Geomatik und Planung

Masterarbeit

Der SCHWEIZER WELTATLAS interaktiv als Webapplikation

Ein Prototyp realisiert mit ArcGIS API for JavaScript

Abgabedatum
27. Januar 2014

Leitung
Prof. Dr. Lorenz Hurni

Autorin
Barbara Born
born.barbara@bluewin.ch

Betreuung
Thomas Koblet
Philipp Marty

IKG
Institute of Cartography and Geoinformation

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

I. Vorwort und Dankesworte

Die vorliegende Arbeit entstand im Rahmen einer Masterarbeit am *Institut für Kartografie und Geoinformation (IKG)* im *Departement Bau, Umwelt und Geomatik* der *Eidgenössischen Technischen Hochschule Zürich (ETH)*. Sie wurde im Herbstsemester 2013 im 9. Semester des Studiengangs Geomatik und Planung geschrieben. Die Masterarbeit steht unter der Leitung von Prof. Dr. Lorenz Hurni und wurde von Thomas Koblet vom *IKG* und Philipp Marty von *Esri Schweiz* betreut. Das Thema wurde zusammen mit den beiden Betreuern definiert.

Für die Zurverfügungstellung der benötigten Infrastruktur danke ich dem *IKG* unter der Leitung von Prof. Dr. Lorenz Hurni und Prof. Dr. Martin Raubal.

Ein grosser Dank geht an meine beiden Betreuer Thomas Koblet und Philipp Marty. Sie begleiteten mich von der Themenfindung bis zur Abgabe dieser Arbeit und unterstützten mich sowohl bei fachlichen, als auch bei technischen Fragen und Problemen.

Tabea Probst danke ich fürs Korrekturlesen und die konstruktiven Verbesserungsvorschläge. Auch allen weiteren hier nicht namentlich genannten Personen, die zum Gelingen dieser Arbeit beigetragen haben, möchte ich für ihre Unterstützung und Rückmeldungen danken.

Barbara Born

II. Zusammenfassung

Im Schulunterricht werden gedruckte Atlanten, Wandkarten und Globen zunehmend durch digitale, interaktive Medien ersetzt. Produkte wie interaktive Atlanten oder geografische Informationssysteme erleichtern es den Schülern, den Umgang mit Karten und räumlichen Daten mit Hilfe derjenigen Technologien zu lernen, mit denen sie aufgewachsen sind. Mehrere Herausgeber von gedruckten Atlanten haben den Schritt gewagt, eine Version ihres Produkts und ergänzende Materialien als Webanwendung anzubieten. Zum *SCHWEIZER WELTATLAS* (*SWA*) wurde als Ergänzung der *SCHWEIZER WELTATLAS interaktiv* (*SWAi*), eine kostenlose, interaktive Anwendung, herausgebracht. Der Aufruf dieser plattformunabhängigen Java Web Start Applikation ist an Schulen aufgrund teilweise fehlender Berechtigungen nicht immer möglich. Ausserdem kann der *SWAi* auf zahlreichen mobilen Geräten nicht gestartet werden, weil diese Java nicht unterstützen.

Damit der *SWAi* in Zukunft auch auf diesen Geräten verwendet werden kann, wird im Rahmen dieser Arbeit mit Hilfe der *ArcGIS API for JavaScript* und *ArcGIS for Server* ein Prototyp für den *SWAi* als Webapplikation entwickelt. Dieser ist ebenfalls plattformunabhängig, kann aber im Gegensatz zum aktuellen *SWAi* auf mobilen Geräten verwendet werden. Der Web-Atlas ist mit den üblichen Grundfunktionen ausgestattet. Er enthält zum Beispiel Pan- und Zoomfunktionen, eine Massstabs- und Koordinatenanzeige sowie eine Legende zu jeder verfügbaren Karte. Zusätzlich werden verschiedene Tools angeboten, mit denen der Benutzer die Karte gestalten und analysieren kann. Mit einem Zeichnungstool kann er Punkte, Linien, Flächen und Beschriftungen hinzufügen, bearbeiten und löschen. Am Beispiel von Brunnen wird aufgezeigt, wie der Benutzer Objekte erfassen, editieren und analysieren kann. Für die Anwendung der Webapplikation auf verschiedenen Geräten, deren Bildschirmgrösse stark variiert, ist es unerlässlich, dass sich das Design an die verfügbare Displaygrösse anpasst. Dafür wird das Konzept des *Responsive Web Designs* eingesetzt.

Mit einem Anwendungsfall wird der Prototyp auf mehreren Geräten mit unterschiedlichen Browsern und verschiedenen Bildschirmgrössen getestet. Die erstellte Webapplikation läuft auf allen Geräten und das Design passt sich den jeweiligen Bildschirmgrössen an. Es können jedoch nicht alle Funktionen auf allen Geräten und Browsern erfolgreich ausgeführt werden. Dies liegt hauptsächlich daran, dass die verschiedenen Browser identischen Code unterschiedlich interpretieren und nicht alle HTML-Spezifikationen unterstützen. Zusätzlich treten einige geräte- und steuerungsabhängige Probleme auf, die in einem weiteren Schritt genauer betrachtet und gegebenenfalls behoben werden müssen. Abschliessend kann gesagt werden, dass der *SWAi* als Webapplikation auf mobilen Geräten eingesetzt werden kann und sich mit der *ArcGIS API for JavaScript* in Kombination mit *ArcGIS for Server* interaktive, anspruchsvolle und vor allem vielseitige Karten-Anwendungen entwickeln lassen.

Schlüsselwörter: ArcGIS API for JavaScript, SCHWEIZER WELTATLAS, interaktiver Atlas, Schulatlas, Responsive Web Design, Webapplikation

III. Abstract

In school printed atlases, wall maps and globes are increasingly being replaced by digital, interactive media. Products such as interactive atlases or geographical information systems make it easier for students to learn how to use maps and how to deal with spatial data using the technologies they grew up with. Several publishers of printed atlases have taken the step toward offering a version of their product and supplementary materials as a web application. *SWISS WORLD ATLAS interactive* (*SWAi*), a free and interactive application, was released as a supplement for *SWISS WORLD ATLAS* (*SWA*). Using this platform-independent Java Web Start application is not always possible in schools due to partly insufficient privileges. In addition, *SWAi* cannot be started on numerous mobile devices, because they do not support Java.

In order to make *SWAi* available for these devices the prototype of a web application is developed. It uses *ArcGIS API for JavaScript* and *ArcGIS for Server*. In contrast to the current *SWAi* this platform independent prototype can be used on mobile devices. The web atlas contains the usual basic functions as pan and zoom, a measuring rule and a legend for each map. In addition, various tools are available that allow the user to create and analyze the map. With a drawing toolbox he can add, edit and delete points, lines, surfaces and labels. Using the example of wells it is shown that the user can capture, edit, and analyze objects. In order to use the web application on different devices, whose screen size varies greatly, the design has to adapt to the available screen size. Therefore the concept of *Responsive Web Design* is used.

The prototype is tested with a use case on several devices with different browsers and different screen sizes. The created web application runs on all devices and the design adapts to any screen sizes. However, not all functions can be executed successfully on all devices and browsers. This is mainly because the different browsers interpret the same code differently and do not support all HTML specifications. Additionally some device- and control-related problems occur, which must be considered in more detail in a further step. In conclusion it can be said that *SWAi* can be used as a web application on mobile devices. With the combination of *ArcGIS API for JavaScript* and *ArcGIS for Server* interactive, challenging and versatile map applications can be developed.

Keywords: ArcGIS API for JavaScript, SWISS WORLD ATLAS, Interactive Atlas, School Atlas, Responsive Web Design, Web Application

IV. Inhaltsverzeichnis

V. Abbildungs- und Tabellenverzeichnis

VI. Abkürzungen

Ajax	Asynchronous JavaScript and XML
API	Application Programming Interface
CSS	Cascading Style Sheets
CSV	Comma Separated Values
GIS	Geografisches Informationssystem
GPS	Global Positioning System
HTML	Hypertext Markup Language
PDA	Personal Digital Assistant
SWA	SCHWEIZER WELTATLAS
SWAi	SCHWEIZER WELTATLAS interaktiv
WebGL	Web Graphics Library
WFS	Web Feature Service
WMS	Web Map Service
XML	Extensible Markup Language

VII. Glossar

Ajax (Asynchronous JavaScript and XML)
«Der Begriff Ajax stellt ein Konzept für den Aufbau von interaktiven Webseiten dar. Ajax ermöglicht es, dass Server und Browser Daten austauschen, ohne dass die jeweilige Seite mit der Anfrage komplett neu geladen werden muss. [...] Der Benutzer kann also weitere Aktionen ausführen, während die HTTP (Hypertext Transfer Protocol)-Anfragen noch übertragen und bearbeitet werden.»[1]

API (Application Programming Interface)
«Ein API ist eine Schnittstelle zur Anwendungsprogrammierung, die von einem Betriebssystem oder einer anderen Software bereitgestellt wird. [...] Die zumeist Hardware-unabhängigen API ermöglichen es Entwicklern, Applikationen zu schreiben, die kompatibel mit der Software sind, von der sie zur Verfügung gestellt werden.»[2]

CSS (Cascading Style Sheets)
«CSS ist eine Formatierungssprache, die es Webautoren gestattet, Formatierungen von strukturierten Dokumenten durchzuführen. CSS wird hauptsächlich verwendet, um HTML- und XHTML-Dokumente für die Ausgabe in unterschiedlichen Medien visuell aufzubereiten, beispielsweise für Computerbildschirme, Drucker, akustische Geräte oder Handhelds.»[3]

DOM (Document Object Model)
Das DOM ist eine plattform- und spachunabhängige Schnittstelle. Sie erlaubt Programmen und Scripts dynamisch auf den Inhalt, die Struktur und die Darstellung von Dokumenten, zum Beispiel HTML, zuzugreifen und diese zu verändern.[4]

HTML (Hypertext Markup Language)
HTML ist eine plattformunabhängig Markup-Language, die das Erscheinungsbild eines Textdokuments definiert. HTML-Dokumente werden vom Web-Browser interpretiert und enthalten neben dem eigentlichen Inhalt auch Formatierungshinweise und sonstige Markierungen, wie zum Beispiel Verknüpfungen.[5]

JavaScript Framework
Ein JavaScript Framework bietet eine Bibliothek für die Erstellung von Benutzeroberflächen-Elementen. Daneben bietet das Framework Ajax-Unterstützung, hat Funktionen für Animationen und Effekte, erleichtert die DOM-Manipulation und hilft, mit den verschiedenen Eigenschaften der Browser umzugehen.[6]

[1] Computerworld, (2007), Lexikon: aktuelle Fachbegriffe aus Informatik und Telekommunikation, vdf Hochschulverlag AG, s. 12
[2] Computerworld, (2007), Lexikon: aktuelle Fachbegriffe aus Informatik und Telekommunikation, vdf Hochschulverlag AG, s. 14
[3] Jendryschik, M., (2009), Einführung in XHTML, CSS und Webdesign, Pearson Deutschland GMBH, s. 61
[4] Document Object Model (DOM), (2005), http://www.w3.org/DOM (Zugriff 05.01.2014)
[5] Kobert, T., (2012), Das Einsteigerseminar HTML, Hüthig Jehle Rehm, s. 20
[6] Preston, Scott., (2012), Learn HTML5 and JavaScript for iOS, Apress, s. 79

Markup Language
Eine Auszeichnungssprache (Markup Language) ist eine Computersprache, die mithilfe von Tags Elemente in einem Dokument definiert. Es ist eine für Menschen lesbare Sprache. Die zwei bekanntesten Markup Languages sind HTML und XML.[7]

Mashup
«Der Begriff Mashup bezeichnet Webseiten oder Webapplikationen, die Dienste oder Inhalte von verschiedenen Quellen zu einem neuen Service verschmelzen. Die Integration der Daten erfolgt typischerweise über offene Programmierschnittstellen [APIs, Anmerkung der Verfasserin], die von den Anbietern der ursprünglichen Information zur Verfügung gestellt werden.»[8]

Offenes Format
Dateiformate werden «als offen bezeichnet, wenn der Modus der Darstellung der Daten transparent und/oder seine Spezifikation öffentlich gemacht worden ist.»[9] Das Gegenteil von offenen Formaten sind geschlossene, beziehungsweise proprietäre Formate.

WebGL (Web Graphics Library)
WebGL ist eine plattformübergreifende API für die Erstellung von 3D Grafiken in einem Web-Browser. Sie basiert auf OpenGL ES 2.0. Die Grafiken werden auf einem Canvas-Element von HTML5 angezeigt.[10]

WebMapping
WebMapping ist der Prozess, in dem Karten im Internet designt, implementiert, generiert und veröffentlicht werden[11].

Widget
«Ein Widget ist ein Stück Code, welches zu jeder HTML Seite hinzugefügt werden kann und keinerlei weiterer Bearbeitung benötigt. Die Idee, die dahinter steckt, ist die der Code-Wiederverwendung. Also etwas Neues schaffen, mit etwas, was es schon gibt.»[12]

WMS (Web Map Service)
WMS (*Web Map Service*) ist ein OGC (Open Geospatial Consortium) Standard, um Karten über das Internet anzufordern oder anzubieten. Die Karten werden vom Dienst als Bilddatei zurückgegeben.[13]

WFS (Web Feature Service)
WFS (*Web Feature Service*) ist ein OGC Standard, der es ermöglicht, über das Internet Features zu lesen, zu bearbeiten, zu löschen, zu sperren oder abzufragen.[14]

[7] Markup Language, (2011), http://www.techterms.com/definition/markup_language (Zugriff 15.10.2013)
[8] Computerworld, (2007), Lexikon: aktuelle Fachbegriffe aus Informatik und Telekommunikation, vdf Hochschulverlag AG, s. 215
[9] Offene vs. proprietäre Formate, (2013), http://www.openformats.org/de1 (Zugriff 04.10.2013)
[10] WebGL- OpenGL ES 2.0 for the Web, (2013), http://www.khronos.org/webgl (Zugriff 10.10.2013)
[11] Web Mapping and Web Cartography, (2014), http://www.springerreference.com/docs/html/chapterdbid/63141.html (Zugriff 09. 01. 2014)
[12] HTML Bausteine und Widgets, (2011), http://iwebfaq.org/de/iWeb_HTML_Bausteine.html (Zugriff 06.01.2014)
[13] OGC – Web Map Service, (2014) http://www.opengeospatial.org/standards/wms (Zugriff 18. 01. 2014)
[14] OGC, (2005), Web Feature Service Implementation Specification

1. Einleitung

1.1. Der SCHWEIZER WELTATLAS interaktiv

Der *SCHWEIZER WELTATLAS (SWA)*[15] ist ein weitverbreiteter Schulatlas. Zum hundertjährigen Jubiläum wurde 2010 der gedruckte Atlas mit einer kostenlosen, interaktiven Anwendung, dem *SCHWEIZER WELTATLAS interaktiv (SWAi)*, ergänzt (Cron et al., 2011). In der interaktiven Version (Abbildung 1) können neben zweidimensionalen Karten Globen und Blockbilder dargestellt, Kartenprojektionen visualisiert und die Erdrevolution als Animation abgespielt werden. Der *SWAi* ist dank dieser und weiterer Funktionen ein ideales Lehrmittel für den Geografie-Unterricht an Schulen.

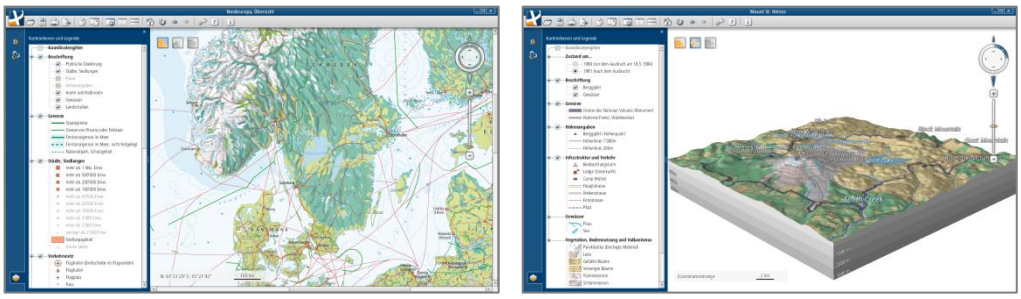


Abbildung 1: SWAi mit 2D Karte (links) und Blockbild (rechts)

Der aktuelle *SWAi* ist eine plattformunabhängige Java Web Start Applikation, die auf der Homepage des *SWA* heruntergeladen werden kann. Der *SWAi* läuft auf Windows, Mac und Linux Rechnern. Um eine Java Web Start Applikation auszuführen, muss Java Runtime Environment installiert sein. Der Aufruf als Java-Applikation an Schulen ist jedoch wegen den teilweise fehlenden Berechtigungen nicht immer möglich. Ausserdem kann der *SWAi* nicht auf Tablets oder Smartphones gestartet werden. Grund dafür ist die fehlende Möglichkeit, Java auf mobilen Geräten zu installierten. Da zahlreiche Hersteller die Java-Funktionalität nicht integrieren, kann der *SWAi* auf folgenden Geräten nicht gestartet werden (Java, 2012):

- Einige Personal Digital Assistants (PDAs) (Blackberry, Palm)
- Tablets (iPad, Android)
- Smartphones (iPhone, Android)
- Spielekonsolen (Nintendo Wii)
- MP3/MP4-Player (iPod)
- Weitere mobile Geräte

[15] SCHWEIZER WELTATLAS: www.schweizerweltatlas.ch

1.2. Ziele der Arbeit

Da eine zukünftige Version des *SWAi* auf Tablets und Smartphones laufen soll (Schweizer Weltatlas, 2014), ist das Hauptziel dieser Arbeit die Konzeptionierung und Implementierung des *SWAi* als Webapplikation. Ein Prototyp wird mit Hilfe der *ArcGIS API for JavaScript* erstellt. Der Atlas bleibt plattformunabhängig und kann auf mobilen Endgeräten[16] wie Tablets und Smartphones verwendet werden.

- Je nach Gerät hat man eine unterschiedliche Bildschirmgrösse. Der Atlas soll so programmiert sein, dass sich die Benutzeroberfläche automatisch an den zur Verfügung stehenden Bildschirm anpasst.
- Der Atlas soll mit folgenden Grundfunktionen ausgestattet werden:
  - Zoom
  - Pan
  - Quellen, Impressum
  - Drucken
  - Legende
  - Karte als Bilddatei exportieren
  - Anzeige von Massstab und Zentrumskoordinaten
  - Button, um Initialansicht zu laden
  - Ein- und Ausschalten von Ebenen
  - Popup für Info zu Feature-Layer
- Dem Benutzer sollen mehrere Tools zur Verfügung gestellt werden, mit denen er eine individuelle Karte gestalten kann. Das Erstellen und Bearbeiten von Punkten, Linien und Flächen soll, ebenso wie das Absetzen von Beschriftungen, unterstützt werden. Für die Ermöglichung von Gruppenarbeiten soll ein WFS-T (transaktionaler Web Feature Service) eingerichtet werden. Die Feature-Klasse «Brunnen» (Punktobjekt) dient der Gruppe für die Erfassung und das Editieren von Objekten als Grundlage.
- Der Anwender soll die erfassten Punktdaten geoverarbeiten können. Die Erstellung einer Dichtekarte soll ermöglicht werden. Eine andere Anwendung ist die Abfrage der Anzahl Punkte, die innerhalb eines vom Benutzer definierten Gebietes oder Radius liegen.
- Falls es die Zeit erlaubt, soll eine thematische Weltkarte erstellt werden. Dazu sollen die Länderpolygone nach einem bestimmten Aspekt eingefärbt werden. Die thematischen Daten sind als CSV (Comma Separated Values) gegeben. Es soll sich um eine Massstababhängige Karte handeln. Mit einer dynamischen Renderung soll die Zeitkomponente der Daten dargestellt werden können.

Die Schwerpunkte der Literaturrecherche werden auf folgende Punkte gelegt:

- Der *SWAi* wird nicht der erste Atlas sein, der den Weg in den Web-Browser schafft. Andere ähnliche Projekte sollen vorgestellt werden.
- Es ist eine Entwicklung von Desktopanwendungen zu Webapplikationen zu erkennen. Die Vor- und Nachteile, welche sich daraus ergeben, sollen aufgezeigt werden.

[16] Als mobile (End-)Geräte werden in dieser Arbeit elektronische Geräte bezeichnet, die sowohl einen Internet-Browser haben, als auch die Möglichkeit besitzen, Anwendungssoftware zu installieren. Dazu gehören Smartphones, iPods und Tablets. Notebooks werden nicht als mobile Geräte betrachtet.

- Die Stärken und Schwächen von HTML5 (Hypertext Markup Language) kombiniert mit CSS3 (Cascading Style Sheets) und JavaScript, nativen Applikationen und hybriden Ansätzen werden erläutert.
- Das Konzept von *Responsive Web Design*, beziehungsweise *Responsive Web Application*, soll vorgestellt werden.

1.3. Inhaltsübersicht

In Kapitel 1 wird auf die Ausgangslage und die Motivation für die Entwicklung eines Prototyps für den *SCHWEIZER WELTATLAS interaktiv* als Webapplikation eingegangen. Aufgrund der daraus gewonnenen Erkenntnisse werden die Ziele dieser Arbeit definiert. Danach werden in Kapitel 2 die Grundlagen erarbeitet. Abschnitt 2.1 beschäftigt sich mit mobilen Geräten im Unterricht, dem Stellenwert von GIS, *WebMapping* und interaktiven Karten in Geografie-Stunden. Andere Atlanten, die neben einer gedruckten Ausgabe bereits interaktive Karten und Anwendungen direkt im Browser anbieten, werden vorgestellt. Der nächste Abschnitt handelt von den für diese Arbeit relevanten theoretischen Grundlagen. Vor- und Nachteile von Webapplikationen werden genannt und mögliche Alternativen sowie das Konzept des *Responsive Web Designs* vorgestellt. Abschnitt 2.3 widmet sich der Technologie, die für die Umsetzung der Webapplikation eingesetzt wird. Unter anderem werden *ArcGIS for Server* und *ArcGIS API for JavaScript* vorgestellt.

Der Aufbau des Prototyps wird in Kapitel 3 beschrieben. Während sich Abschnitt 3.1 mit der Bereitstellung der Dienste für die Webapplikation beschäftigt, zeigt Abschnitt 3.2, wie der Atlas aufgebaut ist. Dazu gehört eine Beschreibung, wie das Konzept des *Responsive Web Designs* in dieser Webapplikation umgesetzt wird. In Abschnitt 3.3 wird auf die Funktionen und Karten des fertigen Prototyps eingegangen. Als Webapplikation sollte der Prototyp in jedem modernen Browser laufen. Der Test, um dies zu prüfen, wird in Abschnitt 3.4 vorgestellt. Die Ergebnisse dieser Arbeit und die Resultate des Tests werden in Kapitel 4 aufgeführt. Es folgen eine Diskussion der Ergebnisse und ein Ausblick.

2. Grundlagen

2.1. Digitale Produkte im Geografie Unterricht

Lange Zeit wurden an Schulen gedruckte Atlanten, Wandkarten und Globen für den Unterricht verwendet. In den letzten Jahren hat sich das geändert. Wie Abschnitt 2.1.1 zeigt, werden zunehmend digitale Produkte eingesetzt. Abschnitt 2.1.2 zeigt die Vorteile des Einsatzes solcher Produkte und die sich dadurch öffnenden Möglichkeiten auf. Es wird darauf eingegangen, wie der *SCHWEIZER WELTATLAS interaktiv* im Unterricht eingebaut wird. In Abschnitt 2.1.3 werden die Online-Portale von vier ausgewählten Atlanten vorgestellt.

2.1.1. Einsatz von Tablets im Schulunterricht

Digitale Medien bestimmen vermehrt den Schulunterricht. Computerräume, Laptops und in den letzten Jahren vor allem Tablets, allen voran das iPad, werden immer häufiger während des Schulunterrichts genutzt. Eine amerikanische Pilotstudie zeigt, dass der Einsatz von Tablets im Unterricht für Schüler mit Autismus eine grosse Bereicherung darstellt (Herbert, 2010). Auch in Regelklassen finden Tablets immer mehr Anklang. Das Projekt *my-Pad an Schulen*[17] versucht, dies zu fördern. Fischer (2014), die Projektleiterin von *my-Pad an Schulen* ist der Meinung, dass das Internet, vor allem *Social Web* Angebote, den Alltag von Schülern stark prägt. Schulen sollen Tablets nutzen, um diese einerseits im Unterricht einzusetzen und andererseits um den Schülern die Gefahren und Risiken des Internets aufzuzeigen. Während die Tablets in vielen Pilotversuchen von der Schule zur Verfügung gestellt werden, bringen die Schüler für die Studie *School-IT-Rhein-Waal* ihre eigenen Geräte mit (Heinen et al., 2013). Apple hat das Bedürfnis nach Tablets in Schulen erkannt und bietet mit der App *iBooks 2* zahlreiche Bücher, interaktive dreidimensionale Objekte, Diagramme, Videos und Fotos an. Apple verspricht, dass «Schüler und Studenten auf eine effizientere und effektivere Art und Weise lernen als jemals zuvor» (Apple, 2013). *Educational App Store*[18] ist ein Portal, auf dem für den Unterricht geeignete Apps für Android, Windows und iOS Geräte gefunden werden können.

Allerdings gibt es kritische Stimmen zum Einsatz von mobilen Geräten im Unterricht. So wird die Handschrift als Kulturgut in Gefahr gesehen und Hirnforscher warnen, dass getippte Begriffe weniger gut gespeichert werden können als geschriebene. Die Probleme beim Einsatz von Tablets sind häufig technischer Natur, wie zum Beispiel ein zu schwaches W-LAN. Auch das Ablenkungspotenzial, welches von einem solchen Gerät ausgeht, kann Lehrer dazu bewegen, im Unterricht auf Tablets zu verzichten. (Wege, 2013)

Der kritische Punkt für den erfolgreichen Einsatz von Tablets im Schulunterricht ist die Art und Weise, wie die Lehrer die mobilen Geräte einsetzen und welche Apps sie wählen. Rittmann (2014) will Lehrern dabei-behilflich sein und bietet seine Unterstützung bei der Einführung von iPads an Schulen an.

