Georg-August-Universität Göttingen

Geographisches Institut

SoSe 2015

Dr. Stefan Erasmi

Praktikumsbericht

über das abgeleistete Praktikum in der Abteilung Kartographie, GIS & Fernerkundung, Goldschmidtstr. 5, 37077 Göttingen

vom 01.04.2015 bis xx.xx.xxxx

Lukas Hoffmann

Wiesenstraße 24

lukas.hoffmann2@stud.uni-goettingen .de

21159171

Bachelorstudiengang Angewandte Informatik

8. Semester

Inhaltsverzeichnis

Abbildungverzeichnis

1 Praktikumsstelle

Ich studiere Angewandte Informatik an der Georg-August-Universität Göttingen und im Rahmen dieses Studiengangs wird das Modul „Vertiefte Angewandte Informatik im forschungsbezogenen Praktikum“ (Modul B.Inf.1811) angeboten. Da empfohlen wird, das Praktikum begleitend zur Bachelorarbeit zu belegen, erfragte ich um einen Platz in derselben Abteilung, in der ich diese verfasse.

Bei der angesprochenen Abteilung handelt es sich um die „Kartographie, GIS & Fernerkundung“ des Geographischen Instituts. Ich wurde der Arbeitsgruppe Campus-GIS zugeordnet, die sich hauptsächlich mit einem webbasierten Gebäude- und Raum-Auskunftssystem für die Georg-August-Universität befasst (GRAS-Geo).

GRAS-Geo vereint Gebäude- und Raumpläne, Belegungspläne aus dem Vorlesungsverzeichnis und freie Geodaten, wie zum Beispiel die „OpenStreetMap[1]“, und bietet somit eine Grundlage für verschiedene Entwicklungen.

Eine Entwicklung ist der Lageplan zur Barrierefreiheit. Da sich die Universität Göttingen bemüht, das Studium für Studierende mit gesundheitlicher Beeinträchtigung zu erleichtern, entstand eine webbasierte Karte, die die Zugänglichkeit hinsichtlich der Barrierefreiheit bzw. Barriereunfreiheit überprüft und darstellt. So können Informationen über Zugänglichkeit zu Räumen ausgehend von Bushaltestellen und barrierefreien Parkplätzen angezeigt werden.

Meine Hauptaufgabe in dem Praktikum war es, den Lageplan für mobile Geräte anzupassen. Das Praktikum lief semesterbegleitend und die Arbeitszeiten waren flexibel, da viele Arbeitsschritte von zu Hause aus erledigt werden konnten.

Zusammengearbeitet habe ich mit meinem Kommilitonen Stefan Siemer, der ebenfalls das gleiche Praktikum absolvierte; mein Hauptansprechpartner war Alexander Winz.

2 Technik und Methodik

2.1 HTML und CSS

Die Hypertext Markup Language, kurz HTML, ist eine textbasierte Aufzeichnungssprache, die benutzt wird, um digitale Inhalte darzustellen. Die in ein Dokument geschriebenen sogenannten

[1] OpenStreetMap ist ein Projekt, das frei unter einer offenen Lizenz verwendbar ist und nutzbare Geodaten sammelt, um sie jedermann verfügbar zu machen.

Tags definieren die komplette Struktur einer Webseite vor. Typische Elemente sind beispielsweise Überschriften, Textabsätze, Listen oder Grafikreferenzen. Der Browser zeigt die Textinhalte, die mit HTML strukturiert wurden, einfach untereinander angeordnet auf dem Bildschirm an. Die genaue Darstellung der Elemente ist durch die Voreinstellungen des Browsers vorgegeben. So ist zum Beispiel die Hintergrundfarbe oder die bestimmte Größe von Überschriften festgelegt. Die genauen Voreinstellungen unterscheiden sich allerdings von Browser zu Browser und so wird meist das visuelle Ergebnis den Ansprüchen nicht gerecht. Es ist die Ergänzungssprache CSS (Cascading Style Sheets, zu Deutsch „Kaskadierende Stilvorlagen“) notwendig. Sie ermöglicht das beliebige Formatieren einzelner HTML Elemente. Mit Hilfe von Stylesheets können zum Beispiel Änderungen der Schriftgröße und Schriftfarbe oder punkgenaues Platzieren auf dem Bildschirm realisiert werden. CSS spezifiziert somit das Layout einer Webseite.
Da HTML und CSS sehr eng miteinander verbunden sind, habe ich im Praktikum beides gleichzeitig erlernt.

