ist eine Webanwendung mit Ansätzen, die aus progressiven WebApps bekannt sind. Somit stützt sie sich in ihrer Funktionalität vollständig auf JavaScript. Browser, in denen JavaScript deaktiviert ist, können somit nicht unterstützt werden. Es wird allerdings eine Fehlermeldung angezeigt, die die dieses Problem kurz erläutert.

Damit eine Interaktion mit der Anwendung in einer unproblematischen Weise möglich ist, ist eine Mindestdisplaygröße erfoderlich. Die Anwendung prüft selbstständig, ob diese gegeben ist und zeigt eine Warnung an, falls die Größe unterschritten sein sollte. Empfohlen ist ein Display, das mindestens die Größe eines durchschnittlichen Tablets hat.

Zum Darstellen der Karte im Hintergrund müssen die Kartendaten von einem Server heruntergeladen werden. Dafür ist eine aktive Internetverbindung zumindest beim ersten Öffnen der Anwendung erforderlich. Für das Laden weiterer Kartenansichten (beispielsweise für die Detailansicht), kann auch während der weiteren Benutzung der Anwendung eine Internetverbindung erforderlich sein. Grundlegend ist die weitere Benutzung auch ohne eine bestehende Internetverbindung möglich. Dann wird die kontextgebende Karte allerdings nicht angezeigt.

Serverseitig

Die einzige serverseitige Voraussetzung ist ein beliebiger installierter Webserver. Dieser muss selbst keine besonderen Bedingungen erfüllen, da keine serverseitigen Berechnungen erfolgen. Er muss lediglich in der Lage sein die entsprechende Menge an Besuchern zu bedienen.

Der WebServer kann prinzipiell auch das clientseitige Caching erlauben. Dafür können jegliche Dateien grundlegend beliebig lange clientseitig gecached werden, da der Build-Prozess die Dateien mit content-hashes versieht. Davon ausgenommen werden sollten folgende Dateien, die gar nicht gechached werden dürfen:

_jegliche Bilder im Ordner **`/images`** (es sei denn diese werden nur hinzugefügt und nie geändert)

_die Datei **`data.js`**, da diese die aktualisierbare Datengrundlage bildet (siehe dazu auch Benutzung und Anpassung)

_die Datei **`index.html`**, da über diese die Cache-Invalidierung beim Deployen neuer Versionen erfolgt

_die Datei **`locations.js`** (hierfür gilt das gleiche wie für **`data.js`**)

_die Datei **`main.css`**, da für diese auf Grund des aktuellen Build-Prozesses keine Cache-Invalidierung durch Content-Hashes möglich ist.

Installationsanweisung

vorgefertigter Release

Im ownCloud des Lehrstuhls befindet sich ein vorgefertigter Release der Anwendung. Die Inhalte dies Ordners müssen lediglich in ein Verzeichnis eines WebServers kopiert und online zugänglich gemacht werden.

Release selbst bauen

siehe dazu „Entwicklungshinweise“

Updates

Bei einem Update auf eine neuere Version der Anwendung ist es lediglich erforderlich folgende Dateien zu ersetzen:

_die Datei **build/index.html**

_der Ordner **build/static**

_der Ordner **build/main.css**

Es stellt allerdings auch kein Problem dar, das gesamte Verzeichnis zu ersetzen. Dabei ist allerdings auf eine Sicherung der Inhalte der Dateien **data.js** und **locations.js** zu achten, um Datenverluste zu vermeiden (siehe dazu auch Anpassung der Inhalte).

Bedienungsanleitung

Die Webanwendung erfordert keine speziellen Kenntnisse zur Bedienung. Es muss lediglich mit einem Webbrowser das entsprechende verzeichnis des Webservers aufgerufen werden, in dem die Anwendung installiert wurde. Zu beachten sind dabei die Systemvoraussetzungen.

Im unkompilierten Zustand sind auch noch Anpassungen der Funkionalität der Anwendung über gesonderte Steuerflags möglich. Dazu muss vor dem Build-Prozess die Datei app/src/settings.js angepasst werden.