[17] my-Pad an Schulen: http://www.my-pad.ch
[18] Educational App Store: http://www.educationalappstore.com

2.1.2. WebMapping, GIS und interaktive Karten im Schulunterricht

WebMapping (→ Glossar), GIS (Geografisches Informationssystem) und interaktive Karten prägen vor allem den Geografie-Unterricht an Schulen und Gymnasien. Nach De Lange (2006) sind GIS sehr gut für den Schulunterricht geeignet, weil sie sehr vielseitig einsetzbar sind, die Schüler den kritischen Umgang mit Daten lernen, die Schüleraktivität gesteigert und das Verständnis von Karten gefördert wird. Er ist der Ansicht, dass GIS im Unterricht in Deutschland noch unzureichend eingesetzt werden. Als mögliche Gründe nennt er die fehlende konkrete Verankerung im Lehrplan, das Festhalten der Lehrer an bewährten Unterrichtsmethoden, zu komplexe oder zu kostenintensive Anwendungen. Hennig und Vogler (2011) sind der Ansicht, dass WebMapping stetig an Bedeutung gewinnt. Digitale Karten begleiten uns im alltäglichen Leben. Sie sehen WebMapping als «aussichtsreiche Brücke zwischen Aneignung und Vermittlung räumlicher Bedeutungen». So können für die unterschiedlichsten Einsatzbereiche eigene Daten und Information in einer WebMapping Anwendung integriert und mit anderen Benutzern geteilt werden. Da viele Schüler ein Smartphone besitzen und Kinder allgemein offener sind für neue Technologien, rät Nuñez (2013), Schulatlanten auf Smartphones anzubieten. Dafür soll nicht der bestehende gedruckte Atlas oder die PC-Version davon kopiert werden, sondern es sollen neue Lernmethoden angeboten werden, die bei Schülern das Interesse an Karten wecken.

Das Schweizer Projekt *Map Your World*[19] setzt an diesem Punkt an. Das Projekt wird für Schulklassen oder Jugendgruppen angeboten. Mit GPS (Global Positioning System)-fähigen PDAs nehmen die Jugendlichen draussen Daten auf und fügen sie in die Open GeoData Projekte ein. *GIS an Schweizer Mittelschulen*[20] versucht Lehrpersonen mit Anleitungen, Kursen, Lehrmitteln und Geodaten eine Unterstützung beim Einsatz von GIS zu bieten. Dieses Angebot richtet sich, wie der Namen schon sagt, an Mittelschulen und baut auf den Produkten von Esri auf. Ein anderes Projekt, welches aufzeigt, wie vielseitig GIS an Schulen eingesetzt werden kann, ist «Bioindikation im Lebensraum Bach und Fluss» von *GLOBE Schweiz*[21]. Beobachtungen zur Biologie, Ökologie und Chemie von Fliessgewässern können in einem WebGIS von Esri erfasst und analysiert werden. Auch andere Länder sind bestrebt, GIS im Unterricht zu verwenden. *PaikkaOppi*[22], ein WebGIS aus Finnland, überlässt es der Lehrperson, zu welchem Thema sie Daten erfassen und auswerten will. Die aufgenommen Daten können in ein digitales Arbeitsbuch exportiert und mit anderen Benutzern geteilt werden (Toivonen & Riihelä, 2011).

Der *SCHWEIZER WELTATLAS interaktiv* wird im Geografie-Unterricht vielseitig eingesetzt, wie Oleggini (2013) zeigt. Ob als interaktive Wandkarte, für die Vorbereitung der Lektionen oder in Kombination mit der gedruckten Version – der *SWAi* wird von zirka 80% der befragten Lehrpersonen der Sekundarstufe verwendet. Alle von Oleggini befragten Lehrpersonen setzen ein interaktives Produkt im Unterricht ein und alle, die angaben, dass sie den *SWAi* benutzen, sind mit dessen Qualität zufrieden.

2.1.3. Der Schulatlas als Webapplikation

In dieser Arbeit soll ein Prototyp für den *SCHWEIZER WELTATLAS interaktiv* als Webanwendung mit der *ArcGIS API for JavaScript* entwickelt werden. Am *IKG* (*Institut für Kartografie und Geo-*

[19] Map Your World: http://www.map-your-world.ch
[20] GIS an Schweizer Mittelschulen: http://schulgis.ch
[21] GLOBE Schweitz: http://www.globe-swiss.ch
[22] PaikkaOppi: http://www.paikkaoppi.fi

information) wurde bereits ein Prototyp[23] mit *OpenLayers* erstellt, welcher Zoom- und Pan-Funktionen unterstützt. Es werden gekachelte 2D-Karten angezeigt und Ebenen können ein- und ausgeschaltet werden. Andere Atlanten haben die Prototyp-Entwicklung abgeschlossen und bieten neben einer gebundenen Ausgabe bereits interaktive Karten im Browser an. Einige werden in diesem Abschnitt vorgestellt[24]. Nicht in diese Kategorie fallen Atlanten, die eine Software, zum Beispiel auf CD-ROM, anbieten, welche nicht im Browser ausgeführt wird. Somit wird auch der *SWAi* ausgeschlossen.

National Geographic Mapping

National Geographic bietet auf ihrer Education-Webseite[25] für Schüler, Lehrer, Familien und Kinder zahlreiche Tools und Karten an. Sämtliche Inhalte sind English. Auf der Seite von *National Geographic* findet man qualitativ hochwertige statische Karten (Abbildung 2). Diese Bilddateien sind unterteilt in politische, historische, thematische und physikalische Karten und können nach verschiedenen Kriterien gefiltert werden. Zu jeder Karte gibt es Informationen zum Zielgruppenalter und zum Thema sowie eine kurze Beschreibung.

Neben statischen Karten, die nur eine Zoom- und Pan- Funktion haben, gibt es einen interaktiven *MapMaker* (Abbildung 2), in welchem Kartenebenen ein- und ausgeschaltet werden können. Daneben besitzt der *MapMaker* ein einfaches Zeichnungstool und viele verschiedene Punktmarker. An diese Zeichnungselemente kann ein Titel, eine Beschreibung und ein Link zu einer Webseite oder ein Bild angehängt werden. Die Karte kann nach Fertigstellung als Bild exportiert, als Karte gedruckt oder als XML (Extensible Markup Language) gespeichert werden. Gespeicherte XMLs können im *MapMaker* wieder geöffnet und bearbeitet werden. Zudem kann die Hintergrundkarte gewechselt, Distanzen gemessen und der Link zu der Karte als E-Mail oder via Social Media geteilt werden. Um den *MapMaker* zu verwenden, muss der *Adobe Flash Player* installiert sein.

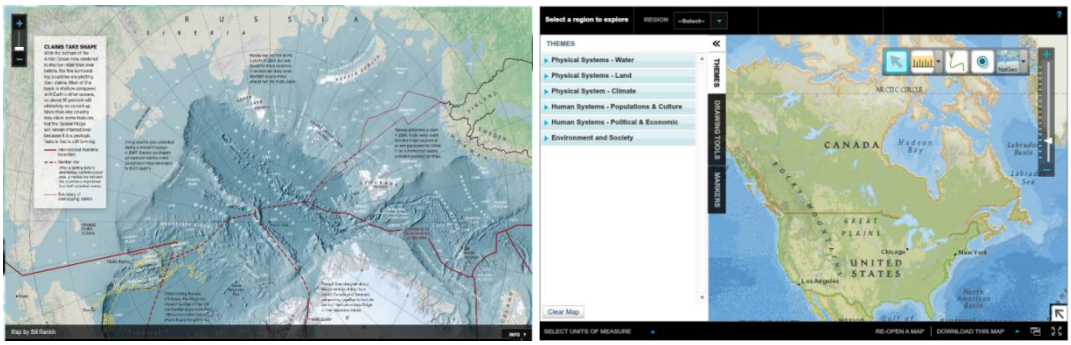


Abbildung 2: National Geographic: Statische Karte (links) und MapMaker (rechts)

National Geogephic Mapping wird durch mehrere interaktive Spiele und ausgearbeitete Lerneinheiten für den Unterricht ergänzt. Weiter kann an einem *FieldScope*-Projekt mitgearbeitet werden, bei dem die Bevölkerung zu unterschiedlichen Themen Daten erfassen kann. Ergänzt wird die Seite von einem Glossar, einer Enzyklopädie, diversen Artikeln und vielen

[23] Schweizer Weltatlas interaktiv- Javascript-Prototyp Version 0.1: http://karatlas.ethz.ch/javascript/swaijs-0.1/
[24] Keine abschliessende Auflistung
[25] National Geographic Mapping: http://education.nationalgeographic.com/education/mapping

weiteren Informationen mit Raumbezug. Ein Beispiel dafür sind *GeoStories*, in denen ein Ereignis oder eine Geschichte mit ihrem räumlichen Bezug dargestellt wird.

National Geographic stellt mehrere gebundene Atlanten her. Neben dem Klassiker *National Geographic Atlas of the World* gibt es den *National Geographic Student Atlas of the World* und Atlanten zu Themenbereichen wie Ozeane oder Imperien. *National Geographic Mapping* steht nicht in einem direktem Zusammenhang mit einem der Atlanten sondern ist ein eigenständiges Produkt.

Diercke

Diercke[26] hat in seiner 130-jährigen Geschichte viele Atlanten veröffentlicht und besitzt eine breite Produktpalette. Auf der Webseite von *Diercke*finden sich zahlreichte Kartenprodukte und atlasverwandte Themen. Einige Bereiche sind nur für registrierte Anwender zugänglich. Einzelne Karten aus den drei Atlanten *Diercke Weltatlas*, *Diercke Weltatlas 2* und *Diercke Drei* können angesehen werden. Dazu gibt es einen beschreibenden Text und je nach Karte zusätzliche Materialien und Grafiken.

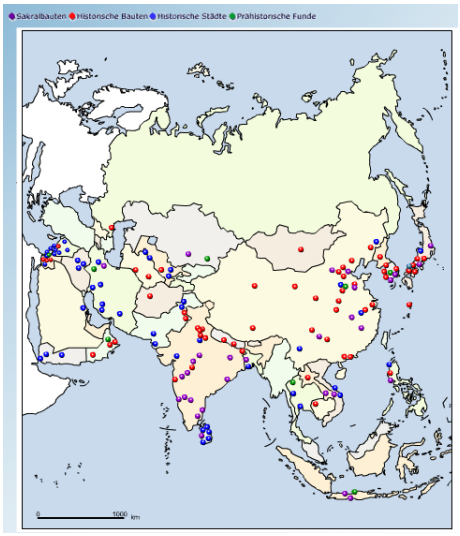


Abbildung 3: Diercke: Web-Karte

Im *Adobe Flash Player* werden Web-Karten angezeigt. Diese zeigen eine sehr einfach gestaltete Karte, auf der die Umrisse der Kontinente, beziehungsweise der administrativen Einheiten, zu sehen sind. Darauf verteilt sind Marker, die auf die jeweilige Homepage weiterleiten. Zoom und Pan werden nicht unterstützt. Manche Regionen können jedoch genauer betrachtet werden, indem man darauf klickt. Dadurch wird eine neue Karte geladen.

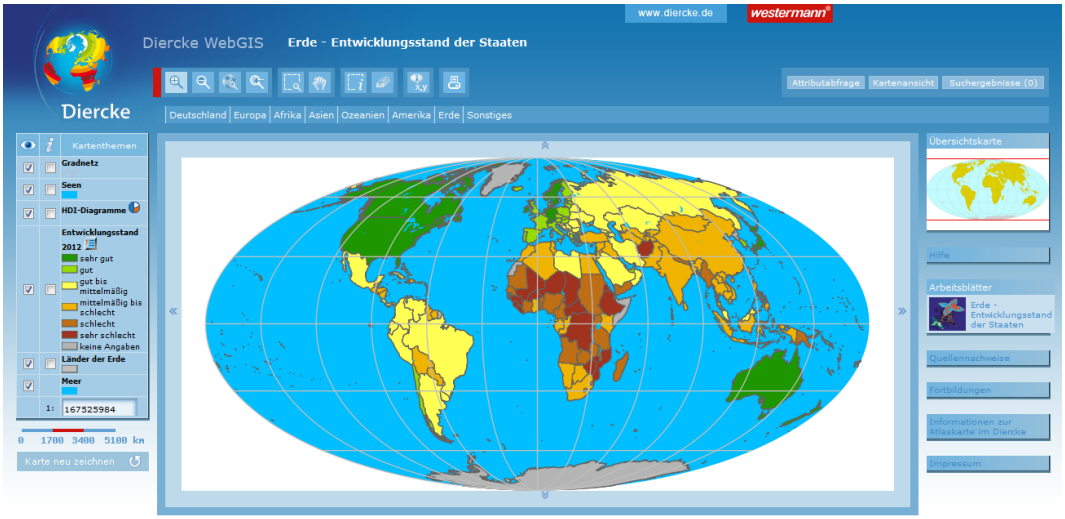


Abbildung 4: Diercke: WebGIS

[26] Diercke: http://www.diercke.de

Interaktiver und vielseitiger als die Web-Karte ist das *Diercke WebGIS*. Dieses WebGIS ist mit der PHP-Anwendung *WebGIS Studio* umgesetzt und bietet zahlreiche Projekte und Funktionen. Neben den Standardfunktionen eines GIS (Zoom, Pan, Objektabfragen, Messen und Drucken), können die Klassifikation von thematischen Ebenen angepasst, Diagramme erstellt und Abfragen definiert werden. Registrierte Benutzer können für jedes GIS-Projekt ein Arbeitsblatt herunterladen. Auch ein Link zum Kartenblatt im gedruckten Atlas ist vorhanden. Dieses umfangreiche GIS wird von einer kurzen und übersichtlichen Hilfe dokumentiert. *Diercke* bietet zudem Schulungen an.

öbv – freytag & berndt Schulatlas

Der *öbv – freytag & berndt Schulatlas*[27] ist ein österreichischer Schulatlas. Die Webseite des Atlas bietet eine umfangreiche Erweiterung zur gedruckten Ausführung. Die meisten Kartenblätter können gratis im pdf-Format heruntergeladen werden. Über eine einfache Benutzeroberfläche wird die Erstellung von «stummen Karten» ermöglicht. Der Benutzer kann angeben welche Elemente, zum Beispiel Flüsse, Staatsgrenzen oder Hauptstädte, auf der Karte gezeichnet werden sollen. Arbeits- und Informationsblätter helfen, den Unterricht vorzubereiten. Das Atlas-Lexikon führt die Symbole der Wirtschafts-, Klima- und Vegetationskarten etwas präziser aus, als dies in der Legende des gedruckten Atlas der Fall ist. Mehrere Karten sind als kmz-Datei erhältlich. Diese können zum Beispiel in *Google Earth* angeschaut werden.

Der ganze Atlas kann mit Hilfe des *Adobe Flash Players* durchgeblättert werden (Abbildung 5). Mit einem Menü kann man direkt zum gewünschten Abschnitt springen. Klickt man auf eine Karte im Atlas, werden rechts die verfügbaren Arbeitsmittel für dieses Thema angezeigt.

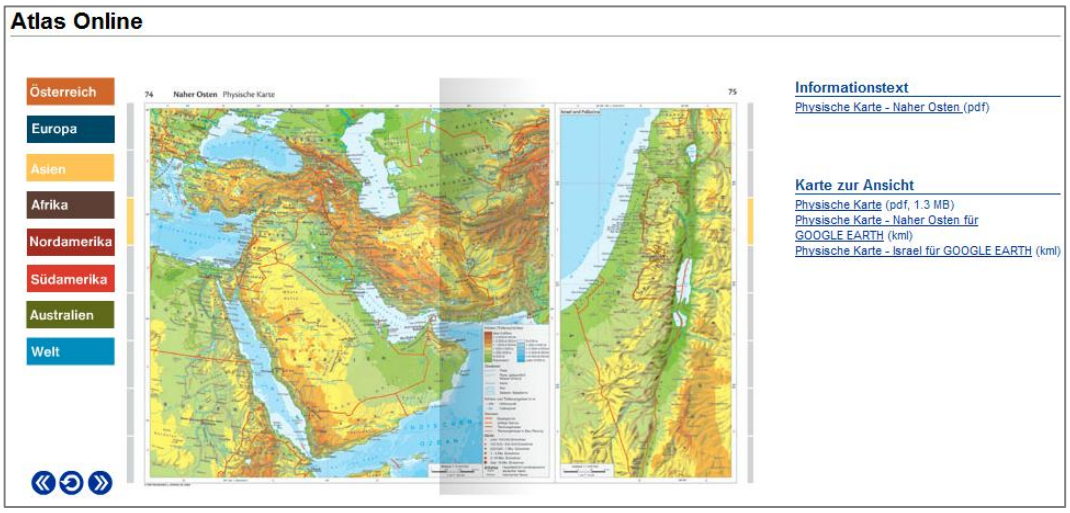


Abbildung 5: öbv – freytag & berndt Schulatlas: Atlas Online

Im interaktiven Atlas von *öbv – freytag & berndt Schulatlas* (Abbildung 6) gibt es für jeden Kontinent eine physische, eine politische und eine Wirtschaftskarte. Für Australien ist keine politische Karte vorhanden. Den Einstieg macht eine Weltkarte, in welcher die Kontinente farblich voneinander getrennt sind. Mit einem Klick auf den Kontinent kommt man zu den Karten. Die Karte kann man verschieben und vergrössern, beziehungsweise verkleinern. Rechts gibt es eine Legende und man hat die Möglichkeit, zwischen den drei zur Verfügung stehenden Karten

[27] öbv – freytag & berndt Schulatlas: http://www.freytagberndt.com/schulatlas

zu wechseln. Durch einen Link kommt man zurück zur Weltkarte. Ebenfalls über einen Link ist die Hilfe erreichbar. Der interaktive Atlas ist mit einem einfachen Zeichnungstool ausgestattet. Damit können Punkte, Polylinien und Polygone gezeichnet und beschriftet und gezeichnete Elemente verschoben oder gelöscht werden. Die Karte kann mit den Zeichnungen als PDF gedruckt werden. Der interaktive Atlas wurde mit *OpenLayers* programmiert.

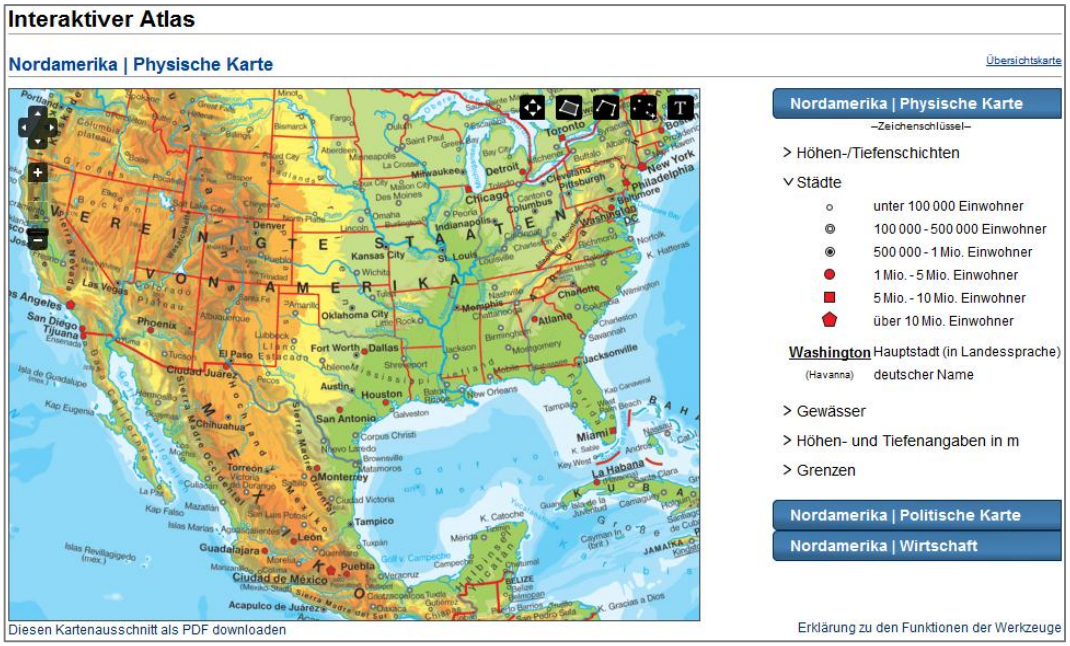


Abbildung 6: öbv – freytag & berndt Schulatlas: Interaktiver Atlas

Haack Weltatlas

Der *Haack Weltatlas*[28] wird vom *Klett Verlag* herausgegeben. Die gebundene Ausgabe wird vom *Haack Weltatlas Online*[29] ergänzt. Darin sind neben den Kartenblättern des Atlas, welche mit dem *Adobe Flash Player* angezeigt werden (Abbildung 7), auch Info- und Arbeitsblätter, Zusatzmaterialien und Links zu den einzelnen Kartenthemen verfügbar. Eine weitere Ergänzung sind die kmz-Dateien, welche mit *Google Earth* visualisiert werden können.

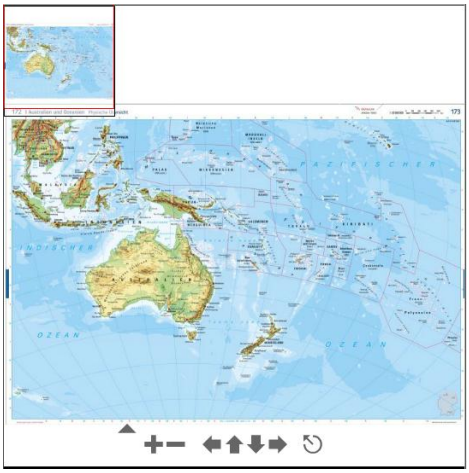


Abbildung 7: Haack Weltatlas: Kartenviewer

Das *Klett-GIS* (Abbildung 8), in welchem mehrere vordefinierte Projekte gestartet werden können, bietet zahlreiche Möglichkeiten. Neben den Grundfunktionen wie Pan, Zoom, Drucken, Messen, Attributabfrage, Legende, Referenzkarte und Ebenen ein- und ausschalten, können Kartenobjekte mit einer Suche, welche mehrere Bedingungen erfüllen muss, gefunden werden. Eine weitere Funktion des GIS ist die Erstellung von thematischen Karten und deren Reklassifizierung. Auch bestehende Daten können reklassifiziert werden (Ernst Klett Verlag, 2010). Das GIS ist mit JavaScript und *OpenLayers* umgesetzt.

[28] Haack Weltatlas: http://www.klett.de/produkt/isbn/978-3-623-49620-7
[29] Haack Weltatlas Online:
http://www2.klett.de/sixcms/list.php?page=lehrwerk_extra&extra=Haack+Weltatlas-Online

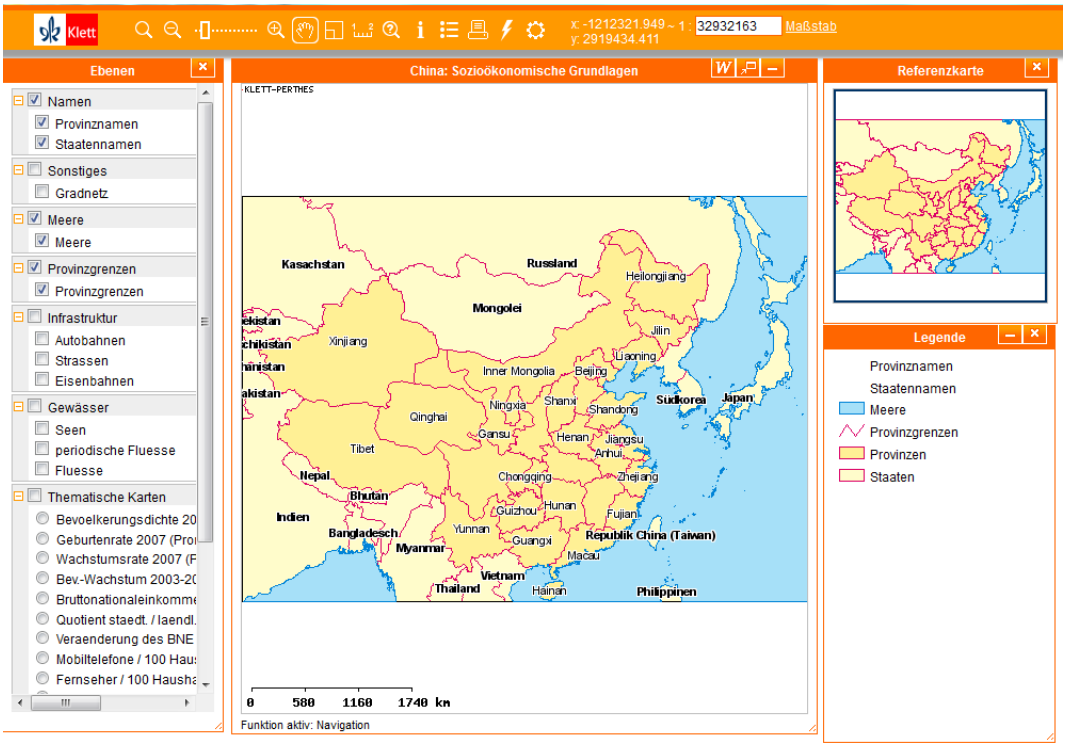


Abbildung 8: Haack Weltatlas: Klett-GIS

Vergleich der vorgestellten Produkte

Tabelle 1 zeigt den Vergleich der Funktionen und Eigenschaften der oben vorgestellten Online-Produkte. Es ist deutlich zu sehen, dass die Herausgeber entweder auf eine interaktive Karte oder auf ein GIS setzen. Bei *National Geographic* ist nicht klar, ob die angebotenen Karten aus einem der gedruckten Atlanten stammen.