2.2 ArcGIS und JavaScript

ArcGIS ist ein von der Firma Esri entwickeltes geografisches Informationssystem, das zur Erstellung und Nutzung von Karten verwendet wird. Weiterhin werden viele Funktionen zur Kompilierung von geografischen Daten, zur Analyse von kartenbezogenen Informationen und zum Datenbankenmanagement zur Verfügung gestellt. Zusammenfassend bietet ArcGIS also eine Infrastruktur, die es möglich macht, Karten und geografische Informationen für eine Organisation, eine spezielle Gemeinschaft oder offen im Internet verfügbar zu machen.
Für Entwickler werden auf verschiedenen Plattformen Programmierschnittstellen (englisch: application programming Interface, API) für ArcGIS angeboten. In meinem Praktikum benutzte ich die ArcGIS API für JavaScript.
JavaScript ist eine Scriptsprache, die zur Programmierung des Verhaltens von Webseiten eingesetzt wird. So können dynamisch Benutzerinformationen ausgewertet, Inhalte verändert, nachgeladen oder neu generiert werden. Da JavaScript im Browser ausgeführt wird, eignet sich diese Sprache zur Manipulation von Internetseiten, ohne dass ein Refresh nötig ist.

3 Aufgaben und Ziele

3.1 Anforderungen an die mobile Webseite

Wie bereits erwähnt, war es meine Aufgabe eine für mobile Geräte angepasste Webseite des Lageplans zu erstellen.

Die Verkaufszahlen für Smartphones sind in den letzten Jahren rasant gestiegen. „Wurden im Jahr 2010 weltweit noch rund 300 Millionen Smartphones ausgeliefert, waren es im Jahr 2013 bereits mehr als eine Milliarde. Im Jahr 2014 belief sich der Smartphone-Absatz auf mehr als 1,3 Milliarden Geräte“ (Statista 2015).
Durch diese Entwicklung wird die Bereitstellung von mobilen Webseiten immer wichtiger.
Das Erstellen einer solchen für den Lageplan für Barrierefreiheit erscheint damit nur logisch, auch wenn man bedenkt, dass viele Studenten die Informationen benötigen, wenn sie bereits unterwegs sind oder auf dem Campus einen Raum suchen.
Eine mobile Webseite sollte die folgenden Punkte gewährleisten. Zum Ersten die Anpassung an unterschiedliche Displaygrößen. Je nach Hersteller und Modell sind diese unterschiedlich. Des Weiteren gilt es die Übersichtlichkeit und Navigation so zu gestalten, dass auch Geräte mit besonders kleinen Bildschirmen keine Probleme haben. Hier musste darauf geachtet werden, dass Funktionen, die die ArcGIS API nutzen und auf der Karte Dinge darstellen, entsprechend angepasst werden.
Als letzten Punkt kann die Geschwindigkeit aufgeführt werden. Nicht alle Smartphones sind sehr rechenstark und lange Ladezeiten behindern den Nutzen einer mobilen Webseite.