Mögliche Anpassungen sind:

_export const maxNumberOfCardsOnMap = 3;:

Definiert die Anzahl an Begriffen, die maximal gleichzeitig auf der Karte angezeigt werden.

_export const maxNumberOfCardsInSideBar = 5;:

Definiert die Anzahl an Begriffen, die maximal gleichzeitig in der Seitenleiste angezeigt werden.

_export const swipeGestures = false;:

In der Detailansicht kann auf mobilen Endgeräten mittels Wischgesten zwischen den einzelnen Abschnitten gewechselt werden. Diese Funktionalität kann hier (de-) aktiviert werden.

_export const mobileWarningMayBeIgnored = false;:

Das ändern dieser Einstellung erlaubt es die Warnung, dass das Dispaly des Gerätes zu klein sei, mit einem Kreuz zu ignorieren.

_export const minimumDeviceWidthNeeded = 750;:

Mittels dieser Eigenschaft kann die Breite eines Displays eingestellt werden, die mindestens benötigt wird, um die Anwendung zu benutzen.

_export const mapOverlayTransparency = 0.65;:

Dieser Wert beeinflusst prozentual die Transparenz der Flächen, die auf er Karte angezeigt werden.

_export const mapOverlayColorEurope = ‚#5F9EA0‘; export const mapOverlayColorArab = ‚#DC143C‘;:

_Mittels dieser beiden Werte lassen sich die Farben für die Zielregionen anpassen.

Die hier aufgeführten Werte sind die Standardwerte.

Grundlegend sollten alle diese Werte synchron in die Datei **app/src/sass/Utilities/ _ variables.scss** eingetragen werden, da die Änderungen sonst nur teilweise angezeigt werden könnten. Die Syntax entspricht jeweils der von JavaScript und SCSS.

Vorbereitung

Es sind keine gesonderten Vorbereitungen nach der Installation erforderlich. Die Anwendung ist vollständig einsatzbereit. Es ist allerdings empfehlenswert nach der Erstinstallation die Inhalte der Anwendung nach der folgenden Anleitung anzupassen:

Anpassung der Inhalte

Die Inhalte der Anwendung können auch nach dem Build noch angepasst werden. Hierfür sind zwei Dateien entschiedend:

In **data.js** befinden sich alle Begriffe, deren Definitionen und die Erklärung der Begrisswandlung

In **locations.js** werden lediglich die einzelnen Orte definiert, die in **data.js** verwendet werden können.

Wichtig ist, dass diese Dateien sowohl im Ordner **app/src/public** und **app/src/build** zu finden sind. Beim Erstellen eigener Builds nach den Installationsanweisungen ist zu beachten, dass die Dateien in **app/src/build** bei jedem Build erneut von denen in **app/src/public** überschrieben werden. Dementsprechend kann es sinnvoll sein, die Dateien außerhalb des Projektes zwischenzuspeichern und erst nach dem Build nach **app/src/build** zu übertragen.

Für die Anpassung der Inhalte ist eine grundlegende Kenntnis der JSON-Notation hilfreich. Am Ende der beiden Dateien befinden sich Vorlagen zum Anlegen eines neuen Begriffs oder eines neuen Ortes. Diese können einfach kopiert werden (ohne die Zeilen, die nur **/*** und ***/** beinhalten) und vor der letzten Zeile]; eingefügt werden. Wichtig ist dabei, dass die Klammern erhalten beliben und die Vorlage nicht innerhalb anderer geschweifter Klammern eingefügt wird.

Anpassung der Begriffe

Jeder Begriff entspricht folgender Struktur:

```
{
    „id“: ,
    „name“: „“,
    „arab“: „“,
    „description“: „“,
    „details“: {
      „type“: „sections/map“,
      „changes“: [
      // for sections
        {
          „teaser“: „“,
          „fullDetail“: „“
        },
        // or for map locations
        {
```

```
            „location“: „name from locations.js“,
            „description“: „“
          }
        ]
      },
       „targetRegion“: „Arab“ / „Europe“
    }
```

Die einzelnen Werte müssen dabei wie folgt vor dem Komma eingefügt werden:

