Multimedia Tour Editor

Name: Philipp Pfeiffer

Matrikelnummer: 0809357

Studienkennzahl: E 033 532

Projektbeschreibung

Aufgabenstellung

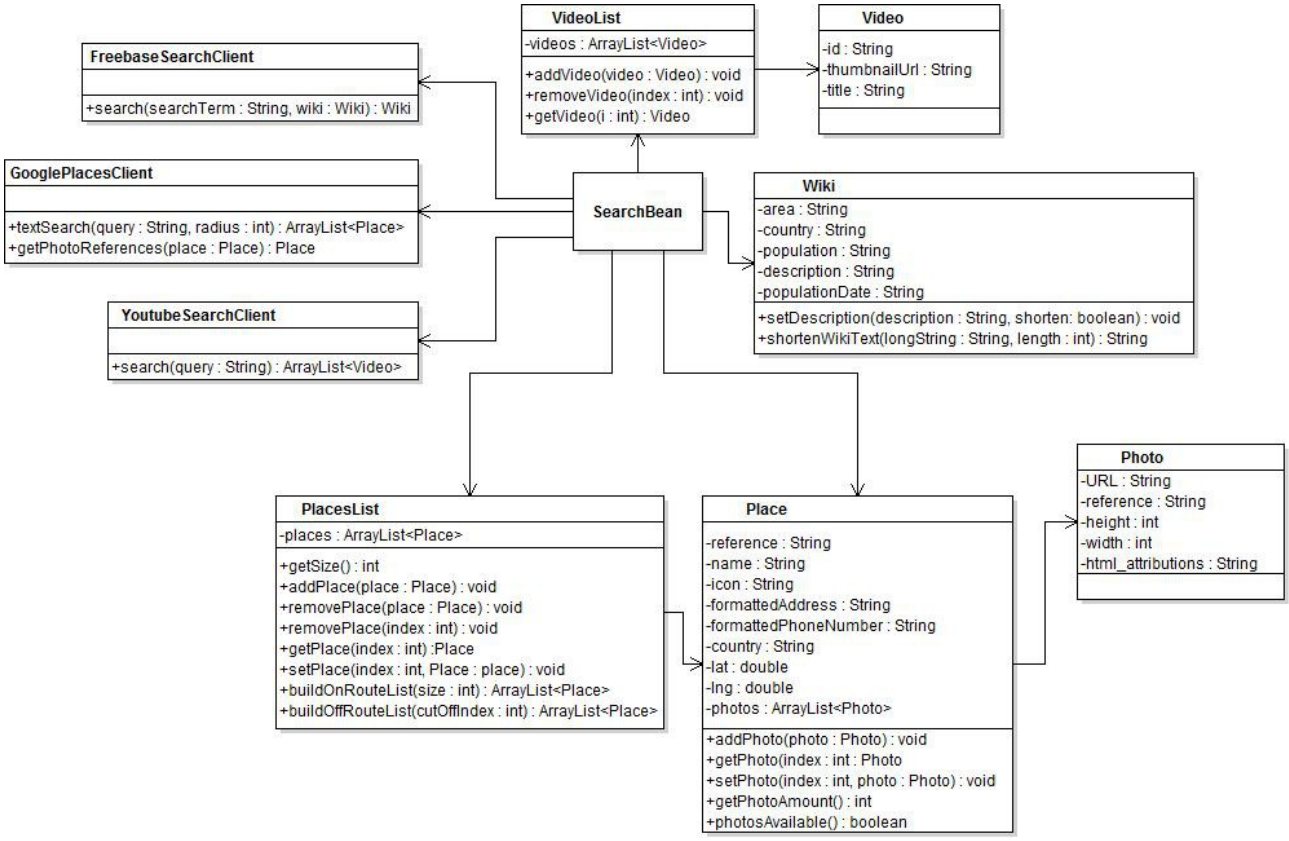
Zu entwickeln ist eine Website, über die Benutzer eine Multimedia Tour durch eine Region oder Stadt unternehmen können. Ein Benutzer soll nach einer Stadt suchen können und als Ergebnis Medieninhalte, die mit der Stadt in Verbindung stehen und für den Benutzer relevant sind, präsentiert bekommen.

Dazu soll zumindest eine Karte zählen, auf der eine Route entlang der wichtigsten, relevanten Orte der Stadt zu sehen ist. Die gefundenen Orte können zum Beispiel für Touristen relevante Sehenswürdigkeiten, Museen oder lokale Besonderheiten sein. Die Route entlang dieser sehenswerten Örtlichkeiten soll außerdem editierbar sein, so dass der Benutzer verschiedene Routen durch die Stadt planen kann.