Tabelle 1: Vergleich der vorgestellten Online-Produkte

	National Geographic Mapping	Diercke	öbv – freytag & berndt Schulatlas	Haack Weltatlas
Katen des gedruckten Atlas	(x)	x	x	x
• Arbeitsblätter zu den Karten		x	x	x
• Zusatzinformationen zu den Karten	(x)	x	x	x
KMZ-Dateien für Ansicht in Google Earth			x	x
Tool um eigene Karten (Stumme Karte) zu erstellen	x		x	
Interaktive Karte	x	x	x	
• Zoom und Pan	x		x	
• Zeichen	x		x	
• Legende	x		x	
• Drucken	x		x	

• Exportieren und neu laden	x			
• Teilen	x			
• Ebenen ein- und ausschalten	x			
• Messen	x			
• Klick-Events		x		
GIS		x		x
• Zoom und Pan		x		x
• Messen		x		x
• Attribut-Abfrage		x		x
• Legende		x		x
• Ebenen ein und ausblenden		x		x
• Drucken		x		x
• Referenzkarte		x		x
• Abfragen (Queries)		x		x
• Projekt speichern / laden				x
• (Re-)Klassifizieren		x		x
• Diagramme erstellen / ändern		x		
• Arbeitsblätter zu GIS Projekt		x		x

2.2. Theoretische Grundlagen

In Abschnitt 2.1 ist der Begriff «Webapplikation» mehrfach gefallen. Warum eine Applikation im Browser und nicht auf dem Desktop laufen soll (Abschnitt 2.2.1) und wie das bei mobilen Geräten aussieht (Abschnitt 2.2.2) zeigen die nächsten zwei Abschnitte. Abschnitt 2.2.3 wird aufzeigen, was unter *Responsive Web Design*, das Konzept, welches für den Prototyp eingesetzt wird, zu verstehen ist.

2.2.1. Von Desktopprogrammen zu Webapplikationen

Desktopanwendungen waren lange nach der Erfindung des Internets die einzigen Programme, die verwendet wurden. Zunehmend lassen sich Pendants zu diesen Programmen als Webanwendungen finden. *Google Drive*[30] oder *Office Web-Apps*[31] ersetzen *Microsoft Office*, *Pixlr*[32] ahmt *Adobe Photoshop* nach und um das 10-Finger System zu erlernen muss man sich keine CD mehr kaufen sondern kann es auf einer der zahlreichen Webseiten machen. Würden wir nicht zuerst den Browser öffnen und eine URL eingeben, würden wir wohl kaum ein Desktopprogramm von einer Webapplikationen unterscheiden können.

Seit der Erfindung des Internets vor beinahe 25 Jahren, hat sich das *World Wide Web* von einer statischen Ansammlung von Dokumenten hin zu einer dynamischen und vielseitigen Technologie entwickelt. Taivalsaari und Mikkonen (2011b) unterteilen die Entwicklung des Webs in drei Phasen. Die Phase bis zur Jahrtausendwende bezeichnen sie als *The Web as Document Environment*. Hyperlinks, Medien und später erste Plugins und dynamische Websites prägen die ersten Jahre. Im neuen Jahrtausend wurde das Web zunehmend als eine Applikationsumgebung gesehen (*The Web as an Application Environment*). Ajax (Asynchronous JavaScript

[30] Google Drive: https://drive.google.com
[31] Office Web Apps: http://office.microsoft.com/de-ch/web-apps
[32] Pixlr: http://pixlr.com

and XML) (→ Glossar) spielte eine Schlüsselrolle in der Entwicklung hin zu einer Applikationsumgebung (Zepeda & Chapa, 2007). HTML5 (→ Glossar) und WebGL (Web Graphics Library) (→ Glossar) verhalfen dem Web 2010 zum Durchbruch als *The Web as the Application Environment*.

Webapplikationen haben gegenüber Desktopanwendungen bedeutende Vorteile (Taivalsaari et al., 2011a). So werden das Wegfallen von Installationen und manuellen Upgrades als grosse Stärke von Web-Apps gesehen. Harjono et al. (2010) sehen das leichtere Erreichen neuer Anwender als direkte Folge daraus. Ausserdem erfolgt die Verteilung direkt über das Internet. Die Applikation kommt auf der ganzen Welt gleichzeitig auf den Markt. Als dritter Vorteil werden die offenen Applikationsformate (→ Glossar) erwähnt. Dank den für Menschen lesbaren Programmiersprachen können Mashups (→ Glossar) erstellt und Daten weltweit geteilt werden.

Harjono et al. (2010) nennen zwei zusätzliche Stärken von Webapplikationen. Auf der einen Seite ist die Applikation nicht mehr an ein spezifisches Gerät gebunden. Sie lebt im Web und kann darum von jedem Gerät mit einem modernen Web-Browser verwendet werden. Auf der anderen Seite ist die betriebssystemübergreifende Konsistenz der Applikation garantiert. Ein weiterer Vorteil ist die Zusammenarbeit, die durch das Internet ermöglicht wird (Anttonen et al., 2011). Menschen auf der ganzen Welt können gleichzeitig mit der gleichen Applikation an den gleichen Daten arbeiten und diese miteinander teilen.

Online Daten stellen gleichzeitig ein Sicherheitsrisiko dar. Sie gegen Verlust und Manipulation durch Unbefugte zu schützen ist eine grosse Herausforderung. Ausserdem können Webapplikationen, welche auf lokale Daten oder auf Hardware-Komponenten zugreifen, missbraucht werden (Wright, 2009). Lange galt die zwingende Internetverbindung als Nachteil von Webapplikationen (Harjono et al., 2010). HTML5 bringt zahlreiche neue Features mit sich, unter anderem *Local Storage* (w3schools, 2013b). Damit ist es möglich, Daten innerhalb des Browsers zu speichern und eine Applikation ohne Internetverbindung zu nutzen.

Wo genau der Unterschied zwischen einer Webseite und einer Webanwendung liegt, ist umstritten. Borodescu (2013) träg die Meinungen von fünf Experten zusammen und beschreibt die Unterschiede aus der Sicht von Entwicklern folgendermassen:

- Web-Apps sind eigenständig und in sich geschlossen
- Web-Apps haben eine umfassende Benutzeroberfläche, wie man sie von Desktopapplikationen kennt
- Web-Apps greifen über APIs (Application Programming Interface) (→Glossar) auf Gerätefunktionen zu
- Webseiten sind informationsorientiert, Web-Apps sind handlungsorientiert
- Web-Apps sind nicht von der Browsernavigation (Back, Forward, Reload,...) abhängig

Ob sich Webanwendungen gegenüber Desktopapplikationen durchsetzen können ist nicht klar. Zahlreiche Autoren ((Taivalsaari & Mikkonen, 2011b), (Wright, 2009), (Anttonen et al., 2011)) vermuten, dass Menschen hauptsächlich mit Webapplikationen arbeiten werden und somit das Web als Betriebssystem verwenden. Hintergrundprozesse und System-Software werden hingegen weiter als Binäre Software entwickelt. Ausserdem wurde laut Taivalsaari et al. (2011a) mit WebGL die „letzte sichere Bastion der konventionellen binären Softwares“ zerschlagen. Mit WebGL können nämlich rechenintensive Games als Web-Apps umgesetzt werden. Allerdings unterstützen erst 61% der Browser WebGL ganz oder zumindest teilweise (Can I use, 2014). Vor allem mobile Browser unterstützen diese Spezifikation noch nicht.

2.2.2. Mobile Apps

Nicht nur auf dem Desktop kommen Webapplikationen zum Einsatz sondern auch auf mobilen Endgeräten werden Web-Apps ausgeführt. Mobile Geräte erfreuen sich in der Schweiz einer grossen Beliebtheit. 59% der zwischen 15 und 74 Jährigen besitzen ein Smartphone, 27% haben ein Tablet und 21% verwenden beides (Beyeler, 2013). Abbildung 9 zeigt den globalen Marktanteil der bekanntesten Smartphone-Betriebssysteme. Die führenden Betriebssysteme sind Android (79.3%) und iOS (13.2%). Windows Phone, BlackBerry OS, Linux und Symbian teilen sich die restlichen 7.5% (Llamas et al., 2013).

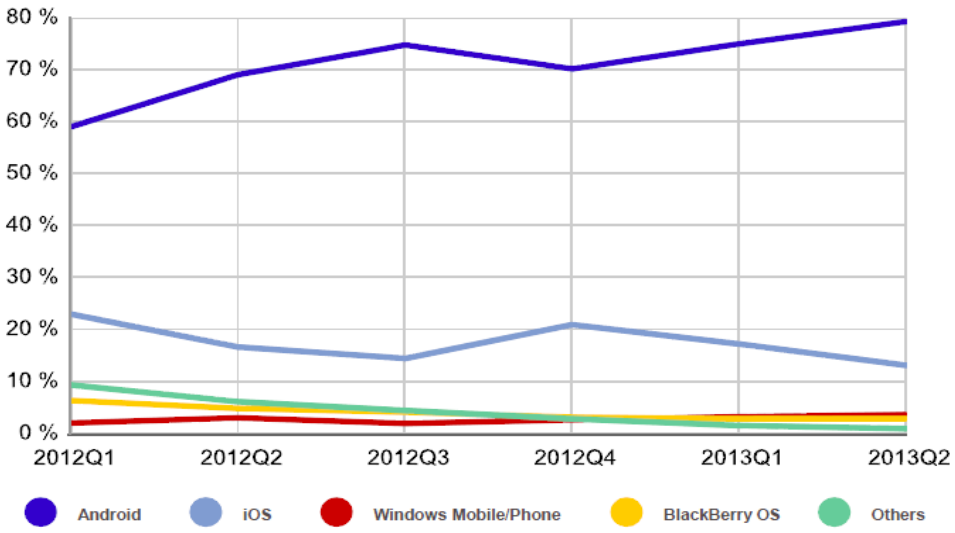


Abbildung 9: Weltmarktanteil der Smartphone-Betriebssystemen zwischen dem ersten Quartal 2012 und dem zweiten Quartal 2013 (Llamas et al., 2013)

Auch die anderen mobilen Geräte laufen mit verschiedenen Betriebssystemen. Dementsprechend schwierig ist es, eine Applikation zu entwickeln, die auf mehreren Plattformen läuft. Die fünf gängigen Ansätze, native Apps, Web-Apps und drei Zwischenlösungen, sind in Abbildung 10 nach Funktionsumfangs / Plattformnähe und Flexibilität / Adaptivität angeordnet. Sie werden im Folgenden kurz vorgestellt.

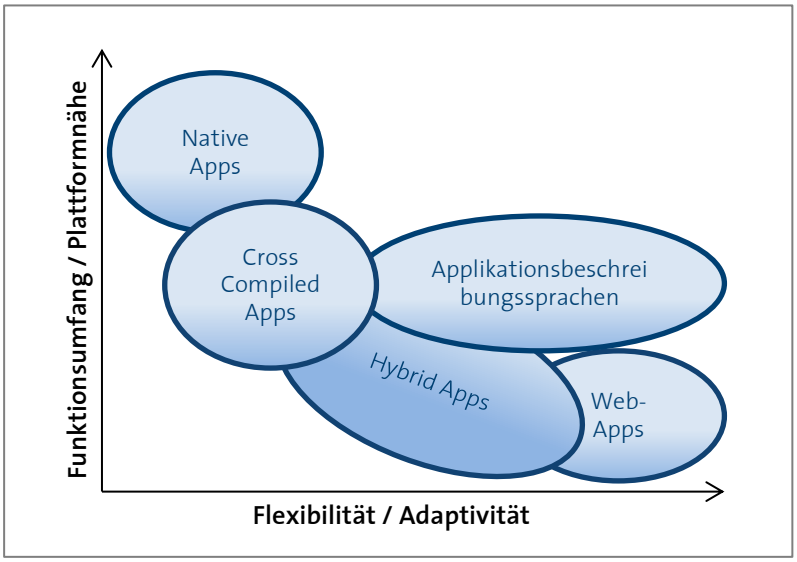


Abbildung 10: Übersicht mobiler Multiplattformumgebungen (nach Willnecker et al. (2012))

Native Apps

Jedes Betriebssystem hat seine eigene Programmiersprache und Entwicklungsumgebung. Der Entwickler einer nativen App muss die jeweilige Sprache beherrschen und mit der Umgebung vertraut sein. Native Apps sind auf ein Betriebssystem zugeschnitten. Darum muss die App für ein anderes Betriebssystem komplett neu programmiert werden. Dieser zusätzliche zeitliche und finanzielle Aufwand wird mit einem hohen Funktionsumfang belohnt und erfüllt höchste Ansprüche an Design und Leistung. Eine native App (Abbildung 11) kann mittels APIs und Bibliotheken auf die Hardware und Gerätesensoren, zum Beispiel Kamera oder Kompass, zugreifen (Willnecker et al., 2012). Die App wird über einen App-Store vertrieben. App-Store wird in dieser Arbeit als Oberbegriff für *Google Play*, *App Store* von iTunes oder *Windows Phone Store* verwendet. Über den Store kann eine allfällige Gebühr für die App verlangt werden. Der Kunde kann ausserdem nach der App suchen und der Store gibt eine gewisse Sicherheit, dass es sich bei der App nicht um eine Malware handelt. Jedoch verzögert sich der Launch der App etwas, da die App vom Store zugelassen werden muss (Mudge, 2012). Weil die App auf dem Gerät installiert ist und nur durch ein Update oder eine erneute Installation angepasst werden kann, kann es sein, dass mehrere Versionen der nativen App im Umlauf sind. Der Benutzer kann wählen, ob er auf die neuere Version umsteigen will oder nicht.

Abbildung 11: Aufbau einer nativen App (nach Jaser (2011))

Web-Apps

In Abschnitt 2.2.1 wurden die Möglichkeiten von Webapplikationen bereits vorgestellt. Mit HTML, CSS und JavaScript wird eine App geschrieben, die mit Hilfe eines beliebigen Browsers ausgeführt werden kann (Abbildung 12). Mobile Web-Apps unterscheiden sich durch das Layout und gewisse Funktionalitäten von Desktop-Web-Apps. So haben beispielsweise die meisten Laptops und Desktops noch keinen Touchscreen. Dementsprechend unterscheiden sich die Events, die ein Anwender auslösen kann. Web-Apps werden nicht auf dem Gerät installiert. Das bedeutet, dass die App jederzeit aktualisiert werden kann, ohne, dass der Benutzer etwas unternehmen muss. Allerdings wird dadurch die Vermarktung erschwert, da Web-Apps nicht in einen App-Store aufgenommen werden, was wiederum bedeutet, dass die App nicht so einfach gefunden werden kann (Demarmels et al., 2012). Einer der grössten Schwachpunkte von Webanwendungen ist der fehlende oder eingeschränkte Zugriff auf das Gerät und dessen Funktionen. Daran wird jedoch gearbeitet. Das *W3C*[33] (*World Wide Web Consortium*) publiziert auf seiner Webseite laufend den aktuellen Status dieser Entwicklungen und gibt an, welche Browser- beziehungsweise Plattformhersteller die neuen Technologien unterstützen. (Albert & Stiller, 2012)

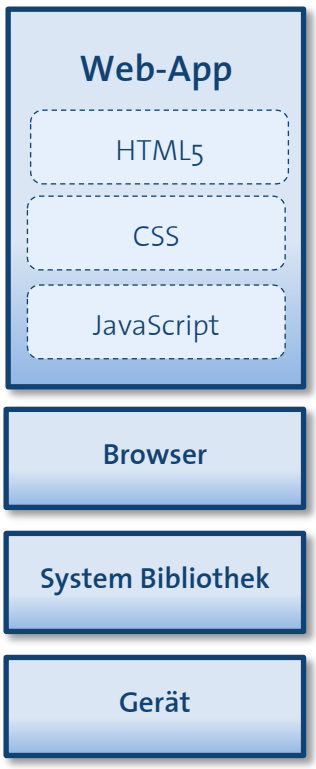


Abbildung 12: Aufbau einer Web-App (nach Jaser (2011))

[33] Standards for Web Applications on Mobile: http://www.w3.org/Mobile/mobile-web-app-state

Hybride Apps

Bei einer hybriden App (Abbildung 13) handelt es sich um eine hybride Lösung aus einer Web-App und einer nativen App (Willnecker et al., 2012). Der Anwender soll eine hybride App nicht von einer nativen App unterscheiden können. «Hybride Anwendungen bestehen aus einer mittels Webtechnologien (HTML, JavaScript und CSS) entwickelten Anwendungsfunktionalität, die innerhalb einer nativen Anwendung gekapselt und ausgeführt wird» (Gerlicher, 2012). Dies geschieht mithilfe von Frameworks wie *PhoneGap*, *Titanium*, *AppMobi* und *Phomobile*. Sie versuchen die Nachteile einer Web-App zu eliminieren, indem sie JavaScript-Bibliotheken zur Verfügung stellen, die den Zugriff auf native Funktionen und Sensoren erlauben (Christ, 2011). Aus einer Web-App kann für jedes vom Framework unterstützte Betriebssystem eine native App erzeugt werden, wobei für jedes Betriebssystem eine eigene hybride App generiert wird. Diese kann gleich wie eine native App im jeweiligen App-Store angeboten und vermarktet werden.

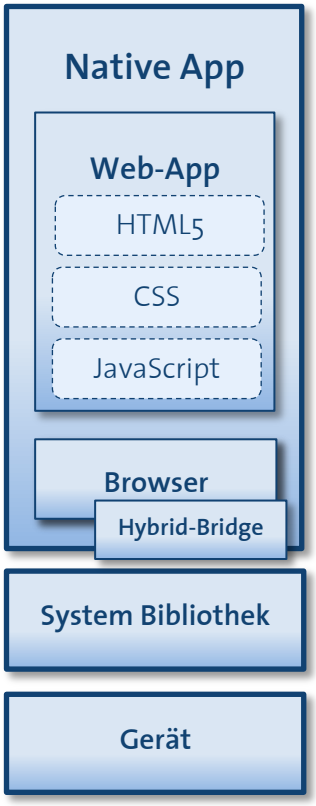


Abbildung 13: Aufbau einer hybriden App (nach Jaser (2011))

Die Web-App wird zusammen mit einer Browserkomponente in einer nativen App gekapselt (Abbildung 13). Mit Hilfe der «Hybrid Bridge», also einer JavaScript API, kann auf die Systembibliothek und damit auf das Gerät selber zugegriffen werden. Als Alternative zur Einbettung einer Browserkomponente, kann die grafische Benutzeroberfläche in JavaScript implementiert werden. Mit einem JavaScript-Interpreter werden aus diesem Code native GUI-Komponenten generiert (Gerlicher, 2012).

Cross Compiled Apps

Bei Cross Compiled Apps handelt es sich um eine Generalisierung von hybriden Apps. Gängige Cross-Compiler sind *MonoTouch*, *Titanium* und *Bedrock*. Statt dass die Anwendung als Web-App mit HTML, JavaScript und CSS geschrieben wird, wird sie zum Beispiel in C# (*MonoTouch*), JavaScript (*Titanium*) oder J2ME (Java Micro Edition) (*Bedrock*) programmiert. Mit dem Cross-Compiler wird aus diesem Code eine native App erzeugt (Willnecker et al. (2012), Demarmels et al. (2012)).

Applikationsbeschreibungssprachen

Applikationsbeschreibungssprachen nutzen anstelle von Webtechnologien eigens für die Smartphone-Welt konzipierte Sprachen. Mit Hilfe der Applikationsbeschreibungssprache wird die Benutzeroberfläche sehr genau aber abstrakt beschrieben. Die App wird nur einmal entwickelt, kann aber in mehrere Zielumgebungen übersetzt werden. Die App wirkt nativ, aber der Zugriff auf das Gerät wird, wie bei den hybriden Apps, von den APIs beschränkt.

Zukunft von Web-Apps

Über die Frage, welcher Ansatz sich durchsetzen wird, wird heftig diskutiert. Heise (2011) ist der Meinung, dass Web-Apps nativen Apps schon bald in nichts mehr nachstehen. Er rät deshalb, auf Web-Apps zu setzen. Sobald die Funktionen, die momentan nur über die nativen Betriebssystem-APIs zur Verfügung stehen, auch über den Web-Browser genutzt werden können, werden nach Gerlicher (2012) Web-Apps auch hybride Apps grösstenteils ersetzen. Auch Albert und Stiller (2012) sind der Ansicht, dass der Erfolg von Web-Apps hauptsächlich von den Zugriffsmöglichkeiten der Web-Apps auf die Gerätefunktionen und die Hardware abhängt. Mit diesen zusätzlichen Möglichkeiten muss das Thema Sicherheit neu betrachtet werden, denn bisher wurden Webapplikationen isoliert ausgeführt. Hinzu kommt die Frage nach der Vermarktung der Web-Apps. Längerfristig sehen Albert und Stiller (2012) die Chance von Web-Apps «vor allem bei geräteübergreifenden Anwendungen, deren Funktionalität überwiegend in der Cloud liegt». Etwas weniger optimistisch sehen Rivera und van der Meulen (2013) die Zukunft von Web-Apps. 2016 werden ihrer Meinung nach über 50% aller Apps hybride Apps sein.

	Geräte-zugriff	Geschwindigkeit	Entwicklungskosten	Zugang zu App Store	Genehmigungsprozess
Nativ	Vollständig	Sehr schnell	Teuer	Verfügbar	Zwingend
Hybrid	Vollständig	Fast native Leistung	Vernünftig	Verfügbar	Tiefer Mehraufwand
Web	Teilweise	Schnell	Vernünftig	Nicht verfügbar	None

Abbildung 14: Vergleich zwischen nativer, hybrider und Web-App (nach Kaminitz (2011))

Abbildung 14 zeigt, dass es im Moment keine Lösung für alles gibt. Je nach Anwendung ist ein anderer Ansatz zu wählen. Wenn es um das Nutzererlebnis und den Funktionsumfang geht, gewinnt die native App klar. Faktoren wie Zielpublikum, Funktionalität der App, Zeit und Budget, verfügbare IT-Ressourcen und die Frage, ob die App auf mehreren Plattformen laufen soll, beeinflussen letztendlich die Entscheidung. (Kaminitz, 2011)

2.2.3. Responsive Web Design

Resonsive Web Design (reagierendes Web Design) ist ein Konzept, welches auf Ethan Marcotte zurückzuführen ist. Marcotte (2011a) bezeichnet es als unsinnig, eine mobile Webseite, zusätzlich zu einer herkömmlichen Webpage, zu designen. Dies sei keine Lösung, da immer mehr Geräte mit unterschiedlichen Bildschirmgrössen und Auflösungen hinzukämen. Stattdessen soll sich das Design selber an das Gerät, auf welchem es ausgegeben wird, anpassen (Abbildung 15).

Abbildung 15: Ein Responsive Web Design passt sich der Bildschirmgrösse an

Entwicklung

```
@media print {
  body { font-size: 10pt }
}
@media screen {
  body { font-size: 13px }
}
@media screen, print {
  body { line-height: 1.2 }
}
```

Abbildung 16: Beispiel für Media Types in CSS (W3C, 2014)

Adams (2006) gestaltete bereits 2004 eine Webseite[34], die sich an die Browserbreite anpasste. Er nannte den Ansatz *Resolution Dependent Layout* und benutzte dazu JavaScript, um je nach Breite das Stylesheet auszutauschen. Mit CSS2 wurden die *Media Types* (Abbildung 16) eingeführt. Im Stylesheet können damit unterschiedliche Formatierungsregeln für die verschiedenen Medientypen festgelegt werden. Als Medientyp gelten beispielsweise *print*, *screen*, *handheld* oder *tv* (W3C, 2014).

Der Gerätetyp sagt aber noch nichts über die Bildschirmgrösse aus. Mit CSS3 wurden nach demselben Prinzip wie die *Media Types* die *Media Queries* eingeführt. Damit können Eigenschaften wie Bildschirmgrösse, Farbtiefe oder die Orientierung eines Geräts abgefragt werden (W3C, 2012). Die *Media Queries* und *Types* können entweder, wie in Abbildung 16, direkt im Stylesheet oder schon beim Aufruf des CSS-Links im HTML Dokument eingefügt werden, wie das in Abbildung 17 gemacht wird.

```
<link rel="stylesheet" type="text/css"
  media="screen and (max-device-width: 480px)"
  href="shetland.css" />
```

Abbildung 17: Beispiel für einen CSS-Link mit Media Query und Media Type (Marcotte, 2011a)

Konzept

Um eine responsive Webseite zu gestalten braucht es laut Marcotte (2011a) neben *Media Queries* auch *Fluid Grids* und *Flexible Images*, beziehungsweise flexible Medien. Das *Grid* beschreibt die grobe Unterteilung einer Webseite in verschiedene Bereiche wie Header, *Navigation*, *News* und *Content*. Marcotte (2009) hält fest, dass man dabei statt mit statischen Werten in der Einheit Pixel mit relativen Werten arbeiten sollte. Ein *Fluid Grid* mit relativen Grössen wird immer die ganze Seite ausfüllen. Auf horizontale Scrollbars kann dadurch verzichtet werden und weisse Ränder ohne Inhalt werden verhindert.

Ein Bild auf einer Webseite bezeichnet Marcotte dann als fexibel, wenn es seine Grösse zusammen mit dem *Grid* verändert. Ein Rasterbild hat eine statische Grösse in Pixel. Dies widerspricht dem Konzept der *Fluid Grids*. Indem man dem Bild die Style-Eigenschaft «max-width: 100%» gibt, stellt man sicher, dass das Bild nie über das ihm zugeteilte *Grid* hinausragen wird (Marcotte, 2011b).

Kritik und Alternativen

Während in unzähligen Büchern, Blogs, Magazinen und Webseiten *Responsive Web Design* als Lösung für die Entwicklung von Webseiten für alle Geräte vorgestellt wird, gibt es auch kritische Stimmen dazu. Chan (2013) ist der Meinung, dass *Responsive Web Design* nicht die ideale, aber im Moment eine gute Lösung ist. Er kritisiert unter anderem die Performance, da auf einem mobilen Gerät die gleiche Datenmenge geladen wird, wie auf dem Desktop. Zwar wäre es möglich dies zu ändern aber das führt zu einer höheren Komplexität, was wiederum einen

[34] The Man in Blue: http://www.themaninblue.com/experiment/ResolutionLayout

höheren zeitlichen und finanziellen Aufwand bedeutet. Und letztendlich verhindere dieses Konzept neue Innovationen, da gute Ideen neben *Responsive Web Design* zu wenig Beachtung fänden.

Eine Alternative zum *Responsive Web Design* ist das *Adaptive Web Design*. Dabei wird ein Web Design auf verschiedene Bildschirmgrössen optimiert. Statt mit *Media Queries* wird mit server- oder clientseitigem Code das Gerät erkannt. Anschliessend bekommt der Client nur die für ihn relevanten Daten. Die dadurch verringerte Datenmenge führt zu einer kürzeren Ladezeit der Webseite. Um das zu ermöglichen, wird für jedes Gerät eine eigene URL mit HTML, CSS und JS (JavaScript) Dateien benötigt. Die Medien der Webseite werden für die Geräte unterschiedlich verkleinert, um die Datenmenge zu minimieren.

