# Architekturbegründungen

Dieses Artefakt wurde nach der Phase der Domänenanalyse durchgeführt. Die Ausgangssituation zur Konzeption der Systemarchitektur bestand in Erkenntnissen über Nutzungskontexte sowie den durch die Veranstaltung gegebenen Rahmenbedingungen. Die getroffenen Architekturentscheidungen und Abwägungen sollen im Folgenden dargelegt werden.

## Inhaltsverzeichnis



# Mobile Clients

Der Gegenstand der Verteilung von Lebensmitteln impliziert bereits ein gewisses Maß an Mobilität der Clients im System. Da die geplante Anwendungslogik vor allem in der Organisation von Abholungen von Lebensmittelspenden durch die Tafeln, dem Sammeln von Lebensmitteln durch Privatpersonen sowie mobiler Sammelstellen besteht ist Mobilität der Clients zur Lokalisierung unverzichtbar. Um eine möglichst weite Verbreitung des Systems sicherstellen zu können wurde Android, welches mit über 80% Marktanteil [1] das meistgenutzte mobile Betriebssystem darstellt, als Zielplattform ausgewählt. Außerdem besteht mit der Programmierung von Java deutlich mehr Erfahrung im Projektteam als mit der Programmierung von Objective C unter iOS, andere mobile Betriebssysteme verfügen über zu geringe Marktanteile um von Relevanz zu sein.

Von einer Konzeption als WebApp wurde aus diversen Gründen abgesehen.
Die Verteiltheit der Anwendungslogik besteht in einer Analyse der aktuellen Fortbewegungssituation (genauere Beschreibung siehe unter "Situationsanalyse") kombiniert mit einem Abgleich der Angebotssituation und

bestehenden Zuständigkeiten auf einem Dienstgeber. Auf Basis dieses Abgleichs werden dem mobilen Teilnehmer dann Vorschläge zur Beteiligung an einer Spendensammelaktion oder einem Spendentransport generiert. In einer Dienstgeber-Dienstnutzer Architektur einer möglichen WebApp, bei der mobile Endgeräte über einen Browser auf das System hätten zugreifen müssen, hätte ständige Übertragung von Standortparametern und weiteren für die Situationsanalyse benötigten (teilweise sensiblen) Daten geschehen müssen. Die Berechnung der Analyseergebnisse wäre auf einer zentralisierten Instanz (dem Dienstnutzer) vollzogen worden, was im Hinblick auf Skalierbarkeit des Systems als kritisch eingestuft wurde. Da gerade in mobilen Nutzungskontexten nicht von einer ständigen Internetverbindung ausgegangen werden kann und das System auch bei Ausfall dieser Verbindung eingeschränkt verwendbar sein sollte kam eine WebApp hier nicht in Frage.

Auch eine Konzeption in einem Peer-to-Peer Netzwerk wurde in Betracht gezogen. Da die geplante Anwendungslogik jedoch auf Aggregation von Daten einzelner Lebensmittelangebote basiert musste es eine Instanz im System geben, der verfügbare Angebote vieler Benutzer bekannt sind. Diese Verwaltungsaufgabe hätte in einem reinen Peer-to-Peer Netzwerk von einer ggf. unzuverlässigen Instanz übernommen werden müssen.

Im Folgenden wird die geplante Anwendungslogik der mobilen Clients erläutert.

### Situationsanalyse

Die mobilen Clients sollen dazu in der Lage sein die Situation des Users möglichst genau und ohne zusätzlichen Aufwand für den User zu erfassen. Dies passiert, indem der Client das eigene Bewegungsprofil auf Verkehrsmittel, oft benutze Routen und geplante oder geschätzte Ankunftszeiten (ggf. bei bekannten Terminen) analysiert. Die Ergebnisse einer solchen Analyse werden als sehr sensibel eingestuft, sodass hier in einem releasefertigen Produkt auf entsprechende Schutzmaßnahmen zu achten wäre.