3.2 Ausgangssituationen

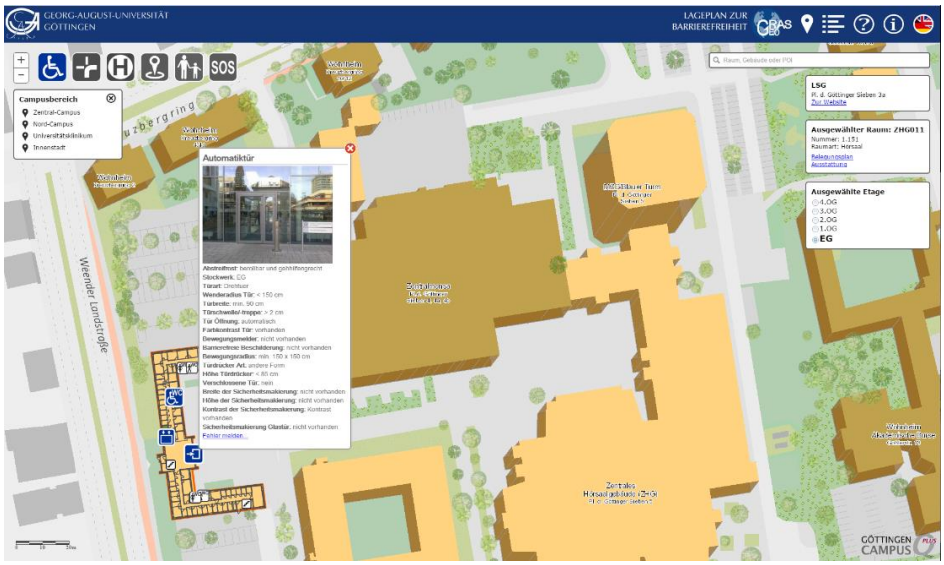

Abbildung 1: Desktopversion des Lageplans zur Barrierefreiheit

Der nicht für mobile Geräte angepasste Lageplan besteht aus mehreren Komponenten. Im oberen Teil ist die Titelleiste mit dem Uni-Logo, dem GRAS-Geo-Logo und fünf Buttons zur Informationsdarstellung (Campusbereiche, Legende, Hilfe, Information und Sprachumstellung auf Englisch). Darunter befindet sich die Karte von Göttingen. Oben links auf der Karte sind sechs weitere Buttons, mit denen Symbole für Barrierefreiheit, Wegenetz, Bushaltestellen, Infrastruktur, familienfreundliche Infrastruktur und Nachtkarte mit Sicherheitsnetz an- und ausgeschaltet werden können.
Wenn die angeschalteten Symbole auf der Karte angeklickt werden, erscheint ein Popup-Fenster, das zusätzliche Informationen bereitstellt.
Auf der rechten Seite tauchen drei Fenster auf, wenn ein Gebäude bzw. ein Raum ausgewählt wird. Im ersten sind Informationen zum Gebäude zu finden, im zweiten zum ausgewählten Raum. Das dritte dient zum Wechseln der Etage.
Zuletzt existieren noch ein Suchfenster, mit dem Gebäude und Räume gesucht werden können, ein Maßstab und ein Zoom-Button.

3.3 Arbeitsergebnisse

3.3.1 Entfernte und verschobene Elemente

Als Erstes habe ich alle Elemente, die ich in der mobilen Version als nicht notwendig erachtet habe, entfernt. Die Buttons der Titelleiste, der Maßstab, der Zoom-Button und das Logo von Gras-Geo gehörten dazu. Die mobile Webseite soll selbsterklärend sein, was genau diese Elemente unnötig macht. Der Zoom-Button wird durch die eingebaute Zoom-Funktion in Smartphones gleichwertig ersetzt. Das Logo der Universität Göttingen habe ich verkleinert, indem ich den Schriftzug abschnitt. Die sechs übrigen Buttons wurden dann der Titelleiste hinzugefügt. Wichtig waren hier, dass der Abstand zum links gelegenen Uni-Logo eingehalten wird (70 Pixel) und die Größe der Buttons so angepasst wird, dass zwei Reihen untereinander möglich sind, falls die Breite des Displays klein ausfällt. Da die Titelleiste 110 Pixel hoch ist, fiel die Wahl auf 35 Pixel Höhe und Breite. So verbleiben noch 40 Pixel für weitere Elemente.

```
.serviceButtonsMobile{
  left: 70px;
  margin-bottom: 0px;
  position: absolute;
  z-index: 2;
}
```

Abbildung 2: Ausschnitt aus der layout.css – Styleklasse „serviceButtonsMobile“

```
<div class="serviceButtonsMobile" style="position: absolute; width:auto; hei
    <img alt="?" id="changingBarrierSymbol" width="35" height="35" style="cu
    <img alt="?" id="changingWegeSymbol" width="35" height="35" style="curso
    <img alt="?" id="changingStopSymbol" width="35" height="35" style="curso
    <img alt="?" id="changingPOISymbol" width="35" height="35" style="cursor
    <img alt="?" id="changingFamilySymbol" width="35" height="35" style="cur
    <img alt="?" id="changingSecuritySymbol" width="35" height="35" style="c
</div>
```

Abbildung 3: Ausschnitt aus der index.html – Buttons der Titelleiste

Die Position der Gebäude- und Raumfenster habe ich nach unten rechts verändert und die Schriftgröße minimiert, sodass mehr Platz für die eigentliche Karte entsteht.