Responsive Web App Design

Die meisten Artikel zu *Responsive Web Design* beschäftigen sich mit Webseiten, nicht aber mit Webapplikationen. Anglin (2012) behauptet sogar, dass dieses Konzept für Web-Apps nicht nützlich sein. Grigsby (2013) und Bricklin (2013) sind anderer Meinung und haben erfolgreich eine responsive Business Web-App entwickelt. Bricklin findet, dass es für *Responsive Web Apps* nicht reicht nur die Bildschirmgrösse zu berücksichtigen, wie das häufig für Webseiten gemacht wird, denn je nach Gerät und Einsatz der App gelten andere Kriterien für die Benutzeroberfläche. Da es jedoch noch keine Standards und nur wenige bekannte Beispiele gibt, hat er eine Webseite[35] aufgebaut, um die Diskussion über *Responsive Web Apps* anzuregen.

2.3. Technologie

Für den Prototyp wird auf ein Client-Server Modell gesetzt. Dabei sendet der Client eine Anfrage an den Server, dieser verarbeitet diese und sendet die Antwort zurück an den Client. In Abschnitt 2.3.1 wird der Server, wie er für diese Arbeit eingesetzt wird, genauer beschrieben und in Abschnitt 2.3.2 werden die clientseitigen Technologien vorgestellt.

2.3.1. Serverseitig: ArcGIS Server und PostgreSQL Geodatenbank

Serverseitig werden in dieser Arbeit ein GIS-Server und eine Geodatenbank benötigt. Als GIS-Server wird der Server vom Softwarepaket *ArcGIS for Server Advanced* von Esri eingesetzt. Darauf können verschiedene Dienste angeboten werden (Law, 2013). Die Dienste können mit *ArcMap* von Esri publiziert und verwaltet werden. Die in dieser Arbeit verwendeten Dienste sind:

- Map Service
- Feature Service
- Geoprocessing Service
- Geometry Service

PostgreSQL ist ein objekt-relationales Open-Source Datenbankmanagementsystem (DBMS). Die ACID-Eigenschaften (Atomicity, Consistency, Isolation, Durability) garantieren die Zuverlässig-

[35] Responsive App Design: http://www.responsiveappdesign.org

keit, Integrität und Korrektheit der Daten. *PostgreSQL* unterstützt die meisten SQL-Datentypen und kann mit weiteren Datentypen ergänzt werden. Dank *PostGIS* können Geodaten in eine *PostgreSQL* Datenbank gespeichert und räumliche Funktionen ausgeführt werden (PostgreSQL, 2014). Mit dem Tool „Create Enterprise Geodatabase" aus der Datenmanagement Toolbox kann in *ArcMap* eine *PostgreSQL Enterprise Geodatabase* erstellt werden (Esri, 2013).

2.3.2. Clientseitig: Browser, HTML, CSS und JavaScript

Es wird vorausgesetzt, dass auf dem Client ein moderner Browser läuft. Die folgenden beiden Abbildungen zeigen die beliebtesten Desktop-Browser (Abbildung 18) und mobilen Browser (Abbildung 19) der Schweizer. Darauf ist zu sehen, dass Chrome auf Kosten vom Internet Explorer (IE) und von Firefox immer häufiger verwendet wird. Auf mobilen Geräten nutzen die meisten Schweizer Safari, welcher der Standardbrowser von allen Apple-Geräten ist, und den Browser von Android. Global wird Internet Explorer nur noch von 10.5% der Internetbenutzer als Desktop-Browser benutzt (Stand November 2013). Firefox kommt auf 26.8% und Chrome dominiert mit 54.8% (w3schools, 2013a).

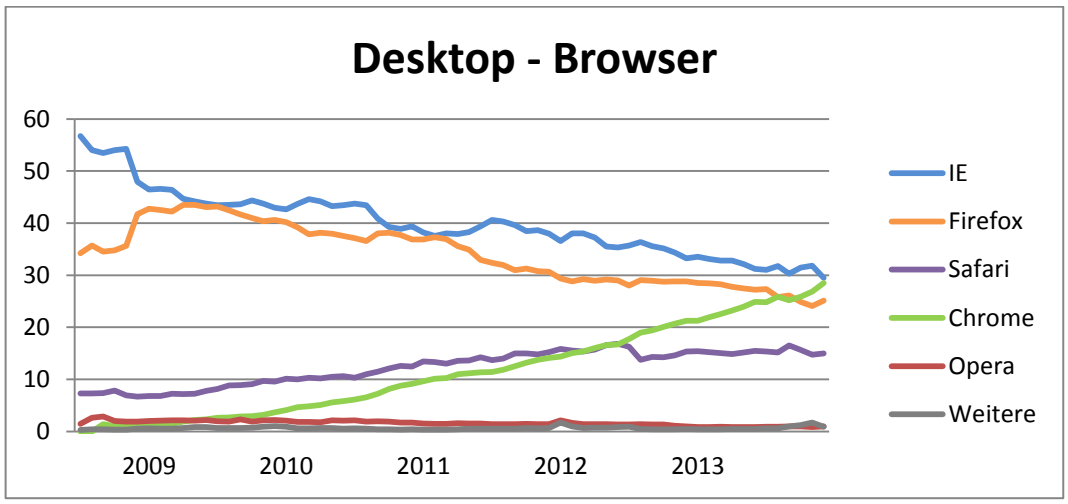


Abbildung 18: Beliebteste Desktop-Browser in der Schweiz zwischen Juli 2008 und Dezember 2013 (StatCounter Global Stats, 2013)

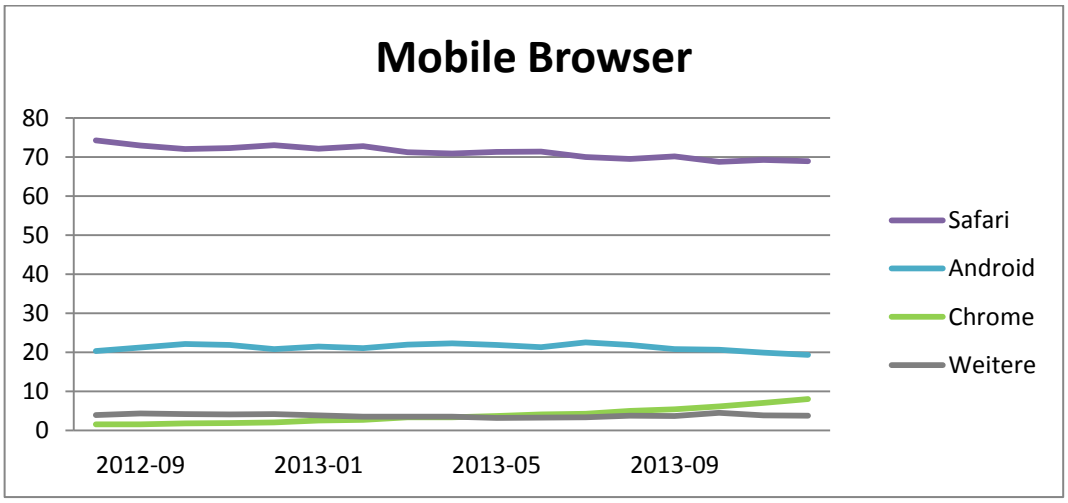


Abbildung 19: Beliebteste mobile Browser (Tablet und Mobiltelefon) in der Schweiz zwischen August 2012 und Dezember 2013 (StatCounter Global Stats, 2013)

Clientseitig wird eine Web-App (siehe 2.2.1 und 2.2.2) erstellt. Neben HTML und CSS wird JavaScript eingesetzt. Durch JavaScript wird eine Interaktion zwischen der Webseite und dem Benutzer ermöglicht. JavaScript ist eine dynamische Skriptsprache, die direkt im Browser ausgeführt werden kann. Objekte werden, basierend auf Prototypen, während der Laufzeit erstellt (Mozilla Developer Network, 2013). *JavaScript Engines*, Programme, welche den JavaScript Code interpretieren und ausführen, waren lange sehr langsam. 2008 begann das «Race for Speed» der Browser. Sie setzten sich zum Ziel, JavaScript Programme gleich schnell wie C-Programme auszuführen (Dalziel, 2013). Die Zukunft von JavaScript sieht vielversprechend aus.

Die *ArcGIS API for JavaScript* ist eine von drei Web-APIs von Esri. Die API (→ Glossar) hilft bei der Erstellung von Karten und Kartenfunktionen in Webanwendungen. Für diese Arbeit wird die Version 3.7 eingesetzt. Alternativen zur *ArcGIS API* wären *Leaflet API*, *OpenLayers*, *Open JavaScript Maps API* (MapQuest), *Google Maps API*, *Bing Maps API* oder *MapBox JavaScript API*.

Ein Vergleich zwischen der *Google Maps API*, *ArcGIS API for JavaScript* und *OpenLayers* hat ergeben, dass die API von Google, gemessen an der Anzahl Objekte, Methoden und Objekteigenschaften, die kleinste ist. Dadurch ist sie übersichtlich und es muss nicht lange nach dem passenden Objekt oder der richtigen Funktion gesucht werden. Ausserdem wurden bei der *Google Maps API* über einen Zeitraum von einem halben Jahr keine Objekte entfernt, was sie zu einer stabilen API macht. Die Entwickler müssen so den Code bei einer neuen Version nicht anpassen. Der grosse Nachteil der *Google Maps API* ist die fehlende Unterstützung bei der Verwaltung von Kartenebenen. *OpenLayers* hingegen hat doppelt so viele Objekte wie die API von Google. Neben dem Zeitverlust für die Suche nach den richtigen Objekten für eine Aufgabe bemängeln die Autoren des Vergleichs vor allem die fehlende Rückwärts-Kompatibilität, die durch das Streichen von Objekten in mehreren Versionen zustande kommt. Auch bei der *ArcGIS API for JavaScript* wurden Objekte aus der API genommen. Sie ist allerdings wesentlich kleiner als *OpenLayers* und dadurch übersichtlicher. (Fernandes et al., 2013)

Dojo ist ein Open-Source JavaScript Framework (→Glossar), für die Generierung von dynamische Benutzeroberflächen für Webseiten (DiWT, 2011). So können beispielsweise Buttons, Trees, Dialoge oder Slider erstellt und angepasst werden. Neben *Dojo* wird *jQuery* eingesetzt. *jQuery* wird hauptsächlich für die DOM (→Glossar) Manipulation verwendet. Statt *Dojo* und *jQuery* könnten *Sencha Ext JS*, *Prototype* oder *MooTools* benutzt werden. Die Wahl fällt auf diese beiden Frameworks, weil *ArcGIS API for JavaScript* selber auf *Dojo* aufbaut. *jQuery* wird verwendet, weil es in der Vorlage, auf welcher der Prototyp aufgebaut ist (Abschnitt 3.2.2), eingesetzt wird.

3. Methode und Vorgehen

Der Prototyp für den *SCHWEIZER WELTATLAS interaktiv* als Webapplikation wird mit der *ArcGIS API for JavaScript* entwickelt. Damit die Benutzeroberfläche der Applikation für alle Bildschirmgrössen dienlich ist, wird das Konzept des *Responsive Web Designs* eingesetzt. Die folgenden Abschnitte beschreiben, wie die Dienste, die für die Applikation benötigt werden, eingerichtet werden (Abschnitt 3.1), wie die Architektur des Atlas aussieht (Abschnitt 3.2.1) und wie das Konzept des *Responsive Web Designs* auf den Prototyp angewendet wird (Abschnitt 3.2.2). Im Abschnitt 3.3 werden die einzelnen Karten und Funktionen des Prototyps vorgestellt. Dabei wird auf Schwierigkeiten oder Probleme eingegangen. Als Webapplikation sollte der Prototyp auf allen Geräten mit einem modernen Browser laufen. Um dies zu prüfen, wird ein Anwendungsfall definiert, welcher auf verschiedenen Konfigurationen ausgeführt wird (Abschnitt 3.4).

Eine Alternative zu diesem Vorgehen wäre die Verwendung von *Esri Thematic Atlas*[36]. Mit dieser Lösung bietet Esri eine konfigurierbare Atlas-Webapplikation an. Damit wird dem Ersteller des Atlas eine Menge abgenommen, da er sich hauptsächlich um die Daten und nicht um die Webapplikation kümmern muss. Allerdings ist diese Webapplikation nicht für kleine Bildschirme optimiert und die Benutzeroberfläche vorgegeben. Da aber der Code dieser Applikation editierbar ist, könnten die Funktionen und die Benutzeroberfläche angepasst werden und aus dieser Vorlage eine *Responsive Web App* erstellt werden. Mit einer neuen Version müssten aber alle diese Anpassungen übernommen werden. Diese Alternative wird darum in dieser Arbeit nicht weiter verfolgt.

3.1. Dienste bereitstellen

Eine Web-Karte kann dank den OGC-Standards mit eigenen oder fremden Quellen erstellt werden. Die folgenden drei Abschnitte zeigen, wie eigene Dienste mit *ArcMap* auf den Server gestellt werden, um später vom Client aus darauf zugreifen zu können. Es werden nur die Dienste behandelt, auf welche der Prototyp für den *SCHWEIZER WELTATLAS interaktiv* zugreift.

3.1.1. Karten und Features publizieren

Die Karte oder einzelne Feature-Klassen werden in *ArcMap* zusammengestellt und bearbeitet. Je nach vorgesehenem Zweck werden die Features und Attribute erstellt, bearbeitet oder gelöscht, Joins ausgeführt, die Objekte gestylet und generalisiert und die Projektion ausgewählt. Die Ausdehnung der Karte sollte begrenzt werden.

Wenn in der Web-Karte das Zoomen über viele Levels ermöglicht werden soll, empfiehlt es sich, die Massstabsabhängigkeit bereits in *ArcMap* umzusetzen, da es einfacher geht als über JavaScript. Dazu kann bei den Layer-Eigenschaften bei den allgemeinen Einstellungen unter Massstabsbereich ein minimaler und ein maximaler Massstab angegeben werden. Wenn man

[36] Esri Thematic Atlas: http://atlas.esri.com

Zoom Level	Karten-Massstab
0	1:591‘657‘550.500000
1	1:295‘828‘775.300000
2	1:147‘914‘387.600000
...	...
19	1:1‘128.497220

Abbildung 20: Diskrete Massstäbe von ArcGIS Online nach Buckley (2009)

bei den Massstabseinstellungen die diskreten[37] Massstäbe von *ArcGIS Online* (Abbildung 20) geladen hat, kann man diese aus der Combobox auswählen, statt die Massstabszahl einzugeben. Auf die Auswahl über die Combobox sollte verzichtet werden. In *ArcMap* wird zwar alles wie gewünscht dargestellt aber in der Web-Karte kann es vorkommen, dass bei einer Zoomstufe nichts angezeigt wird. Setzt man den minimalen und den maximalen Massstab irgendwo zwischen zwei diskrete Massstäbe, entgeht man diesem Problem.

Falls ein Feature Service angeboten werden soll, muss das Feature auf einer auf dem GIS Server registrierten Datenbank angelegt sein. Für einen Map Service spielt die Quelle keine Rolle. Die Daten werden beim Publizieren auf den Server kopiert. Dies ist eine Vorsichtsmassnahme, damit der Server Zugriff auf die Daten hat, wenn sie vom Dienst benötigt werden (Esri, 2013).

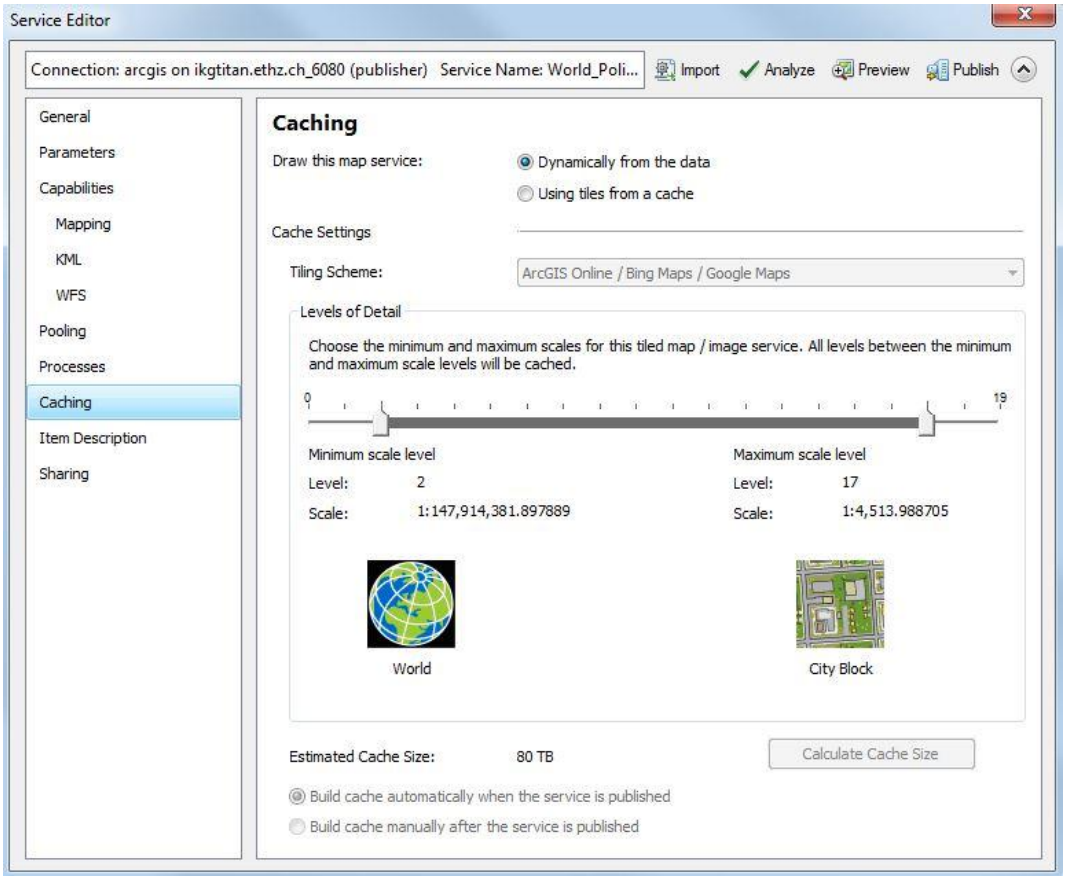


Abbildung 21: Service Editor: Caching-Einstellungen

Wenn die Karte fertig eingerichtet und gespeichert ist, kann unter *Datei* > *Freigeben als* > *Service*. Ein Wizard führt durch den Prozess. Nachdem man den GIS-Server, das Verzeichnis und den Namen des Dienstes gewählt hat, gelangt man zum *Service Editor* (Abbildung 21). In mehre-

[37] In Webkarten, zum Beispiel von Google Maps oder ArcGIS Online, ist es nicht möglich stufenlos zu zoomen. Stattdessen sind Zoom-Levels vorgesehen, die jeweils einem bestimmten Massstab entsprechen.

ren Tabs können die Einstellungen des Dienstes angepasst werden. Unter anderem können folgende Eigenschaften definiert werden:

- Welche Dienste werden angeboten (WFS, WMS, WCS,...)
- Dienstspezifische Parameter
- Caching-Einstellungen
- Metadaten
- Timeouts
- Anti-Aliasing
- Maximale vom Server zurückgegebene Features

Bei den Caching-Einstellungen kann eingestellt werden, ob die Kacheln dynamisch gerechnet werden, wenn sie benötigt werden, oder ob sie im Voraus gekachelt werden sollen. Weist die zu veröffentlichende Karte nicht die WGS 84 Web Mercator (Auxiliary Sphere) Projektion auf, kann nicht das Default-Kachelschema eingesetzt werden, sondern muss ein eigenes Schema definiert werden, falls die Kacheln im Voraus erstellt werden sollen. Dieses Schema sollte die diskreten Massstäbe von *ArcGIS Online* (Abbildung 20) aufweisen. Zusätzlich zu den Massstäben, in welchen die Karte vorgängig gekachelt werden soll, kann der räumliche Ausschnitt angegeben werden.

Bringt die Analyse des Dienstes keinen Fehler hervor und ist man gewillt, die Warnungen zu akzeptieren und nichts dagegen zu unternehmen, kann der Dienst freigegeben werden. Dies dauert je nach Datenmenge, die auf den Server zu kopieren ist und je nach Cache-Einstellungen mehrere Minuten bis Stunden. Weil das Caching die CPU des Servers stark auslastet, sollte gewartet werden, bis die Kacheln erstellt sind, bevor der Server für andere CPU-hungrige Prozesse verwendet wird. Nach erfolgreicher Freigabe stehen der Dienst und dessen Dokumentation auf dem Server bereit und können verwendet werden.

3.1.2. Geoverarbeitungs-Dienst publizieren

Die Analysefunktionen von *ArcGIS* können als Dienst im Internet freigeschaltet werden. Ein Geoverarbeitungs-Dienst (Geoprocessing Service) kann aus einem oder mehreren Tasks beziehungsweise Routinen bestehen. Ein solcher Task ist ein Geoverarbeitungswerkzeug, welches auf dem Server ausgeführt wird. Die Tasks können Standardwerkzeuge aus der *ArcGIS Toolbox* oder Werkzeuge, die mit dem *ModelBuilder* oder Python-Scripts erstellt wurden, sein. (Esri, 2013)

Nachdem die Tasks in *ArcMap* ausgeführt wurden, kann das Resultat, welches im Menü «Geoverarbeitung» unter «Ergebnisse» zu finden ist, mit Rechtsklick als Geoverarbeitungs-Dienst freigegeben werden. Dies startet, wie bei 3.1.1 einen Wizard, der zum *Service Editor* führt, wo der Dienst angepasst und publiziert werden kann. (Esri, 2013)

Falls die Ergebnisse als Karten-Dienst angezeigt werden sollen (Parameter im *Service Editor*), muss die Symbologie des Outputs des Geoverarbeitungs-Dienst zuerst angepasst werden (Esri, 2013). Alternativ können die Daten des Outputs clientseitig dargestellt werden. Je nach Geoverarbeitungs-Dienst, sollte der Ausführungsmodus (Parameter im *Service Editor*) als asynchron oder synchron definiert werden. Dieser Parameter legt fest, ob der Client wartet, bis der Task beendet ist (synchron) oder ob der Client mit anderen Prozessen weitermachen kann und den Server regelmässig nach dem aktuellen Stand des Dienstes abfragt (asynchron) (Esri,

2013). Letzteres empfiehlt sich bei rechenintensiveren Geoverarbeitungen und entspricht dem Ajax-Konzept (→ Glossar).

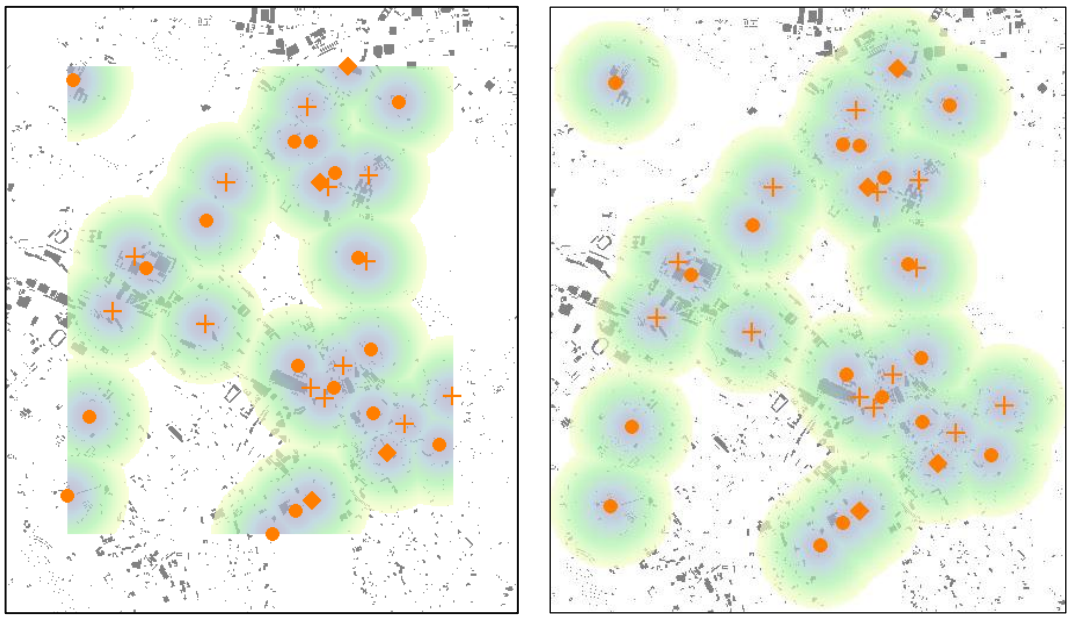


Abbildung 22: Auswirkung der Anpassung der Umgebungseinstellungen für die Geoverarbeitung

Für die Webapplikation wurde das Werkzeug «Euklidische Entfernung» aus dem Spatial Analyst freigeschaltet. Während die direkten Werkzeugparameter wie «Ausgabe Zellengrösse» oder «Maximale Entfernung» auch bei der Ausführung auf dem Server angepasst werden können, müssen die Umgebungseinstellungen für die Geoverarbeitung (Environment Settings) in *ArcMap* angepasst werden, bevor das Werkzeug ausgeführt und publiziert wird. Abbildung 22 zeigt, wie sich zum Beispiel die Verarbeitungsausdehnung (Processing Extent) auf den Output der euklidischen Entfernung auswirkt. Für das linke Bild gelten die Default-Einstellungen. Dazu wird als Ausdehnung jene der Input-Ebene verwendet. Rechts wurde die Ausdehnung auf jene des Projekts gesetzt. Damit werden die Entfernungskreise überall vollständig ausgegeben.

3.1.3. Benutzerdefinierten Print Service einrichten

Die *ArcGIS API for JavaScript* enthält unter anderem ein Print-Widget (→Glossar). Diese leicht einzubauende Funktion erlaubt es dem Anwender, die Karte als Bilddatei oder PDF zu drucken. Darin integriert sind einige Drucklayouts. Diese werden von Esri zur Verfügung gestellt. Um ein benutzerdefiniertes Layout zu verwenden, muss ein Dienst zum Drucken (Print Service) auf dem Server freigegeben werden. Dieser Dienst ist ein Geoverarbeitungs-Dienst (Abschnitt 3.1.2). Grundlage für diesen Dienst sind mxd-Dateien ohne Daten-Ebenen, in denen das Seitenlayout definiert ist. Neben den typischen Layout-Elementen wie Datenrahmen oder Massstab können dynamische Texte verwendet werden. Diese können in der Webapplikation entweder vom Benutzer gesetzt oder im Hintergrund angepasst werden. Die mxd-Dateien müssen sich in einem Ordner befinden.