_id: eine fortlaufend um 1 inkrementierte Zahl

_name: der Begriff selbst

_arab: die arabische Übersetzung des Begriffs (aus Entwicklungsgründen wird dieser aktuell noch nicht in der Anwendung angezeigt)

_description: eine kurze Beschreibung des Begriffs

_details: beinhaltet innerhalb der geschweiften Klammern alle Schritte der Begiffserklärung in einem von zwei Formaten:

_type: wählt den Typ der Detailansicht, kann entweder „**sections**“ oder „**map**“

_changes:

_wenn als type „sections“ gewählt wurde, muss changes Objekte wie folgt beinhalten:

```
{
        „teaser“: „“,
        „fullDetail“: „“
 },
```

_teaser: beschreibt die kurze Überschrift des Textes dieses Abschnittes in der Detailansicht

_fullDetail: beinhaltet den gesamten Text des Abschnittes

_die Abschnitte werden in der Reihenfolge aus dieser Datei angezeigt

_Zur Anpassung der Bilder der einzelnen Abschnitte siehe „Anpassung der Bilder“

_wenn als **type „map“** gewählt wurde, müssen Objekte vom folgenden Typ eingefügt werden:

```
{
        „location“: „name from locations.js“,
        „description“: „“
},
```

_location: ist der Name des Ortes aus der Datei **locations.js**, siehe dazu „Anpassung der Orte“

_description: ist der Text, der neben dem Ort angezeigt wird

_targetRegion: ist die Einordnung des Begriffs in den jeweiligen Sprachraum und kann entweder „Arab“ oder „Europe“ sein.

Anpassung der Orte

Jeder Ort entspricht der folgenden Struktur:

```
{
    „name“: „“,
    „coordinates“: {
      „lat“: ,
```

```
        „lng“:
      }
    },
```

Die einzelnen Werte müssen dabei wie folgt vor dem Komma eingefügt werden:

_name: der Name des Ortes, wie er angezeigt werden soll und in der Datei **data.js** verwendet wird

_coordinates: beinhaltet die Koordinaten des Ortes auf der Karte

Anpassung der Bilder

Bilder sind im Ordner **images** abzulegen. Jeder Begriff erhält dafür einen Unterordner mit dem Namen des Begriffs.

Das Bild, das in der Hauptansicht in der Karte angeigt wird und auch als Marker auf der Karte verwendet wird, muss den gleichen Namen wie der Begriff tragen.

Falls für die Detailansicht der Modus „sections“ in „Anpassung der Begriffe“ gewählt wurde, müssen die Bilder den Namen des Begriffs und eine fortlaufende bei 0 beginnende Nummer enthalten. Beide Teile werden durch einen - voneinander getrennt. Die Bilder werden in der Reihenfolge der Nummern für die Abschnitte verwendet.

Als Format für die Bilder steht ausschließlich .jpg zur Verfügung.

Benutzung

Grundlegend kann die Anwendung auf jedem Gerät, das den Systemvoraussetzungen entspricht genutzt werden, sobald es Zugriff auf den Webserver hat.

Systemarchitektur

Verzeichnisstruktur

Grundlegend gibt es folgende relevanten Verzeichnisse im Projekt:

app/build:

beinhaltet den fertigen Build

app/public:

beinhaltet Dateien, die 1-zu-1 in den Build-Ordner kopiert werden

das sind Inhalte und Bilder

app/src:

beinhaltet jeglichen Quellcode der Seite

app/src/Components:

alle in der Anwendung verwendeten Komponenten

app/src/Redux:

das Speichermodell der Anwendung

app/src/sass:

die Styles der Anwendung

Hardware

Es wird keine spezielle Hardware verwendet.

Software (Aufbau und Funktionsweise)

Die Anwendung verwendet zum Rendern React.

Das Datenmodell wird dabei von Redux gehändelt.

Die Karte wird von Leaflet gerendert. Das Styling dieser kommt von Mapbox.

Das Styling erfolgt mittels SCSS. Dabei wird eine lose Mischung von BEM und Atomic Design verwendet.