3.3.2 Das Menü und die Suchleiste

In dem neuen Design sollten die Suchleiste und ein Menü unterhalb der Informationsbuttons zu sehen sein. Für die Suchleiste mussten dafür Anpassungen in der layout.css vollzogen werden. Ich entschied mich dafür, dass die Suchleiste 60% und das Menü 40% der Bildschirmbreite einnimmt.

```
#search {
    width: 60%;
    display: block;
    position: absolute;
    z-index: 3;
    margin-top: 80px;
    left: 1px;

  }
```

```
.gebmenu {
  width: 20%;
  position: absolute;
  z-index: 1;
  top: 36px;
  right: 20%;
  margin: 0px auto;
  margin-top: 42px
}
.gebmenu > ul {
  list-style: none;
  padding: 0;
  margin: 0 0 0px 0;
}
```

```
.floormenu {
  width: 20%;
  position: absolute;
  z-index: 1;
  top: 36px;
  right: 0%;
  margin: 0px auto;
  margin-top: 42px
}
.floormenu > ul {
  list-style: none;
  padding: 0;
  margin: 0 0 0px 0;
}
```

Abbildung 4: Ausschnitt aus der layout.css und der menu.css – Style der Suchleiste und der Menüreiter

Für das Menü wurden drei neue JavaScript-Klassen und ein neuer CSS-Stylesheet angelegt (menu.css). Das Menü besteht aus zwei Reitern, dem Inforeiter und dem Etagenwechselreiter. Ersteres soll die Gebäude- und Rauminformationsfenster an- und ausschalten und zweiteres das Etagenwechselfenster ersetzen. Eine der neu angelegten Javascript-Klassen (widget_toggle.js) enthält die Funktion für den ersten Reiter. Wenn das Gebäudefenster nicht zu sehen ist, wird das Räumfenster geschlossen und das Gebäudefenster angezeigt, ansonsten werden beide geschlossen. Wenn das Raumfenster nicht zu sehen ist, werden sowohl das Raum- als auch das Gebäudefenster angezeigt, ansonsten werden beide geschlossen. Die Erkennung, ob das jeweilige Fenster angezeigt wird oder nicht, wird durch boolean-Variablen organisiert.

```
function toggle_widget(widget_toggle){

    switch(widget_toggle){
    case 'gebinf':

        if(gebinf_toggle == false){
            gebinf_toggle = true;
            roominf_toggle = false;
            RoomInfo.collapse();
            BuildingInfo.expand();
        }else{
            gebinf_toggle = false;
            roominf_toggle = false;
            BuildingInfo.collapse();
            RoomInfo.collapse();
        }
        break;
    case 'roominf':
        if(roominf_toggle == false){
            gebinf_toggle = false;
            roominf_toggle = true;
            BuildingInfo.expand();
            RoomInfo.expand();
        }else{
            gebinf_toggle = false;
            roominf_toggle = false;
            BuildingInfo.collapse();
            RoomInfo.collapse();
        }
        break;
    default:
        alert("DEFAULT");
    }
}
```