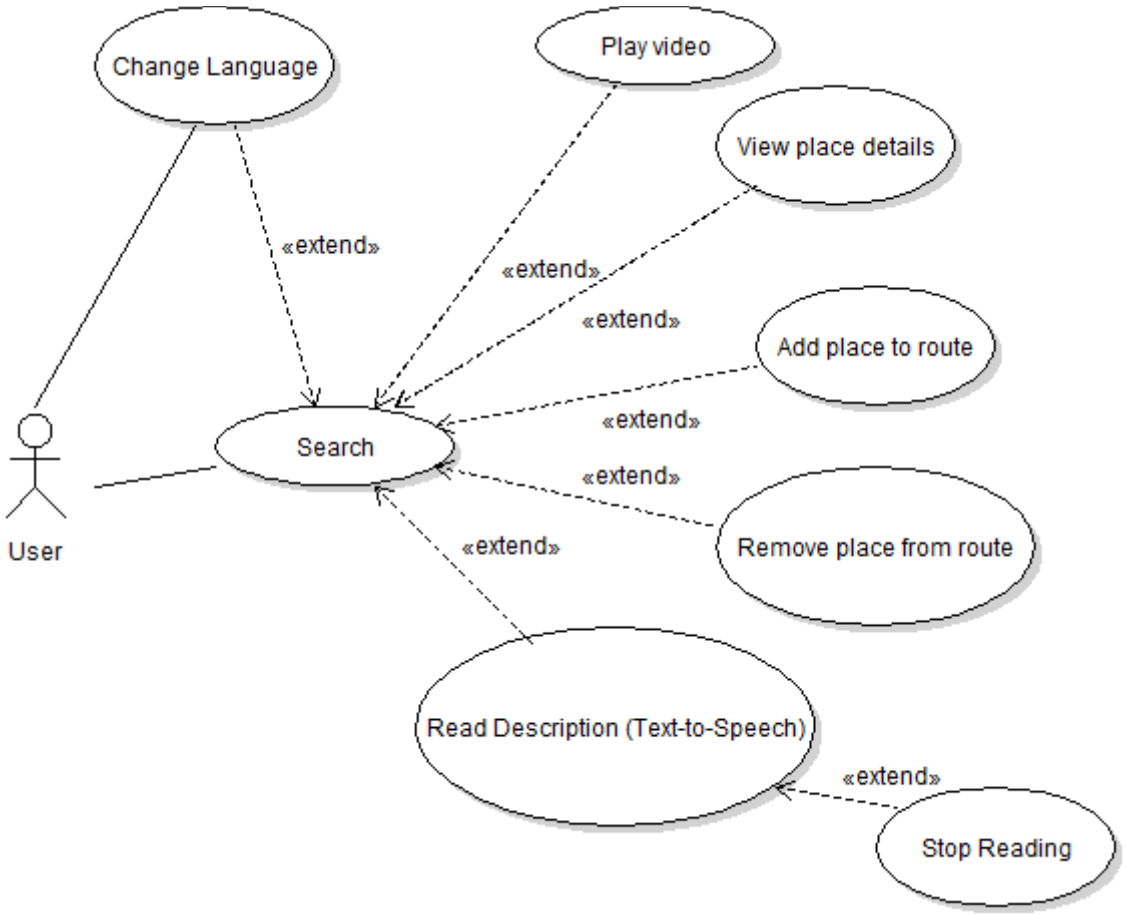
Neben der editierbaren Route durch die Stadt sollen noch weitere relevante Medieninhalte präsentiert werden. So sollen, zum Beispiel, relevante Videos, Bilder oder Informationen über die gesuchte Stadt sichtbar sein.