Entwicklungshinweise

Für den Buildprozess werden lediglich Yarn und Sass benötigt. Diese können nach der Anleitung auf der jeweiligen Website oder mittels eines Packagemanagers wie brew installiert werden.

Bauen der Anwendung

Die weitere Installation erfolgt wie folgt:

```
git clone git@gitlab.mg.inf.tu-dresden.de:komplexpraktika/InfoVis/InfoVis_17-18_Arabismen.git
cd app
yarn
```

Der Entwicklungsserver wird ebenfalls mittels yarn gestartet:

```
yarn start
```

Ein Build kann mit folgendem Befehl erstellt werden:

```
yarn build
```

Optional kann die Installation auch mittels **npm** erfolgen.

Die entsprechenden Befehle sind **npm run start** für den Developmentserver und **npm run build** für das erstellen eines Builds.

Kompilieren der Styles

Das Kompilieren der Styles ist nicht in den oben genannten Schritten enthalten. Es wird jedoch das Package **node-sass** installiert, dessen Binaries zum Kompilieren der Styles genutzt werden können.

Zum Kompilieren der Styles mittels **sass** muss folgender Befehl im Ordner **app** ausgeführt werden:

```
sass src/sass/main.scss build/main.css
```

Für kontinuierliches Neukompilieren geänderter Stylesheets kann folgender Befehl verwendet werden:

```
sass --watch src/sass/main.scss:build/main.css
```

Abschließend kann der erzeugte Ordner app/build zur Installation der Anwendung auf einem WebServer verwendet werden.

Motivation

Die grundlegende Motivation dieser Anwendung ist es, die Bedeutung des Einflusses der arabischen Sprache und Kultur auf unsere europäischen Sprachen und Kulturen zu vermitteln, und ihren Präsenz im Alltag zu unterstreichen. Dabei fügt sie sich in den Rahmen der Kooperation des Komplexpraktikums mit der Austellung des Damaskuszimmers der SKD ein.

Aufgabenstellung

Das Projekt beschäftigt sich mit dem Damaskuszimmer im Japanischen Palais Dresden. Das Zimmer befindet sich dort losgelöst von seinem ursprünglichen Standort als Teil einer entstehenden Ausstellung. Die Restauratorin Anke Scharrahs bedauert dabei den Verlust des Kontextes, in dem sich dieses Zimmer eins befand. Das bezieht sich nicht nur auf den räumlichen Bezug, sondern auch auf die Geschichte, die Nutzung und die kulturelle Bedeutung des Zimmers. Das Projektes dient dementsprechend dazu dem Besucher der Ausstellung einen Sinn für diesen Kontext zu verschaffen.

Aufgabe des Projektes ist es jedem Besucher diesen Kontext in einer für ihn geeigneten Form leicht zugänglich zu machen. Dabei muss auf die verschiedenen Bedürfnisse der Benutzer, wie deren Zeit, Lernstil und Hingabe, eingegangen werden. Außerdem muss der museale Kontext Beachtung finden. Nutzer dürfen nicht durch die vornagegangene Nutzung der Anwendung durch andere Besucher beeinflusst werden.

Das vorliegende Projekt versucht diesen Kontext konkret durch das Aufzeigen des Einflusses der arbabischen Kultur auf die europäische Sprache herzustellen.

Zielstellung

Ziele

Die Ziele unsere Anwendung sind es, die Anwendung für möglichst alle Besucher der Ausstellung zugänglich zu machen, das heißt, sie sollte nicht zu komplex sein, leicht verständlich, aber auch tiefer führende Informationen für interessiertere Besucher bieten. Selbstverständlich soll sie für den Besucher relevante Informationen liefer, dabei interessant gestaltet sein und einen intuitiven Einblick verschaffen. Eine sehr wichtige Eigenschaft ist auch die Zustandslosigkeit. Sie ermöglicht es, das Besucher nicht in einem begonnen „Spiel" eines vorherigen Nutzers landen, und dann nicht wissen, was geschieht - die Anwendung soll also immer bereit für einen neuen Besucher sein.