### Angebotsfilter

Der Client soll die Angebote nach bestimmten Parametern filtern. Diese Parameter müssen teilweise vom Benutzer selbst angegeben werden, wie z.B Unverträglichkeiten und Allergien oder werden vom Client durch die Situationsanalyse automatisiert erfasst. Da diese Daten teilweise sensibel sind, werden sie auf dem eigenen Client gespeichert und müssen somit nicht über ein Netz an den Dienstgeber übertragen werden. (Diese Funktionalität ist nach der Fokussierung auf den Spendentransport nicht mehr in Betracht gezogen worden)

## Pub-Sub Middleware

Nach Analyse des Problemraums wurden einige Vorgänge identifiziert, die eine Event-basierte Architektur nahelegen. Das Anbieten von Lebensmitteln, die Organisation von Sammelaktionen sowie mobilen Essensausgaben bedürfen einer Benachrichtigung eines bestimmten Adressatenkreises im Vorfeld ,bei Start oder Durchführung der Aktion. Zu diesem Zwecke wurde eine Pub-Sub fähige Middleware in der Architektur eingeplant. Auch der Client für die Tafelmitarbeiter ist an das Pub-Sub System angebunden, damit die Tafeln zeitnah Benachrichtigt werden wenn Spenden zur Abholung zur Verfügung stehen. Damit können Spenden gegebenenfalls noch in aktive Abholrouten der Tafeln eingeplant werden. Außerdem haben die Tafeln so

eine Kommunikationsplattform mit der sie viele Leute erreichen können, um bspw. zu Spendenaktionen aufzurufen, wie es beispielsweise vor Weihnachten oft getan wird.

Da im Projektteam bereits Erfahrung mit dem PubSub System Faye[2] besteht sollte diese Technologie eingesetzt werden. Außerdem würde der PubSub Server hier selbst gehostet werden können, was vor allem aus datenschutztechnischer Perspektive als Vorteil bewertet wurde. Faye unterstützt allerdings keine Push Notifications, was zunächst kritisch bewertet wurde. Sollten weitere Modellierungsschritte aufdecken, dass Push-Notifications unverzichtbar sind, müsste auch die Nutzung eines externen Dienstes in Betracht gezogen werden.

Der PubNub-Dienst[3] bietet nach erster Beurteilung alle erforderlichen Eigenschaften und Fähigkeiten für das zu entwickelnde System. Kommunikation nach dem PubSub-Paradigma, Message Queueing sowie Push Notifications sind mit diesem Dienst möglich. Das zurückgreifen auf externe Dienste beraubt das eigene System jedoch ggf. um Potenziale und ist aus wirtschaftlicher und datenschutztechnischer Sicht nicht optimal. Nach einem Feedbackgespräch wurde die Auswahl der Technologien zur Umsetzung dieser Kommunikationsvorgänge daher noch einmal überdacht.

Als Wahlalternativen wurden die folgenden Technologien, Protokolle und Dienste beleuchtet:

**XMPP (Protokoll(e))** [4]

| Vorteile | Nachteile |
|---|---|
| Push Notifications möglich | |
| XML -Unterstützung eines der gängigsten Datenformate im Web | Stark an XML gebunden, Projektteam ist nicht so geübt im Umgang mit diesem Format wie mit Alternativformaten |
| Pubsub mit extension möglich | |
| Unterstützung von in späteren Iterationen (nicht im Prototyp) nützlichen Funktionalitäten (Chats, Echtzeit Übertragungen) | |
| Gewisses Maß an Vorwissen im Projektteam vorhanden | Keine konkrete Implementierungserfahrung im Projektteam |
| Presence Information über aktive Teilnehmer, für Verfolgung von Sammelaktionen interessant | Ist im wesentlichen eine Technologie zum streaming von XML, es müsste daher(wie auch bei einigen anderen Alternativen) eine „stehende Verbindung" zum XMPP-Server aufrecht gehalten werden, was sich negativ auf die beschränkte Akkulaufzeit des mobilen Endgerätes auswirkt |
| In Implementierungen wie ejabberd auch eine Form von Message Queueing möglich | |
| Offener Standard mit vielen interessanten XEPs | |

