1 Architekturbegründung

Aufgrund der Tatsache, dass die Mobilität der Reiter unterstützt werden soll, bietet sich uns die einzige Option einer mobilen Anwendung. Vielfältige Betriebssysteme bieten sich auf dem Markt zur Programmierung von mobilen Anwendungen an, unter Anderem Google Android, IOS, Windows Phone, Blackberry OS, Firefox OS uvm.

Blackberry hat ihr Betriebssystem perfekt auf Business-Anwender abgestimmt. Zudem ist die Nutzung der E-Mail Dienste leistungsstark. Eingeschränkte Multimedia-Eigenschaften benachteiligen das Betriebssystem. Die Vorteile von Blackberry OS sind nicht für unser Projekt von Vorteil.

Das Windows Phone bietet eine sehr hohe Performance und eine simple intuitive Bedienung. Jedoch ist die App-Auswahl deutlich kleiner als bei Android und Apple.

Das Betriebssystem IOS von Apple bietet auch eine sehr hohe Performance. Jedoch ist die Nutzung verschiedener Schnittstellen von Google nicht nutzbar. Eine Einarbeitung in die Programmiersprache Swift kann enorm viel Zeit in Anspruch nehmen.

Da bei Android**?** mit der Programmiersprache Java programmiert werden kann und die Google Produkte perfekt eingebunden werden können fällt die Entscheidung auf Android. Im Zusammenhang mit Android ist eine Programmierung mittels Java und XML die beste Möglichkeit, das Projekt umzusetzen, da das Projekt Java voraussetzt und XML in dem Modul Web-basierte Anwendungen praktisch schon umgesetzt wurde.

Die zentrale Datenerhaltung soll auf dem Server stattfinden. Dadurch können Datenverluste vermieden werden. Durch das Architekturmodell REST - Representational State Transfer - für verteilte Systeme, soll die Anwendung ihre Daten publizieren und abgreifen können. Die Clients sollen auf die vom Server bereitgestellten Ressourcen durch das Transportprotokoll HTTP zugreifen können. Im Gegensatz zur Webservice SOAP - Simple Object Access Protocol - bietet REST nicht so komplexe XML-Nachrichten, welches ein wesentlicher Nachteil von SOAP ist. **?**

Als Kommunikationsmittel und gleichzeitig als Push-Dienst soll Google Cloud Messaging **?** verwendet werden. Die Gründe hierfür beruhen sich auf der Tatsache, Google Cloud Messaging ein kostenloser Dienst ist und es genügend Informationen zur Implementierung vorhanden sind, die bei der Umsetzung von Bedeutung sein können, da es sich um einen erstmaligen Kontakt mit Android handelt. Durch Google Cloud Messaging wird die synchrone und asynchrone Kommunikation zwischen Serverinstanzen und der mobilen Endgeräte ermöglicht.

Da die Kommunikation der Benutzer im System untereinander ermöglicht werden soll, ist XMPP - Extensible Messaging and Presence Protocol - eine Möglichkeit, dies geschehen zu lassen. XMPP eignet sich nicht nur für große, öffentliche Nutzergruppen zur Kommunikation, sondern auch für kleine Nutzergruppen.

XMPP-CLients sind für alle gängingen Architekturen und Systeme frei verfügbar. Die Interaktion mit den XMPP-Servern sollte keine Problematik aufweisen, da aufgrund der freien Verfügbarkeit von XMPP, für alle Programmier- und SKriptsprachen vorhandene Klassen und Funktionen verfügbar sind. Die Tatsache, dass sich XMPP nicht nur als Instant-Messaging-Protokoll eignet darf nicht aussen vor gelassen werden. Zugleich handelt es sich um einen Status- und Informationsprotokoll.

Als Framework bietet sich am Besten die asmack an, da diese für die Programmierung mittels Android angepasst wurde.**?**

Die Bereitstellung der Wetterdaten soll durch die Schnittstelle openweather API **?** realisiert werden. Die openweather API ermöglicht einen kostenlosen Zugriff auf die Wetterdaten, solange es sich nicht um ein kommerzielles Produkt handelt. Es soll lediglich die Herkunft der Wetterdaten im Endprodukt sichtbar sein. Ein Nachteil ist, dass die kostenlose Schnittstelle auf 10000 Zugriffe im Monat eingeschränkt ist. Jedoch sollten die Zugriffe völlig ausreichen. Ein weiterer Grund für die Wahl der Wetter API ist, dass für die Anfragen an die Schnittstelle ein einfaches REST-Format verwendet wird. Die Wetterinformationen sollen durch die HTTP-Operation GET abgerufen und im XML-Format herausgegeben werden.

Zur Routenplanung stehen zwei Alternativen zur Verfügung, zum Einen Google Maps und zum Anderen die OpenStreetMap. Die OpenStreetMap ist flexibler im Gegensatz zu Google Maps und bietet eine grosse Bandbreite an Informationen in den Karten. Jedoch überragt Google Maps im Design- und Farb-Schema. Die Städte-Namen werden in Google Maps so angezeigt, dass Menschen, die nicht die Europäischen Buchstaben beherschen, in ihren eigenen Ländern keine Probleme haben, da die Buchstaben sich automatisch je nach Land anpassen. Die OpenStreetMap verfügt eine solche Funktionalität nicht. Ausserdem sind die Karten von der OpenStreetMap eng gehalten, sodass die Karten einige Adressen, Parks oder beispielsweise sogar wichtige Gebäude wie Krankenhäuser nicht enthalten. Daraus ist ab zu leiten, dass eventuell wichtige Routen zum Reiten fehlen können. Diese relevanten Fakten sollten für das Planen der Routen, durch die Verwendung von Google Maps API sprechen. Die API ist weit verbreitet und stellt Karten für mobile Anwendungen zur Verfügung. Zu dem ist die API völlig kostenlos und ermöglicht als Routenplaner 2500 Anfragen pro Tag. Karten können individuell angepasst werden. Das heisst, dass Daten wie zum Beispiel vorhandene Routen oder Bilder beliebig hervorgehoben werden können. Zudem sollte erwähnt werden, dass bereits vorhanden Routen zum Reiten in Google Maps vorhanden sind.**?**