Ergebnisse

Im aktuellen Stand existiert eine funktionale, aber inhaltlich nicht ausgeschmückte Anwendung. Das vorhandene Konzept ist nicht vollkommen auf diese Anwendung abgebildet, es fehlen noch eine funktionale Darstellung der geographischen Wanderung von Begriffen und auch das Styling ist noch nicht komplett abgeschlossen. Die Zustandslosigkeit ist jedoch gelungen. Außerdem sollte man beachten, dass die Anwendung sich kontextuell im Rahmen der Ausstellung eingliedert. Sie hat keinen direkten inhaltlichen Bezug auf das Damaskuszimmer, eignet sich aber trotzdem gut für den Einstieg in die Ausstellung, oder um von zu Hause aus nochmal über das Thema zu reflektieren.

Teilschritte

Am Anfang stand die Auseinandersetzung mit dem Damaskuszimmers, welches wir besuchten und uns von Anke Scharrahs, der leitenden Restauratorin präsentiert wurde. Dort sammelten wir Inspirationen, erhielten Einblicke in den Restaurationsprozess, die Geschichte des Zimmers und allgemein zur Kultur. Dabei zeigte uns Scharrahs eine Tafelm auf der 25 Begriffe waren, und fragte, vvwelche aus dem arabischen Raum stammten und ließ die Gruppe Schätzungen abgeben. Nur wenige kamen an den tatsächlichen hohen Wert heran. Später griffen wir dieses Thema wieder auf. Nach dem Besuch des Zimmers sammelten und bündelten wir unsere Eindrücke, und unterteilten sie in unterschiedliche Themenbereiche und bildeten Gruppen zu diesen. Dabei legten wir anfangs noch unseren Fokus auf die Kalligrafie im Zimmer und weniger auf Arabismen. Die Auswertung eines über die vergangengen Semesterferien geführter Fragebogen gab uns Einblicke in die Besucher und deren Erwartungshaltung und Anforderung gegenüber einer mulitmedialen Ausstellung. Um eine für den Nutzer nicht nachvollziehbare Unterteilung der Anwendung in zwei doch so unterschiedliche Themen zu vermeiden, legte sich der Fokus dann komplett auf die Arabismen. Es folgten diverse neue Fragestellungen: Wie stellt man die Trennung zwischen Orient und Okzident dar? Wie kann man die Häufigkeit & Bedeutung der Arabismen demonstrieren? Wie ihre Geschichte präsentieren? Wie überkommen wir das Problem der Zustandslosigkeit? Nach der Auseinandersetzung mit diesen Themen lag die Entwicklung eines Prototypen nahe. Dieser wurde mit React, einem Javascript Framework realisiert. Anhand des Prototypen wurden diverse Konzepte dann ausprobiert, verworfen oder verfeinert, wie z.B. das Teaser-Konzept: anfangs wenig Information oder Text, sondern Grafiken. Wecken diese das Interesse des Nutzers, kann dieser sich in den inhalten vertiefen, um mehr zu erfahren.

Um die Anwendung tatsächlich zu erleben, muss sie noch mit mehr echten Daten gefüllt werden, da sie im Moment nur zwei beispielhaft eingepflegte Begriffe enthält. Des weiteren muss die Ansicht des geographischen Wandels noch ausgearbeitet werden, für welchen es aktuell noch kein befriedigendes Konzept gibt. Außerdem sollte sie natürlich mit echten Nutzern getestet und evaluiert werden, um Probleme identifizieren und beheben zu können.

Einleitung

Anwendungsfall

Der Kontext der Anwendung ist der Kontext des Zimmers - sie liefert keine konkreten Informationen zum Ausstellungsstück, sondern zu dessen kulturellen Hintergrund. Sie eigent sich dabei gut als einleitendes Element, fördert das Verständnis und Bewusstsein zu der Thematik und lässt den Besucher auch noch von zu Hause oder unterwegs sich mit dem Thema auseinandersetzen.

Dabei wird der Einfluss des Orients auf den Okzident durch den kulturellen Austausch über Jahrhunderte auch in sprachlicher Form vermittelt, die sprachliche und inhaltliche Wandlung beschrieben und dargestellt.