Sind die Layouts vorbereitet, kann das Werkzeug «Webkarte exportieren» ausgeführt werden. Dieses ist unter *System-Toolboxes* > *Server Tools* > *Drucken* zu finden. Der Parameter «Webkarte als JSON» ist leer zu lassen, da dieser gesetzt wird, wenn der Dienst aufgerufen wird. Dem Werkzeug ist vor der Ausführung anzugeben, in welchem Verzeichnis sich die Layoutvorlagen

befinden. Nach der Ausführung wird das Werkzeug analog zum Geoverarbeitungs-Dienst (Abschnitt 3.1.2) freigegeben. (Esri, 2013)

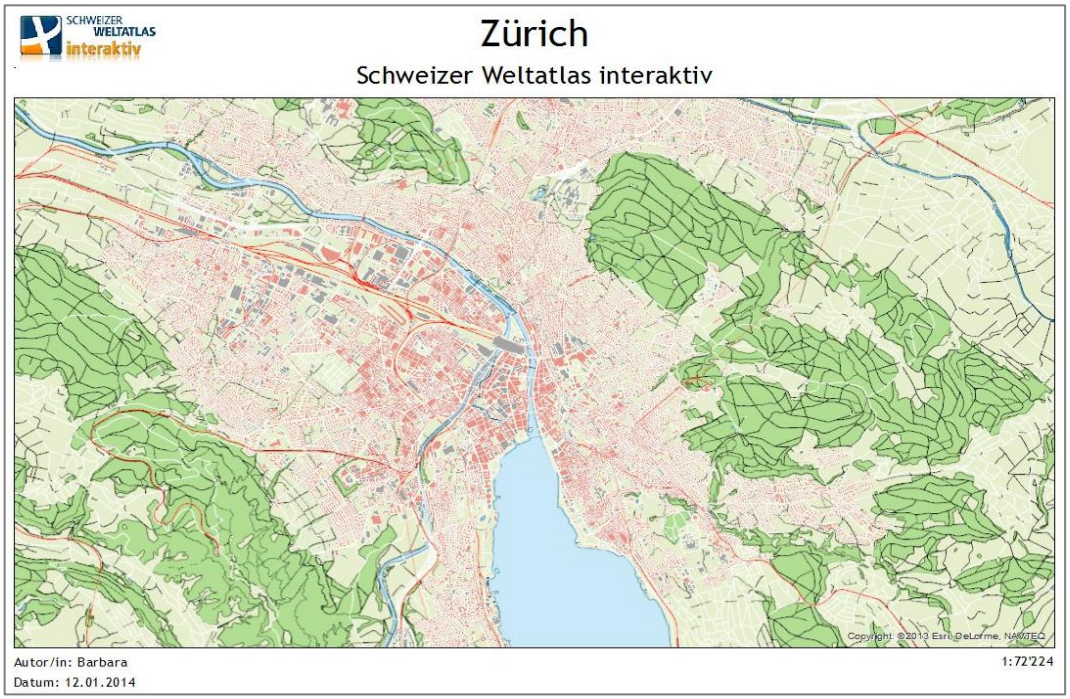


Abbildung 23: Benutzerdefiniertes Printlayout mit einer Karte des Prototyps

3.2. Aufbau des Atlas

Im nächsten Abschnitt ist beschreiben, wie der Prototyp aufgebaut ist und wie andere mit der *ArcGIS API for JavaScript* erstellte Atlanten gegliedert sind. Dazu gehört die Navigation zwischen den Karten innerhalb des Atlas. In Abschnitt 3.2.2 darauf eingegangen, wie das in Abschnitt 2.2.3 vorgestellte *Responsive Web Design* auf den Prototypen übertragen wird.

3.2.1. Atlas-Architektur

Der Prototyp für die Webanwendung des *SCHWEIZER WELTATLAS interaktiv* besteht aus einem HTML-File. Wechselt man die Karte im Atlas, so ändern sich nur einige Elemente des DOM (→ Glossar). Weil nicht ein neues HTML-File geöffnet wird, ist es nicht möglich, mit dem Back-Button im Browser zur letzten Karte zurück zu gehen. Die Webapplikation ist so aufgebaut, dass mit wenigen Anpassungen eine weitere Karte hinzugefügt werden kann. Welche Ergänzungen wo nötig sind, wird in Anhang B beschrieben.

Sobald die Karte gewechselt wird, wird die alte Karte zerstört und ein neues Kartenobjekt erzeugt. Häufig wird ein anderer Ansatz gewählt, wenn mehrere Karten darzustellen sind. Das Kartenobjekt und die Basiskarte werden belassen und es werden nur die darübergelegten Kartenebenen darüber ausgetauscht, wenn eine neue Karte geladen wird[38]. Weil unterschiedliche Referenzsysteme eingesetzt werden und diese Eigenschaft nicht nachträglich

[38] Beispiele:
Scholars GeoPortal: http://geo2.scholarsportal.info/#r/tab/browseTab/browse
Atlas of New South Wales: http://atlas.nsw.gov.au/public/nsw/home/map/base.html
Forest Atlas of Cameroon: http://www.wri.org/tools/atlas/map.php?maptheme=cameroon

geändert werden kann, ist dieser Ansatz nicht möglich. Ausserdem besteht aus Sicht der Autorin ein Atlas aus eigenständigen Karten. Ein Kartenobjekt für alle Karten zu verwenden, würde nicht dieser Philosophie entsprechen.

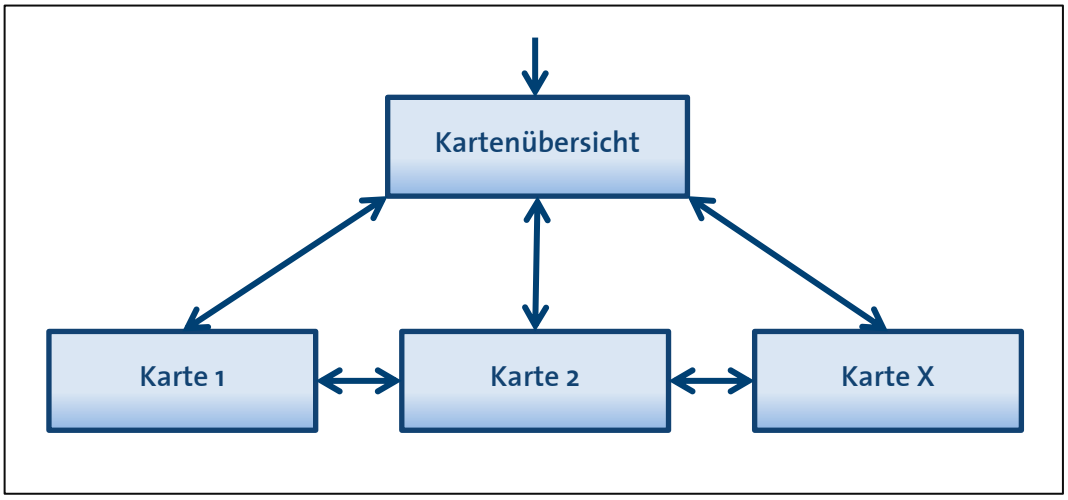


Abbildung 24: Navigation zwischen den Karten innerhalb des Atlas

Wird die Applikation geladen, erscheint zuerst die Kartenübersicht (Map Sheet Index) (Abbildung 24). Von da aus gelangt man zu den Karten des Atlas, indem man entweder in der Kartenübersicht einen Kartenumriss wählt oder eine Karte aus dem räumlichen Verzeichnis (Abschnitt 3.3.1) lädt. Von den einzelnen Karten kann man zurück zur Kartenübersicht oder über das räumliche Verzeichnis direkt eine andere Karte aufrufen.

Für die Navigation zwischen den Karten kann zwischen der Kartenübersicht und dem räumlichen Verzeichnis gewählt werden. Mit einem Verlauf könnten die zuletzt angesehenen Karten schnell geladen oder beliebte Karten erkannt werden. Ein weiteres Konzept wäre ein Verzeichnis, geordnet nach dem Alphabet oder einem Thema. Auch die Stichwortsuche über ein Eingabefeld oder eine Anzeige von thematisch oder geografisch verwandten Karten wäre denkbar. (Cron et al., 2009)

3.2.2. Umsetzung des Responsive Web Designs

Wie bereits in Abschnitt 2.2.3 angesprochen, ist *Responsive Web Design* ein heiss diskutiertes Thema, nicht aber wenn es um eine Web-App geht. *con terra* hat mit *map.apps* ein WebGIS-Produkt herausgebracht, welches ein responsives Design hat. Allerdings setzt map.apps drei statt ein HTML-File ein (Austerschulte & Kuhlmann, 2013), um den Aufbau der Web-App für die verschiedenen Bildschirmgrössen zu definieren und setzt somit auf eine Mischung aus *Responsive Web Design* und *Adaptive Web Design* (Abschnitt 2.2.3).

Der Prototyp wird auf einer Vorlage[39] für ein responsives Kartendesign von Esri aufgebaut. Der Code dieser Vorlage kann von GitHub heruntergeladen werden. Abbildung 25 zeigt diese Vorlage, bevor sie für den Prototyp angepasst wird. Rechts ist die Vorlage optimiert für die Stufe Smartphone, links sieht man, wie sie auf einem Laptop aussehen würde. Die Vorlage unterscheidet zwischen vier verschiedenen Bildschirmbreiten:

[39] Responsive Map Viewer Boilerplate for App Developers: http://maps.esri.com/SP_DEMOS/responsive

- >=1280px Desktop
- 1025-1279px Laptop
- 768-1024px Tablet
- <=767px Smartphone

Diese Grenzwerte richten sich nach häufig verwendeten Bildschirmauflösungen. Allerdings kann man mit der Bildschirmauflösung nicht bestimmen, um welchen Gerätetyp es sich handelt. So hat beispielsweise das iPhone 5 eine Auflösung von 1136×640 Pixel, und würde je nach Orientierung als Tablet eingestuft.

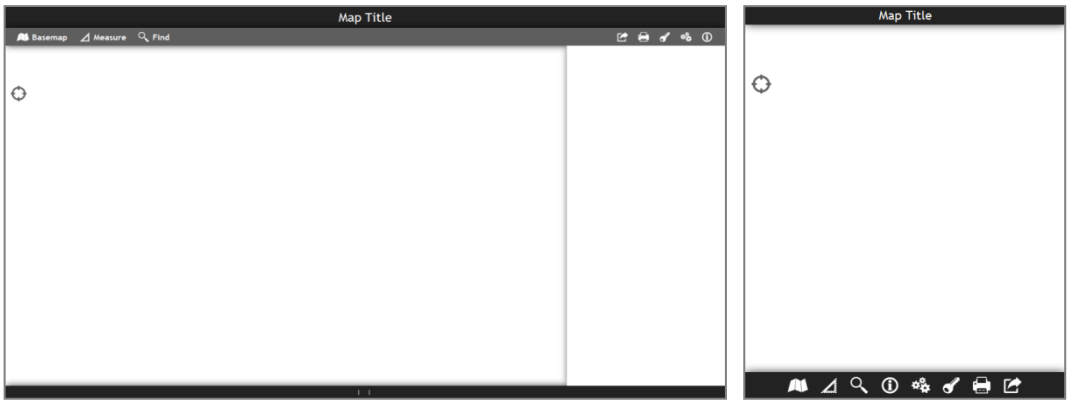


Abbildung 25: Responsive Web Design Vorlage von Esri

Die Benutzeroberfläche des Prototyps (Abbildung 26) unterscheidet sich hauptsächlich durch die Farbwahl, die Funktionen und dessen Icons sowie durch den Header von der Esri-Vorlage. Die Farbwahl der Benutzeroberfläche orientiert sich am *SWAi*. In der Vorlage lässt sich das Info-Panel (Abbildung 25 (links) zeigt das Panel am rechten Bildschirmrand) nur auf Tablets und Smartphones schliessen. Bei grösseren Bildschirmen ist es fix. Im Prototyp lässt es sich zusätzlich im Desktop- und Laptop-Design schliessen. Die Vorlage sieht vor, dass für die kleinste Bildschirmgrösse Icons mit einer kleineren Auflösung eingesetzt werden. Darauf wird im Prototyp verzichtet da die Datenmenge für die Applikation so gross ist, dass die insgesamt 87KB für die Icons nicht ins Gewicht fallen. Beim Starten des Prototyps werden allein für alle JavaScript Dateien und Libraries über 2MB geladen[40].

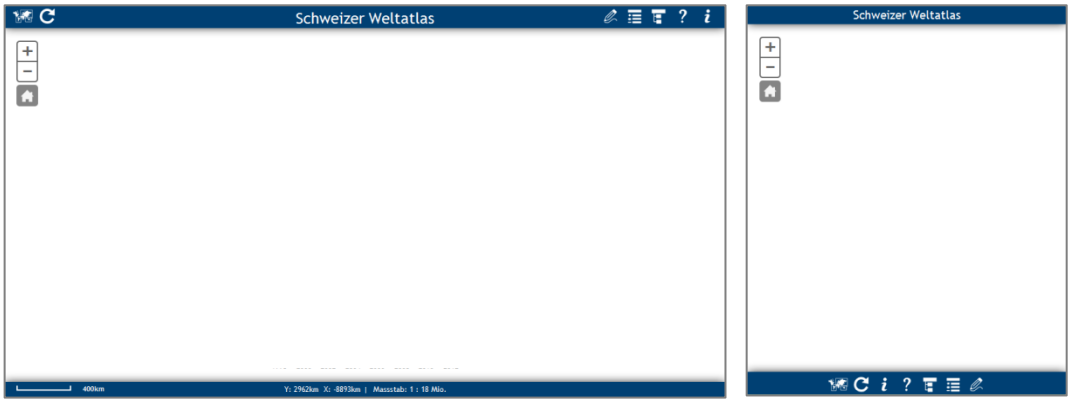


Abbildung 26: Angepasste Vorlage für den Prototyp

[40] Angabe aus Netzwerkanalyse (Entwickler-Tool) in Firefox

3.3. Karten des Atlas und ihre Tools

Die folgenden Abschnitte beschreiben detailliert die einzelnen Karten und deren Funktionen. Den Einstieg machen Tools und Funktionen, die in mehreren Karten verwendet werden. Bei den einzelnen Karten wird erwähnt, welche Tools für die Karte zur Verfügung stehen. Die Karten wurden so gewählt, dass die Daten einfach beschafft werden können und der Zielerreichung (Abschnitt 1.2) dienen. Die Ansprüche an die kartografische Qualität werden eher tief gesetzt, um mehr Zeit für die Implementierung der Webapplikation zu haben.

3.3.1. Kartenübergreifende Tools

Insgesamt sind elf kartenübergreifende Funktionen und Tools im Prototyp zu finden. Diese Tools können bei zusätzlichen Karten einfach eingebaut werden, zum Teil über einen einzelnen Funktionsaufruf. In diesem Abschnitt werden diese Funktionen vorgestellt.

Hilfe

Die Hilfe zeigt, wie man die aktuelle Karte verwenden kann und erklärt kurz die in der Karte verfügbaren Tools. Der Inhalt der Hilfe wird in einem HTML-Dokument verwaltet und in einem Iframe im Prototyp angezeigt. Obwohl die Hilfe für alle Karten in einem Dokument gespeichert ist, werden immer nur die Elemente angezeigt, die für die aktuelle Karte relevant sind. Damit das möglich ist, hat jedes Div-Element im Class-Attribut die Namen der Karten, bei welchen es angezeigt werden soll. Wird eine Karte geladen, so werden immer nur die entsprechenden Divs sichtbar gesetzt. Die anderen erhalten als Style-Attribut «display: none» und sind damit unsichtbar.

Impressum und Quellen

Das Impressum und die Quellen sind gleich wie die Hilfe umgesetzt. Hier wird ebenfalls ein HTML-File mit einem Iframe eingebettet. Wie bei der Hilfe, werden die Quellen je nach Karte ein- oder ausgeblendet. Das Impressum bleibt gleich.

Massstab und Koordinaten

Ausser beim Smartphone-Design wird in jeder Karte unten links ein Massstabsbalken angezeigt. Dieser Balken kann mit dem *Scalebar-Widget* der *ArcGIS API* einfach eingebaut werden. Der Massstab wird ausserdem numerisch in der Mitte der Fussleiste eingeblendet. Links daneben sind die Koordinaten des Mauszeigers, beziehungsweise bei Touchscreens der Kartenmittelpunkt, angegeben. Alle diese Funktionen waren in der Vorlage von Esri vorgesehen und wurden für den Prototyp leicht angepasst.

Karte neu laden

Die aktuelle Karte wird neu geladen. Dabei gehen sämtliche Anpassungen und Zeichnungselemente verloren.

Zeichnen und Bearbeiten

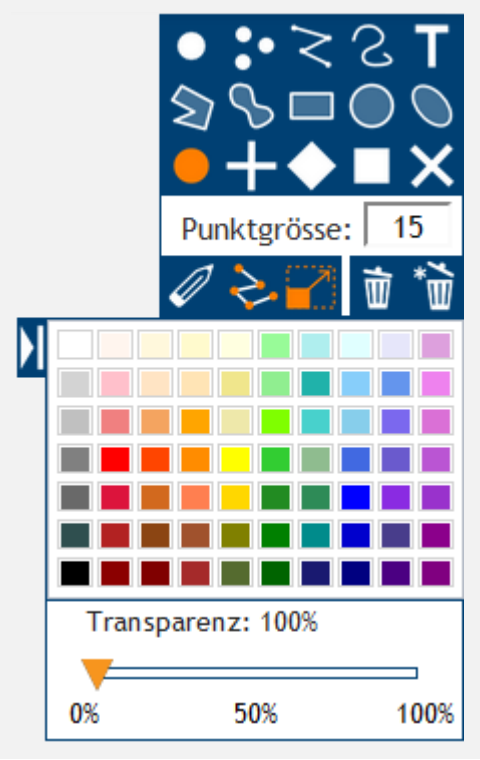


Abbildung 27: Werkzeugkasten - Zeichnen und Editieren

Der Werkzeugkasten mit den Zeichnungs- und Bearbeitungswerkzeugen baut auf den beiden Toolbars «Draw» und «Edit» der *ArcGIS API for JavaScript* auf. In der Webapplikation werden diese zusammen in einem Werkzeugkasten vereint und mit zusätzlichen Funktionen erweitert. Die Objekte werden in die Ebene map.graphics gezeichnet. Diese Kartenebene gehört zum Kartenelement der API und wird zusammen mit diesem Kartenelement zerstört. Wenn eine neue Karte geladen wird oder die aktuelle Karte aktualisiert, gehen alle Zeichnungen verloren.

In den obersten zwei Reihen des Werkzeugkastens sind die Zeichnungswerkzeuge verstaut. Der Benutzer kann auswählen, welche Geometrie er erstellen will. Zur Auswahl stehen Punkt, Multipunkt, Polylinie, freihändige Polylinie, Text, Polygon, freihändiges Polygon, Rechteck, Kreis und Ellipse.

Je nach Geometriettyp, bekommt man in den Reihen drei und vier andere Einstellungsmöglichkeiten. Für Punkt und Multipunkt können das Punktsymbol (Punkt, Kreuz, Diamant, Quadrat oder X) und die -grösse gewählt werden. Bei den zwei Linienformen und den Flächen können die Linienart (Ausgezogen, Gestrichelt oder Gepunktet) und –dicke eingestellt werden. Für die Flächen wirkt sich diese Einstellung auf den Rahmen aus. Wählt der Anwender das Texterstellungstool, können der Text, der geschrieben werden soll, und dessen Schriftgrösse angegeben werden.

Bevor der Benutzer mit dem Zeichnen beginnt, kann er die Farbe und die Transparenz festlegen. Bei Flächen wirkt sich die Transparenz nur auf die Füllung aus. Die Farbpallette mit dem Transparenzregler kann eingeklappt werden, damit die Karte weniger verdeckt wird. Bei sehr kleinem Bildschirm, also wenn die Darstellung für ein Smartphone geladen wird, ist dieser Teil standardmässig eingeklappt.

Multipunkte, Polylinien und Polygone werden mit einem Doppelklick abgeschlossen. Weil es mit einem Touchscreen recht schwierig ist, das Doppelklick-Event auszulösen, werden bei Geräten mit Touchscreen links oben zwei zusätzliche Knöpfe angezeigt, wenn einer dieser Geometrietypen erstellt wird. Einer der Knöpfe schliesst die Zeichnung ab und ersetzt den Doppelklick. Der andere verwirft das bereits Gezeichnete.

In der fünften Reihe sind die Bearbeitungswerkzeuge angeordnet. Mit dem Stift kann man die zu bearbeitende Geometrie selektieren. Je nach Geometrietyp kann das selektierte Element verschoben, skaliert oder rotiert werden. Bei Linien und Flächen können die einzelnen Stützpunkte verschoben und neue hinzugefügt werden. Dies ist nur möglich, wenn «Stützpunkte bearbeiten», die Option direkt neben dem Editierstift aktiv ist. Mit der Option «Gleichmässig skalieren» (rechts von «Stützpunkte bearbeiten») kann gesteuert werden, ob das Verhältnis von Höhe zu Breite gesperrt ist. Wenn ein Objekt für die Bearbeitung aktiviert ist, können auch

Farbe, Transparenz und die Optionen von Reihe drei und vier geändert werden. Die gezeichneten Objekte können mit «Löschen» einzeln gelöscht werden. Mit dem Tool ganz rechts, werden nach erfolgter Bestätigung alle Objekte von der Zeichnungsebene gelöscht.

Drucken

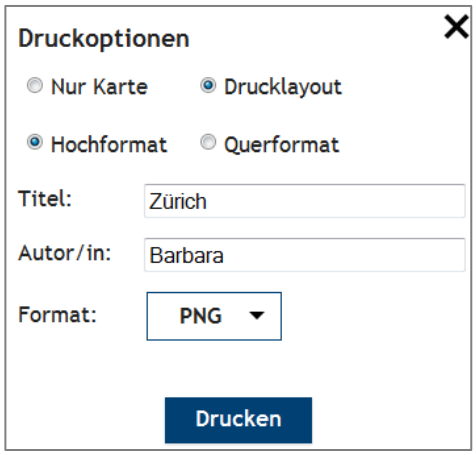


Abbildung 28: Karte aus Webapplikation drucken

In Abschnitt 3.1.3 wurde erläutert, wie ein benutzerdefinierter Print Service eingerichtet wird. Diesen Dienst nutzt die Druck-Funktion der Webapplikation (Abbildung 28). Der Benutzer erhält die Möglichkeit, aus vier Dateiformaten eines auszuwählen und mit Radio-Buttons zu entscheiden, ob nur die Karte gedruckt werden soll oder ob die Karte mit einem Karten-Layout exportieren wird. Wählt er letzteres kann er zusätzlich zur Seitenausrichtung auch einen Titel und den Namen des Autors erfassen. Die in dem gewählten Format exportierte Karte kann nach Abschluss des Exportvorgangs über einen Hyperlink, dessen Ziel sich in einem neuen Tab öffnet, angesehen werden. Die Druckfunktion baut auf der *Print*-Klasse der *ArcGIS API* auf.

Startansicht

Die Startansicht der Karte wird geladen. Dabei werden nur das Zoomlevel und das Kartenzentrum geändert. Die anderen Karteneinstellungen, zum Beispiel die sichtbaren Ebenen, und die Zeichnungselemente bleiben erhalten. Für diese Funktion wird die Klasse *HomeButton* der *ArcGIS API for JavaScript* verwendet

Geografisches Kartenverzeichnis

Das geografische Kartenverzeichnis ist eine Alternative zur Kartenübersicht (Abschnitt 3.3.2). In einer Baumstruktur werden die Karten des Atlas geografisch geordnet. Die einzelnen Ordner können auf- und zugeklappt werden. Sobald der Anwender auf ein Baum-Blatt, also eine Karte, klickt, wird diese geladen. Das Verzeichnis ermöglicht die Navigation zwischen den Karten, ohne den Umweg über die Kartenübersicht zu gehen (siehe Abbildung 24). Für das Verzeichnis wird *dijit.Tree* vom *Dojo Toolkit* verwendet.

Abbildung 29: Geografisches Kartenverzeichnis

Kartenübersicht laden

Von jeder Karte kann direkt zurück zur Kartenübersicht (Abschnitt 3.3.2) navigiert werden. Bei dieser Interaktion werden die Zeichnungen gelöscht.

Lokalisierung

Unterstützt der Browser die Geolokalisierung und willigt der Anwender darin ein, kann der aktuelle Standort ausfindig gemacht werden. Damit ist der Zugriff auf allfällig vorhandene GPS-Sensoren möglich. Diese Funktion baut auf der *ArcGIS API* - Klasse *Locator* auf. Um den Punkt, welcher in der Karte den Standort markiert, wieder zu entfernen, kann ein zweites Mal auf das GPS-Symbol geklickt werden.

Legende

Die Legende im Prototyp erfüllt gleich zwei Funktionen. Einerseits dient sie als klassische Legende, in welcher der Atlasnutzer die Bedeutung der Symbole und Farben erfahren kann, andererseits können Ebenen ein- und ausgeschaltet werden. Die Legende ist auf dem *Table of Content* Widget von Liu (2013) aufgebaut. Die vom Widget erzeugte Legende wird in den meisten Karten nachträglich angepasst. So wird beispielsweise in der Legende von Abbildung 30 bei Industrie und Wohngebäude das Symbol erst nachträglich eingefügt. Mit dem Widget von Liu kann gewählt werden, welche Kartenebenen in welcher Reihenfolge in der Legende erscheinen sollen. Ausserdem kann jeder Ebene ein Transparenzregler angehängt werden oder angegeben werden, ob sie defaultmässig eingeklappt oder aufgefaltet ist. Es lässt sich nachträglich noch ein Ebene hinzufügen. Dies ist zum Beispiel nötig, wenn das Ergebnis eines Geoverarbeitungswerkzeugs, welches nicht bei der Erzeugung der Legende vorhanden ist, in die Legende aufgenommen werden soll.

