Hochschule Neubrandenburg
University of Applied Sciences

Fachbereich Landschaftsarchitektur, Geoinformatik, Geodäsie und Bauingenieurwesen

Studiengang Geoinformatik

Entwicklung einer WAP-Version von KLEKsOnline

Bachelorarbeit

vorgelegt von: *Sebastian Gritzka*

geboren am: 30.12.1986

Zum Erlangen des akademischen Grades

„Bachelor of Engineering“ (B.Eng.)

urn:nbn:de:gdv:519-thesis2010-0489-4

Erstprüfer: Prof. Dr.-Ing. Andreas Wehrenpfennig
Zweitprüfer: Dr. Maik Stöckmann

Abgabetermin: 09.09.2010

Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Bachelorarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Neubrandenburg, den 09. September 2010

Unterschrift

Danksagung

An dieser Stelle möchte ich mich bei allen bedanken, die mich bei der Erstellung dieser Bachelorarbeit unterstützt haben.

Zum einen bedanke ich mich bei Herrn Prof. Dr.-Ing. Andreas Wehrenpfennig und bei Herrn Dr. Maik Stöckmann dafür, dass Sie die Betreuung dieser Arbeit übernommen haben sowie für die Unterstützung während der Bearbeitungszeit.

Besonderer Dank geht an meine Familie und Freunde, die mich stets ermuntert und unterstützt haben.

Des Weiteren möchte ich mich bei Christina Möller (B.Eng.) und Manuel Prager (B.Eng.) bedanken, die sich zum Korrekturlesen Zeit genommen haben.

Kurzfassung

Das mobile Internet hat in den letzten Jahren stark an Bedeutung gewonnen, was auf die gesunkenen Datentarife und schnellere Übertragungstechniken zurückzuführen ist.

Das Ziel der vorliegenden Bachelorarbeit ist die Entwicklung einer WAP-Version des KLEKsOnline-Viewers (http://embed.kleks-online.de, vgl. auch www.kleks-online.de). Dabei besteht die Aufgabe in der Online-Visualisierung von Inhalten der KLEKs-Datenbank für Handys. Dazu müssen die Ausgaben des derzeitigen KLEKsOnline-Viewers für eine Anwendung auf Mobiltelefonen angepasst werden. Clientseitig ist die Anwendung nach WAP-Standards zu entwickeln, serverseitig eine Lösung mit PHP und dem UMN-MapServer zu konzipieren. Bestehende WMS sollen hierbei genutzt und über die Server-Anwendung gebündelt werden. Die Grafik-Ausgabe und Bedienung muss an die Größe eines üblichen Handy-Displays angepasst sein.

Abstract

In recent years the mobile internet has become increasingly important, due to the lower prices of data rates and faster transmission technologies.

Aim of this bachelor thesis is to develop a WAP version for KLEKsOnline-Viewer (http://embed.kleks-online.de, see also www.kleks-online.de). During to this the task is to set-up the online visualization of the KLEKs database for mobile devices. The current application has to be adapted to mobile phones. At client side the application has to be developed to WAP standard, at server side is a need of using PHP and UMN-MapServer. Existing WMS have to be used and be bundled at the server application. The graphic output and handling have to be adapted to the display size of a usual mobile phone.

Inhaltsverzeichnis

1 Einführung

Das mobile Internet verzeichnet ein exponentielles Wachstum. Aufgrund der zunehmenden Verbreitung von Smartphones, den gesunkenen Datentarifen und der verbesserten mobilen Breitbandverfügbarkeit erobert das mobile Internet den Massenmarkt. Mittlerweile nutzt jeder zehnte Deutsche mit dem Mobiltelefon das Internet. Laut Berechnung des Bundesverbandes für Informationswirtschaft, Telekommunikation und neue Medien e.V. (BITKOM) werden 8,2 Millionen Smartphones im Jahr 2010 verkauft. Das entspricht einem Wachstum von 47 Prozent [BI10]. Zudem wurden in diesem Jahr freigewordene Frequenzen für die mobile Breitbandversorgung versteigert. Diese sollen zum einen der Träger für die nächste Generation des mobilen Internets sein, welche deutlich höhere Übertragungsraten ermöglichen, zum anderen die Lücken in der Breitband-Internet-Versorgung besonders in ländlichen Regionen schließen.

Die erhöhte Nutzung des mobilen Internets hat auch Auswirkungen auf die Entwicklung und Präsentation von Webinhalten. Hier setzt diese Arbeit an. Die bestehende Webanwendung des KulturLandschaftsElementeKataster (KLEKs) Online-Viewer[1] soll für eine geeignete Ausgabe auf dem Mobiltelefon angepasst werden. Dabei sollen bestehende Web Map Services (WMS) genutzt werden ebenso wie die Kommunikation über das Wireless Application Protocol (WAP).

Das zweite Kapitel dieser Arbeit behandelt die theoretischen Grundlagen der zu verwendenden Technologien.

Gegenstand des dritten Kapitels ist die Analyse des KLEKsOnline-Viewer, Besonderheiten bei der Entwicklung von Mobilseiten und bestehende Einflüsse auf die Übertragungstechniken.

Der Entwurf der WAP-Version für KLEKsOnline wird im vierten Kapitel behandelt.

In dem fünften Kapitel wird die Umsetzung der WAP-Version sowohl client- als auch serverseitig beschrieben.

Darauf folgt im sechsten Kapitel die Schilderung der Testphase.

Den Abschluss bilden eine Zusammenfassung und ein Ausblick des Projektes.

[1] http://embed.kleks-online.de, vgl. auch www.kleks-online.de

2 Grundlagen

2.1 Serverseitig

2.1.1 PHP

Serverseitig sollen hinter dieser Anwendung der UMN-MapServer (University of Minnesota) und die Open Source Programmiersprache PHP stehen. Als rekursives Akronym steht PHP zunächst für „PHP:Hypertext-Preprocessor“ und darauf folgend für „Personal Home Page“. Die Sprache ist für die Nutzung auf einem Internetserver gedacht und kann im Scriptstil eingesetzt werden. Dabei wird vor allem auf PHP zurückgegriffen, wenn dynamische Inhalte im Internet dargestellt werden sollen, wie z. B. bei der Interaktion mit dem Nutzer oder zur Abfrage von Datenbanken.

Als serverseitige Scriptsprache wird PHP deshalb bezeichnet, weil eine Anfrage an ein PHP-Script auf einem Internetserver bearbeitet wird. Mit der Anfrage werden XHTML-Seiten von dem Internetserver durch die Interpretation eines PHP-Scripts erstellt und diese dann an den Browser gesendet, der zuvor eine entsprechende Anfrage gestellt hat. Der Browser ist dann fähig, den ihm zurückgelieferten Code darzustellen. Dazu gehören u.a. XHTML (Extensible HyperText Markup Language), XML oder Javascript-Code. Aufgrund dessen, dass der PHP-Code in eine XHTML-Seite eingebettet wird, wird PHP auch als Scriptsprache bezeichnet [Avc03].

2.1.2 Web Map Service

Ein Web Map Service (WMS) ist eine von dem Open Geospatial Consortium (OGC) verabschiedete Spezifikation zur Visualisierung von Geodaten. Aus räumlich referenzierten Daten erstellt dieser Karten mit geographischen Informationen. Dabei definiert ein WMS eine Karte als Abbildung geographischer Informationen ebenso wie von einem Computerbildschirm die Abbildung ein digitales Bild ist. Der Datenbestand dabei ist nicht die Karte selbst. Die durch einen WMS generierte Karte ist üblicherweise gerendert in einem piktographischen Format wie PNG (Portable Network Graphics), GIF (Graphics Interchange Format) oder JPEG (Joint Photographic Experts Group), zum Teil jedoch auch als vektorbasierte graphische Elemente in SVG- (Scalable Vector Graphics) oder WebCGM- (Web Computer Graphics Metafile) Formaten. Dabei spielt es keine Rolle, von welchem Kartenserver die Kartenbilder geliefert werden, der Server muss lediglich einen OGC konformen WMS anbieten.

Ein OGC WMS definiert drei Anfrageoperationen, wobei in diesen mindestens die Operationen *getCapabilities* und *getMap* integriert sein müssen. Optional enthält dieser auch die Operation *getFeatureInfo*.

GetCapabilities dient der Abfrage der Fähigkeiten eines WMS. Als Antwort auf die gestellte Anfrage wird ein wohlgeformtes und gültiges XML-Dokument (Extensible Markup Language) geliefert. Es enthält die unterstützten Ausgabeformate, allgemeine Angaben zum Anbieter des

WMS und die in der Karte enthaltenen Layer sowie die räumlichen Referenzsysteme.

GetMap liefert ein georeferenziertes Rasterbild vom Server als Karte zurück. Die in der Karte vorkommenden Layer, die gewünschte Darstellung der Layer, das Koordinatenbezugssystem, der Kartenausschnitt, die Größe der Karte und das Ausgabeformat können u.a. innerhalb dieser Anfrage spezifiziert werden.

Die Tabelle 1 zeigt eine Übersicht über die gängigsten Parameter einer GetMap-Anfrage. Weitere optionale GetMap-Parameter sind in der OGC 06-042 Web Map Service Implementation Specification zu finden.[2]

Tabelle 1: Übersicht über GetMap-Parameter, eigene Darstellung auf Basis von [dlB04]

Parameter = [Parameterwert]	**Status**	**Beschreibung**
SERVICE = [WMS]	zwingend	Service Typ
VERSION = [version]	zwingend	Version des WMS
REQUEST = [GetMap]	zwingend	Angabe des WMS Abfragetyps
LAYERS = [layer_2list]	zwingend	kommaseparierte Liste der Kartenebenen
STYLES = [style_list]	zwingend	mit einer Zeichenvorschrift kommaseparierte Liste je angefragtem Layer
SRS = [EPSG:code]	zwingend	räumliches Bezugssystem
BBOX = [minx,miny,maxx,maxy]	zwingend	Kartenausschnitt in der Einheit des SRS
WIDTH = [integer]	zwingend	Breite des Kartenbildes in Pixeln
HEIGHT = [integer]	zwingend	Höhe des Kartenbildes in Pixeln
BGCOLOR = [0xRRGGBB]	optional	Hintergrundfarbe der Karte in RGB (hexadezimal)
TRANSPARENT = [TRUE,FALSE]	optional	Transparenz des Hintergrundes der Karte
FORMAT = [output_format]	zwingend	Ausgabeformat des Kartenbildes

Mittels der optionalen getFeatureInfo-Anfrage können festgelegte thematische Informationen der zugrunde liegenden Daten geliefert werden. Anfragen von WMS-Operationen können mittels des Uniform Resource Locators (URLs) über einen Webbrowser aufgerufen werden. Je nach Anfrageoperation ändert sich der Inhalt einer URL. Besonders bei der GetMap-Anfrage sind innerhalb der URL die Karteninformationen sichtbar. Bei dem Aufruf von mehreren Karten mit der gleichen Ausgabegröße und den gleichen geographischen Parametern kann eine zusammengesetzte und exakt überlagerte Karte erzeugt werden. Zudem bleiben die darunterliegenden Karten sichtbar, wenn die Ausgabeformate transparenten Hintergrund unterschützen, wie dies z. B. bei PNG oder GIF der Fall ist. Des Weiteren können mittels WMS von verschiedenen Servern Karten angefordert werden, was die Schaffung eines verteilten Netzwerkes von MapServern ermöglicht und somit auch als eine Aufteilung der Rechenlast auf die Server genutzt werden kann

[2] http://www.opengeospatial.org/standards/wms

[Fis03]. Clients können somit folglich individualisierte Karten erstellen. Für einen WMS, dessen Hauptaufgabe das Erstellen von Karten ist und der nicht seine Fähigkeit in den Zugriff von spezifischem Datenbestand legt, gilt dieser internationale Standard. Dabei klassifiziert ein grundlegender WMS seine geographischen Informationen in Layer und bietet zudem vordefinierte Styles für die Visualisierung an. Es werden nur bezeichnete Layer und Styles unterstützt und es enthält keinen Mechanismus für die benutzerdefinierte Symbolisierung von Daten. [dlB04]

2.2 Clientseitig

2.2.1 Wireless Application Protocol

Clientseitig basiert die Anwendung auf dem Wireless Application Protocol (WAP). Mobile Endgeräte verfügen nur über kleine Display- und Speicherkapazitäten, geringere Rechenleistung und niedrigeren Stromverbrauch bedingt durch die kommerziellen Randbedingungen auf dem Massenmarkt im Mobilfunksektor. Zudem haben Mobilfunknetze eine geringere Bandbreite aufgrund von frequenzbedingten Einschränkungen und der Mobilität der Anwender, geringere Stabilität und Verfügbarkeit als auch größere Kommunikationsverzerrungen bei den Teilnehmerverbindungen.