Neben dem Hauptthema wurden auch Sprachgeschichte für die Inhalte und Webentwicklung für den Prototypen betrachtet.

Verwandte Arbeiten

Neben dem kleinen, aber sehr schön ausgearbeiteten Werk „Von Algebra bis Zucker" gab es keine ähnlichen Werke oder Projekte, die herangezogen wurden.

Hauptteil

Methodisches Vorgehen

Beim Entwurfsprozess wurde auf diverse Methoden zurückgegriffen. Am grundlegendsten war dabei das Affinitätsdiagramm, es half enorm dabei, den Umfang des Themas zu erkennen, Bereiche zu schaffen, und Inhalte zu ordnen und zu priorisieren. Für die Usability wurden Personas erstellt, sie vermittelten dabei, dass durchaus unterschiedliche Nutzer die Anwendung nutzen möchten, etwa weniger technisch affine, dafür aber interessierter an ausführlichen Inhalten, im Vergleich zu medial erfahrenen Nutzern, die aber eher hastig den Kern der Anwendung erfassen wollen und dann weiter ziehen. So konnte ein Kompromiss zwischen Komplexität und inhaltlichem Umfang ausgearbeitet werden. Szenarios halfen dabei, andere Probleme zu entdecken, hier konkret die Zustandslosigkeit: Nutzer sollen die Anwendung zu jedem Zeitpunkt „betreten" und „verlassen" können, ohne auf Artefakte von vorherigen Nutzern zu stoßen oder selber welche zu hinterlassen. Mit Skizzen, Screens und Mockups konnten schnell und einfach verschiedene Konzepte ausgearbeitet und verglichen werden, wie zum Beispiel die Karten- und die Spaltenansicht. Um zu erfassen, welche Informationen überhaupt dargestellt werden sollen, half die Recherche. Hier stellte sich die Differenzierung zwischen geographischem und inhaltlichem Wandel als wichtiger Faktor heraus. Leider gab es jedoch wenig Literatur, die hier zur Hilfe gezogen werden konnten.

Konzeption

Bei der Konzeption gab es folgende Themen: Das Interface generell. Hier wurde ohne viele andere Möglichkeiten zu betrachten eine einfache Unterteilung des Screens in Karte und Sidebar am rechten Bildrande ausgewählt, da dieses unkompliziert ist und eine gute Metapher von Quelle (die Arabismen befinden sich in der Sidebar) und Ziel (sie sollen auf die Karte gezogen werden) darstellt. Für die Trennung von Orient und Okzident bot sich eine Karte an - die wohl naheliegendste Methode, um eine geographische Einordnung zu ermöglichen. Um die Regionen zu unterscheiden, werden sie mit zwei unterschiedlichen Tönen eingefärbt. Eine umstrittene Frage war das Verdeutlichen der Gewichtung der Arabismen. Hierbei wurden zwei Konzepte betrachtet, die visuelle und die mentale Distinktion: bei der visuellen befinden sich viele Arabismen auf der Karte und der Nutzer erfasst dies auf Anhieb, während bei der mentalen nur wenige, insgesamt maximal fünf Begriffe auf der Karte liegen, und der Nutzer selber durch aktives Einordnen die Erkenntnis erlangen soll. Bei der Detailansicht für die inhaltliche Wandlung wurden die meisten Konzepte betrachtet. Mit dem Stichwort „Collage" wurden Ansichten wie eine Comicbook Seite oder eines Fotoalbums betrachtet, aber zu Gunsten von Spalten verworfen, da diese am besten dynamisch gestaltbar sind: Sollen mehr, detailliertere Inhalte angezeigt werden, kann die ausgewählte Spalte breiter werden und die anderen verdrängen.