Entwurfsdiagramme

Klassendiagramm des Servers



Use Case Diagramm



Zeitliste

Datum	Arbeitsstunden	Aktivität
02.08.2013	1:15	Research, Planung
10.09.2013	2:00	Research, Planung, Errichtung der Grundstruktur im IDE
11.09.2013	3:30	Research, Implementierung: SearchBean
12.09.2013	5:30	Research, Implementierung: SearchBean, Web Service Clients
13.09.2013	1:30	Research, Implementierung: SearchBean, Web Service Clients
17.09.2013	4:00	Implementierung: SearchBean, Web Service Clients
19.09.2013	4:30	Research, Implementierung: SearchBean, Web Service Clients
20:09.2013	4:00	Implementierung: SearchBean, Web Service Clients
23.09.2013	0:30	Research
25.09.2013	2:30	Implementierung: SearchBean, Web Service Clients
26.09.2013	1:00	Research, Debugging
16.10.2013	1:30	Debugging
19.10.2013	2:15	Debugging, Implementierung: SearchBean, Web Service Clients
20.10.2013	2:30	Implementierung: SearchBean, Web Service Clients, HTML
22.10.2013	4:30	Implementierung: Web Service Clients, HTML
23.10.2013	1:00	Research, Planung
24.10.2013	0:30	Research, Planung
27.10.2013	4:00	Implementierung: Web Service Clients, HTML
19.01.2014	2:00	Research,
20.01.2014	2:00	Implementierung: Web Service Clients, HTML
22.01.2014	3:00	Implementierung: HTML, Javascript
23.01.2014	1:30	Implementierung: Javascript, HTML
23.02.2014	5:00	Implementierung: HTML, Javascript
26.02.2014	4:00	Implementierung: HTML, Javascript
27.02.2014	2:30	Debugging, Implementierung: HTML, Javascript
28.02.2014	0:30	Debugging
22.04.2014	3:45	Implementierung: HTML, Javascript
23.04.2014	1:45	Implementierung: Web Service Clients, HTML
15.05.2014	1:00	Implementierung: HTML
16.05.2014	1:30	Implementierung: HTML
20.05.2014	2:00	Implementierung: HTML, Javascript
21.05.2014	3:15	Debugging, Implementierung: HTML
24.05.2014	0:45	Research, Planung
02.06.2014	1:30	Planung, Implementierung: HTML
03.06.2014	1:00	Implementierung: HTML
05.06.2014	1:30	Implementierung: HTML
07.06.2014	1:30	Implementierung: HTML, Javascript
09.06.2014	1:00	Implementierung: HTML, Javascript
24.06.2014	1:30	Implementierung: HTML, Javascript
19.07.2014	2:00	Debugging, Implementierung: HTML, Javascript

20.07.2014	1:30	Debugging, Implementierung: HTML, Javascript
21.07.2014	2:00	Implementierung: HTML, Javascript
23.07.2014	1:30	Debugging
24.07.2014	2:00	Debugging
31.07.2014	2:30	Implementierung: HTML
01.08.2014	1:00	Implementierung: HTML
02.08.2014	2:30	Implementierung: HTML
03.08.2014	2:00	Implementierung: HTML
04.08.2014	3:00	Implementierung: HTML, Javascript
07.08.2014	1:30	Implementierung: HTML
11.08.2014	2:00	Debugging, Implementierung: HTML
12.08.2014	1:00	Debugging, Implementierung: HTML
13.08.2014	1:30	Debugging
14.08.2014	3:00	Implementierung: HTML
17.08.2014	2:00	Debugging
18.08.2014	1:30	Debugging, Implementierung: HTML
29.08.2014	1:00	Code Cleanup
30.08.2014	3:00	Code Cleanup
01.09.2014	2:30	Debugging
03.09.2014	4:00	Code Cleanup, Debugging
04.09.2014	3:00	Code Cleanup, Kommentare vervollständigen
05.09.2014	6:00	Code Cleanup, Kommentare vervollständigen, Abschlussbericht
06.09.2014	1:30	Abschlussbericht
07.09.2014	4:00	Abschlussbericht
08.09.2014	4:30	Abschlussbericht, Image Attributions hinzufügen
09.09.2014	5:00	Debugging
10.09.2014	5:00	Text-to-Speech hinzugefügt, Debugging
05.10.2014 bis 26.10.2014	25:00	Debugging, Distanzzähler hinzugefügt, Farbassoziation bei Bildern hinzugefügt
Total:	186:00 Stunden	

Implementierung

Systemumgebung

OS: Windows 7 Professional, 64 Bit

Prozessor: Intel(R) Core(TM) i5-2500K CPU @ 3.30GHz 3.30GHz

RAM: 8 GB

IDE: NetBeans 7.3.1

Server: Glassfish 4

Bibliotheken

Verwendete Bibliotheken und Toolkits sind in der Datei pom.xml spezifiziert und werden im Build Prozess geladen. Manche Bibliotheken werden nur auf XHTML seiten verwendet und werden daher dort spezifiziert (z.B.: JSF2, Google Maps API, etc).Verwendet werden:

- Java(TM) EE 7 Web Specification APIs
- Google API Client 1.17.0-rc
- JSON 20090211
- JSON Simple 1.1.1
- JSON Path 0.8.1
- Youtube Data V3-rev40-1.13.2-beta
- Google HTTP Client 1.17.0-rc
- jquery v1.9.1
- JSF 2
- meSpeak
- jquery
- Google Maps API

Besonderheiten

Im Groben besteht das Projekt aus zwei Hauptbestandteilen: Dem Serverteil, der Anfragen übernimmt, im Hintergrund Web Services kontaktiert und Daten kapselt und dem Browserteil, in dem die gefundenen Daten dem Benutzer mithilfe von HTML Webpages präsentiert werden und mithilfe dessen der Benutzer mit dem System interagiert.