Für die Bereitstellung von text- und grafikbasierten Informationen und Diensten definiert WAP einen Industriestandard für mobile Endgeräte wie Mobiltelefone, Smartphones und PDAs. Dabei spezifiziert WAP eine Anwendungsumgebung und Kommunikationsprotokolle von mobilen Endgeräten. Die Open Mobile Alliance (OMA) definierte, noch unter dem Namen WAP-Forum, die Hauptziele für die WAP-Standardisierung. Der Internetzugang sollte unabhängig von den Netztechniken bestehender digitaler Mobilfunknetze durch eine globale Protokollarchitektur ermöglicht werden. Zu den Mobilfunknetzen, auf denen das WAP-Protokoll arbeitet, gehören u.a.:

- GSM - Global System for Mobile Communications
- GPRS - General Packet Radio Service
- EDGE - Enhanced Data Rates for GSM Evolution
- UMTS - Universal Mobile Telecommunications System
- HSDPA - High Speed Downlink Packet Access [WAP10]
- zukünftig LTE - Long Term Evolution

Ziel ist es Zugriffstechniken und Anwendungen zu realisieren, die auf die besonderen Eigenschaften von mobilen Geräten inklusive ihrer Benutzer angepasst sind. Des Weiteren wird angestrebt existierende Standards so weit wie möglich zu integrieren und zu erweitern. Zwischen

den Endgeräten der unterschiedlichen Hersteller, deren Betriebssystemen und den Basisdiensten der verschiedenen Mobilfunkstandards soll eine Interoperabilität bestehen. Zudem soll es eine Garantie der geforderten Kommunikationsmerkmale im Bereich von Dienstgüte (Quality of Service), Zuverlässigkeit und Sicherheit geben [Dul05].

Um die Darstellung auf den kleinen Displays zu gewährleisten und gleichzeitig die Menge der übertragenen Daten klein zu halten, behält WAP die offene Form der Auszeichnungssprache (XHTML) bei, übermittelt den Text aber in kompilierter Form an den WAP-Client. Bei der Kommunikation zwischen Webserver und WAP-Client ist ein Proxy zwischengeschaltet - ein sogenannter WAP-Gateway. Dieser übersetzt die von WAP-Client ankommenden binären Anfragen in Klartext an den Webserver. Die Antworten des Servers wiederum werden im WAP-Client kompiliert und im MIME-Typ WMLC (Wireless Markup Language Compiled) an den Client übertragen. Der Gateway übernimmt die Aufgaben, die im Web der Browser ausführt wie u.a. die syntaktische Analyse der WML-/XHTML-Seiten. Zwischen Server und WAP-Gateway wird mithilfe des HyperText Transfer Protocol (HTTP) kommuniziert, zwischen Gateway und WAP-Client via Wireless Session Protocol (WSP) bis WAP 1.2.

Seit WAP 2.0 wird weitestgehend auf Mobilfunk-Spezifika verzichtet. Die Protokolle WSP, WTP (Wireless Transaction Protocol) und WTLS (Wireless Transport Layer Security) wurden durch HTTP, TCP/IP (Transmission Control Protocol/Internet Protocol) und SSL (Secure Socket Layer) ersetzt. So sind mittels WAP 2.0 weitestgehend alle Internetseiten erreichbar. Zudem wurde das Proxy-Konzept gelockert. Auf diese Weise kann der Gateway umgangen werden und folglich ist eine direkte Kommunikation zwischen Client und Server möglich [wik10].

2.2.2 XHTML Mobile Profile

Bei der Gestaltung einer Webseite für den mobilen Bereich sollte die erste Überlegung sein, ob es von Vorteil ist den bestehenden Auftritt anzupassen oder zu der bestehenden Webseite eine parallele Seite zu pflegen. Zu Ersterem Fall eignet sich XHTML Basic, sofern für den Entwickler ein- und dieselbe Auszeichnungssprache für das Desktop- und Mobil-Layout in Frage kommt. Für den zweiten Fall einer parallelen Seite empfiehlt sich XHTML Mobile Profile (XHTML-MP), eine speziell für mobile Endgeräte entwickelte Auszeichnungssprache. Diese Sprache hat die Nachfolge von WML (Wireless Markup Language) angetreten und ist Bestandteil des WAP 2.0. XHTML-MP ist fast identisch mit dem W3C-Standard (World Wide Web Consortium) XHTML Basic und wurde durch das Standardisierungsgremium Open Mobile Alliance entworfen. Die Sprache ergänzt XHTML Basic um zusätzliche Präsentationselemente und bietet Unterstützung für interne Stylesheets. Funktionen von cHTML (compact HTML) und WML wie acronym, address, br, b, big, hr, i, small, dl, fieldset und optgroup wurden implementiert [the07]. Als Untermenge von XHTML ist sie die am meisten unterstützte Sprache auf modernen mobilen Endgeräten. Einige komplexere Bestandteile wurden entfernt, jedoch funktioniert

die Struktur der Dokumente, Metadaten, Stylesheet-Einbindung, Links, Listen, Bilder und Multimediaobjekte wie gewohnt. Tabellen und Formulare sind hingegen nur in stark vereinfachter Form zulässig und Frames fallen komplett weg. Seit der Version 1.1 von XHTML MP ist auch die JavaScript-Einbindung möglich.

Bei der Verwendung von XHTML-MP wie auch bei anderen Auszeichnungssprachen empfiehlt es sich, die Dokumente mit einem XML-MIME-Typ (Multipurpose Internet Mail Extensions) auszuliefern. Inklusive des Internet Explorer Mobile rendern die gängigen Browser solche gekennzeichneten Dokumente.

Des Weiteren gibt es auch abgespeckte Standards für CSS (Cascading Style Sheets). Der Entwurf von der Open Mobile Alliance heißt Wireless CSS (WCSS), ursprünglich WAP CSS und ließ Sprach- und Druckausgabe sowie einige fortgeschrittene Positionierungsangaben weg. Die gängigsten CSS-Eigenschaften blieben erhalten. Der Gegenentwurf des W3C ist das CSS Mobile Profile (CSS MP), enthält jedoch keine wesentlichen Unterschiede zum Wireless CSS [Bra09].

2.2.3 JavaScript

HTML bzw. XHTML - im Folgenden (X)HTML genannt - sind häufig nicht alleiniger Bestandteil einer Webseite, da Webseiten dynamisch gestaltet werden sollen oder auf Nutzeraktionen reagiert werden soll. Clientseitig findet ebenso die Programmiersprache JavaScript Anwendung. Diese wurde geschaffen, um (X)HTML-Autoren die Möglichkeit zu geben, Webseiten optimieren zu können. Dabei wird der Code zum Client übertragen und lokal ausgeführt. Dies spart Übertragungskapazität und ermöglicht ein schnelleres Ausführen von Aktionen. Zudem können Ausgaben ohne oder mit reduzierter Datenübertragung generiert werden.

JavaScript wurde ursprünglich von Netscape entwickelt und lizenziert. Später kam SUN hinzu. Die Sprache dient der clientseitigen Programmierung und wurde an Java angelehnt. Als Mehrzwecksprache dient sie ebenso der Programmierung von Algorithmen und Ereignisbehandlung. Des Weiteren können Nutzereingaben unabhängig vom Server verarbeitet werden [Weh08]. JavaScripts können direkt oder als externe Datei in eine (X)HTML-Datei integriert werden. Diese werden vom Browser während der Laufzeit mittels einer adäquaten Interpreter-Software interpretiert. Dabei gibt es jedoch Unterschiede zwischen den verschiedenen Browsern.

JavaScript wurde unter dem Namen ECMAScript bzw. ECMA-262 durch die European Computer Manufacturers Association (ECMA) als Industriestandard deklariert. Sie wird von den meisten Browsern unterstützt [jav07].

Für XHTML MP gibt es ebenfalls eine clientseitige Scriptsprache. Diese heißt ECMAScript Mobile Profile. Jedoch ähnelt es JavaScript so sehr, dass ECMASript Mobile Profile keine Rolle in der Praxis spielt [Bra09].

2.2.4 MIME-Typ

Multipurpose Internet Mail Extensions (MIME) war ursprünglich für E-Mails mit Anhang gedacht. Die dabei zu übertragenden Daten waren alle in einer Datei vorhanden. Um die Daten voneinander trennen zu können, wurde ein Schema entwickelt, welches sicher stellt, dass die einzelnen Datentypen unterschieden werden können. Dabei wurde der interpretierenden Software mitgeteilt, um welchen Datentyp es sich bei welchem Teil der E-Mail handelt. Dieses Schema findet seine Anwendung aber nicht nur bei E-Mails. Bei der Kommunikation zwischen entfernten Programmen, wie zwischen Browser und Webserver, geht es auch um die Art der zu übertragenden Daten. Hier hat sich im Internet das Schema der MIME-Typen durchgesetzt. MIME-Typen sind folglich festgelegte Dateitypen mit festgelegten Extensions, d.h. Dateiendungen. Damit dieser Mechanismus funktioniert, müssen MIME-Typen bei dem Sender und bei dem Empfänger bekannt sein und weiter verarbeitet werden können. So führen alle Web-Browser und -server eine Liste der ihnen bekannten MIME-Typen. Die gängigen Web-Browser akzeptieren in der Regel jeden MIME-Typ. Ist dem nicht so, wird dem Nutzer angeboten, die empfangenen Daten als Download zu speichern. Bei Webservern ist dies kritischer. Unbekannte MIME-Typen werden nicht verarbeitet. Um dem entgegen zu wirken, sollten entweder nur möglichst stark verbreitete Dateitypen verwendet werden oder es ist dafür Sorge zu tragen, dass unbekannte MIME-Typen in der Konfigurationsdatei des Webservers notiert werden. Das MIME-Typ-Schema besteht aus der Angabe des Medientyps, getrennt durch einen Schrägstrich von der Angabe des Subtyps [MIM07]. Beispiele hierfür sind:

- text/html - für HTML
- application/vnd.wap.xhtml+xml - für XHTML MP

Folgende Medientypen gibt es:

- application - für Dateien, die an ein bestimmtes Programm gebunden sind
- audio - für Audiodateien
- image - für Grafikdateien
- message - für Nachrichten
- model - für Dateien, die mehrdimensionale Strukturen repräsentieren
- multipart - für mehrteilige Daten
- text - für Textdateien
- video - für Videodateien

Eine ausführliche Liste für entsprechende Subtypen ist zu finden unter:
http://www.iana.org/assignments/media-types/

2.3 KulturLandschaftsElementeKataster (KLEKs)

KLEKs ist ein Geoinformationssystem zur Digitalisierung von Kultur- und Landschaftselementen. Das KulturLandschaftsElementeKataster ähnelt dabei der Plattform Wikipedia und zeichnet sich ebenso durch das Gemeinschaftsprinzip aus. Nutzbar für Forschung, Planung, Bildung und Tourismus enthält es drei Informationsebenen, die miteinander verknüpft sind. Dies sind zum Ersten die Fachdaten, zum Zweiten die allgemein beschreibenden Texte und zum Dritten die Mediendatenbank, welche u.a. Fotos (Geo-Images) und Audios enthält.