Abbildung 30: Legende der Karte von Zürich

3.3.2. Kartenübersicht (Map Sheet Index)

Die Kartenübersicht (Abbildung 31) ist die Startkarte. Der Atlasnutzer sieht, von welchen Regionen Karten vorhanden sind. Als Hintergrundkarte wird die *World Light Gray* Karte von Esri eingesetzt. Die Kartenübersicht verwendet *WGS84 Web Mercator (Auxiliary Sphere)* als geografisches Referenzsystem. Diese winkeltreue Projektion eignet sich für die Darstellung der Kartenausdehnungen, da diese als Rechtecke abgebildet werden. Andere winkeltreue Projektionen wären möglich. Allerdings bietet Esri die Kacheln für diese Basiskarte nur in der Web Mercator Projektion an.

Die Rechtecke werden als Feature-Layer angezeigt. Die Grundlage, also der Web Feature Service, wurde, wie in Abschnitt 3.1.1 erläutert, auf dem GIS-Server freigeschaltet. Ändert sich die Anzahl Karten im Atlas oder der Ausschnitt einer Karte, so ist im JavaScript Code der Kartenübersicht eine Funktion eingebaut, die bei Bedarf aufgerufen werden kann. Diese Funktion löscht zuerst alle Features der Kartenebene und erstellt dann die Rechtecke aufgrund der Angaben in einem Array. Dieser Array, der ebenfalls in diesem Skript und kann beliebig mit neuen Karten erweitert werden. In diesem Array ist neben dem Kartennamen auch der Kartenumriss als *Extent* angegeben. Damit kann der Feature-Layer schnell aktualisiert werden.

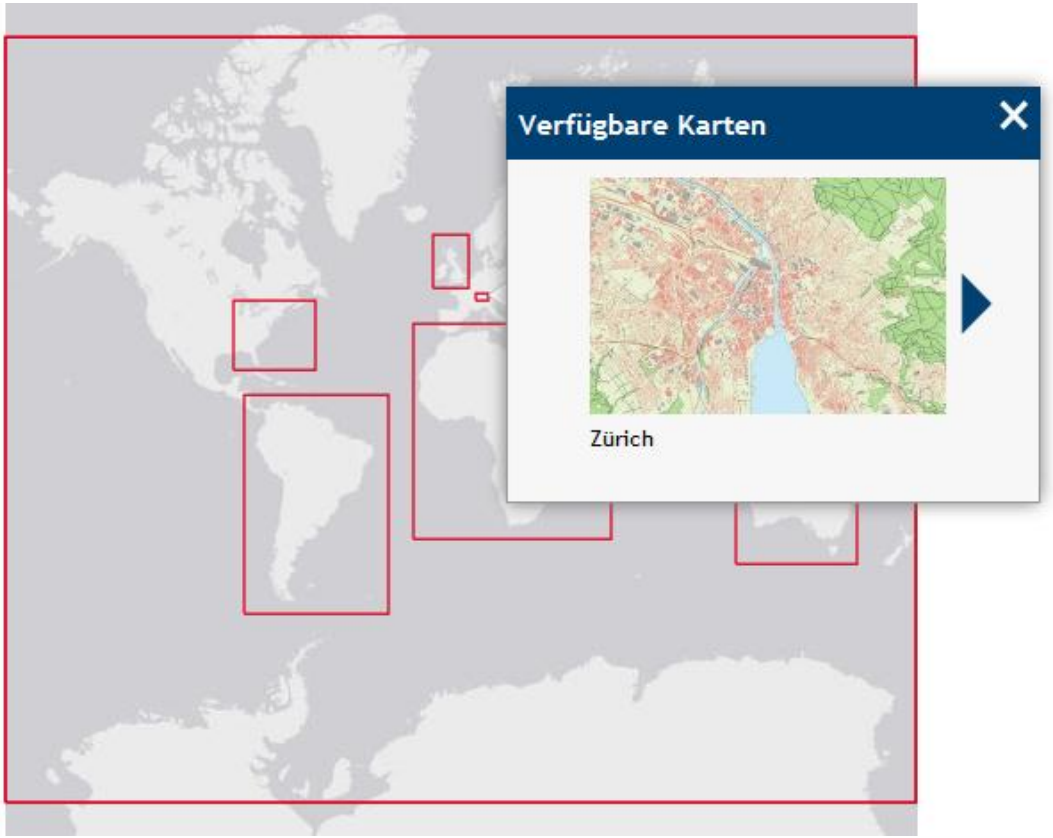


Abbildung 31: Kartenübersicht mit Infofenster

Klickt man in eines der Rechtecke, öffnet sich ein Infofenster. Darin werden die Kartenvorschau und der Name der Karte angezeigt. Sind für den geklickten Punkt mehrere Karten vorhanden, kann man sich durch die Karten klicken. Dieses Karussell basiert auf dem *Tiny Carousel* Plugin von Baijs (2010). In der Kartenübersicht wird der jeweilige Umriss farblich hervorgehoben. Die Karten, die den geklickten Punkt beinhalten, werden mit einem *Identify Task*, einer Funktion der *ArcGIS API*, gefunden. Damit auch mit einem Touchscreen der Umriss einer kleinräumigen Karte selektiert werden kann, ist die Toleranz für Touchscreens höher eingestellt.

Statt die verfügbaren Karten in einem Karussell anzuzeigen, könnten alle gleichzeitig in einer Galerie dargestellt werden. Da aber immer die Karte mit der kleinsten Ausdehnung zuoberst liegt, wird diese Karte an der ersten Stelle im Karussell erscheinen. Wenn der Benutzer zum Beispiel auf die Schweiz klickt, wird die Schweizerkarte vor der Europakarte und der Weltkarte angezeigt. Der Atlasnutzer muss nicht die gewünschte Karte in einer Galerie suchen.

Die Kartenübersicht besitzt als Zusatzfunktionen eine Hilfe, ein Impressum mit Quellenangabe, eine Massstabs- und Koordinatenangabe und das geografische Verzeichnis. Die Karte kann neu geladen werden.

3.3.3. Politische Karte

Die politische Karte (Abbildung 32) verwendet einen für diese Arbeit erstellten gekachelten Karten-Dienst. Die Daten dafür stammen aus dem *SCHWEIZER WELTATLAS*. Alle Ebenen sind in einer Kachelebene zusammengefasst. Aus diesem Grund können nicht einzelne Objektklassen ausgeblendet werden. Bei der politischen Karte handelt es sich um eine massstabsabhängige

Karte in der Robinson-Projektion. Der Name von Staaten, Flüssen, Seen und Städten kann mit einem Klick auf das Objekt angezeigt werden. Im Hintergrund wird, wie bei der Kartenübersicht, ein *Identify Task* ausgeführt. Sind für den geklickten Ort mehrere Objekte innerhalb der Toleranz, kann man mit einem Klick auf die Pfeile des Infofensters durch die Namen navigieren. Für Bildschirme, die der Kategorie Smartphone zugeordnet werden (Abschnitt 3.2), wird ein mobiles Popup, welches auf kleine Bildschirme optimiert ist, angezeigt.

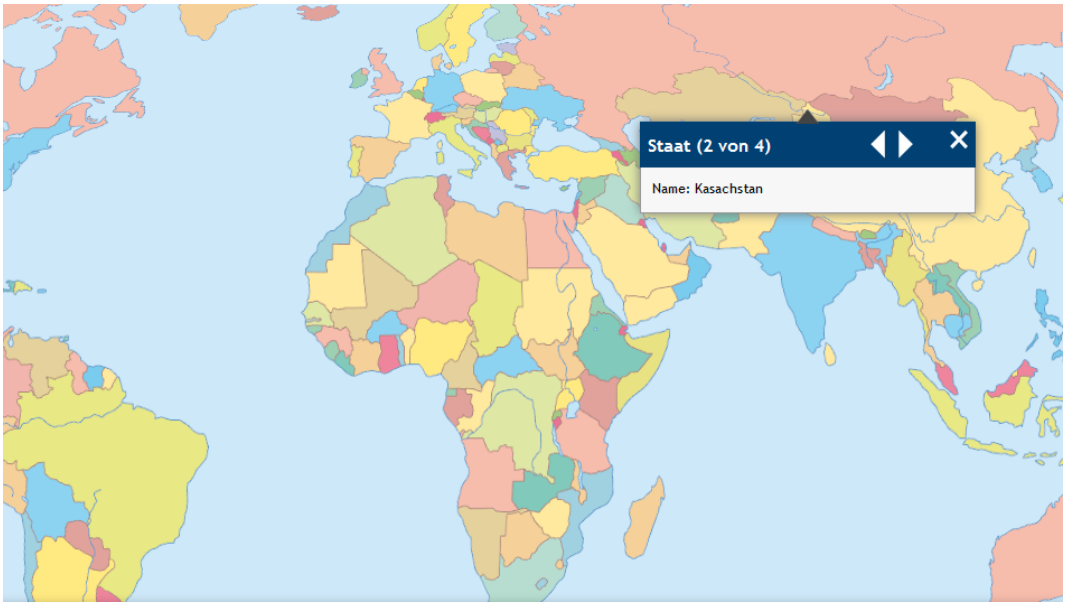


Abbildung 32: Politische Weltkarte mit Infofenster

In der politischen Karte können Zeichnungen erstellt und bearbeitet werden. Damit der *Identify Task* nicht ausgeführt wird, wenn man während dem Zeichnen auf einen Staat, See, Fluss oder eine Stadt klickt, wird er für die Zeit in welcher der Werkzeugkasten aktiv ist, deaktiviert. Die politische Karte hat ausser der Lokalisierung alle kartenübergreifenden Tools.

3.3.4. Karte von Zürich

Die Karte von Zürich (Abbildung 33) zeigt die Gebäude der Stadt Zürich und die Topografie der näheren Umgebung und hat alle kartenübergreifenden Tools eingebunden. Für diese Karte werden die *Vector25* Daten von *swisstopo* verwendet. Der Zeitraum, in welchem die Wohngebäude erstellt wurden, ist bekannt. Sie stammen aus der Gebäude- und Wohnungsstatistik (GWS) des *Bundesamts für Statistik*. Mit diesen beiden Datensätzen lässt sich die Entwicklung der Stadt Zürich darstellen. Da dort aber mehrere Tausend Wohngebäude stehen, kann diese Entwicklung nicht mit einem WFS visualisiert werden. Die Datenmenge, die vom Server an den Client geschickt werden müsste, wäre riesig. Die standardmässig eingestellte Anzahl Objekte liegt bei Tausend. Aus diesem Grund wird die Stadtentwicklung mit WMS-Layern dargestellt. Für jede Bauperiode wird ein WMS-Layer erstellt. Legt man alle Bauperioden übereinander, sieht man das aktuelle Stadtbild. In der Legende sind die Wohngebäude nur einmal aufgeführt. Ein Listener, der angesprochen wird, sobald der Benutzer die Sichtbarkeit der Wohngebäude ändert, ermöglicht es, alle WMS-Layer der Wohngebäude gleichzeitig ein- oder auszublenden.

Mit einem Schieberegler mit zwei Reglern kann die anzuzeigende Bauperiode ausgewählt werden. Dieser Schieberegler kann, wie die Farbpallette des Zeichnungswerkzeugkastens,

eingeklappt werden. Eine besondere Schwierigkeit stellen die ungleich langen Perioden der vorhandenen Daten dar. Während einige nur fünf Jahre lang sind, dauern andere 26 Jahre. Der Schieberegler muss darum in ungleiche Intervalle eingeteilt werden. Je nach Bildschirmgrösse wird der Schieberegler in der Karte oder im Panel bei der Legende angezeigt. Sobald einer der Regler verschoben wird, werden die sichtbaren WMS-Layer der Wohngebäude angepasst.

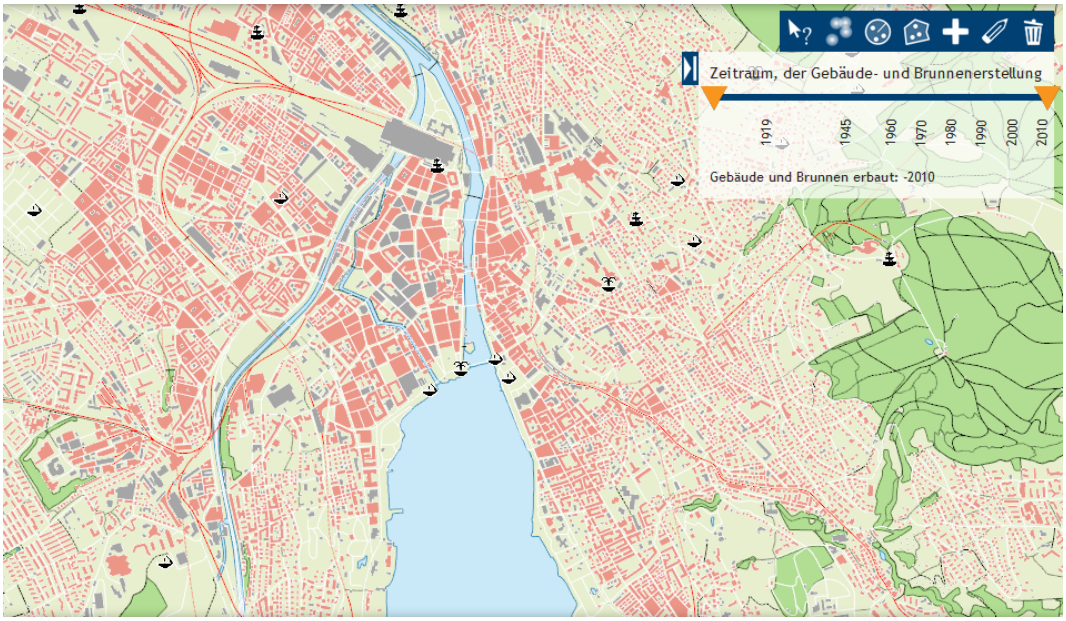


Abbildung 33: Karte von Zürich mit Brunnen-Toolbar und Schieberegler

Gleichzeitig werden die Brunnen aufgrund ihres Baujahres ein- und ausgeblendet. Die Brunnen sind in einem Feature-Layer, der auf einem WFS aufbaut und können vom Atlasnutzer bearbeitet werden. Mit der speziell dafür eingerichteten Toolbar können Brunnen erstellt, bearbeitet, verschoben und gelöscht werden. Jedem Brunnen können folgende Attribute zugeordnet werden: Name, Beschrieb (optional), Baujahr (optional), Brunnenart, Wasserart, und Zierart, falls die Brunnenart «Zierbrunnen» ist. Bei den drei Letztgenannten handelt es sich um Aufzähltypen. Für diese Attribute kann aus einer Combobox, die vorgegebene Werte enthält, der passende gewählt werden. Bevor ein neuer Brunnen gespeichert wird oder eine Änderung in die Datenbank geschrieben wird, werden die Attribute aufgrund verschiedener Kriterien geprüft. Ein Brunnen muss zum Beispiel einen Namen haben und das Baujahr muss eine Zahl sein, falls es eingegeben wurde. Die Attribute können mit dem «Brunnen Inspektor» abgefragt werden. Diese Abfrage basiert auf einem *PopupTemplate*, einer Klasse aus der *ArcGIS API*, mit dem ausgewählte Attribute eines Features angezeigt werden können, wenn der Anwender auf das Objekt klickt. Damit mehrere Benutzer gleichzeitig Brunnen bearbeiten können, wird der Feature-Layer jede Minute neu geladen. So werden Veränderungen durch andere Leute sichtbar.

Das Datenmodell für die Brunnen wurde für diese Arbeit erstellt. Die Objektklasse «Brunnen» wurde gewählt, weil Zürich eine der weltweit brunnenreichsten Städte ist. Mit 1200 Brunnen (Stadt Zürich, 2008) sollte jeder Zürcher einige Standorte kennen. Diese können ausserdem im ZüriPlan[41] unter dem Thema «Infrastruktur» angezeigt werden. Die Attribute und die Aufzähl-

[41] ZüriPlan: http://www.stadtplan.stadt-zuerich.ch

typen werden von der Autorin dieser Arbeit gewählt, die in diesem Fachgebiet nur ein laienhaftes Wissen aufweist

Die so erfassten Geodaten können mit drei Tools analysiert werden. Ein Werkzeug zählt die Brunnen, die innerhalb eines frei definierbaren Radius liegen. Der Puffer, der für diese Aufgabe benötigt wird, wird mit dem Geometry-Service vom verwendeten GIS-Server erstellt. Ein anderes Werkzeug lässt den Benutzer ein Polygon aufspannen und zählt die darin enthaltenen Brunnen. Das dritte Tool baut auf einem für diese Arbeit eingerichteten Geoverarbeitungs-Dienst auf, welcher die euklidische Distanz berechnet (Abschnitt 3.1.2). Dem Dienst kann unter anderem die maximale Distanz übermittelt werden. Das Resultat (Abbildung 34) dieses Tools ist eine Visualisierung der Brunnen-Abdeckung und wird als Ebene in die Legende aufgenommen. Die Farbe gibt Auskunft darüber, wie weit der nächste Brunnen entfernt ist. Anfangs war geplant, eine Dichtekarte aus den Brunnen zu erstellen. Weil das Resultat dieser Geoverarbeitungsfunktion bei wenigen Punktobjekten schwierig zu interpretierend ist, da die Rasterzellen gross gewählt werden müssten, wurde von der Realisierung einer Dichtekarte abgesehen.

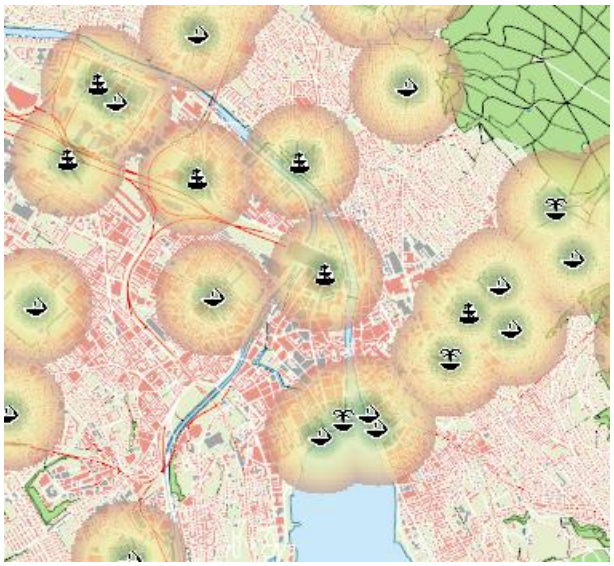


Abbildung 34: Geoverarbeitungs-Dienst: Euklidische Distanz angewendet auf Brunnen

3.3.5. Dieselpreis-Karte

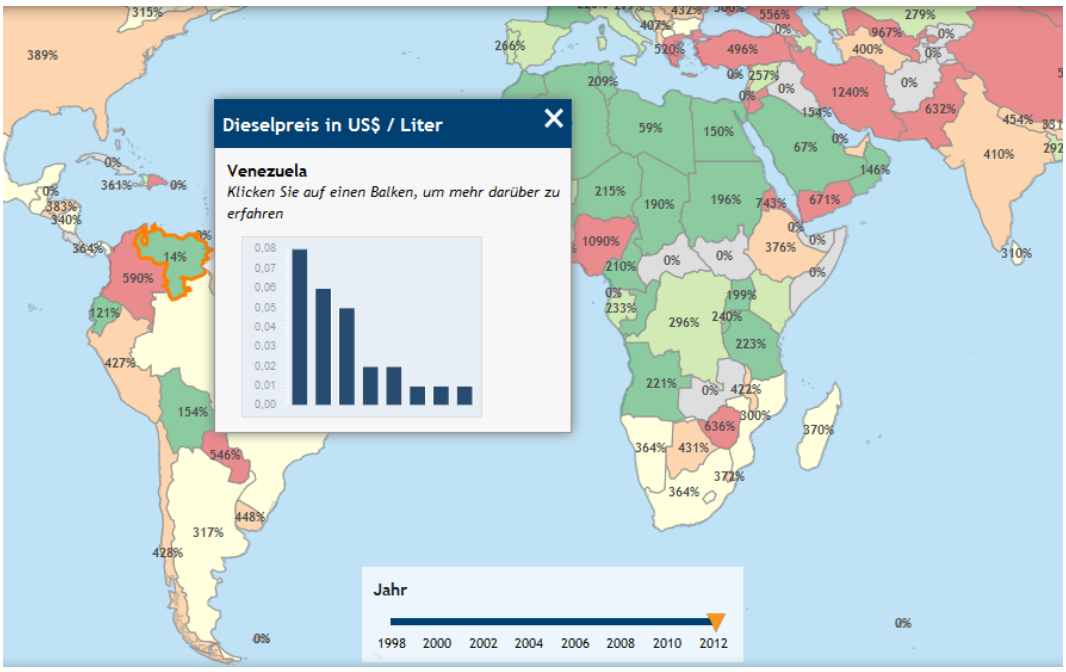


Abbildung 35: Dieselpreis-Karte mit Infofenster

In der Dieselpreis-Karte (Abbildung 35) wird die Veränderung des Dieselpreises zwischen 1998 und einem über den Schieberegler wählbaren Jahr angezeigt. Die massstabsabhängigen Geometriedaten werden den Daten des *SCHWEIZER WELTATLAS* entnommen. Die thematischen Daten, nämlich die absoluten Dieselpreise, stammen von der *World Bank Group*[42] und sind für alle geraden Jahreszahlen vorhanden. Sie werden als csv-File in *ArcMap* importiert und mit der Geometrie verbunden. Die Dieselpreisveränderung, die in der Karte dargestellt wird, wird ebenfalls in *ArcMap* berechnet.

Den Staaten wird vor der Freigabe auf dem Server eine einheitliche, attributunabhängige Symbologie gegeben. In der Webapplikation werden sie als Feature-Layer eingebunden. Die thematische Karte wird erst erstellt, wenn der Benutzer die Karte lädt. Der Workflow, der eingesetzt wird, um die Klassen zu berechnen und zu symbolisieren ist in Abbildung 36 schematisch dargestellt. Zuerst werden alle Dieselpreisveränderungsdaten nach Jahresdifferenz (zum Beispiel 1998-2000 oder 1998-2002) getrennt eingelesen und ausgewertet. Die Veränderung der jeweiligen Jahresdifferenz wird in einem Array gespeichert und nach Grösse sortiert. Anschliessend werden das Minimum, das Maximum und die 20%, 40%, 60% und 80% Quantile ausgelesen und gespeichert. Sind alle Veränderungswerte eingelesen, kann für jede Jahresdifferenz ein sogenannter *ClassBreaksRenderer* erstellt werden. Dieser *ClassBreaksRenderer* ist ein Objekt der *ArcGIS API*, das Geometrien einer Ebene aufgrund eines Attributwertes symbolisiert. Jeder Klasse des Renderers werden eine Farbe, eine Beschriftung für die Legende und eine obere und untere Grenze zugewiesen. Ausserdem wird dem Renderer das Attribut, nach welchem er den Style setzen soll, mitgeteilt. Alle *ClassBreaksRenderer* werden gespeichert und nach Bedarf auf die Geometrien angewendet.

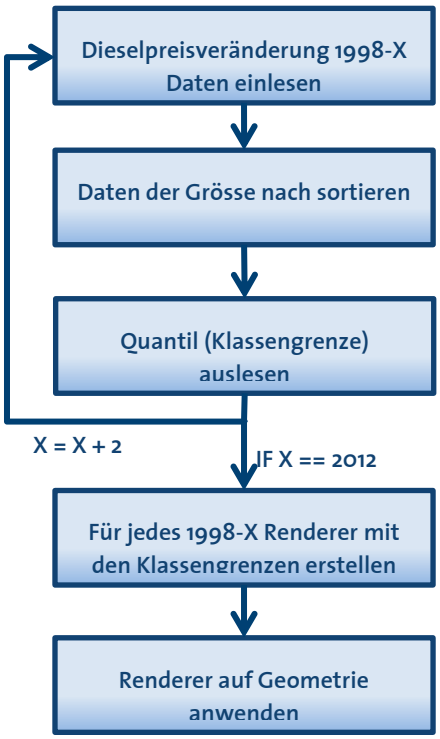


Abbildung 36: Erstellung der Renderer für die Dieselpreis-Karte

Es muss für jede Jahresdifferenz ein Renderer erstellt werden, weil die Daten der einzelnen Perioden sehr unterschiedlich sind. Während für alle Jahresdifferenzen die minimale Veränderung des Dieselpreises zwischen 12% und 40% liegt, beträgt die maximale Veränderung 290% bis 1241%. Würde man den Renderer auf die Veränderung zwischen 1998 und 2000 optimieren, würde die Karte für die Veränderung zwischen 1998 und 2012

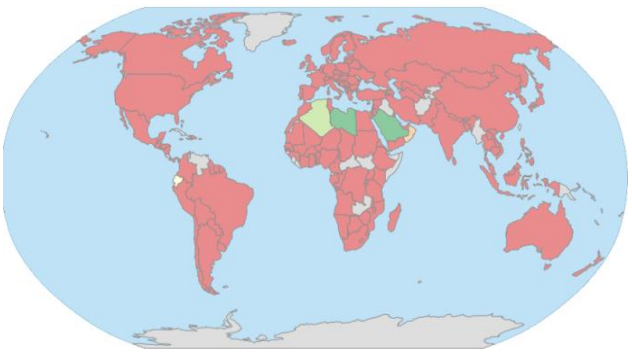


Abbildung 37: Dieselpreisveränderung 1998-20012 mit dem Renderer für 1998-2000

[42] The World Bank Group: http://www.worldbank.org

wie in Abbildung 37 aussehen. Diese thematische Karte lässt sich nicht mehr lesen. Darum wird für jede Jahresdifferenz ein anderer Renderer eingesetzt. Die Folge davon ist, dass sich die Klassengrenzen ändern. Die Beschriftungen werden ebenfalls von einem Renderer erstellt. Dieser Renderer ist allerdings einfacher aufgebaut, da er keine Klasseneinteilung vornehmen muss. Ihm werden nur ein Textsymbol und das anzuzeigende Attribut übergeben. In der Dieselkarte kann sowohl der absolute Dieselpreis als auch die Dieselpreisveränderung als Beschriftung angezeigt werden - allerdings nicht gleichzeitig. Verändern der Benutzer den Wert des Schiebereglers, wird im Hintergrund ein neuer Renderer auf die Geometrien angewendet. Gleichzeitig wird die Legende aktualisiert, damit die aktuellen Klassengrenzen angezeigt werden. Bei den Beschriftungen wird dem Renderer das neue Attribut, welches für die Beschriftung verwendet werden soll, angegeben. Der Schieberegler wird bei kleinen Bildschirmen, die für das responsive Design als Smartphones kategorisiert werden, in der Legende angezeigt.