Abbildung 5: Die JS-Klasse „widget_toggle.js“

Damit der Etagenwechselreiter funktioniert, muss er als Erstes mit der richtigen Anzahl an Ober- und Untergeschossen gefüllt werden. Dies geschieht in der Funktion „updatefloorselect“ in der nächsten neuen JavaScript-Klasse „floorselectmenu.js“. Die Funktion bekommt die Anzahl der Geschosse übergeben und iteriert über sie, um sie dann in den Reiter zu schreiben; „upadatefloorselect“ wird jedes Mal aufgerufen, wenn ein neues Gebäude ausgewählt wird.

```
function updatefloorselect(ugmax,ogmax){
    menubuilding = "";
    for(var i = ugmax; i < 0; i++){
        var etage = i*(-1) + ".UG";
        menubuilding = menubuilding +
            "<li><a onClick =\"setfloorselect('" + etage +"');\">" + etage + "</a></li>";
    }

    menubuilding = menubuilding + "<li><a onClick =\"setfloorselect('EG');\">EG</a></li>";

    for(var j = 1; j <= ogmax; j++){
        var etage = j + ".OG";
        menubuilding = menubuilding +
            "<li><a onClick =\"setfloorselect('" + etage + "');\">" + j +".OG </a></li>";
    }
    document.getElementById("floorselection").innerHTML = menubuilding;
}
```

Abbildung 6:Die Funktion „updatefloorselect“ der JS-Klasse „floorselectmenu.js“

Wenn eine Etage angeklickt wird, wird die Funktion „setfloorselect“ ausgeführt, welche die Etage übergeben bekommt und sie dementsprechend zeichnet.

3.3.3 Informationen zu Symbolen

Wenn in der nicht mobilen Version des Lageplans ein Symbol auf der Karte angeklickt wurde, beispielsweise eine Eingangstür, dann erschien ein Popup-Fenster, das Informationen zu dieser Tür bereitstellte. Diese Datenstruktur gehört zur ArcGIS API und bietet keine annehmbare Möglichkeit zur Skalierung auf dem Bildschirm. Da die Symbolinformationen allerdings ein essentieller Bestandteil des Lageplans sind, musste eine Lösung zur Anzeige gefunden werden. Ich entschied mich für ein komplett neues Fenster, das die Inhalte des Popupfensters anzeigt und mit einem „Zurück"-Button ausgestattet ist, der es ermöglicht, zum Lageplan zurückzukehren. Um das Vorhaben umzusetzen, nutzte ich die bereits vorhandenen „InfoString"-Methoden, die alle Informationen zu einem Symbol in eine Zeichenkette speichern.

```
// Informationen ins neue Fenster schreiben und zurück Button erstellen
function openInfoWindow(inputString) {

var w = window.open("", "_self", "copyhistory=yes");
//Spinge auf letzte bekannte URL zurück
w.document.write(inputString + "
    <p /><button onclick='javascript:window.location.href = actualWebsite;return false;'>Zurück</button>");

}
```

Abbildung 7: Die Funktion „openInfoWindow" der index.html

Diese Zeichenketten werden dann der neuen Methode „openInfoWindow" übergeben, die ein Fenster in demselben Tab wie der Lageplan öffnet, das die Inhalte und den „Züruck"-Button anzeigt. Um das Zurückspringen zu bewerkstelligen, wird sich vor dem Fensterwechsel die aktuelle Netzwerkadresse gemerkt. Nach dem Klicken des Buttons wird genau diese dann wieder aufgerufen. Sie enthält die Informationen, welches Gebäude, welcher Raum und welche Etage zuletzt ausgewählt waren. Die naheliegendste Lösung, einfach im Verlauf einen Schritt zurückzugehen, war nicht möglich, da alle Bewegungen auf der Karte ein dynamischer Prozess mit Hilfe von JavaScript sind und somit kein wirklicher Verlauf aufgebaut wird.

3.3.4 Verschiedenes und Abschluss

Smartphones haben zwei verschiedene Orientierungen – Portrait (Hochkant) und Landscape (quer). Im Laufe der Arbeit an dem Lageplan wurde klar, dass die Umsetzung im Querformat keinen Sinn ergab, da zu viel Platz der Karte verloren geht. Ich konzentrierte mich auf eine Optimierung im Portrait-Modus und implementierte, dass den Nutzern der mobilen Webseite dies immer dann mitgeteilt wird, wenn das Smartphone quer gehalten wird.

```
window.addEventListener("orientationchange", function() {

    if (window.orientation == 90 || window.orientation == -90 ) {

        alert("Bitte Portrait Modus benutzen (Hochkant)");
    }
}, false);
```

Abbildung 8: Funktion zur Erkennung von Änderung der Orientierung in der index.html

In Kapitel 3.1 habe ich die schnelle Geschwindigkeit als eine Anforderung an eine mobile Webseite genannt. Um genau diese in der neuen Version des Lageplans zu erhöhen, sollte nun überflüssig gewordener Code entfernt werden. Dieses Themenfeld übernahm mein Kommilitone Stefan.