Die Informationen dabei stammen von verschiedenen Quellen wie z. B. aus den öffentlichen Denkmallisten, von Heimatforschern, Vereinen, Landschaftsplanungsbüros und engagierten Bürgern. Zu dem Datenbestand gehören u.a. Gebäude, Starkbäume, Rad- und Wanderwege, alte Verkehrswege sowie historische Stätten [PDHB08].

3 Analyse

3.1 Bestandteile des KLEKs

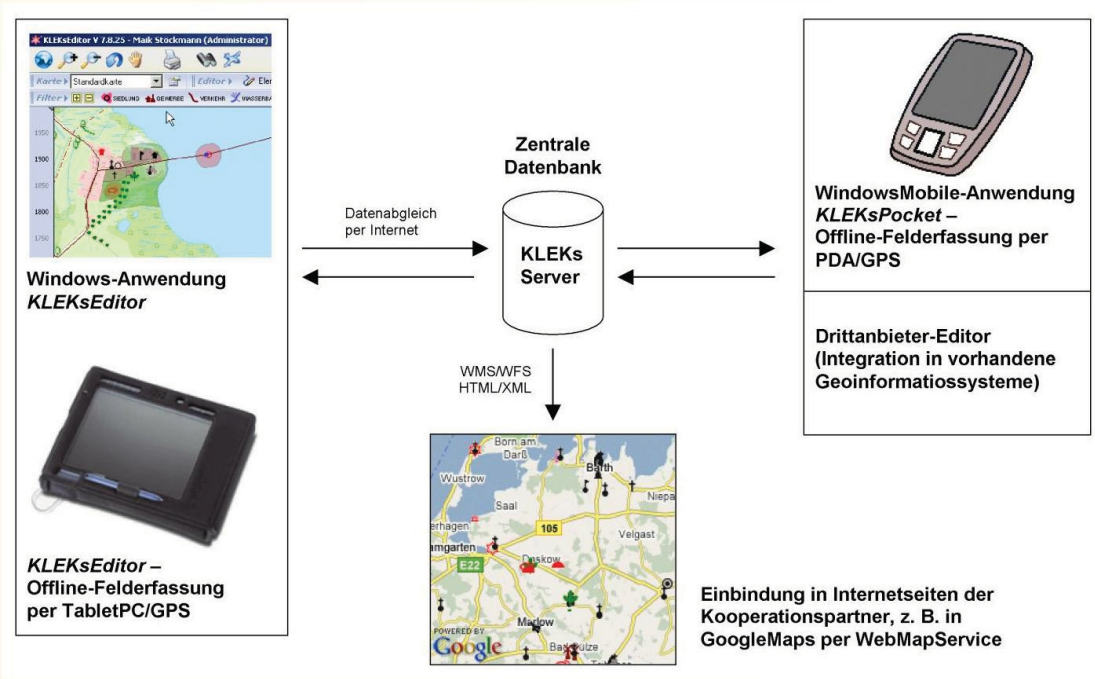

Abbildung 1: KLEKs-Anwendung [PDHB08]

Das KLEKs-System, dargestellt in Abbildung 1 besteht aus einer zentralen Datenbank, sowie den Ein- und Ausgabe-Komponenten. Bestandteil der Eingabe-Komponenten sind der KLEKs-Editor[3] und die Server-Schnittstellen. Der KLEKsEditor ist eine kostenlose Windows-Anwendung, die speziell für dieses Projekt entwickelt wurde und jedem Nutzer zur Verfügung steht. Dabei ist der Editor ein Desktop-Geoinformationssystem (GIS) und verwaltet eine lokale Kopie der Server-Datenbank, welche über das Internet automatisch mit dem zentralen Server synchronisiert wird. Die Landkarten-Module KLEKsOnline und KLEKsAktiv[4] sowie die Server-Schnittstellen gehören zu den Ausgabe-Komponenten. Dabei ermöglichen die Server-Schnittstellen die Verknüpfung der KLEKs-Datenbank mit Geoinformationssystemen oder mit Internetseiten von Kooperationspartnern[5]. Des Weiteren können stark komprimierte Datensätze über die Schnittstellen übertragen werden und zudem stehen Schnittstellen nach internationalen Standards zur Verfügung wie WMS (Web Map Service) und XML (eXtensible Markup Language). Dadurch ist es möglich, die Kulturlandschaftselemente über den WMS des KLEKs-Servers mit anderen Kartenanwendungen zu kombinieren. Beispiele hierfür sind z. B. Google Maps,

[3] http://www.kleks-online.de/
[4] http://www.kleks-online.de/
[5] http://www.touristikkarte.de/

OpenLayers und OpenStreetMap, wobei Letzteres die Grundkarte von KLEKsOnline bildet [PDHB08].

3.2 Mobiltauglichkeit des KLEKsOnline-Viewer

Im Folgenden wurde der KLEKsOnline-Viewer mit einem Mobiltelefon und -browser aufgerufen. Dies diente zum einem dazu, die Darstellung zu testen, Probleme zu erkennen bzw. Besonderheiten bei der Entwicklung der WAP-Version auszumachen, zum anderen die Dauer des Seitenaufbaus festzustellen, um ein Gefühl zu entwickeln, wie stark die bestehende Desktop-Version verschlankt werden muss. Die folgende Tabelle beinhaltet Eigenschaften und Technologien des bei diesem Test verwendeten Mobiltelefons.

Tabelle 2: Eigenschaften des Testhandys

Modell	**Sony Ericsson K800i**
Displaygröße [Auflösung B x H]	240 x 320 Pixel
Netzstandards	UMTS, GSM
Frequenz	UMTS: SingleBand 2100MHz GSM: TriBand 900, 1800, 1900 MHz
Datenübertragung: Upload max. / Download max.	UMTS: 64 /384 KBit/s GPRS: 28,8 / 57,6 KBit/s HSCSD: kA / 115,2 KBit/s
Darstellbare Bildformate	BMP, GIF, JPEG, PNG
WAP Version	2.0
Browser	NetFront 3.3, Opera Mini 5.1

Bei diesem Test wurden zwei mobile Browser verwendet: Zum einen der vorinstallierte Browser NetFront in der Version 3.3 und zum anderen der Browser Opera Mini 5.1.

Bei NetFront 3.3 zeigten sich eklatante Schwächen. Keine einzige Grafik konnte dargestellt werden wie in Abbildung 2 zu sehen ist, was eine weitere Verwendung sinnlos machte. Aus diesem Grund wurde an dieser Stelle der Test abgebrochen.

Opera Mini 5.1 ermöglichte eine genauere Betrachtung. In der ersten Darstellung sorgt der Browser dafür, dass die Karte lediglich ein Viertel der Bildschirmgröße einnimmt und somit optisch so gut wie nichts zu erkennen. Die untere Menüleiste ist zu breit gestreckt. Diese nimmt der Browser als Maßstab für die maximale Breite der Seite, weshalb die Karte verschwindend klein wird. Jedoch bietet der Browser eine Vergrößerungsfunktion, mit der in die Anwendung hinein gezoomt werden kann. Dies erfolgt sehr schnell. In der jetzigen Ansicht (Abbildung 3) nimmt die Karte inklusive der Navigationspfeile die gesamte Displaybreite ein. Allerdings wird die halbe Karte von dem Hinweis für die Benutzung der Desktop-Anwendung verdeckt. Die Buttons und Navigationspfeile sind gut sichtbar und treffsicher mit dem Cursor des Mobiltelefons anzuzielen. Die Karte ist jedoch nicht vollständig. Besonders bei dem Zoomen fällt auf, dass

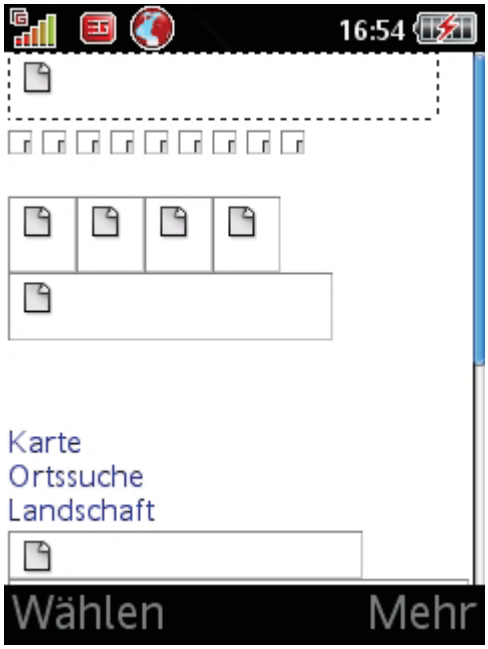

Abbildung 2: Startansicht KLEKsOnline-Viewer in NetFront 3.3

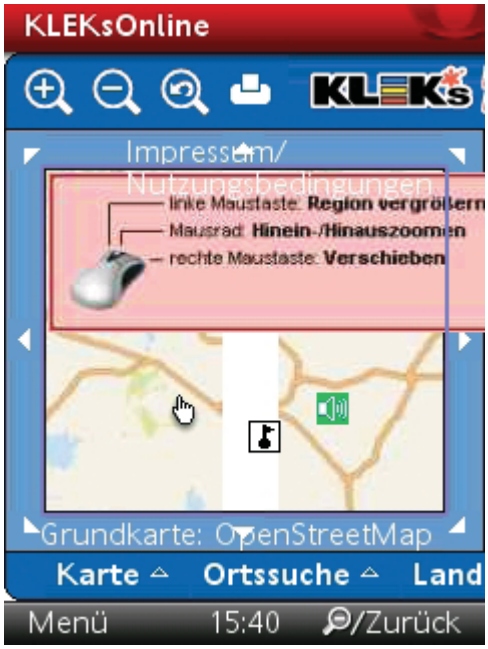

Abbildung 3: Startansicht KLEKsOnline-Viewer in Opera Mini 5.1

sich weiße Streifen innerhalb der Karte befinden und diese einen Teil der Karteninformationen verschwinden lassen. Symbole innerhalb der Karte sind gut sichtbar. Das sich hinter den Symbolen etwas verbirgt, lässt sich aber nur erahnen. Bei dem Bewegen des Cursors auf ein Symbol findet allerdings keine Aktion statt, sodass dem Nutzer eine intuitive Bedienung nicht ermöglicht wird. Bei dem Klicken auf ein Symbol wird die Seite neu geladen und es wird eine Grafik über die vorhergehende Ansicht gelegt, wie dies in der Abbildung 4 zu sehen ist. Diese Grafik kann Text oder ein Bild mit Text enthalten. Jedoch verschwindet diese des Öfteren außerhalb der Karte. Geschieht dies am linken Bildrand, so ist ein Teil der Information verschwunden, da ein Scrollen nach links nicht möglich ist. Bei dem Anklicken von Audio-Symbolen wird die Seite lediglich mit der aktuellen Ansicht neu geladen. Eine Information, ein Abspiel-Button oder ein Klang sind weder zu sehen noch zu hören. Bei wiederholtem Klicken auf das Audio-Symbol erscheint eine Grafik mit dem Hinweis, dass der Adobe Flash Player nicht installiert ist. Flash ist zur Zeit nur auf wenigen Smartphones neuerer Generation verwendbar.

