TECHNISCHE UNIVERSITÄT WIEN

Sighted! - die mobile Lösung für das interaktive Mapping von Gebäuden

Fakultät für Mathematik und Geoinformation
am
Department für Geodäsie und Geoinformation
der Technischen Universität Wien

Technischer Bericht

vorgelegt von

Christoph Weichselbaum, 1126113
Christof Kocer, ???

am 29. Februar 2016

Inhaltsverzeichnis

1 Einleitung

Im Rahmen der LVA Mobile GIS Anwendung wird eine Applikation entwickelt. Dabei sollen alle während des Semesters gelernten elementaren Bestandteile einer mobilen Anwendung in Betracht gezogen werden. Es soll, unter Verwendung von gängigen Web-Technologien, speziell auf die Bedürfnisse von mobilen Anwendungen eingegangen werden.

1.1 Entwicklungsziele

Auf Basis der im Semesterstoff erarbeiteten Lehrziele, soll die fertige Lösung folgende elementare Komponenten implementieren.

- Verwendung einer gängigen Javascript-API für responsive mobile Design
- Verwendung der Overpass-API für die Abfrage von OSM-Daten
- Feature-Implementierung unter der Verwendung der Smartphone-Sensorik
- Datenbankanbindung an CouchDB
- Offline-Synchronisation der gespeicherten Daten

Des weiteren wurde bei der Implementierung stets auf folgende Unterpunkte geachtet.

- Implementierung des MVC-Pattern
- Modularisierte Architektur
- Skallier- und Erweiterbarkeit
- Konfigurierbarkeit
- Automatisiertes Assembling der Android-APK Datei

1.2 Methodik

Die Entwicklungsziele bilden die Grundlage und Rahmenbedingungen zur Implementierung für den Entwurf einer Android-Applikation namens 'Sighted!'. Die Methodik zur Entwicklung der vorgestellten Problemstellung umfasst folgende Punkte:

- Aufsetzen einer Hybriden-Android Applikation, die den Zugriff auf hardwarenahe Sensorikfunktionen mittels Javascript ermöglicht.
- Konfiguration des Assembling-Prozesses der Applikation mittels Gradle
- Aufsetzen der grundlegenden Projektstruktur der Web-Applikation

- Konfiguration der Web-Applikation mittels Grunt
- Einbindung von gängigen Web-Frameworks: jquery (mobile), bootstrap, requireJS, ol3, pouchdb
- Implementierung des Grundgerüsts der Web-Applikation
- Implementierung eines Wrappers für die Verwendung der Overpass-API
- Implementierung eines Wrappers für die Verwendung der pouchDB-API
- Implementierung der Anwendungslogik und Design der Anwendung

2 Technolien

Im Rahmen der Implementierung der Anwendung werden eine Reihe von Frameworks und APIs verwendet um die Anwendungslogik effizient nach gängigen Patterns zu strukturieren. Dies ist besonders hilfreich, wenn man - wie auch im Enterpris-Umfeld - im Team an Projekten arbeitet. Die Teammitglieder können sich auf die Feature-Entwicklung konzentrieren und müssen sich nicht mit komplexen internen Anwendungsprozessen beschäftigen (e.g Templating-Engine), die bestimmte Frameworks out-of-the-box unterstützen.

2.1 Back-End

Das Endprodukt soll eine hybride Android-Applikation sein. Da diese Applikation lediglich auf den GPS-Sensor des Smartphones zugreift und keine komplexen hardwarenahen Operationen ausführt, wird die WebView-API von Android verwendet. Aus diesem Grund müssen die Webinhalte von einem internen Assets-Ordner aus geladen werden. Dieser Prozess wird mittels des Build-Management-Tools Gradle automatisiert.

2.1.1 Gradle

Gradle ist ein auf Java basierendes Build-Management-Tool und wurde für Builds von Softwaresystemen entworfen. Besonders während der Entwicklungszeit werden sehr viele Änderungen am Quellcode vorgenommen, was eine ständiges Kopieren von neu erstellten Dateien in den Assets-Ordner bedeuten würde. Gradle ermöglicht es nur die Teile einer Software zu bauen, welche verändert wurden oder auf veränderten Teilen beruhen. Des weiteren können bestimmte Tasks angelegt werden, die beim Build parallel laufen (e.g. Tests). Resultat ist eine wesentlich höhere Geschwindigkeit beim Entwicklungsprozess.