Als Letztes programmierte ich noch eine Umleitung auf die mobile Webseite ein. Die Erkennung, ob der Nutzer der Webseite ein Smartphone benutzt, läuft über die Breite des Displays. Wenn diese unter 800 ist, wird zur mobilen Version weitergeleitet.

```
<script type="text/javascript">
    if (screen.width <= 800) {
    window.location = "http://www.geodata.uni-goettingen.de/mobileplan";
    }
</script>
```

Abbildung 9: Codeschnipsel zur Erkennung ob Weiterleitung zur mobilen Webseite

Nachdem alle diese Schritte erfolgreich implementiert wurden, war die mobile Version des Lageplans abgeschlossen (siehe Abb.9).

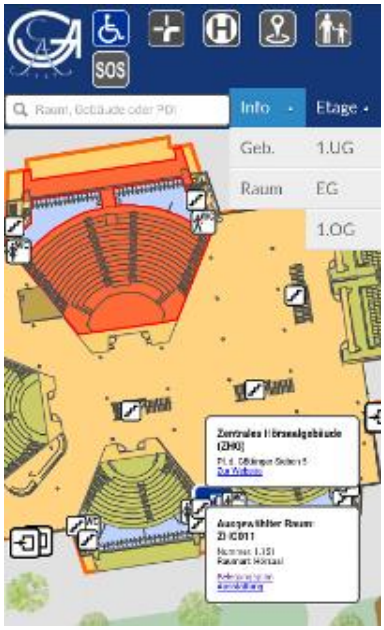

Abbildung 10: Layout der fertigen mobilen Version des Lageplans zur Barrierefreiheit

4 Fazit

Nachdem die Einarbeitung in JavaScript, HTML, CSS und in die ArcGIS API erfolgreich verlief und die Anforderungen an die mobile Webseite klar gemacht wurden, war die Umsetzung der Ziele ohne große Schwierigkeiten möglich. Vor allem das Arbeiten mit ArcGIS verhalf mir zu vielen Einblicken, die ich für meine Bachelorarbeit nutzen konnte. Dort arbeite ich mit der ArcGIS API für Android, welche ebenfalls in einer anderen Abschlussarbeit verwendet wurde, um den Lageplan als Applikation zur Verfügung zu stellen.

Da ich im Studium ausschließlich mit der höheren Programmiersprache Java und etwas in C programmiert habe, konnte ich mit dem Arbeiten einer Scriptsprache meinen Horizont erweitern. Gleichzeitig war es aber auch eine Herausforderung, da nur Interpreter in Form des Browsers und keine Compiler für sie existieren.

Im Laufe des Projekts konnte ich immer wieder theoretisches Wissen aus dem Studium in der Praxis anwenden. So hat mein Wissen über Datenbanken geholfen, die SQL-Anfragen an den ArcGIS-Server zu verstehen und selber zu programmieren.

Abschließend lässt sich sagen, dass ich sehr viel Wissen aus dem Praktikum mitgenommen habe und mir in Zukunft vorstellen kann, weiter mit ArcGIS in seinen zahlreichen Einsatzbereichen weiterzuarbeiten.

Literaturverzeichnis

SELFHTML, (1998-2014). *Grundlagen/Technologien/CSS*, unter: http://wiki.selfhtml.org/wiki/Doku:Grundlagen/Technologien/CSS.

STATISTA, (2015). *Statistiken und Studien zum Thema Smartphones*, unter: http://de.statista.com/themen/581/smartphones/.

UNIVERSITÄT GÖTTINGEN, (2012). *Lageplan zur Barrierefreiheit*, unter: http://www.uni-goettingen.de/de/lageplan-zur-barrierefreiheit/476871.html.