Der Serverteil ist um die Klasse SearchBean.java aufgebaut. Searchbean.java dient als Session Scoped Bean und verwaltet sowohl Suchanfragen als auch jene Daten die gefunden wurden und zur Verfügung gestellt werden sollen. Dafür stellt Searchbean.java Methoden zur Verfügung, über die der Browserteil auf Daten zugreifen kann. Searchbean instanziert alle verwendeten Web Service Clients, verwaltet und speichert alle verwendeten Entities über den Lauf der Session.

Der Browserteil besteht im Grunde aus index.xhtml und editor.xhtml. Index.xhtml beinhaltet ein Suchfeld und soll beim ersten Besuch der Seite angezeigt werden. Von dieser Seite aus sucht der Benutzer nach einer Stadt und wird dann zu editor.xhtml geführt. Editor.xhtml bietet eine Übersicht über alle gefundenen Daten und beherbergt Javascript Code der für das Laden der Karte notwendig ist. Der Benutzer kann die Karte auch über diese Seite editieren, oder eine neue Suche starten, die

wiederum zu editor.xhtml führt, jedoch mit neuen Daten.

Sowie index.xhtml als auch editor.xhtml bestehen aus verschiedenen Komponenten. Index.xhtml inkludiert die Komponente searchComponent.xhtml. Editor.xhtml inkludiert infoComponent.xhtml, mapComponent.xhtml, navComponent.xhtml, placeImageComponent.xhtml und videoComponent.xhtml.

Da keine Daten permanent gespeichert werden, wird keine Datenbank benötigt. Daten werden serverseitig in Entities gekapselt und nur für die Länge der Session gespeichert und von SearchBean.java verwaltet. Wird eine neue Suche getätigt, werden sämtliche vorher gefundenen Daten überschrieben und mit neuen ersetzt.

Es werden folgende Web Services verwendet (mit Anzahl der erlaubten Requests pro Tag):

- Google Freebase API (100 000 Requests)
- Google Maps JavaScript API v3 (1 000 000 Requests)
- YouTube Data API v3 (50 000 000 Requests)
- Google Places API (100 000 Requests)

Die Web Services unterliegen Nutzungslimitierungen. So gibt es für jeden Web Service eine tägliche Nutzungsquote, die man nicht überschreiten darf. Das hat anfangs ein Problem beim Testen dargestellt, da die tägliche Quote für Google Places zuerst bei 1000 lag und das Programm sehr stark vom Google Places Service nutzen macht. Die Quote wurde auf 100000 erhöht und sollte für Testzwecke keine Probleme bereiten.

MeSpeak.js wird verwendet um Beschreibungen der gefundenen Städte als Sprache auszugeben. Klickt man unterhalb der Beschreibung auf „Sprechen“ oder „Speak“, wird der Text als Sprache ausgegeben. Es kann, je nach länge der Beschreibung ein paar Sekunden dauern, bis die Sprache ausgegeben wird.

Known Bugs

Verwendet man das Text-To-Speech Feature, kann es sein, dass es zu kurzen Wartezeiten kommt. Je nach Browser kann auch vorkommen, dass hier eine Warnung angezeigt wird. In dem Fall sollte man das Skript seine Arbeit verrichten lassen. Sobald die Bearbeitung des Strings fertig ist, sollte die Sprachausgabe folgen.

Potenziale für Weiterentwicklungen

Es sollte relativ leicht möglich sein weitere Web Services einzubinden um neue Medieninhalte anzuzeigen. Eine Möglichkeit wäre zum Beispiel Soziale Medien wie Twitter oder Facebook einzubauen. Vor allem Twitter wäre eine gute Möglichkeit, da man eine liste von Posts anzeigen könnte, die einen Hashtag verwenden, der relevant zur momentanen Suche ist.

Ein Problem beim Ausbau des Systems mit weiteren Medieninhalten ist jedoch der Platzmangel. Da sich das System momentan stark auf das Anzeigen von Bildern stützt, um dem Benutzer einen Eindruck der Örtlichkeiten zu verschaffen, wird viel Platz eingenommen. Man könnte editor.xhtml auf verschiedene Arten strukturieren und neu anordnen um zu sehen welche Anordnung den besten Überblick verschafft, ohne auf Informationen zu verzichten. Übertreibt man es jedoch mit den angezeigten Medieninhalten, riskiert man Feature Overload und die Website wird unangenehm zu benutzen.