2.1.2 WebView

Bei WebView handelt es sich um eine Java-Klasse, die das Rendering von Webinhalten ermöglicht. Es verwendet die WebKit rendering engine für die Darstellung und implementiert Basisfunktionen eines Webbrowsers.

2.2 Front-End

Während der Entwicklung werden die Vorteile von Web-Technologien ausgenützt. Anstatt jedes mal die Inhalte in die Android-Applikation zu laden und die Software dort zu testen, wird ein eigenes Submodul angelegt und das Front-End-Build-Management-Tool Grunt integriert. Des weiteren werden gängige APIs wie requireJS, jquery-mobile, bootstrap für die Anwendungslogig und Responsive-Design integriert. Für die Datenbankanbindung wird pouchDB und für die Abfrage von Geodaten wird ol3 integriert.

2.2.1 Grunt

Grunt ist ein auf node.js basierender Taskrunner und ermöglicht es eine Vielzahl an Plugins zu Nutzen, die die Entwicklung vereinfachen und beschleunigen. Im Rahmen dieser Applikation wird speziellen von der Browser-Synchronisation Gebrauch gemacht. Dabei werden Änderungen im Quellcode sofort im Browser reflektiert. Die auszuführenden Tasks werden mittels einer zentralen Konfigurationsdatei (gruntfile.js) initialisiert.

2.2.2 requireJS

Eine freie Javascript API für die Implementierung von asynchroner Moduldefinition (AMD). Dies ermöglicht die objektorientierte Codestrukturierung und das Laden der Dateien, wenn sie wirklich im Browser benötigt werden.

2.2.3 jQuery Mobile

Eine freie Javascript API, die DOM-Navigation und - Manipulation speziell für mobile Anwendungen optimiert. Diese API wird im Rahmen der Applikation speziell für die Navigation und bestimmte Design-Elemente verwendet.

2.2.4 Bootstrap

Eine freie Javascript API für die gezielte Manipulation von CSS-Elementen. Diese API wird im Rahmen der Applikation speziell für bestimmte Design-Elemente verwendet.

2.2.5 PouchDB

Eine freie Javascript API die für die Ergänzung der CouchDB API zur Unterstützung von Offline-Funktionalitäten verwendet wird. Die API läuft im Browser und speichert die Dokumente im Web Storage des Browsers.

2.2.6 Overpass

Eine freie Javascript API für die gezielte Abfrage von OpenStreetMap-Daten. Diese API wird im Rahmen der Applikation speziell für die Abfrage von Gebäudeattributen verwendet.

3 Datemanagement

INTRO

3.1 Datenmodel

Wie schaut unser Datenmodell am client aus. Wie schaut unser Datenmodell in coouchdb aus
Wie schaut das Mapping aus?

3.2 Prozess der Abfrage

welche prozesse werden durchlaufen

3.3 Prozess der Speicherung

welche prozesse werden durchlaufen

4 Implementierung

Die Implementierung stützt sich auf die in 1.1 definierten Entwicklungsziele. Zuerst soll auf die Projektstruktur und das Assembling eingegangen werden. Danach wird die Integration verschiedener APIs und die eigentliche Anwendungs-Architektur besprochen.

4.1 Projektstruktur und Assembling

Die Anwendung wird in zwei grundlegende Module gegliedert. Diese bilden das Back-End und Front-End und sind anhand Graphik