Die Massstabsabhängigkeit dieser Karte führt zu Schwierigkeiten. Um die drei Generalisierungsstufen anzeigen zu können, müssen drei Feature-Layer mit den Geometrien der Staaten geladen werden. Das führt dazu, dass es drei Beschriftungsebenen braucht, um die Dieselpreisveränderung als Prozentwert anzeigen zu können. Insgesamt hat man also sechs Beschriftungsebenen, je drei für die Veränderung und den absoluten Preis, und drei Feature-Layer. Damit sich der Benutzer nicht um die Massstabsabhängigkeit kümmern muss, wird nur je eine Ebene in der Legende aufgeführt. Mit dem gleichen Prinzip wie in der Karte von Zürich für die Wohngebäude werden für die Dieselkarte die Kartenebenen im Hintergrund ein- und ausgeblendet, wenn der Anwender die Sichtbarkeit der Ebene in der Legende ändert.

In Abbildung 35 wird ein Infofenster für den Staat Venezuela angezeigt. Neben dem Namen des Landes ist die Entwicklung des Dieselpreises als Säulendiagramm dargestellt. Dieses Infofenster baut, wie der «Brunnen Inspektor» der Karte von Zürich, auf einem *PopupTemplate* auf. Der Unterschied liegt in der Art der Anzeige der Attribute. Während die Attribute der Brunnen als Text angezeigt werden, wird die Dieselpreisentwicklung als Diagramm dargestellt. Damit dieses Infofenster nicht während dem Zeichnen aufgeht, wird die gleiche Logik wie in der politischen Karte eingesetzt (Abschnitt 3.3.3).

Neben dem Schieberegler und dem Popup-Fenster, in dem die Preisentwicklung dargestellt wird, besitzt diese Karte die meisten kartenübergreifenden Funktionen. Nicht eingebaut sind die Lokalisierung und die Druckfunktion. Die Druckfunktion wird weggelassen, weil das Resultat unbefriedigend ist (Abbildung 38). Die Opazität der Flächensymbole wird für den Druck, unabhängig vom Ausgabeformat, nicht berücksichtigt. Auch die Beschriftungsebenen werden nicht gedruckt. Die Ursache für dieses Verhalten ist unklar.

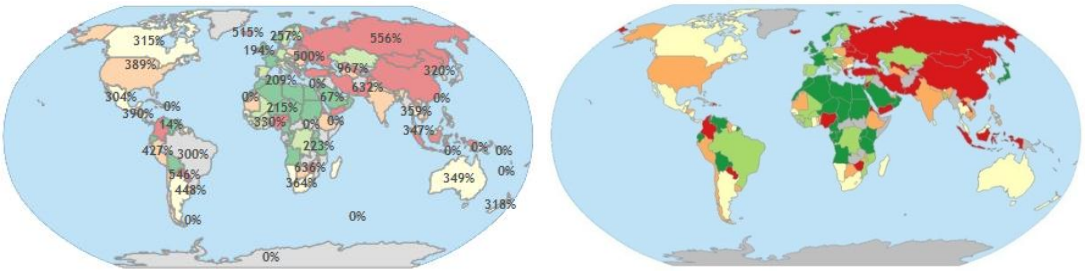


Abbildung 38: Dieselpreiskarte, wie sie im Browser dargestellt wird (links) und Ausdruck der Dieselpreiskarte als JPG-Datei (rechts)

3.3.6. Thematische Karte der Schweiz

Die Karte der Schweiz hat einen ähnlichen Aufbau wie die Dieselpreis-Karte. Statt Geometrien von Staaten werden jene der Kantone angezeigt. Diese haben, wie die Länder, eine einheitliche Symbologie und werden als Feature-Layer in der Karte eingefügt. Allerdings sind die Attribute, nach denen die Kantone eingefärbt werden, nicht als Attribut des Features gespeichert, sondern stammen vom Bundesamt für Statistik.

Abbildung 39: Kantone der Schweiz mit einheitlichem Style und Startdialog

Die Daten, welche die thematische Karte der Schweiz darstellen soll, kann der Benutzer selber bereitstellen. Im Startdialog (Abbildung 39) erhält er die Möglichkeit, eine lokale Datei einzulesen. Damit die Datei eingelesen werden kann, muss der Browser die Web API *FileReader* unterstützen. Die Daten der eingelesenen Datei werden über das Autokennzeichen mit den Geometrien der Kantone verbunden und damit eine Choroplethenkarte mit fünf Klassen erstellt. Die Klassen werden so eingeteilt, dass in jeder Klasse etwa gleich viele Features sind. Damit diese lokale Datei eingelesen werden kann, muss sie ein bestimmtes Format, welches im Anhang D dokumentiert ist, aufweisen.

Im Gegensatz zur Dieselpreis-Karte wird für diese Karte nicht die Symbologie der vorhandenen Geometrien mit einem Renderer überschrieben, sondern ein Grafik-Layer erstellt, in dem die mit dem Renderer symbolisierten Kantone als Grafiken hinzugefügt werden. Gemäss der *ArcGIS API* wäre die Anpassung der Symbologie mit einem Renderer möglich, auch wenn das Attribut, nach dem die Klassen gebildet werden, nicht zum Feature gehört. Anstelle des Attributs wird dann eine Funktion angegeben, die den Wert für den Renderer zurückgibt. Der Ansatz mit dem Grafik-Layer wurde gewählt, um das Druck-Problem, welches bei der Dieselkarte auftaucht, zu umgehen. Der Grafik-Layer wird mit transparenten Flächensymbolen richtig gedruckt.

In der Legenden wird mit dem Widget der Datei-Name aufgeführt. Die anschliessende Klasseneinteilung wird per JavaScript erstellt. Der Grund dafür ist, dass der verwendete Code für die *Table of Contents* für Grafik-Layer nur den Namen der Ebene aufführt, nicht aber die Symbole

und deren Beschriftung. Abgesehen von der Geolokalisierung können in der thematischen Karte der Schweiz alle kartenübergreifenden Funktionen verwendet werden.

3.4. Testen der Applikation

Zahlreiche Studien und Bücher befassen sich damit, wie man eine (Web) Applikation testen soll. Dieses Thema beschäftigt so sehr, dass es eine jährliche Konferenz, die *International Conference on Testing Software and Systems*, zu diesem Thema gibt. Ausserdem stehen zahlreiche Tools zur Verfügung, mit denen Webanwendungen getestet werden können. So kann beispielsweise mit *Cross Browser Testing*[43] die Browser Kompatibilität getestet werden, *WAPT*[44] ermöglicht Belastungstests und *Am I Responsive*[45] zeigt dem Web Designer, wie seine Webseite auf verschiedene Bildschirmgrössen reagiert. Wichtig ist, dass man die Applikation ganzheitlich testet. Ein Ansatz, die Applikation zu testen ist die Unterteilung in Kategorien (Abbildung 40), welche dann in sich getestet werden (Software Testing Class, 2012). Von den acht Kategorien Funktionalität, Benutzbarkeit, Kompatibilität, Datenbank, Belastung, Schnittstelle, Performance und Sicherheit werden in dieser Arbeit nur die Funktionalität und die Kompatibilität getestet. In beschränktem Umfang werden dadurch auch die Bereiche Datenbank und Schnittstelle abgedeckt. Die drei Bereiche Belastung, Performance und Sicherheit werden nicht betrachtet, weil sie für diese Arbeit nicht relevant sind. Auch die Benutzbarkeit wird nicht getestet, da dafür umfangreichere Tests nötig wären.

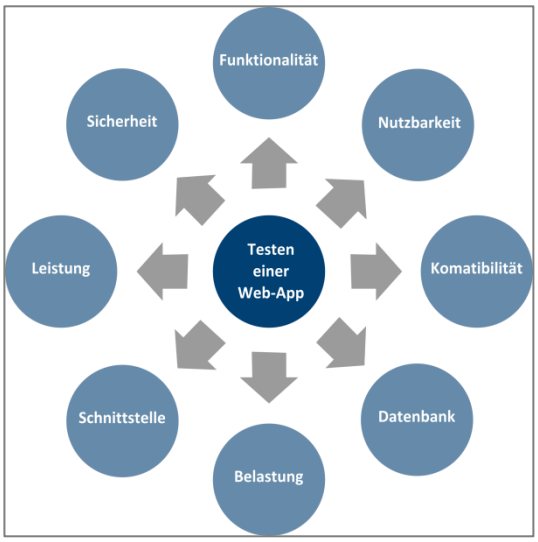


Abbildung 40: Testen einer Web-App (nach Software Testing Class (2012))

Um eine Aussage über die Funktionalität und die Browser- und Geräte-Kompatibilität zu machen, wird der Prototyp mit verschiedenen Browsern und Geräten getestet. Dazu wird ein Anwendungsfall mit mehreren Schritten definiert (Anhang C), welcher in den verschiedenen Umgebungen durchgespielt wird. Vor den Tests wird der Browser-Cache gelöscht. Beim Test werden die einzelnen Schritte beurteilt und in folgende Kategorien eingeteilt:

- Bestanden: Es funktioniert wie es sollte
- Bestanden aber Verbesserung nötig: Einzelne Teile funktionieren nicht / Es wird nicht so wie in den anderen Browsern dargestellt / Es ist schwierig zu bedienen
- Nicht bestanden: Die getestete Funktion funktioniert nicht
- Nicht testbar: Aufgrund einer Gegebenheit kann diese Funktion nicht getestet werden

Wenn ein Testschritt als nicht bestanden gilt oder nicht testbar ist, wird das Problem kurz beschrieben. Die Auswahl der Browser orientiert sich an Abbildung 18 und Abbildung 19. Diese

[43] Cross Browser Testing: http://crossbrowsertesting.com
[44] WAPT: http://www.loadtestingtool.com/index.shtml
[45] Am I Responsive: http://ami.responsivedesign.is/

beiden Abbildungen zeigen die beliebtesten Browser auf dem Desktop und auf mobilen Geräten. Bei den Geräten wurde darauf geachtet, dass verschiedene Betriebssysteme, Bildschirmgrössen und Eingabemöglichkeiten getestet werden. Für den Funktionstest werden folgende Geräte und Browser eingesetzt:

- HP ProBook 5320m, Windows 7 Professional, Steuerung: Touchpad und Tastatur
    - Google Chrome 31
    - Firefox 26
    - Safari (Windows) 5.1.4

- Fujitsu Celsius W510, Windows 7 Enterprise, Steuerung: Maus und Tastatur
    - Internet Explorer 11

- Samsung Galaxy SIII I9300, Android-Version 4.3, Steuerung: Touchscreen
    - Android-Browser 4.3
    - Firefox 26
    - Google Chrome 31

- Apple iPad Retina (4. Generation) (MD511), Version 7.0.4, Steuerung: Touchscreen
    - Mobile Safari 7
    - Google Chrome Mobile iOS 31

- ASUS Transformer Pad TF700T, Android-Version 4.2.1, Steuerung: Touchscreen
    - Android Browser 4
    - Firefox 23
    - Google Chrome 31

4. Ergebnisse

In Abschnitt 4.1 wird auf den in dieser Arbeit entwickelten Prototypen für den *SCHWEIZER WELTATLAS interaktiv* als Webapplikation eingegangen und die wichtigsten Eigenschaften hervorgehoben. Die Ergebnisse des Funktionstests werden in Abschnitt 4.2 vorgestellt.

4.1. SCHWEIZER WELTATLAS interaktiv als responsive Webapplikation

Mit der *ArcGIS API for JavaScript* wurde ein Prototyp für eine Webapplikation für den *SWAi* entwickelt. In Abschnitt 3.3 sind die einzelnen Teile des Atlas genauer beschrieben. Als Webapplikation ist der Prototyp, aufgebaut mit HTML, CSS und JavaScript, unabhängig von Plugins und plattformunabhängig. Dank dem *Responsive Web Design* passt sich die Benutzeroberfläche an die Bildschirm- oder Fenstergrösse an, falls diese nicht der Bildschirmgrösse entspricht (Abbildung 41). Theoretisch kann mit jedem Gerät, das einen modernen Browser installiert hat, auf den Atlas zugegriffen werden. Wie das in der Praxis aussieht, wird in Abschnitt 4.2 gezeigt.

Abbildung 41: Karten des Prototyps dargestellt auf verschiedenen Geräten (Visualisiert mit www.ami.responsivedesign.is)

Der Atlas besteht aus fünf Karten. Eine davon ist die Kartenübersicht, auf welcher die Umrisse der Karten dargestellt sind. Diese Übersicht ist das «Inhaltsverzeichnis» des Atlas. Zahlreiche Funktionen, wie der Zeichnungs- und Bearbeitungswerkzeugkasten, die Druckfunktion, eine Legende, in der Ebenen ein- und ausgeblendet werden können oder die Geolokalisierung des Gerätes, laden den Benutzer ein, den Atlas nicht nur zu betrachten, sondern zu erleben. So kann beispielsweise die Kartenkomplexität verringert werden, indem einige Themen ausgeblendet

werden. Die Zeichnungsfunktion kann eingesetzt werden, um etwas hervorzuheben oder um Ergänzungen zur Karte einzuzeichnen, und mit der Druckfunktion kann diese benutzerdefinierte Karte mit einem Kartenlayout exportiert werden.

Mit der Brunnen-Toolbar in der Karte von Zürich wurde ein Toolbar erschaffen, mit der über eine grafische Oberfläche Geodaten in einer Datenbank erstellt, bearbeitet und gelöscht werden können. Eine Attributabfrage und einfache Analysefunktionen sind möglich. Allerdings wären komplexere Geoverarbeitungs-Tools ohne grossen zusätzlichen Aufwand realisierbar. Diese Bearbeitungs- und Analysefunktionen machen den Prototyp zu einem GIS, in dem räumliche Daten erfasst, verwaltet, analysiert und präsentiert werden können.

4.2. Ergebnisse des Applikationstests

Der Test wurde, wie in Abschnitt 3.4 beschrieben, durchgeführt und beschäftigt sich mit der Funktionalität und der Kompatibilität. Die Ergebnisse sind in Anhang C tabellarisch aufgeführt. Einige Probleme sind browserabhängig, andere geräte- oder steuerungsabhängig. Der Prototyp läuft mit allen getesteten Konfigurationen. Auf einige ausgewählte Probleme wird nun kurz eingegangen:

- In einigen Karten wurde darauf verzichtet, für kleine Bildschirme ein mobiles Popup anstelle eines normalen Popups zu verwenden. Auf dem Samsung Galaxy öffnen sich darum einige Popups ausserhalb des sichtbaren Bereichs und können nur nach dem Verschieben der Karte angeschaut werden.
- In den zwei getesteten mobilen Firefox-Browsern können Multipunkte, Polylinien und Polygone mit mehr als drei Stützpunkten nicht richtig gezeichnet werden. Ausserdem stimmt der auf dem Touchscreen getippte Ort nicht mit der gezeichneten Position überein. Dieses Verhalten beeinflusst sowohl die Funktionalität des Zeichnungstools, als auch die Funktion, mit der die Brunnen innerhalb des gezeichneten Polygons gezählt werden können.
- Es ist sehr schwierig mit dem Finger die Werte der Schieberegler zu verändern.
- Weil die Datenbank und die Webapplikation nicht die gleiche Domain nutzen, wirkt das Sicherheitskonzept *Same-Origin-Policy*. Die Webapplikation hat dadurch keinen Zugriff auf die Datenbank. Um dieses Konzept zu umgehen, kann eine Proxy-Seite eingerichtet werden. Eine andere Möglichkeit ist *CORS* (*Cross-Origin Resource Sharing*), falls dies von beiden Servern (Datenbank-Server und Webapplikations-Server) und dem Browser unterstützt wird (Esri, 2011). Im Prototyp wird die zweite Variante eingesetzt. Damit scheint Chrome Mobile iOS ein Problem zu haben und verlangt nach einer Proxy-Page.
- Sowohl mit dem Android-, als auch mit dem Chrome-Browser, kann das ASUS Tablet nicht geolokalisiert werden. Dies ist ungewöhnlich, da beide Browser die benötigte API unterstützen und mit dem Firefox-Browser die Position gefunden werden kann.
- Mit den Chrome-Browsern auf dem ASUS Tablet und dem Samsung Galaxy kann die Datei für die thematische Karte der Schweiz nicht geöffnet werden. Mit den anderen Browsern konnte das gleiche File jedoch problemlos eingelesen werden.
- Das iPad kennt keine lokale Dateistruktur, wie man sich das von den meisten Geräten gewohnt ist. Die thematische Karte der Schweiz kann mit der aktuellen Implemen-

tierung nicht für iPads verwendet werden, da das zu visualisierende File in einem lokalen Ordner abgelegt sein muss.

- Beim Internet Explorer kam es bei mehreren Funktionen und Tools zu Problemen. Die Dieselkarte konnte nicht angezeigt werden. Internet Explorer schneidet generell bei der HTML5 Kompatibilität schlecht ab. Während die aktuelle Chrome-Version auf 503 von 555 Punkten kommt, schafft Internet Explorer 11 gerade mal 363 Punkte (HTML5 Test, 2014). Allerdings ist zu beachten, dass der Prototyp während der Entwicklung nie mit Internet Explorer getestet wurde.

Die meisten im Webapplikations-Test erkannten Probleme könnten mit zusätzlichem Aufwand behoben oder wenigstens abgefangen werden. Bei einigen ist die Ursache der Probleme unbekannt und müsste zuerst analysiert werden. Mit dem durchgeführten Funktionstest kann aber gezeigt werden, dass die Webapplikation nicht an ein Betriebssystem, einen Browser oder einen Gerätetyp gebunden ist, obwohl sie auf einem Code aufbaut. Grundsätzlich kann gesagt werden, dass die Standardbrowser (Safari für iOS, Android-Browser für Android) für die mobilen Geräte besser abgeschnitten haben, als die anderen installierten Browser. Bei Windows-Rechnern ist es genau andersrum. Google Chrome, Firefox und ein älterer Safari-Browser unterstützen die Webapplikation besser als Internet Explorer.

5. Diskussion und Ausblick

Die folgende Diskussion konzentriert sich auf die Stärken, Schwächen und Chancen des *SWAi* als Webapplikation und befasst sich mit der Zielerreichung dieser Arbeit. In Abschnitt 5.2 werden Verbesserungs- und Erweiterungsmöglichkeiten vorgeschlagen und auf die mögliche Zukunft des *SWAi* als Webapplikation eingegangen.

5.1. Diskussion

In Abschnitt 2.2.1 konnte gezeigt werden, dass Webapplikationen immer mächtiger werden und viele Vorteile mit sich bringen. Auf mobilen Geräten sind Webanwendungen eine Alternative zu hybriden und nativen Applikationen. Als Nachteile von Web-Apps werden häufig die Geschwindigkeit, der begrenzte Zugriff auf die hardwarenahen Funktionen und Gerätesensoren die Sicherheit genannt. Da die Funktionen nicht rechenintensiv sind, dürfte die Geschwindigkeit nicht relevant sein. Der Zugriff auf das Gerät wird sich verbessern, da in HTML5 viele APIs vorgesehen sind, die den Zugriff auf die Gerätesensoren ermöglichen. Die Sicherheit müsste näher betrachtet werden, vor allem wenn der Anwender, wie bei den Brunnen, Transaktionen auf einer Datenbank ausführen kann. Der durchgeführte Applikationstest zeigt eine weitere Schwäche von Webapplikationen auf: Die Web-App läuft nicht auf allen Browsern und Geräten gleicht. Als Entwickler ist man von HTML Spezifikationen und deren Unterstützung durch die Browseranbieter abhängig. Dazu kommt die Abhängigkeit von den verwendeten APIs.

Mit einer Webanwendung kann der *SCHWEIZER WELTATLAS interaktiv* auf alle Geräte mit einem Browser gebracht werden. Weil dafür nur ein Code-Satz benötigt wird, werden die Kosten sowie die Zeit für die Entwicklung und den Unterhalt des *SWAi* minimiert. Die Atlasnutzer haben immer die neuste Version und Updates können ohne Zulassung durch den App-Store und ohne Zutun des Benutzers herausgebracht werden. Zu prüfen wäre, ob für Tablets und Smartphones eine hybride App geeigneter wäre als eine Web-App. Als hybride App könnte die Applikation im App-Store angeboten werden. Die grosse Datenmenge[46], die bei der Webapplikation anfällt, ist besonders für mobile Geräte ohne W-LAN ein finanzielles Problem. Diese Datenmenge könnte möglicherweise mit einer hybriden App reduziert werden.

Obwohl die *ArcGIS API for JavaScript* häufig für einzelne Karten mit mehreren Themen eingesetzt wird (siehe Fussnote 38, Seite 25), eignet sie sich für mehrere eigenständige Karten innerhalb einer einzelnen Webapplikation. Allerdings kann die *ArcGIS API* mit *ArcGIS for Server* im Hintergrund viel mehr, als nur eine Karte mit mehreren Ebenen und Tooltips anzuzeigen. Den *SWAi* als Webapplikation anzubieten bringt viele Chancen mit sich. Wie in Abschnitt 3.1.2 erwähnt, können beliebige Geoverarbeitungs-Tools eingesetzt werden. Echtzeitkarten können aktuelle Ereignisse oder Daten visualisieren. Mit einem Feature-Service im Hintergrund und einer Erfassungsfunktion in der Applikation sind *Public Participation* Projekte realisierbar, wie das am Beispiel der Brunnen gezeigt wird. Andere interaktive Module, die nicht auf einer Karte

[46] In einer Test-Session, welche 40 Minuten dauerte, wurden 22.5MB empfangen und 2.5MB gesendet

aufbauen, können mit JavaScript eingebaut werden. Je nach Angebot, wäre der Einsatz des *SWAi* als Webapplikation ausserhalb der Schule denkbar. Diese Erweiterungen wären auch im Sinne von Nuñez (2013). Er betont, dass für mobile Schulatlanten nicht der gedruckte Atlas oder die PC-Version kopiert, sondern neue Lernmethoden, welche bei Schülern das Interesse an Karten wecken, entwickelt werden sollen.

Die in Abschnitt 1.2 gesetzten Ziele können mehrheitlich erreicht werden. Das direkte Drucken der Karte wurde zum Beispiel nicht umgesetzt und die Darstellung sowie der Aufbau der Legende, in der Ebenen ein- und ausgeschaltet werden können, kann zu wenig beeinflusst werden. In den meisten Karten sind Nachbearbeitungen der Legende nötig. Beim Export der Karte als Bilddatei mit Kartenlayout wäre es wünschenswert, dass anstelle des Kartenmassstabs ein Massstab gesetzt werden kann. Ausserdem sollte der Ausschnitt, welcher exportiert wird, in der Karte sichtbar sein. Die restlichen Ziele werden alle erreicht. Obwohl die Applikation nicht in allen Systemumgebungen und Browsern gleich gut läuft, kann gezeigt werden, dass die Web-App plattformunabhängig ist.

5.2. Ausblick

Tablets und mobile Geräte begleiten uns täglich und machen auch vor dem Unterricht keinen Halt (Abschnitt 2.1.1). Immer mehr Lehrpersonen setzen die Geräte in ihren Stunden ein, weshalb die Nachfrage nach pädagogisch sinnvollen Applikationen, die sich im Unterricht einsetzen lassen, steigt. Der *SCHWEIZER WELTATLAS interaktiv* läuft als Java Web Start Applikation weder auf Android noch auf iOS Geräten. Eine zukünftige Version wird Tablets und Smartphones unterstützen (Schweizer Weltatlas, 2014). Den Nachfolger oder eine Ergänzung zum bestehenden *SWAi* als Webapplikation anzubieten, würde dies ermöglichen.

Die im Applikationstest erkannten Probleme müssen, soweit möglich, behoben oder zumindest mit einer Fehlermeldung abgefangen werden. Dabei sollten die Geräteeigenschaften, welche die Applikation beeinflussen, besser berücksichtigt werden. So stellte sich zum Beispiel heraus, dass die im Prototyp verwendeten Schieberegler für Touchscreens nicht optimal sind oder dass es mit dem iPad nicht möglich ist, eine lokale Datei anzugeben, die visualisiert werden soll. Auch Laptops, die zusätzlich zur üblichen Steuerung über einen Touchscreen verfügen, müssen berücksichtigt werden, da Elemente der grafischen Benutzeroberfläche von der Eingabemethode abhängig sind. Es sollte eine Möglichkeit gefunden werden, die Datenmenge für mobile Geräte zu minimieren. Ein Ansatz, der vielleicht zum Speichern von Applikationsdaten verwendet werden kann, ist *Local Storage*, ein interner Speicher des Browsers.

Anschliessend an die Arbeit von Müller (2009) wäre die Integration des Kommentarbandes des *SCHWEIZER WELTATLAS* eine sinnvolle Erweiterung, wenn Karten aus dem *SWA* eingebaut werden. Lernprogramme und Quiz, die «einen spielerischen Umgang mit der Thematik ermöglichen» (Marty, 2007, S. 99) ist ein Bedürfnis von Lehrern. Mögliche Beispiele sind *Storytelling with Maps*[47] und *GeoStories*[48], zwei Angebote, bei denen ein geschichtliches oder gesellschaftliches Ereignis mit Hilfe einer interaktiven Karte erzählt wird. Ein weiteres Beispiel ist *Geo-*

[47] Storytelling with Maps: http://storymaps.esri.com/home
[48] GeoStories: http://education.nationalgeographic.com/education/topics/geostories/?ar_a=1

Games[49]*,* ein Portal mit Spielen für verschiedene Altersstufen, die den Umgang mit Karten und Geografie vermitteln.

In Abschnitt 5.1 werden als Chancen des *SWAi* Echtzeitkarten erwähnt. Mögliche Themen wären die Darstellung von Naturereignissen wie Tornados, Überschwemmungen, Busch- und Waldbrände oder von langfristigen Daten wie Umwelt- oder Wetterindikatoren. Allerdings unterstützt die *ArcGIS API for JavaScript* nur die OGC-Standards KML, WMS und WMTS (Esri, 2013). Der Zugriff auf Daten von Drittanbietern beschränkt sich auf diese drei Dienste und Anbieter, die auf ihrem Server *ArcGIS for Server* installiert haben. Die Frage, ob mit diesen Einschränkungen interessante Echtzeitkarten angeboten werden können, wäre eine Grundlage für eine zukünftige Arbeit.