Der Browser erlaubt Multitasking mit Tabs. Bei dem Test wurde ausschließlich der KLEKs-Online-Viewer aufgerufen.

Abbildung 4: Kulturelement im KLEKsOnline-Viewer in Opera Mini 5.1

3.3 Mobile Web Best Practices

Im Zuge der stärkeren Verbreitung von nativen Applikationen und Webseiten für Mobilgeräte wurden Praktiken und Empfehlungen für die Entwicklung des mobilen Webs veröffentlicht.

Darunter die „*Mobile Web Best Practices*“[6] des W3C und die „*Global Authoring Practices for the Mobile Web*“[7] von *Luca Passani*. Diese sollen bei der Entwicklung von Applikationen für mobile Geräte helfen, um vor allem dem mobilen Nutzer ein homogenes Niveau in der „User Experience“ gewährleisten zu können. Damit ist ein ausgeglichenes Verhältnis von Funktionalität, Design und Interaktion für das Benutzungserlebnis mit dem jeweiligen Gegenstand gemeint.

Generell gilt, was bei der Entwicklung einer Anwendung für das Web beachtet werden muss, sollte auch für das mobile Web gelten. Im Folgenden wird ein Ausschnitt der *Mobile Web Best Practices* des W3C gegeben [Alb08][RM08]

ACCESS KEYS: Bei Verfügbarkeit der Funktion der Access Keys können Links und Funktionalitäten, die häufig benutzt werden, mit den Access Keys verknüpft werden, um einen schnelleren Zugriff dem Benutzer zu ermöglichen. So könnten die Zoom- oder Scrollfunktionen mit Access Keys verknüpft werden um dem Benutzer eine einfachere Bedienung zu ermöglichen.

AUTO REFRESH: Das automatische Neuladen von Seiten ist zu unterbinden; mit Ausnahme, dass der Benutzer darüber informiert ist und es eine Abstellmöglichkeit des Neuladens gibt.

CACHING: Um Ladezeiten zu verringern, ist es von Vorteil Caching zu nutzen. Dabei sollte in den HTTP Responses angegeben werden, welche Elemente gecached werden können. Diesbezüglich bedarf es der angemessenen Angabe von Cache-Expiration-Daten.

CAPABILITIES: Bei der Entwicklung sollte nicht der kleinste gemeinsame Nenner der Fähigkeiten aller Mobiltelefone gewählt werden. Dies könnte sonst zur Folge haben, dass viele Fähigkeiten den Nutzern vorenthalten werden, weil eine kleine Anzahl an Mobiltelefonen zur optimalen Darstellung nicht fähig ist, ein Großteil dies aber problemlos bewältigt.

CHARACTER ENCODING USE: Es muss angegeben werden, in welchem Encoding die Inhalte ausgegeben werden.

COLOR CONTRAST: Ein Mobiltelefon hat nicht immer ein gutes kontrastreiches Display und es wird nicht immer unter idealen Lichtbedingungen genutzt. Daher sollten konstrastreiche Bilder verwendet werden.

CONTENT FORMAT SUPPORT: An ein Gerät sollten nur solche Formate gesendet werden, die es auch versteht.

[6] http://www.w3.org/TR/mobile-bp/
[7] http://www.passani.it/gap/

CONTENT FORMAT PREFERRED: Die bevorzugten Formate des Gerätes sollten gesendet werden. Mittels des HTTP Accept Headers, des HTTP User-Agent Headers, UAProf oder WURFL können Informationen über geeignete Formate ausgelesen werden. Die Richtigkeit dieser Angaben ist jedoch nicht zu 100 Prozent gegeben. So kann es vorkommen, dass einige Geräte keinen HTTP Accept Header senden, andere falsche Fähigkeiten angeben oder User-Agents ein Gerät nicht eindeutig identifizieren. Auch können UAProf-Informationen unvöllständig sein. Weiterführende Informationen zu UAProf und WURFL sind in den Kapiteln 3.4.1 und 3.4.2 zu finden.

CONTROL LABELLING: Bedienelemente müssen deutlich gekennzeichnet sein.

ERROR MESSAGES: Fehler sind gänzlich nicht zu vermeiden. Daher ist es ratsam, Fehlermeldungen aussagekräftig zu erklären und einen Link anzubieten, der den Nutzer zu der gewünschten Seite weiterleitet, da Mobilbrowser häufig keinen schnell zu findenden „Zurück“-Button besitzen.

EXTERNAL RESOURCES: Externe Ressourcen sollten so wenig wie möglich eingebunden werden. Jedes Objekt hat sonst eine weitere Abfrage zur Folge, was die Ladezeit verlängern kann.

GRAPHICS FOR SPACING: Das Verwenden von pixelgroßen Grafiken für die Positionierung von Elementen ist zu unterlassen. Dies funktioniert auf mobilen Geräten häufig nicht.

IMAGE RESIZING: Nicht auf dem Client, sondern auf dem Server sollte die Größe der Bilder verändert werden um Ladezeiten einzusparen. Außerdem wird auch die Batterie des Gerätes geschont, wenn die Bearbeitungszeit wegfällt.

IMAGE SPECIFY SIZE: Die Größe eines Bildes sollte im Markup angegeben werden. Nach dem Herunterladen muss der Browser damit eine Seite nicht neu rendern.

LARGE GRAPHICS: Um Bandbreite zu sparen empfiehlt es sich, auf große Grafiken mit hoher Auflösung und vielen Farben zu verzichten. Zudem können auch manche Geräte solche Grafiken nicht rendern.

LINK TARGET ID: Damit der Benutzer Zeit und Kosten sparen kann, ist es wichtig, dass dieser weiß, was sich hinter einem Link verbirgt. Dies muss klar und deutlich signalisiert werden.

MEASURES: Mit Ausnahme von Grafiken, sollten bei allen Elementen keine absoluten, sondern relative Maßangaben gesetzt werden, damit der Browser den Inhalt der Seite an die Displaygröße anpassen kann.

MINIMIZE: Um die Ladezeit zu verkürzen, sollte der Code optimiert werden. Dazu zählt u.a. die Verwendung von CSS und die Entfernung von Leerzeichen, Zeilenumbrüchen und Kommentaren.

NAVBAR: Im Kopf der Seite sollte nur eine minimale Navigation vorliegen. Es empfiehlt sich, die wichtigsten Links auf einer Linie zu platzieren. Zudem sollte der Inhalt einer Seite nach dem Laden zu sehen sein, ohne dass gescrollt werden muss.

NO FRAMES: Frames werden von den meisten Mobilbrowsern nicht unterstützt, so dass auf diese verzichtet werden sollte.

OBJECTS OR SCRIPT: Einige mobile Endgeräte unterstützen keine Verwendung von Scripten. Scripte sollten nur verwendet werden, wenn es keine andere Möglichkeit zur Realisierung gibt. Zudem sind verbrauchen Scripte viel Strom.

PAGE SIZE LIMIT: Mobile Geräte haben üblicherweise Einschränkungen für die maximale Größe einer Seite. Diese darf nicht überschritten werden. Das W3C empfiehlt eine maximale Seitengröße von 20 Kilobyte.

PAGE SIZE USABLE: Um die Ladezeit zu verkürzen, sollten große Seiten in kleinere aufgeteilt werden. Zudem erspart dies dem Nutzer einiges Scrollen, was durchaus auf manchen Geräten eine knifflige Angelegenheit sein kann.

PAGE TITEL: Der Titel sollte kurz und aussagekräftig sein. Zwar zeigen einige Mobilbrowser den Seitentitel nicht an, jedoch wird dieser für die Erstellung von Lesezeichen benötigt.

POP UPS: Pop ups und weitere Fenster dürfen nur geöffnet werden, wenn der Nutzer zugestimmt hat.

REDIRECTION: Redirects sollten nur über Server Redirects ablaufen, nicht über das Markup z. B. durch Meta-Tags. Bei einem Markup-Redirect wird der Befehl erst dann ausgeführt, wenn die Seite schon einmal geladen worden ist. Dies hätte andernfalls zur Folge, dass die Seite zwei Mal geladen werden muss.

SCROLLING: Gescrollt werden sollte immer nur in eine Richtung. Erfordert ein Objekt jedoch ein weiteres Scrollen, sollte dies nicht auch zwangsweise für andere Objekte gelten, damit diese ohne ein weiteres Scrollen lesbar bzw. sichtbar bleiben.

STYLE SHEET USE: Sofern die Geräte Stylesheets unterstützen, ist es ratsam diese für die Gestaltung des Layouts zu verwenden. Es ist jedoch zu beachten, dass die Unterstützung von Stylesheets je nach Gerät variiert. Werden diese nicht unterstützt, müssen Inhalte trotzdem gut lesbar sein.

TABLES SUPPORT: Tabellen werden nicht von allen Mobilbrowsern unterstützt, zudem funktionieren sie auf den kleinen Displays häufig nicht einwandfrei. D.h. Tabellen sollten nur dann verwendet werden, wenn das Gerät diese auch unterstützt. Dies gilt auch für ineinander verschachtelte Tabellen. Des Weiteren ist es ratsam, Tabellen nicht für das Layout zu verwenden.

TESTING: Es gilt, je mehr Geräte getestet werden, desto aussagefähiger sind die Ergebnisse. Dabei ist es wichtig, dass nicht nur Emulatoren zum Einsatz kommen, sondern auch reale Geräte. Ebenso empfiehlt es sich verschiedene Funktionen bei dem Testen zu deaktivieren.

URIS: Das Tippen von URIs ist nicht nur mühselig für den Nutzer, sondern auch fehlerträchtig. Daher sollen URIs so kurz wie möglich gehalten werden.

VALID MARKUP: Die Syntax der erstellten Dokumente muss korrekt sein. Bei ungültigem Markup können unerwartete und unvollständige Ergebnisse entstehen.

Um die Funktionalität der Mobilversion von KLEKsOnline zu gewährleisten, müssen einige der oben genannten Punkte missachtet werden. So ist die Verwendung von JavaScript u.a. unabdingbar für die Realisierung der Zoomfunktionen. Zudem wird die Karte als Bild ausgeliefert, was vor allem die Seitengröße erheblich beeinflusst und vergrößert.

3.4 Möglichkeiten der Geräteerkennung

Ein Produkt zeichnet sich durch seine Eigenschaften und Fähigkeiten aus. Um sich gegen die Konkurrenz durchzusetzen, wird versucht, kontinuierlich die Produkteigenschaften zu verbessern oder sich auf bestimmte Elemente zu konzentrieren und somit den Kunden über den Preis zu gewinnen. Nicht anders sieht es in der Mobilfunkbranche aus. Aufgrund dessen gibt es viele verschiedene Geräte mit unterschiedlichsten Funktionen. Je nach Art des Telefons besitzen die Geräte einen Touchscreen, eine QWERTZ- oder 12er Tastatur. Einige Geräte können fast

alle Audioformate abspielen, andere nur ganz wenige. Manche Geräte können Millionen Farben darstellen, andere nur wenige Tausend. Dies sind nur einige wenige Beispiele für die unterschiedlichen Funktionalitäten.