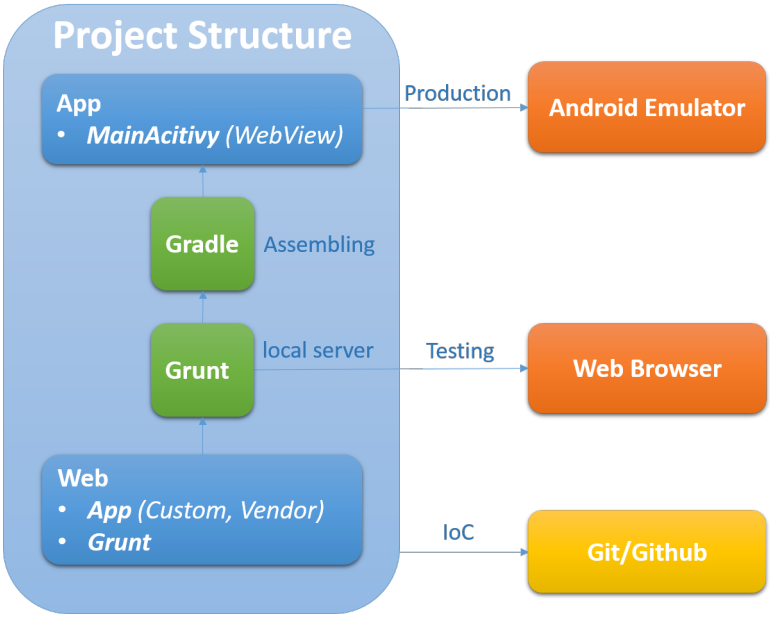

Abbildung 4.1: Sighted! - Projektstruktur

Das erste Modul 'App' bildet das Back-End. Dieses startet das Hauptprogramm mittels der Java-Klasse MainActitivy. Dort wird der Webview initialisiert und verschiedene hardwarenahe Konfigurationen getroffen. Zu diesen Konfigurationen zählen beispielsweise bereitgestellt Rechte, die die Steuerung der Anwendung durch Javascript betreffen.
Das zweite Modul 'Web' bildet das Front-End. Dort werden alle Inhalte des Web-Projekts gespeichert, wie Anwendungslogik und verschiedene Javascript-APIs. Dieses Modul kann direkt mittels eines bereitgestellten HTTP-Servers in den Browser geladen werden. Speziell

für Entwicklungszwecke wurde der vom Build-Management-Tool Grunt zur Verfügung gestellte HTTP-Server verwendet. Dieser erlaubt durch Browser-Synchronisation eine direkte Reflektion von geändertem Quellcode im Browser. Die hauptsächliche Entwicklung der Anwendungslogik findet auf diese Weise statt.
Für das Laden des Webinhalts des Front-End-Moduls in das Back-End-Modul wird das Build-Management-Tool Gradle verwendet. Es wurde ein Task definiert, der die Webinhalte automatisiert in den Assets-Ordner der Android-Applikation lädt und eine ausführbare Android-APK Datei liefert.
Da die Entwicklung hauptsächlich im Team stattfindet, wird das Versionsverwaltungssystem Git in Kombination mit Github verwendet. Dort sind alle Beiträge zum Projekt dokumentiert.

GitHub-Link:
https://github.com/ChriWe/MobileGIS

4.2 Architektur

Die Architektur der Anwendung folgt dem Model-View-Controller Pattern. Dabei wurden logisch unabhängige Teile des Quellcodes physisch getrennt um die Anwendung zu modularisieren. Dies soll in weiterer Folge die Skallier- und Erweiterbarkeit gewährleisten.

4.2.1 MVC-Pattern

Das MVC-Pattern wird als Strukturierungsmodell verwendet um die Einheiten des Datenmodells, der Programmsteuerung und der Präsentation voneinander zu trennen. Ziel ist ein flexibler Programmentwurf, der spätere Änderungen oder Erweiterungen erleichtert und die Wiederverwendbarkeit von Einzelkomponenten gewährleistet. Die Geschäftslogik wird in dieser Applikation weitestgehend im Controller implementiert.
Die grundlegende Vorgehensweise des Musters wird anhand folgender Graphik verdeutlicht:

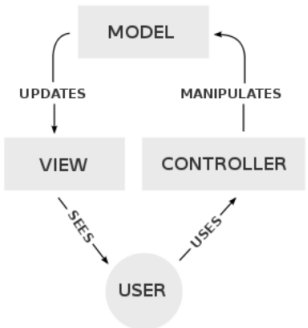

Abbildung 4.2: MVC - Pattern

- Model: enthält die zur Verfügung gestellten Daten
- View: ändert die Präsentation anhand von Änderungen im Model
- Controller: Manipuliert das Model und die View

4.2.2 Anwendungs-Architektur

5 Diskussion

Abbildungsverzeichnis