Damit die Webapplikation eingesetzt werden kann, ist es zwingend nötig, die Karten und Funktionen an die Bedürfnisse und möglichen Anwendungen der Benutzer anzupassen. Es muss entschieden werden, ob die Webapplikation als Ergänzung des aktuellen *SWAi* dienen soll, oder ob sie ihn ablöst. Zudem muss die Frage nach dem Zielpublikum geklärt werden.

.

[49] GeoGames: http://education.nationalgeographic.com/education/media/geogames/?ar_a=1

6. Literaturverzeichnis

Adams, C. (2006). *Resolution dependent layout*. http://www.themaninblue.com/writing/perspective/2004/09/21 (Zugriff 01. 01. 2014)

Albert, K., & Stiller, M. (2012). Der Browser als mobile Plattform der Zukunft–Die Möglichkeiten von HTML5-Apps. *Smart Mobile Apps* (s. 147-160): Springer.

Anglin, T. (2012). *Universal Mobile Apps with HTML5 and Kendo UI* http://docs.telerik.com/kendo-ui/howto/universal-mobile-apps-html5 (Zugriff 09. 11. 2013)

Anttonen, M., Salminen, A., Mikkonen, T., & Taivalsaari, A. (2011). *Transforming the web into a real application platform: new technologies, emerging trends and missing pieces*. Proceedings of the 2011 ACM Symposium on Applied Computing, TaiChung, Taiwan.

Apple. (2013). *Mit iBooks 2 für iPad erfindet Apple das Schulbuch neu*. http://www.apple.com/chde/pr/library/2012/01/19Apple-Reinvents-Textbooks-with-iBooks-2-for-iPad.html (Zugriff 17. 01. 2014)

Austerschulte, M., & Kuhlmann, J. (2013). *Responsive Web Design in map.apps* http://developernetwork.conterra.de/content/responsive-web-design-mapapps (Zugriff 18. 01. 2014)

Baijs, M. (2010). *Tiny Carousel*. http://baijs.nl/tinycarousel (Zugriff 13. 01. 2014)

Beyeler, R. (2013). *Doppelt so viele Tablet-Nutzer wie im Vorjahr*. http://www.comparis.ch/comparis/press/medienmitteilungen/artikel/2013/telekommunikation/smartphone-verbreitung/tablet-verbreitung.aspx (Zugriff 20. 10. 2013)

Borodescu, C. (2013). *Web Sites vs. Web Apps: What the experts think* http://www.visionmobile.com/blog/2013/07/web-sites-vs-web-apps-what-the-experts-think (Zugriff 12. 10. 2013)

Bricklin, D. (2013). *Responsive Web App Design*. http://www.bricklin.com/responsiveapps.htm (Zugriff 04. 01. 2014)

Buckley, A. (2009). *How can you tell what map scales are shown for online maps?* http://blogs.esri.com/esri/arcgis/2009/03/19/how-can-you-tell-what-map-scales-are-shown-for-online-maps (Zugriff 06. 01. 2014)

Can I use. (2014). *Can I use...* http://caniuse.com (Zugriff 22. 01. 2014)

Chan, J. (2013). *Responsive Web Design is Not the Future*. Six Revisions. http://sixrevisions.com/mobile/responsive-web-design-not-the-future (Zugriff 03. 01. 2014)

Christ, A. M. (2011). *Bridging the mobile app gap*. Connectivity and the User Experience*, 11*(1), 27.

Cron, J., Marty, P., Bär, H. R., & Hurni, L. (2009). *Navigation in school atlases: Functionality, design and implementation in the "Swiss world atlas interactive"*. 24th international cartographic conference ICC, Santiago, Chile.

Cron, J., Moll, P., Häberling, C., & Hurni, L. (2011). *100 Jahre "Schweizer Weltatlas" und neuer "Schweizer Weltatlas interaktiv"*. Kartographische Nachrichten*, 61-1*, 32-35.

Dalziel, J. (2013). *The race for speed*. http://creativejs.com/2013/06/the-race-for-speed-part-1-the-javascript-engine-family-tree (Zugriff 23. 12. 2013)

De Lange, N. (2006). *Geoinformationssysteme in Schulen: derzeitiger Stand und zukünftiger Einsatz*. Lernen mit Geoinformationen. Herbert Wichmann Verlag, Heidelberg, 11-22.

Demarmels, M., Neumann, K., & Memmel, T. (2012). *Mobile Engineering–Welche Umsetzungsstrategie ist die richtige?* http://www.zuehlke.com/fileadmin/pdf/others/whitepaper_mobile_engineering.pdf (Zugriff 28. 10. 2013)

DiWT. (2011). *What is Dojo Toolkit ?* http://diwt.wordpress.com/2011/08/17/what-is-dojo-toolkit

Ernst Klett Verlag. (2010). *Kurzanleitung Klett-GIS*. http://www2.klett.de/sixcms/media.php/229/GIS-Informationsblatt_2_0.doc (Zugriff 06. 01. 2014)

Esri. (2011). *Cross Origin Resource Sharing (CORS) with the ArcGIS API for JavaScript* http://blogs.esri.com/esri/arcgis/2011/09/29/cross-origin-resource-sharing-cors-with-the-arcgis-api-for-javascript (Zugriff 14. 01. 2014)

Esri. (2013). *ArcGIS Resources - ArcGIS Help 10.1*. http://resources.arcgis.com/en/help/main/10.1

Fernandes, A. I., Goulão, M., & Rodrigues, A. (2013). *A Comparison of Maps Application Programming Interfaces*. arXiv preprint arXiv:1305.3485.

Fischer, C. (2014). *my-Pad – mobiles, kooperatives Lernen im Unterricht*. http://www.my-pad.ch/das-projekt (Zugriff 17. 01. 2014)

Gerlicher, A. (2012). Die Grenzen des Browsers durchbrechen. *Smart Mobile Apps - Mit Business-Apps ins Zeitalter mobiler Geschäftsprozesse* (s. 161-176): Springer.

Grigsby, J. (2013). *Responsive Design for Apps.* http://blog.cloudfour.com/responsive-design-for-apps-part-1 (Zugriff 18. 01. 2014)

Harjono, J., Ng, G., Kong, D., & Lo, J. (2010). *J.(2011) Building smarter web applications with HTML5.* Conference of the Center for Advanced Studies on Collaborative Research, Toronto (Canada).

Heinen, R., Kerres, M., Karpa, D., Eickelmann, B., & Graf, S. (2013). *Bring your own device: Private, mobile Endgeräte und offene Lerninfrastrukturen an Schulen*. Digitale Medien und Schule. Zur Rolle digitaler Medien in Schulpädagogik und Lehrerbildung. Schriftenreihe „Theorie und Praxis der Schulpädagogik *", 19*.

Heise, D. (2011). *Web-Apps und native Apps: ein Vergleich*. http://www.zdnet.de/41555607/web-apps-und-native-apps-ein-vergleich (Zugriff 30. 12. 2013)

Hennig, S., & Vogler, R. (2011). *WebMapping: Der Einsatz von digitalen, interaktiven Karten in Schule und Bildung*. GW-Unterricht*, 123*, 86-99.

Herbert, M. (2010). *The iPad—Breaking new ground in special education*. District Administration: New and Noteworthy.

HTML5 Test. (2014). *HTML5 Test - How well does your browser support HTML5?* . http://html5test.com/results/desktop.html (Zugriff 13. 01. 2014)

Jaser, M. (2011). *Evaluation, Bewertung und Implementierung verschiedener Cross-Platform Development Ansätze für Mobile Internet Devices auf Basis von Web-Technologien.* (Bachelorarbeit), Hochschule Augsburg - University of Applied Sciences.

Java. (2012). *Wie erhalte ich Java für mobile Geräte?* . http://www.java.com/de/download/faq/java_mobile.xml (Zugriff 08. 12. 2013)

Kaminitz, S. (2011). *Native Web or Hybrid Mobile App Development*. http://de.scribd.com/doc/50805466/Native-Web-or-Hybrid-Mobile-App-Development (Zugriff 31. 12. 2013)

Law, D. (2013). *ArcGIS for Server - Understanding architecture, deployment, and workfl ows*. 42-47. http://www.esri.com/esri-news/arcuser/spring-2013/~/media/Files/Pdfs/news/arcuser/0313/arcgis101.pdf (Zugriff 03. 01. 2014)

Liu, N. (2013). *Table of Contents (TOC)/Legend Widget for JavaScript API*. http://www.arcgis.com/home/item.html?id=9b6280a6bfb0430f8d1ebc969276b109 (Zugriff 13. 01. 2014)

Llamas, R., Reith, R., & Shirer, M. (2013). *Apple Cedes Market Share in Smartphone Operating System Market as Android Surges and Windows Phone Gains, According to IDC*. http://www.idc.com/getdoc.jsp?containerId=prUS24257413 (Zugriff 28. 10. 2013)

Marcotte, E. (2009). *Fluid Grids*. A list Apart(279).

Marcotte, E. (2011a). *Responsive Web Design*. A list Apart(306).

Marcotte, E. (2011b). *Fluid Images*. A list Apart(328).

Marty, P. (2007). *Analyse der Nutzeranforderungen an den «Schweizer Weltatlas interaktiv».* (Diplomarbeit), Universität Zürich, Zürich.

Mozilla Developer Network. (2013). *About JavaScript*. https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript (Zugriff 23. 12. 2013)

Mudge, J. (2012). *Native App vs. Mobile Web App: A quick Comparison*. Six Revisions. http://sixrevisions.com/mobile/native-app-vs-mobile-web-app-comparison (Zugriff 09. 10. 2013)

Müller, C. (2009). *Erarbeitung von Vorschlägen zur Integration von Inhalten des Kommentarbandes in den SCHWEIZER WELTATLAS interaktiv.* (Bachelorarbeit), ETH Zürich, Zürich.

Nuñez, J. J. R. (2013). *Smartphone-Based School Atlases?* Cartographica: The International Journal for Geographic Information and Geovisualization, *48*(2), 126-133.

Oleggini, L. (2013). *Einsatz des Schweizer Weltatlas interaktiv auf der Sekundarstufe II.* (Besondere Arbeit Monofach Geografie), PH Bern, Bern.

PostgreSQL. (2014). *PostgreSQL - About*. http://www.postgresql.org/about (Zugriff 03. 01. 2014)

Rittmann, T. (2014). *iPad Schule*. http://www.ipad-schule.ch (Zugriff 17. 01. 2014)

Rivera, J., & van der Meulen, R. (2013). *Gartner Says by 2016, More Than 50 Percent of Mobile Apps Deployed Will be Hybrid*.

Schweizer Weltatlas. (2014). *FAQ*. http://schweizerweltatlas.ch/faq (Zugriff 14. 01. 2014)

Software Testing Class. (2012). *Web Testing: Complete Guide To Your Web Application Testing*. http://www.softwaretestingclass.com/web-application-testing

Stadt Zürich. (2008). *Brunnen* http://www.stadt-zuerich.ch/dib/de/index/wasserversorgung/ueber_die_wasserversorgung/wasser_u_natur/brunnen.html (Zugriff 02. 01. 2014)

StatCounter Global Stats. (2013). *StatCounter Global Stats*. http://gs.statcounter.com (Zugriff 03. 01. 2014)

Taivalsaari, A., & Mikkonen, T. (2011b). *The web as an application platform: The saga continues.* 37th EUROMICRO Conference on Software Engineering and Advanced Applications, Oulo (Finland).

Taivalsaari, A., Mikkonen, T., Anttonen, M., & Salminen, A. (2011a). *The death of binary software: End user software moves to the web.* Ninth International Conference on Creating, Connecting and Collaborating through Computing, Kyoto (Japan).

Toivonen, T., & Riihelä, J. (2011). *PaikkaOppi brings GIS to schools*. positio(ICC SPECIAL ISSUE), 5-7.

W3C. (2012). *Media Queries*. http://www.w3.org/TR/css3-mediaqueries (Zugriff 01. 01. 2014)

W3C. (2014). *Media Types*. http://www.w3.org/TR/CSS2/media.html (Zugriff 01. 01. 2014)

w3schools. (2013a). *Browser Statistics*. http://www.w3schools.com/browsers/browsers_stats.asp (Zugriff 23. 12. 2013)

w3schools. (2013b). *HTML5 Web Storage*. http://www.w3schools.com/html/html5_webstorage.asp (Zugriff 10. 10. 2013)

Wege, B. (2013). *Der Neue im Klassenzimmer*. Die Zeit(30).

Willnecker, F., Ismailović, D., & Maison, W. (2012). Architekturen mobiler Multiplattform-Apps. *Smart Mobile Apps - Mit Business-Apps ins Zeitalter mobiler Geschäftsprozesse* (s. 403-417). Berlin Heidelberg: Springer.

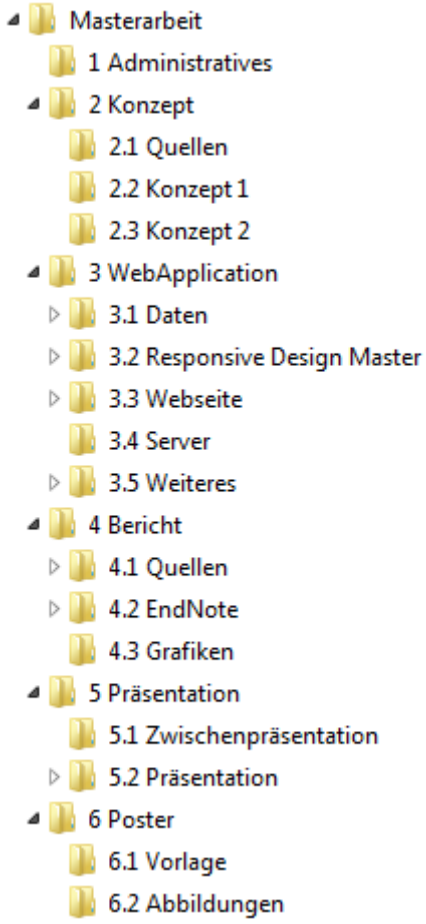
Wright, A. (2009). *Ready for a Web OS?* Communications of the ACM, *52*(12), 16-17.

Zepeda, J. S., & Chapa, S. V. (2007). *From desktop applications towards ajax web applications.* 4th International Conference on Electrical and Electronics Engineering, Mexico City (Mexico).

Anhang

A CD-Inhalt

Die folgende Abbildung zeigt die Daten-Struktur auf der CD.



Daten-Struktur der CD

B Einbau einer zusätzlichen Karte

Die Webapplikation wurde so aufgebaut, dass mit wenigen Anpassungen eine weitere Karte hinzugefügt werden kann. Hier werden die Stellen im Code und im Verzeichnis angegeben, an denen etwas angepasst werden muss. Jede Karte hat einen eindeutigen Kartennamen.

- Die CSS-Datei und das JavaScript-File, welche die Karte, das Layout und die Funktionen erstellen und steuern, müssen in index.html referenziert werden.
- Im Ordner images/mapPreviews muss eine Bilddatei mit der Kartenvorschau abgelegt werden. Das Bildformat beträgt 200x133 Pixel. Der Name der Datei entspricht dem eindeutigen Kartennamen.
- In JavaScript File map.js muss das Switch-Statement der Funktion loadMap() mit einem Fall ergänzt werden. Das Case-Kriterium entspricht dem eindeutigen Kartennamen, die auszuführende Funktion jener Funktion, die das map-Objekt für die neue Karte erstellt, zurückgibt und die neue Karte erstellt.
- In mapSheetIndex.js muss die neue Karte im Array swaiMaps und im Memory-Objekt mapTreeStore eingetragen werden. Dabei ist der eindeutige Kartenname zu verwenden.
- In mapSheetIndex.js muss die Funktion recreateSwaiMapExtents() einmal ausgeführt werden. Damit werden die Extents für die Kartenübersicht aktualisiert (Abschnitt 3.3.2). Um diese Funktion auszuführen, müssen die Kommentar-Zeichen beim Aufruf im Code entfernt und der Prototyp einmal im Browser geöffnet werden.
- Je nach Referenzsystem der Karte muss in page.js die Objekt-Funktion showCoords() mit einem zusätzlichen Fall ergänzt werden, damit die Koordinaten des Mauszeigers / des Kartenmittelpunktes richtig angezeigt werden.
- Die beiden HTML-Dateien help.html und imp_src.html müssen angepasst werden, damit Hilfe und Quellen auch für die neue Karte angezeigt werden können.

C Funktionstest

Bestanden Bestanden aber Verbesserung nötig Nicht bestanden Nicht testbar	Firefox 26	Chrome 31	Safari 5.1.4	IE 11	Android 4.3	Firefox 26	Google Chrome 31	Mobile Safari 7	Chrome Mobile iOS 31	Android 4	Firefox 23	Google Chrome 31
Gerät	HP Laptop			PC	Galaxy SIII			iPad		ASUS Tablet		
Kartenübersicht												
Hilfe anzeigen												
Impressum und Quellen anzeigen												
Kartenübersicht neu laden												
Geografisches Verzeichnis anzeigen												
Baum öffnen / schliessen					12					12		
Politische Karte öffnen												
Politische Karte												
Legende anzeigen, auf- und zuklappen				2								
Namen von Flüssen, Seen, Städten, Staaten anzeigen				3	13	13	13					
Zwei Geometrien zeichnen, bearbeiten und löschen				4		15					15	
Nur Karte in beliebigem Format drucken												
Zoom und Pan				5								
Zurück zu Kartenübersicht												
Kartenübersicht												
Auf einen Extent klicken und Dieselkarte laden					13	13	13					
Dieselkarte												
Absolute Beschriftung einschalten				6								
Regler verschieben				6	14					14		

Entwicklung des Preises anschauen (auf ein Land klicken)				6								
Kartenausschnitt ändern				6								
Home Button anklicken				6								
Geografisches Verzeichnis öffnen und Zürich laden				6								
Zürich												
Brunnen erfassen				7					19			
Attribut von Brunnen abfragen												
Brunnen verschieben und Attribut ändern				7					19			
Brunnen löschen				8					19			
Brunnen innerhalb beliebigem Radius zählen												
Brunnen innerhalb Polygon zählen						16					16	
Brunnenverteilung berechnen												
Transparenz von Resultat-Ebene ändern					14					14		
Zeitintervall ändern				9	14	14	14	14	14	14	14	14
Gerät lokalisieren										20		20
GPS-Punkt löschen										20		20
Karte mit Layout, Titel und Autor drucken				10								
Geografisches Verzeichnis öffnen und Schweiz laden												
Schweiz												
Daten mit Bevölkerungsdichte (Anhang D) einlesen			1	11			17	18	18			17

1. Die Karte benötigt die FileReader API. Diese wird von der Safari-Version 5.1.4 nicht unterstützt. Der Anwender erhält allerdings die Meldung, dass er einen anderen Browser nehmen soll.
2. Legende wird nicht angezeigt.

3. Es ist keine Abfrage möglich. Es passiert nichts, wenn man auf ein Feature klickt.
4. Der Werkzeugkasten wird nicht angezeigt.
5. Mit dem Mausrad kann nur vergrössert werden, egal in welche Richtung man das Rad dreht.
6. Dieselkarte wird im Hintergrund geladen aber der Spinner (Rad, welches anzeigt, dass die Karte am Laden ist) verschwindet nicht.
7. Attribut-Felder werden nicht angezeigt.
8. Meldung kommt, dass Brunnen nicht gelöscht werden kann. Vermutlich Zusammenhang mit Punkt 18. Fehlermeldung: Error: esri.config.defaults.io.proxyUrl is not set.
9. Darstellungsproblem: Die Jahreszahlen werden nicht rotiert.
10. Meldung kommt, dass Karte nicht gedruckt werden kann.
11. Es passiert nichts, wenn man das File einlesen möchte.
12. Teilweise kommt es vor, dass mehrere Baum-Elemente hervorgehoben sind, nachdem man auf mehrere gedrückt hat. Tritt bei beiden getesteten Android-Browsern auf.
13. Popups gehen zum Teil ausserhalb des sichtbaren Bereichs auf, so dass man zuerst die Karte verschieben muss, um den Inhalt des Popups zu sehen.
14. Mit dem Finger kann der Regler nicht gut verschoben werden.
15. Multipunkte, Polylinien und Polygone werden nicht richtig gezeichnet. Die Stützpunkte liegen nicht da, wo man getippt hat.
16. Polygon kann nicht gezeichnet werden. Hängt wahrscheinlich mit Punkt 15 zusammen.
17. Fehlermeldung: „Fehler beim Öffnen der Datei“. Die Datei ist jedoch die gleiche wie für den Firefox und den Android-Browser
18. Es ist nicht möglich, mit dem iPad lokale Dateien anzulegen. Demnach kann diese Funktion nicht getestet werden und die Karte kann auf iPads nicht genutzt werden.
19. Die Daten können nicht bearbeitet werden. Vermutlich Zusammenhang mit Punkt 8. Fehlermeldung: Error: esri.config.defaults.io.proxyUrl is not set.
20. Geolocation wird nicht gefunden, obwohl Zugriff erlaubt wird.

D Datenformat für thematische Karte der Schweiz

Um die thematische Karte der Schweiz nutzen zu können, muss ein lokales File eingelesen werden. Das File muss folgendermassen formatiert sein:

[Autokennzeichen des Kantons],[Zu visualisierende Zahl];

Die zu visualisierende Zahl muss ein Integer, also eine Ganzzahl sein. Dezimalzahlen werden mit der JavaScript-Funktion parseInt() in eine Ganzzahl umgewandelt. Zwischen Autokennzeichen und dieser Zahl ist ein Komma zu setzen. Am Schluss der Zeile ist ein Semikolon. Es dürfen keine Leerzeichen verwendet werden. Hier ist ein Beispiel, wie ein solches File ausschauen könnte. Es zeigt die Bevölkerungsdichte im Jahr 2010[50]. Die Reihenfolge der Kantone ist nicht relevant.

```
ZH,804;
BE,159;
LU,253;
UR,33;
SZ,159;
OW,72;
NW,147;
GL,56;
ZG,468;
FR,145;
SO,325;
BS,5027;
BL,508;
SH,254;
AR,218;
AI,87;
SG,232;
GR,27;
AG,396;
TG,249;
TI,119;
VD,224;
VS,59;
NE,214;
GE,1486;
JU,84;
```

[50] Schweizer Kantone, http://schweizer-kantone-orte.websieb.info (Zugriff 11.01. 2014)

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

E Eigenständigkeitserklärung

Ich erkläre hiermit, dass es sich bei der von mir eingereichten schriftlichen Arbeit mit dem Titel

Der SCHWEIZER WELTATLAS interaktiv als Webapplikation
Ein Prototyp realisiert mit ArcGIS API for JavaScript

um eine von mir selbständig und in eigenen Worten verfasste Originalarbeit handelt.

Verfasserin

Name	*Vorname*
Born	Barbara

Betreuer

Name	*Vorname*
Koblet	Thomas
Philipp	Marty

Leitung

Name	*Vorname*
Hurni	Lorenz

Mit meiner Unterschrift bestätige ich, dass ich über fachübliche Zitierregeln unterrichtet worden bin und das Merkblatt "Zitier-Knigge" gelesen und verstanden habe. Die im betroffenen Fachgebiet üblichen Zitiervorschriften sind eingehalten worden.

Eine Überprüfung der Arbeit auf Plagiate mithilfe elektronischer Hilfsmittel darf vorgenommen werden.

Ort, Datum Unterschrift

Durch die Unterschrift bürge ich für den vollumfänglichen Inhalt der Endversion dieser schriftlichen Arbeit.

ETH-Bibliothek
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

ETH-Bibliothek
ETH Zürich
Rämistrasse 101
8092 Zürich

F Nutzungsbestimmungen für Swisstopo-Geodaten für ETH-Angehörige

1 Allgemein

Analoge und digitale Geodaten der Swisstopo dürfen nur für Unterrichts-und Forschungszwecke innerhalb des ETH-Bereichs (**ETH Zürich, ETH Lausanne, WSL, PSI, EAWAG**, ohne EMPA) verwendet werden. Die Originaldaten müssen nach Abschluss des Forschungsprojektes gelöscht werden.

2 Abgabe an Dritte

Geodaten und geodätische Produkte dürfen nicht an Dritte ausserhalb des ETH-Bereichs abgegeben werden. Bei gemeinsamen Forschungsprojekten mit anderen Hochschulen und Instituten ausserhalb des ETH-Bereichs müssen die Institutionen des ETH-Bereichs vorab eine Bewilligung bei Swisstopo einholen.

3 Reproduktionsbedingungen

Analoge Reproduktion: Ausschnittgrösse: A5-A3 (Originalmassstab), 1:1, vergrössert oder verkleinert, einfarbig oder mehrfarbig
Digitale Reproduktion: Ausschnittgrösse: 2 Mio. Pixel, A5-A3 (Originalmassstab), nur statische Bilder oder PDF, Publikation nur innerhalb Domain ethz.ch, Ausdrucke ab Webseite nur für den Eigenbedarf

4 Bewilligungsvermerk

Analoge Reproduktion: **Reproduziert mit Bewilligung von swisstopo (JA100120)**
Digitale Reproduktion: **© 2013 swisstopo (JD100042)**

5 Belegexemplar

Bei Publikationen ist der Swisstopo ein Belegexemplar bis Ende des laufenden Jahres zuzustellen. Bei Forschungsberichten mit Bezug auf die Qualität und die Nutzung der Swisstopo-Geodaten innert drei Monaten zuzustellen.

Swisstopo, Bundesamt für Landestopografie
Seftigenstrasse 264
Postfach
3084 Wabern

6 Publikationen mit spezieller Bewilligung

Für folgende Fälle ist eine spezielle schriftliche Bewilligung der Swisstopo erforderlich:

- Veröffentlichungen von Produkten von Swisstopo in direkter oder abgeleiteter grafischer analoger Form sowie deren Teile oder Derivate, sofern die Auflage für ausschliesslich wissenschaftliche Veröffentlichungen 500 Exemplare übersteigt.
- Direkte Digitalisierungen (Vektorisierungen) des Karteninhaltes oder Teilen davon, welche nicht reinen Unterrichts- und Forschungszwecken dienen.
- Veröffentlichungen von Produkten oder Diensten im Internet.

7 Weitere Bestimmungen

Im Übrigen gelten die Bestimmungen der Verordnungen des Bundesrates über die Geoinformation und die Landesvermessung vom 21. Mai 2008 (SR 510.620 und 510.626).

Quelle: Jahresbewilligungen Swisstopo 2013