Bei der Entwicklung einer Applikation für Mobilgeräte sollte in der Regel nicht der kleinste gemeinsame Nenner den Ausschlag für den inhaltlichen Kern geben. Nur weil z.B. einige wenige Mobiltelefone keine Videos abspielen können, ein Großteil hingegen schon, ist es nicht zwangsläufig sinnvoll, auf deren Einbettung zu verzichten. So würde einem Großteil der Nutzer das Benutzungserlebnis vorenthalten werden, nur weil ein paar wenige Telefonmodelle die Videoformate nicht verstehen. Dem entsprechend ist es vorteilhaft, je nach Eigenschaften und Fähigkeiten der Mobilgeräte, angepasste Inhalte zu präsentieren. Im Web fällt dies nicht allzu schwer, da es nur wenig stark verbreitete Browser gibt und die Rechner keine allzu großen Unterschiede aufweisen. Im Mobilfunksektor gibt es hingegen zahlreiche verschiedenartige Geräte mit unterschiedlichsten Fähigkeiten und vielen verschiedenen Browsern. Um dem entgegenzuwirken, gibt es einige Ansätze zur Erkennung der Geräte und deren Eigenschaften.

In den folgenden Abschnitten werden einige dieser Ansätze vorgestellt. Darunter das User Agent Profile, das Wireless Universal Resource File und der DeviceAtlas von der dotMobi mTLD Top Level Domain Limited.

3.4.1 User Agent Profile

User Agent Profile, kurz UAProf, ist ein Datenprofil mit den Eigenschaften von Mobilgeräten. UAProfs werden von den Geräteherstellern bereitgestellt. Diese UAProf-Datenblätter befinden sich unter der im optionalen http-Anfrageheader http-X-WAP-Profile angegebenen URL. Dabei handelt es sich um XML-Dateien, die dem Resource Description Framework (RDF) Standard folgen. Je nach Umfang der Herstellerangaben sind die Dateien unterteilt in verschiedene Angaben zu: HardwarePlatform, SoftwarePlatform, NetworkCharacteristics, BrowserUA, WapCharacteristics, PushCharacteristics, MMSCharacteristics, Streaming und weiteren. Einige dieser Angaben werden auszugsweise im Folgenden für das Sony Ericsson K800i vorgestellt.[8]

Wichtige Informationen für den Entwickler sind dabei z. B. Angaben zum Display - wie Größe, Auflösung, Farbdarstellung und Standardzeichengröße - und die Art der Tastatur des Telefons:

```
<prf:component>
<rdf:Description rdf:ID="HardwarePlatform">
<prf:Vendor>Sony Ericsson Mobile Communications</prf:Vendor>
<prf:Model>K800i</prf:Model>
<prf:ScreenSize>240x320</prf:ScreenSize>
<prf:ColorCapable>Yes</prf:ColorCapable>
```

[8] http://wap.sonyericsson.com/UAprof/K800iR101.xml

```
<prf:BitsPerPixel>18</prf:BitsPerPixel>
<prf:PixelAspectRatio>1x1</prf:PixelAspectRatio>
<prf:ImageCapable>Yes</prf:ImageCapable>
<prf:ScreenSizeChar>17x16</prf:ScreenSizeChar>
...
<prf:Keyboard>PhoneKeypad</prf:Keyboard>
...
</rdf:Description>
</prf:component>
```

Verständliche MIME-Typen der Geräte listet der UAProf-Eintrag CCPPACCEPT (Composite Capabilities/Preference Profiles ACCEPT) auf. Im Gegensatz zum Standard-HTTP-Header HTTP_ACCEPT gibt dieser nach einigen MIME-Typen keine Wildcard */* aus, sondern listet diese weiter auf:

```
<prf:component>
<rdf:Description rdf:ID="SoftwarePlatform">
...
<prf:CcppAccept>
<rdf:Bag>
<rdf:li>image/gif</rdf:li>
<rdf:li>image/jpeg</rdf:li>
<rdf:li>image/vnd.wap.wbmp</rdf:li>
<rdf:li>image/bmp</rdf:li>
<rdf:li>image/png</rdf:li>
<rdf:li>image/cvg</rdf:li>
<rdf:li>image/svg+xml</rdf:li>
<rdf:li>text/x-imelody</rdf:li>
<rdf:li>text/x-emelody</rdf:li>
...
<rdf:li>application/vnd.wap.connectivity-wbxml</rdf:li>
<rdf:li>application/x-x509-ca-cert</rdf:li>
<rdf:li>application/x-www-form-urlencoded</rdf:li>
</rdf:Bag>
</prf:CcppAccept>
```

Es lassen sich ebenso Informationen über die Datenübertragungstechniken finden, wie über den User Agent und seine Fähigkeiten, darunter z. B. JavaScript und Java-Applets:

```
<prf:component>
```

```
<rdf:Description rdf:ID="BrowserUA">
<prf:BrowserName>NetFront</prf:BrowserName>
<prf:BrowserVersion>3.3</prf:BrowserVersion>
...
<prf:FramesCapable>No</prf:FramesCapable>
<prf:TablesCapable>Yes</prf:TablesCapable>
<prf:PreferenceForFrames>No</prf:PreferenceForFrames>
<prf:JavaAppletEnabled>No</prf:JavaAppletEnabled>
<prf:JavaScriptEnabled>Yes</prf:JavaScriptEnabled>
</rdf:Description>
</prf:component>
```

Als nächstes werden die WAP-Fähigkeiten des Gerätes beschrieben:

```
<rdf:Description rdf:ID="WapCharacteristics">
<!-- General -->
<prf:WapDeviceClass>C</prf:WapDeviceClass>
<prf:WapVersion>2.0</prf:WapVersion>
<!-- WML -->
<prf:WmlVersion>
<rdf:Bag>
<rdf:li>1.1</rdf:li>
<rdf:li>1.2</rdf:li>
<rdf:li>1.3</rdf:li>
</rdf:Bag>
</prf:WmlVersion>
<prf:WmlDeckSize>45000</prf:WmlDeckSize>
...
```

Weiterhin folgen Informationen zu PushCharacteristics, MMSCharacteristics und Streaming, auf die an dieser Stelle jedoch verzichtet werden soll, da diese keine weiteren wichtigen Angaben für die hier behandelte Thematik enthalten.

Mit UAProf lassen sich eine Vielzahl wichtiger und aktueller Informationen zu einem Gerät gewinnen, die z.T. erscheinen, bevor neue Geräte verfügbar sind. Jedoch keine Gewährleistung für die Richtigkeit und Vollständigkeit der Herstellerangaben.

In dem Beispiel des K800i liegen über UAProf keine Informationen zur Unterstützung von XHTML oder der Version von JavaScript vor. Darüber hinaus kann es vorkommen, dass es für ein Gerät kein UAProf gibt oder die entsprechende URL nicht korrekt ist. Außerdem fehlt ein einheitlicher Standard für die UAProf-Einträge. Nachteilig an UAProf ist zusätzlich die Tatsache, dass die Daten auf verschiedenen Servern liegen [Alb08].

3.4.2 Wireless Universal Resource File

Bei dem Wireless Universal Resource File (WURFL[9]), hingegen werden alle Informationen zu Mobilgeräten in einer XML-Konfigurationsdatei gesammelt. Im Gegensatz zu UAProf, bei dem die Daten von den Herstellern kommen, werden bei WURFL die Daten von Nutzern und Entwicklern zusammengetragen.

Die einzelnen Einträge der Geräte (<device>) sind in Gruppen (<group>) gegliedert. Die Gruppen listen die Eigenschaften (<capability>) der Mobilgeräte auf. Für das Sony Ericsson K800i sieht der Eintrag wie folgt aus:

```
<device id="sonyericsson_k800i_ver1" user_agent="SonyEricssonK800i"
fall_back="sonyericsson_k800_ver1" actual_device_root="true">
```

Für den Fall, dass keine Beschreibung eines Gerätes vorliegt, wird über das Attribut fall_back auf die Eigenschaften eines ähnlichen Gerätes zugegriffen, dessen Eigenschaften „geerbt“ und verwendet werden.

Gruppen umfassen Eigenschaften, die zusammengefasst sind zu z. B. „product_info“, „markup“ oder „display“. Hier lassen sich u.a. die Eigenschaften zu:

- dem UAProf „uaprof“
- der XHTML-Unterstützung „xhtml_support_level“, die von -1 (kein XHTML) bis 4 (vollwertige HTML/CSS/Ajax-Unterstützung) reicht
- dem bevorzugtem Markup „preferred_markup“
- der Bildschirmgröße „resolution_width“ und „resolution_height“
- und der unterstützten Grafikformate „image_format“

gewinnen.

```
<group id="product_info">
<capability name="mobile_browser_version" value="3.3"/>
<capability name="uaprof" value=
"http://wap.sonyericsson.com/UAprof/K800iR201.xml"/>
<capability name="model_name" value="K800i"/>
</group>
<group id="markup">
<capability name="html_wi_oma_xhtmlmp_1_0" value="true"/>
<capability name="html_wi_w3_xhtmlbasic" value="true"/>
```

[9] http://wurfl.sourceforge.net/index.php

```
<capability name="wml_1_1" value="true"/>
<capability name="wml_1_2" value="true"/>
<capability name="xhtml_support_level" value="1"/>
<capability name="wml_1_3" value="true"/>
<capability name="preferred_markup" value="html_wi_oma_xhtmlmp_1_0"/>
</group>
<group id="display">
<capability name="physical_screen_height" value="41"/>
<capability name="columns" value="17"/>
<capability name="physical_screen_width" value="30"/>
<capability name="rows" value="16"/>
<capability name="max_image_width" value="232"/>
<capability name="resolution_width" value="240"/>
<capability name="resolution_height" value="320"/>
<capability name="max_image_height" value="300"/>
</group>
<group id="image_format">
  <capability name="jpg" value="true"/>
  <capability name="gif" value="true"/>
  <capability name="bmp" value="true"/>
  <capability name="wbmp" value="true"/>
  <capability name="png" value="true"/>
  <capability name="colors" value="262144"/>
</group>
```

Anhand dieser Informationen kann der Inhalt einer Seite an die entsprechenden Fähigkeiten eines Gerätes angepasst werden. Um die WURFL-Informationen verwenden zu können, stehen mehrere APIs (Application Programming Interface) zur Verfügung: PHP, Java und .NET. Darüber hinaus gibt es noch weitere APIs für Perl, Ruby, Python, und C++, welche jedoch als veraltet gelten und in Zukunft entfernt werden [Pas10].

3.4.3 DeviceAtlas

DeviceAtlas [10], erarbeitet von dotMobi, ist ein Datenbestand für mobile Geräte. DeviceAtlas wurde entworfen um auf die besonderen Anforderungen der Entwicklung von mobilen Anwendungen eingehen zu können. Die Daten stammen von verschiedenen Quellen wie z. B. von Telefonherstellern und Netzbetreibern.

[10] http://deviceatlas.com/

Zusätzlich beinhaltet DeviceAtlas ein Analysewerkzeug, mit dem sich Informationen über bestimmte Geräteeigenschaften gewinnen lassen, z. B. wie viele Mobilbrowser XHTML MP 1.0 unterstützen. Ebenso ist eine Geräteselektion nach bestimmten Kriterien möglich. Es besteht auch die Möglichkeit anhand des User-Agent-String zu überprüfen, ob der Datenbestand ein Gerät enthält.