Entwurf

Bild 01 Anwendung intialisiert

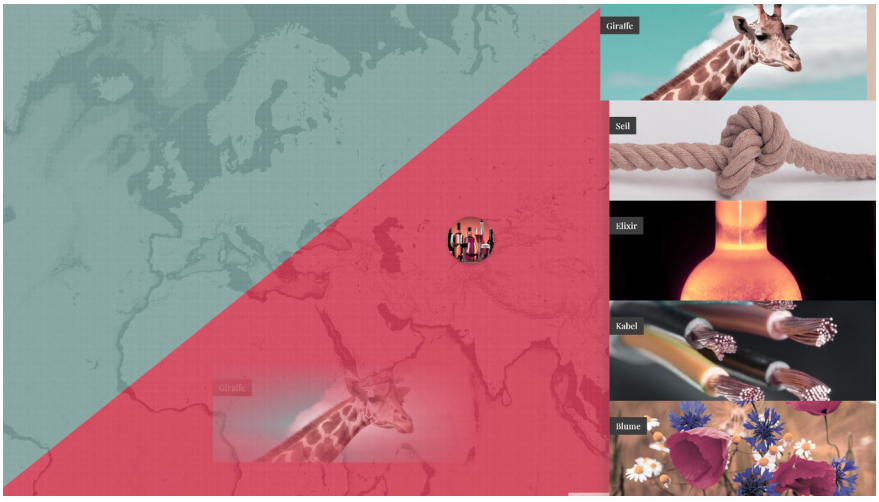

Bild 02 Ziehen eines Begriffes auf die Karte - Unterteilung der Regionen

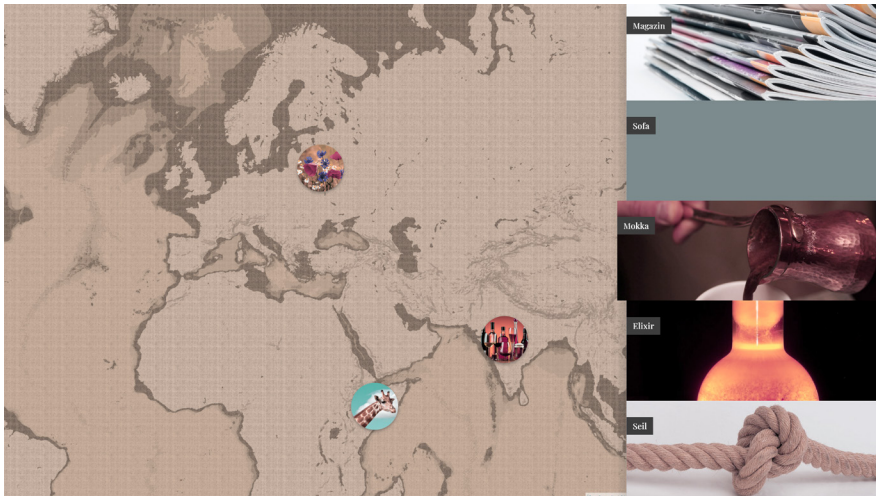

Bild 03 Begriffe auf der Karte

Anwendung intialisiertv

Bild 04 Detailansicht inhaltiche Wandlung

Bild 05 Detailansicht geographische Wandlung

Eine aktuelle Version des Prototypen kann unter leonardfollner.de/infovis gefunden werden.

Fazit

Abschließend lässt sich sagen, dass das Ergebnis der ursprünglichen Aufgabe gerecht wird. Zwar ist sie nicht vollständig und lässt Fragen, die sich während des Entwicklungsprozesses gestellt haben, noch unbeantwortet, aber die ausgänglichen Anforderungen werden erfüllt. Auch hat sich der Wandel von Kalligrafie und Arabismen zu nur Arabismen sich positiv bemerkmar gemacht. Er führte zu einer kompakteren und inhaltich konzentrierten Anwendung, welche nicht viel, aber das gut umsetzt.

Ausblick

Wie schon häufig erwähnt, ist die Anwendung nicht fertig - die größte Baustelle ist hier wohl die Detailansicht für den geographischen Wandel von Begriffen, hier muss noch ein gutes Konzept erarbeitet und implementiert werden. Zusätzlich stellt sich die Frage, ob die Anwendung in ihrer kontextuellen Niche bleiben soll, oder noch anders einen Bezug zum Damaskuszimmer hergestellt werden kann. Auch lässt sich noch diskutieren, ob man die Inhalte noch aus anderen Winkeln neben der geographischen und inhaltlichen Wandlung betrachten kann.