Für DeviceAtlas gibt es sechs verschiedenen Lizenzen. Darunter befindet sich eine freie Lizenz, welche nach einer Registrierung auf mobiForge [11] erhältlich ist. Mit dieser „Evaluation Licence“ ist eine JSON (JavaScript Object Notation)-Datei erhältlich, die eine umfangreiche Datensammlung von Mobilgeräten beinhaltet. Zur Identifizierung von Geräteeigenschaften ist eine API für PHP, Java, .NET, Ruby und Python verfügbar. Die komplette Datensammlung ist hingegen erst mit einer kostenpflichtigen Version vorhanden. [mTLDL08]

3.5 Einflüsse auf die Übertragungsgeschwindigkeit

Bei der Übertragung von Webseiten mittels den mobilen Datenträgern (wie z. B. UMTS) zu Clientgeräten kommt es mehr als nur auf die Datenverbindung an. Der Datendurchsatz wird in vielen Fällen bei Weitem nicht erreicht, selbst bei optimalen Netzbedingungen. Browser sind hierbei ein entscheidender Faktor für die Schnelligkeit der Darstellung. Aktuelle Mobilbrowser verlangsamen das Darstellen von Mobil- und Webseiten erheblich. Die UMTS-Datenraten spielen deshalb nur eine untergeordnete Rolle. [Pau10]

Des Weiteren sind die schnellen Datenübertragungstechniken UMTS und HSDPA nicht überall verfügbar; besonders in ländlichen Regionen nicht. Hier gibt es häufig Lücken in der Abdeckung und es ist lediglich GPRS verfügbar. Die Mobilfunkinfrastruktur bestand zum Ende des ersten Quartals 2009 u.a. aus rund 39.000 UMTS-Funkbasisstationen und etwa 120.000 aktiven UMTS-Funkzellen. So können UMTS-Dienste theoretisch an knapp 70 Prozent aller Standorte genutzt werden. Die Netzabdeckung variiert jedoch. Je nach Netzbetreiber liegt die Abdeckung zwischen 59 Prozent und 81 Prozent [Bun10].

Da Kultur- und besonders Landschaftselemente verstärkt in ländlichen Räumen anzutreffen sind, kann es gerade hier passieren, dass nicht immer die schnellen Datenübertragungstechniken verfügbar sind. Umso mehr ist bei der Entwicklung von KLEKs-WAP darauf zu achten, dass die übertragenen Inhalte für kurze Ladezeiten der Seite möglichst klein gehalten werden.

[11] http://mobiforge.com/

4 Entwurf von KLEKs-WAP

Für den Betrieb einer Mobilversion der Internetpräsenz wird kein eigener WAP-Server benötigt. Denn dieser ist im Grunde genommen nichts anderes als ein gewöhnlicher Webserver. Was für den Betrieb einer Webseite von Nöten ist, wird ebenso für die Mobilseite benötigt. Seit WAP 2.0 sind auch die Kommunikationsprotokolle die gleichen.

Basierend auf dem World Wide Web Architektur-Modell (Abbildung 5) folgt das WAP-Modell (Abbildung 6) den Web-Standards soweit dies möglich ist. Zudem kommen Optimierungen und Erweiterungen hinzu, um den Charakteristiken der „over the air“ Übertragungen gerecht zu werden. Die existierenden Standards wurden für WAP übernommen oder vorübergehend als Start für die WAP-Technologie genutzt. WAP Inhalte und Applikationen basieren auf den bekannten Formaten, die im WWW zum Einsatz kommen.

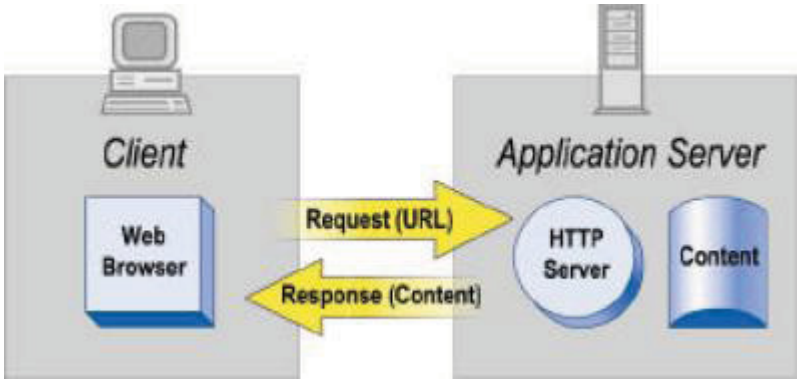

Abbildung 5: World Wide Web Modell [WAP01]

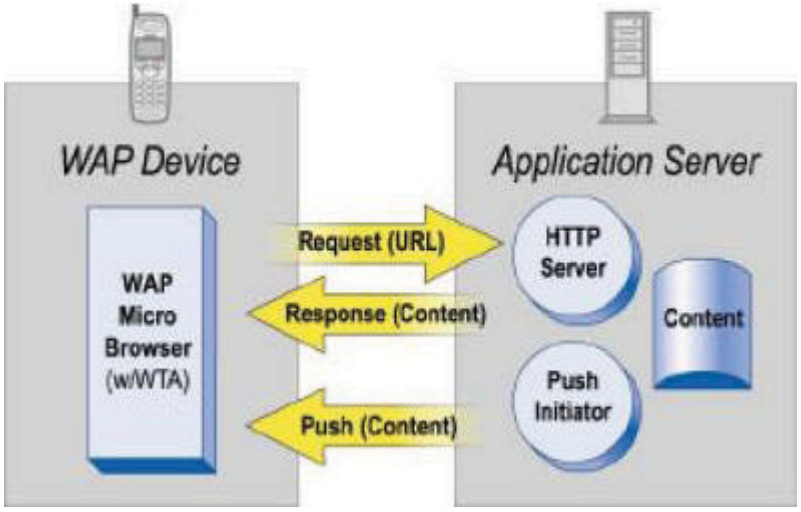

Abbildung 6: WAP-Modell [WAP01]

Zu den bedeutendsten Erweiterungen des WAP-Modells gehören die Push-Funktionalitäten und der Zugriff auf Telefonfunktionen.

Das WAP-Modell ist durch Proxies und Supporting Server erweiterbar. Die Proxy-Technologie ermöglicht eine Optimierung und Verbesserung der Verbindung zwischen dem drahtlosen Gerät und dem Server. Ein WAP-Proxy stellt eine Vielzahl von Funktionen zur Verfügung, zu denen u.a. der Protocol Gateway, Content Encoders und Decoders, User Agent Profile Management und ein Caching Proxy gehören.

Durch diesen Aufbau wird es ermöglicht, eine Vielfalt von Internetanwendungen auf den mobilen Geräten dem Nutzer präsentieren zu können. Die WAP-Architektur ist jedoch nicht an den üblichen Aufbau von Webserver, WAP-Proxy und WAP-Client gebunden, andere Konfigurationen sind ebenso möglich. Es können Supporting Server integriert werden, welche verschiedene Dienste für Geräte, Proxies und Applikationen unterstützen. Eine Möglichkeit besteht z. B. in der Integration eines UAProf Servers, wodurch gerätespezifische Informationen gewonnen werden können. Ein möglicher Aufbau einer WAP-Architektur mit einen WAP-Proxy und einem Supporting Server ist in der Abbildung 7 dargestellt. Durch die Eingabe einer URL in den WAP-Browser startet dieser eine Anfrage an den Webserver. Die Anfrage kann, in Abhängigkeit der Einstellungen des Mobiltelefons, über einen WAP-Proxy geleitet werden. Der Proxy kann den Kommunikationsprozess durch Kodier- und Dekodierfunktionen optimieren und bietet Erweiterungen für mobile Dienste an [WAP02]. Die Anfrage wird von dem Proxy an den Server weitergeleitet, welcher seinerseits wiederum eine Anfrage an einen Supporting Server stellen kann, um beispielsweise Geräte spezifische Informationen zu erhalten. Nach Erhalt dieser Informationen übermittelt der Server die Daten an den Proxy und dieser überträgt die Daten zum Client.

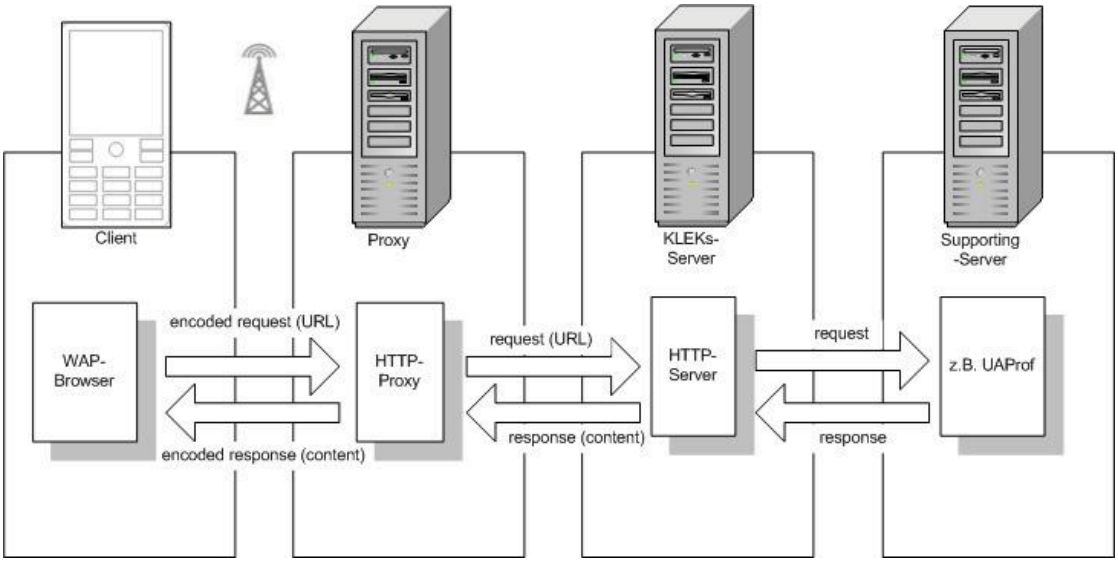

Abbildung 7: KLEKs-WAP-Anwendung, eigene Darstellung auf Basis von [WAP01]

Bei der Architektur von Client, WAP-Proxy und Webserver wird der WAP-Proxy von den Mobilfunk-Netzbetreibern bereitgestellt. Da auf diesen kein Einfluss genommen werden kann, entfällt dieser in der folgenden Abbildung. Anhand des Komponentendiagramms der KLEKs-WAP-Anwendung, dargestellt in Abbildung 8, sollen die Funktionalitäten und die verschiedenen Beziehungen zwischen den einzelnen Komponenten dargestellt werden. Die Funktionalität der Komponenten kann aus beliebig vielen Funktionen bzw. Dateien bestehen.

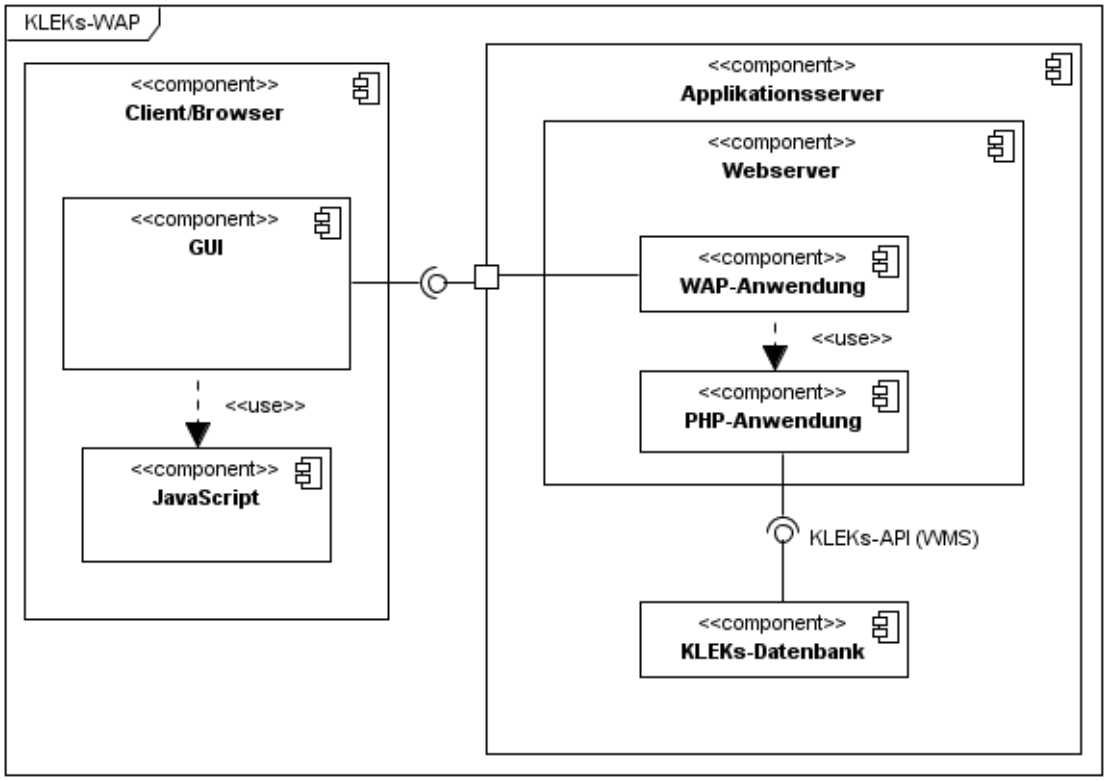

Abbildung 8: Komponentendiagramm der KLEKs-WAP-Version

Die KLEKs-WAP-Anwendung unterteilt sich dabei nach dem Client-Server-Prinzip. Auf Software-Ebene ist der Browser hierbei der Client und fungiert als Kommunikationsschnittstelle mit dem Nutzer. Über eine Schnittstelle mit dem Server greift der Client auf die WAP-Anwendung zu. Innerhalb der Komponente des Applikationsservers befindet sich die Komponente der WAP-Anwendung, welche den Inhalt der WAP-Seite realisiert. Diese greift auf die PHP-Anwendung, die die Visualisierung der Karteninformationen realisiert, zu. Über die Schnittstelle der KLEKs-API wird auf die Inhalte der Komponente der KLEKs-Datenbank zugegriffen. Die WMS des KLEKs bilden hierbei die Schnittstellen. Nachdem durch den Browser auf die WAP-Anwendung zugegriffen worden ist, können innerhalb der WAP-Seite (Komponente GUI) verschiedene Funktionen ausgeführt werden. Diese Funktionen werden durch die Komponente JavaScript realisiert.

5 Umsetzung

Die KLEKs-WAP-Anwendung erfolge als prototypische Umsetzung unter Verwendung der Scriptsprachen PHP und JavaScript sowie der Auszeichnungssprache XHTML-MP. Die Anwendung wurde auf dem Paket *MapServer for Windows*[12] in der Version 2.3.1 realisiert. Dieses Paket besteht aus Apache 2.2.10, PHP 5.2.6, MapServer 5.2.1 und weiteren Bestandteilen.

5.1 Serverseitig - PHP

Bei der Entwicklung einer WAP-Seite muss zuerst sichergestellt werden, dass der Webserver auch WAP-Inhalte versteht und darstellen kann, d.h. der Webserver muss Dokumente im WML und XHTML Format verstehen. Dem wiederum müssen die MIME-Typen der WAP-Dokumente in der Konfigurationsdatei des Webservers korrekt gesetzt sein. Ist dies nicht der Fall, können WAP-Browser die WAP-Seite nicht darstellen. Der Webserver der KLEKs-Anwendung ist ein Apache in der Version 2.2.9. Aus diesem Grund wird sich lediglich darauf beschränkt, wie MIME-Typen im Apache gesetzt werden und es wird auf Verwendung weiterer Webserver verzichtet. Überlicherweise befindet sich die Konfigurationsdatei für MIME-Typen unter [Apache_Home]/conf/mime.types. Hier sind alle verständlichen MIME-Typen des Webserver in der Form MIME-Typ gefolgt von einer oder mehreren Dateiendungen aufgelistet.

```
MIME-Typ Dateiendung1 Dateiendung2 Dateiendung3 ...
```

Hier muss kontrolliert werden, ob die MIME-Typen für die WAP-Dokumente gesetzt sind oder nicht und für den zweiten Fall müssen diese eingetragen werden. Jedoch werden für diese Variante Administratorenrechte benötigt. Da aber nicht immer Adminstratorrechte vorliegen, z. B. weil die Seiten auf den Servern einer Hosting-Firma liegen, bedarf es einer Variante ohne Adminstratorrechte. Es besteht die Möglichkeit der Nutzung einer .htaccess Datei mit der AddType Direktiven. Mit

```
AddType MIME-Typ Dateiendung1 Dateiendung2 Dateiendung3 ...
```

werden neue MIME-Typen gesetzt und schon bestehende überschrieben. Die .htaccess Datei sollte dabei von jedem lesbar sein - world-readable. Es besteht jedoch die Möglichkeit, dass der Webserver so konfiguriert ist, dass .htaccess Dateien ignoriert werden. Deshalb sollte für das Setzen von MIME-Typen eine serverseitige Technologie verwendet werden. Dazu werden weder Administratorenrechte benötigt, noch muss die Konfigurationsdatei geändert werden. In diesem Fall wird dies mit PHP erledigt. Durch die Funktion header() kann in folgender Form ein MIME-Typ gesetzt werden:

```
<?php header('Content-type: MIME-Typ'); ?>
```

[12] http://maptools.org/ms4w/index.phtml

Diese Zeile Code muss am Anfang des Dokumentes stehen und die Funktion header() muss aufgerufen werden, bevor die Ausgabe zum Client gesendet wird. [dev08]

Serverseitig bestand die Aufgabe in der Bündelung der bestehenden WMS mittels PHP. Dies sind zum einem der WMS mit der Hintergrundkarte und zum anderen der WMS mit den Landschaftselementen und den Audio-Dateien. Die benötigten Parameter der WMS wurden in einer config-Datei gesetzt und anschließend ausgelesen. Der Vorteil hierbei liegt darin, dass zu einem späteren Zeitpunkt die WMS möglichst einfach austauschbar sind.

5.2 Clientseitig - XHTML Mobile Profile

Aufgrund des Umfangs dieser Arbeit, wurde beschlossen eine einheitliche Version zu erstellen und auf eine gerätespezifische Anpassung zu verzichten. Anhand dieser soll exemplarisch gezeigt werden, wie eine Umsetzung der WAP-Version von KLEKsOnline funktionieren könnte. Die Seite wurde im Folgenden für Mobiltelefone mit einem 12er Tastenfeld und einer Bildschirmauflösung von 240x320 Pixel entworfen. Als Auszeichnungssprache findet XHTML MP als WAP-Standard Anwendung. Dies in der Version 1.1 oder höher, damit die Einbindung von JavaScript gegeben ist. Dabei wurde ein einfaches Layout favorisiert, um in der Darstellung zu garantieren, dass möglichst viele Informationen dargeboten werden.

Die Abbildung 9 zeigt die Startansicht der WAP-Seite von KLEKs. Das Layout ist in eine Kopfzeile und in den Karteninhalt unterteilt. Die Kopfzeile beinhaltet die Schaltflächen für die Zoomfunktionen und eine Verlinkung zum KLEKsOnline-Viewer. Die Navigation innerhalb des Kartenmaterials wurde nicht an den KLEKsOnline-Viewer angelehnt, um die begrenzte Displaygröße nicht mit Navigationselementen zu verbauen. Hier wurde eine möglichst platzsparende Variante bevorzugt. Wie bei GoogleMaps[13] oder BingMaps[14] befindet sich die Steuerung der Karte in der oberen linken Ecke. Mit der Auswahl einer Funktion wird im Hintergrund eine JavaScript-Funktion aufgerufen, die dafür sorgt, dass die gewünschte Aktion stattfindet. Dabei wurden bisher nur die Funktionen des Hinein- und Hinauszoomens realisiert. Über die Symbole zum Zoomen wird jeweils eine Funktion aufgerufen, welche den Wert „Meter pro Pixel“ halbiert bzw. verdoppelt. Der neu berechnete Wert wird innerhalb einer Funktion gerundet, um die Zoomstufen zu realisieren und anschließend zum Server übermittelt, um die Karte mit einer neuen Ausdehnung anzufordern.

[13] http://maps.google.de
[14] http://www.bing.com/maps

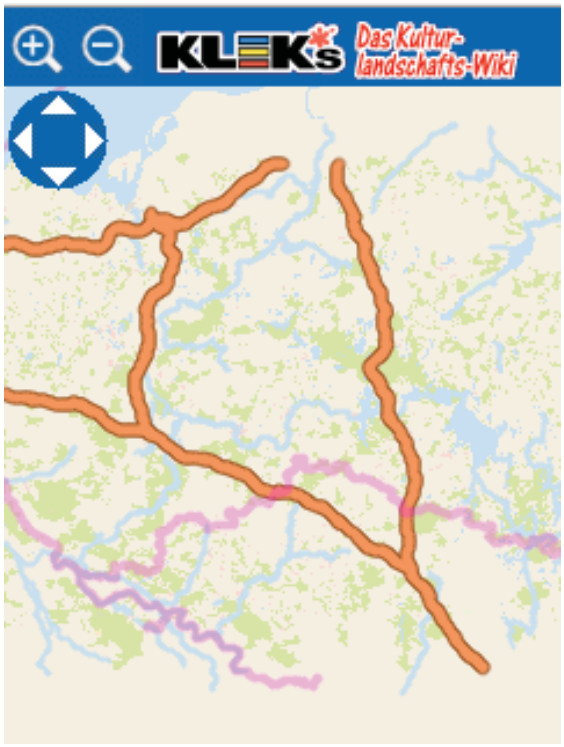

Abbildung 9: Startansicht der KLEKs-WAP-Seite

6 Testphase

Nicht nur während, sondern ganz besonders am Ende der Entwicklung muss jede Anwendung getestet werden. Dabei gilt, je mehr getestet wird, desto besser. Für mobile Webanwendungen sollte in drei verschiedenen Phasen getestet werden. Zum Ersten auf Desktop-Browsern, um sehen zu können, ob die Anwendung überhaupt funktioniert. Zum Zweiten mit Emulatoren, um die für Mobilgeräte angepasste Webapplikation auf einer möglichst großen Bandbreite von Mobiltelefonen und Mobilbrowsern zu testen. Dies ist besonders in Hinblick auf die Wirtschaftlichkeit empfehlenswert, da keine Hardware benötigt wird um die Funktionalität zu prüfen und so Kosten gespart werden können. Jedoch verhalten sich Emulatoren zum Teil anders als reale Geräte. Zum Schluss folgt dann der reale Test auf am besten mehreren Telefon- und Smartphonemodellen. Dabei empfiehlt es sich auch einzelne Funktionen der Mobilgeräte und Browser zu deaktivieren und erneut zu testen [Alb08].

Getestet wurde die KLEKs-WAP-Seite mit je zwei Emulatoren und Geräten. Als Emulatoren kamen das iPhone[15] und das PalmPre[16] zum Einsatz. Bei beiden war die Darstellung so wie gewünscht und wie in Kapitel 5.2 beschrieben.

Auf den realen Geräten verlief der Test hingegen nicht vollständig wie gewünscht. Anwendung fanden das Sony Ericsson K800i und das Smartphone Samsung S8500 Wave. Bei dem K800i (siehe hierzu Tabelle 2) wurde die KLEKs-WAP-Seite über den Browser Opera Mini 5.1 aufgerufen. Der Browser leitet die Anfrage über eigene Server um, sodass Grafiken vor dem Senden zum Client kleiner gerechnet werden, um Übertragungskapazität zu sparen. Folglich wurde die Karte der Anwendung zu klein angezeigt, die Kopfzeile hingegen wie gewünscht.

Das Samsung hat eine Auflösung von 480x800 Pixel. Die WAP-Seite wurde wie oben beschrieben auf eine Größe von 240x320 Pixel festgesetzt. So wäre zu erwarten, dass die Seite nur in der linken oberen Ecke angezeigt wird. Doch der Dolfin-Browser des Samsung stellt die WAP-Seite angepasst an die Displaybreite dar. Hier zeigt sich, dass moderne Mobilbrowser teils selber versuchen Seiten angepasst an Mobilgeräte darzustellen. Da es sich hierbei jedoch um eine angepasste Seite für Mobilgeräte handelt, sind die Versuche der Browser zur Anpassung unerwünscht. Abhilfe könnte hier der „viewport meta tag“ schaffen. Der Tag zeigt an, das diese Seite für Mobilgeräte optimiert wurde und mobile Browser keine automatischen Optimierungen durchführen [Sch10].

Durch das Klicken auf die Zoomsymbole wird die Karte mit neuen Ausdehnungsparameter angefordert. Dies funktionierte sowohl in den Emulatoren, als auch auf den beiden realen Geräten. Weitere Funktionalitäten der WAP-Seite konnten leider nicht getestet werden, jedoch funktionieren die zu implementierenden Funktionen auf dem KLEKsOnline-Viewer innerhalb der Mobilgeräte. Eine Ausnahme bildet dabei das Abspielen von Audio-Dateien, da das Abspie-

[15] http://testiphone.com/
[16] http://www.genuitec.com/mobile/download.html

len über einen Flash-Player läuft. Je nach Mobilbrowser ist Flash standardmäßig ausgeschaltet und muss erst aktiviert werden oder ist nicht verfügbar.

Zusätzlich zu den oben beschriebenen Tests empfehlen sich noch zwei Kompatibilitätstests, ob die Seite den besonderen Gegebenheiten des mobilen Webs entspricht. Dazu bieten das W3C und die mTLD jeweils einen Validator an. Der Validator des W3C ist der „mobile OK checker“[17] und das Pendant von der mTLD heißt „mobiReady“[18]. Hier lässt sich die Entwicklung nach Industriestandards und auf „Best Practices“ für Mobilgeräte testen. Hinter den Tests verbirgt sich im Großen und Ganzen eine Überprüfung der Einhaltung der in Kapitel 3.2 vorgestellten „Mobile Web Best Practices“. Die Tests geben u.a. Hinweise zur Einhaltung der „Best Practices“ und liefern detaillierte Fehlerberichte. Der mobiReady Validator schätzt zudem die Aufrufgeschwindigkeit der Seite für die Trägerdienste WLAN (Wireless Local Area Network), 3G (3rd Generation) - dazu gehören UMTS und HSDPA - und GPRS. Darüber hinaus werden die ungefähren Kosten für einen Seitenaufruf bei einem gewöhnlichen Datentarif geschätzt.

[17] http://validator.w3.org/mobile/
[18] http://ready.mobi/

7 Zusammenfassung

Ziel dieser Arbeit war eine prototypische Umsetzung einer WAP-Version des KLEKsOnline-Viewers. Dabei hat sich gezeigt, dass sich WAP als Bestandteil des mobilen Webs dem Web sehr stark genähert hat und dies auch in Zukunft weiter tun wird. Die Grenzen zwischen dem Web und dem mobilen Web werden mehr und mehr in einander übergehen.

Für die Gestaltung der WAP-Seite war ein möglichst einfacher Aufbau wichtig. Der Inhalt sollte dabei im Vordergrund stehen und die Elemente zur Bedienung möglichst wenig Platz einzunehmen. Innerhalb der WAP-Seite sollte die Möglichkeit des Hinein- und Hinauszoomens bestehen, was prototypisch umgesetzt wurde.

Funktionalitäten zum Verschieben der Karte, zum Erhalt von Informationen und Bildern zu Landschaftselementen sowie eine Abspielmöglichkeit für Audios konnten nicht implementiert werden. Hier ist in Zukunft anzusetzen, um die WAP-Version des KLEKsOnline-Viewers zu erweitern.

In Erweiterung des Projektes besteht die Möglichkeit der Entwicklung einer nativen Applikation (App). Mit dieser könnte auf ein - in das Mobiltelefon - integriertes GPS (Global Positioning System)-Modul zugegriffen werden und sich der eigene Standpunkt in der Karte angezeigt werden. Mit einer App ließen sich auch die Karten direkt auf dem Gerät speichern, womit eine Menge an Daten bei der Übertragung gespart werden kann. Unter Nutzung der im Mobiltelefon eingebauten Kamera und GPS-Empfänger ließen sich Fotographien von Landschaftselementen mit geographischen Koordinaten erstellen. Diese wiederum könnten zum Ausbau des Datenbestandes genutzt werden, um so den KLEKsOnline-Viewer noch attraktiver zu gestalten.

Abkürzungsverzeichnis

3G	3rd Generation
API	Application Programming Interface
BITKOM	Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e.V.
CCPPACCEPT	Composite Capabilities/Preference Profiles ACCEPT
cHTML	compact HTML
CSS	Cascading Style Sheets
CSS MP	CSS Mobile Profile
ECMA	European Computer Manufacturers Association
EDGE	Enhanced Data Rates for GSM Evolution
EPSG	European Petroleum Survey Group
GIF	Graphics Interchange Format
GIS	Geoinformationssystem
GPRS	General Packet Radio Service
GPS	Global Positioning System
GSM	Global System for Mobile Communications
HSDPA	High Speed Downlink Packet Access
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IP	Internet Protocol
JPEG	Joint Photographic Experts Group
KLEKs	KulturLandschaftsElementeKataster
LTE	Long Term Evolution
MIME	Multipurpose Internet Mail Extensions
OGC	Open Geospatial Consortium
OMA	Open Mobile Alliance
PDA	Personal Digital Assistant

PHP	PHP Hypertext Preprocessor
PNG	Portable Network Graphics
RDF	Resource Description Framework
SSL	Secure Socket Layer
SVG	Scalable Vector Graphics
TCP	Transmission Control Protocol
UAProf	User Agent Profile
UMN	University of Minnesota
UMTS	Universal Mobile Telecommunications System
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WAP	Wireless Application Protocol
WCSS	Wireless CSS
WebCGM	Web Computer Graphics Metafile
WLAN	Wireless Local Area Network
WML	Wireless Markup Language
WMLC	Wireless Markup Language Compiled
WMS	Web Map Service
WSP	Wireless Session Protocol
WTLS	Wireless Transport Layer Security
WTP	Wireless Transaction Protocol
WURFL	Wireless Universal Resource File
WWW	World Wide Web
XHTML	eXtensible HyperText Markup Language
XHTML MP	XHTML Mobile Profile
XML	eXtensible Markup Language

Literatur

[Alb08] ALBY, TOM: *Das mobile Web*. Carl Hanser Verlag, München, 2008.

[Avc03] AVCI, ORAL: *Web-Programmierung*. Werner Mellis, Wiesbaden, 2003.

[BI10] BUNDESVERBAND INFORMATIONSWIRTSCHAFT, TELEKOMMUNIKATION UND NEUE MEDIEN E.V.: *Etwa jeder vierte Internetnutzer surft mobil*. http://www.bitkom.org/files/documents/BITKOM-Presseinfo_mobiles_Internet_05_04_2010.pdf, April 2010. Zugriff am 07. August 2010.

[Bra09] BRAUN, HERBERT: *Nett zu den Kleinen*. c't extra, Januar 2009.

[Bun10] BUNDESNETZAGENTUR: *Jahresbericht 2009*. http://www.bundesnetzagentur.de/cae/servlet/contentblob/152206/publicationFile/6683/Jahresbericht2009Id18409pdf.pdf, 2010. Zugriff am 24. August 2010.

[dev08] *Tutorial about Setting up WAP Servers for Hosting WAP 1.x or WAP 2.0 Sites*. http://www.developershome.com/wap/wapServerSetup/, September 2008. Zugriff am 25. August 2010.

[dlB04] BEAUJARDIERE, JEFF DE LA: *OGC Web Map Service Interface*. http://portal.opengeospatial.org/modules/admin/license_agreement.php?suppressHeaders=0&access_license_id=3&target=http://portal.opengeospatial.org/files/index.php?artifact_id=4756, Januar 2004. Zugriff am 22. Juli 2009.

[Dul05] DULZ, WINFRIED: *WAP Wireless Application Protocol*. *http://www.gi-ev.de/no_cache/service/informatiklexikon/informatiklexikon-detailansicht/meldung/wap-wireless-application-protocol-91.html*, Juli 2005. Zugriff am 18. Juli 2010.

[Fis03] FISCHER, THORSTEN: *UMN MapServer 4.0*. MapMedia GmbH, Berlin, 2003.

[jav07] *Einführung in JavaScript und DOM*. http://de.selfhtml.org/javascript/intro.htm, 2007. Zugriff am 25. August 2010.

[MIM07] *MIME-Typen*. http://de.selfhtml.org/diverses/mimetypen.htm, 2007. Zugriff am 24. August 2010.

[mTLDL08] TOP LEVEL DOMAIN LTD MTLD: *DeviceAtlas*. http://deviceatlas.com/, 2008. Zugriff am 30. August 2010.

[Pas10] PASSANI, LUCA: *WURFL*. http://wurfl.sourceforge.net/index.php, Juli 2010. Zugriff am 28. August 2010.

[Pau10] PAULER, WOLFGANG: *Sieben Handys im großen Browser-Test*. http://www.chip.de/artikel/Sieben-Handys-im-grossen-Browser-Test_42796426.html, Mai 2010. Zugriff am 24. August 2010.

[PDHB08] PROF. DR. HERMANN BEHRENS, PROF. DR. LUTZ VETTER, DR. MAIK STÖCKMANN: *Flyer zum Projekt KLEKs*. http://www.kleks-online.de/FlyerA4.pdf, Juli 2008. Zugriff am 15. Juli 2010.

[RM08] RABIN, JO und CHARLES MCCATHIENEVILE: *Mobile Web Best Practices 1.0*. http://www.w3.org/TR/mobile-bp/, Juli 2008. Zugriff am 21. August 2010.

[Sch10] SCHWARZ, MICHAEL: *Wozu der viewport meta tag?* http://www.mobilewebdesign.de/?p=54, Februar 2010. Zugriff am 06. September 2010.

[the07] *What is the difference between XHTML MP, XHTML Basic, WML, i-mode, and HDML?* http://www.thewirelessfaq.com/what_is_the_difference_between_xhtml_mp_xhtml_basic_wml_imode_and_hdml, Januar 2007. Zugriff am 18. Juli 2010.

[WAP01] *WAP Architecture*. http://www.openmobilealliance.org/tech/affiliates/LicenseAgreement.asp?DocName=/wap/wap-210-waparch-20010712-a.pdf, Juli 2001. Zugriff am 25. Juli 2010.

[WAP02] *Wireless Application Protocol*. www.wapforum.org/what/WAPWhite_Paper1.pdf, Januar 2002. Zugriff am 30. Juli 2010.

[WAP10] *WAP (wireless application protocol)*. http://www.itwissen.info/definition/lexikon/wireless-application-protocol-WAP-WAP-Protokoll.html, Zugriff am 18. Juli 2010.

[Weh08] WEHRENPFENNIG, ANDREAS: *dynamische Web-Programmierung: clientseitig*. Hochschule Neubrandenburg, Bachelor-Studiengang Geoinformatik, Modul B123, 2008.

[wik10] *Wireless Application Protocol*. http://de.wikipedia.org/wiki/Wireless_Application_Protocol, Zugriff am 18. Juli 2010.

Abbildungsverzeichnis

Tabellenverzeichnis

Anhang