**AMQP (Protokoll)** [5]

| Vorteile | Nachteile |
|---|---|
| Clientimplementierungen in den im Projekt benötigten Programmiersprachen | Noch wenig Vorwissen im Projektteam vorhanden |
| Message Queueing und Pubsub möglich | |
| AMQP wird beschrieben als „wire level | |

| | |
|---|---|
| protocol", es ist daher nicht gebunden an bestimmte Message Broker oder Client Implementationen. Es existieren viele , teilweise open source Implementationen (RabbitMQ,Qpid, Storm MQ) dieser Komponenten | |
| | |

**MQTT (Protokoll)** [6]

| Vorteile | Nachteile |
|---|---|
| Entworfen für Kommunikation über Verbindungen mit geringen Bandbreiten und unzuverlässigen Netzwerken sowie Situationen in denen der Stromverbrauch kritisch ist | Diese Situationen meinen in erster Linie Kommunikation zwischen Maschinen im Kontext des „Internet of Things", für Sensoren und ähnliche Kleinstgeräte |
| Auch hier viele Implementierungen von Brokern/Servern und Clients in Java und Javascript | |

**JMS (Schnittstellenspezifikation)** [7]

| Vorteile | Nachteile |
|---|---|
| Unterstützt Point To Point (Message Queues) und PubSub Kommunikation | |
| Gibt lediglich eine Schnittstelle vor , ermöglicht dadurch eigene Implementierung der Schnittstelle | Schnittstellenspezifikation stark an Java gebunden |
| | Schlechte Kompatibilität mit gewählten Dienstgebertechnologien |
| | |

**FAYE (messaging System auf Basis des Bayeux Protokolls)** [2]

| Vorteile | Nachteile |
|---|---|
| ClientAPIs in Java (Android) und javascript (Node.js Plugin) | Keine Unterstützung von Message Queues , lediglich transiente Nachrichten |
| unkompliziert und leichtgewichtig | keine unterstützung von Push Notifications |
| Verwendung sowohl mit XML als auch mit JSON Payloads möglich | |

**GCM (kommerzieller Dienst)**[8]

| Vorteile | Nachteile |
|---|---|
| Alle benötigten Interaktionen möglich | Kommerziell motivierter Dienst , nicht selbst gehostet |
| Gute Interoperabilität mit der Android Plattform | Datenschutztechnisch als kritisch zu bewerten |

**PubNub (kommerzieller Dienst)**

| Vorteile | Nachteile |
|---|---|
| Alle benötigten Interaktionen möglich | Kommerziell motivierter Dienst , nicht selbst gehostet |
| Sehr gute Dokumentation | Datenschutztechnisch als kritisch zu bewerten |
| Viele SDKs | |
| Viele Features (u.a Presence, Streaming,Access Management) | Undurchsichtige Implementierung , bezeichnet sich selbst als „Protokollunabhängig" , was bedeuten soll das zwischen einer Vielzahl von Protokollen gewechselt werden kann |

Nach Abwägung der Vor;- und Nachteile der Alternativen sowie der Fähigkeiten einzelner Implementierungen der in Frage kommenden Protokolle fiel die Wahl auf die Nutzung einer open source Implementierung des AMQP Protokolls. RabbitMQ erfüllt alle in diesem Projekt benötigten Eigenschaften einer MoM, das Risiko der unzureichenden Erfahrung mit dieser Technologie wird in folgenden Artefakten (Risiken und PoCs) angegangen.

# Zentrale Instanz - Dienstgeber

Die zentrale Instanz des Systems beinhaltet einen großen Teil der Anwendungslogik und verfügt über eigenes Ressourcenmanagement. Auch der Dienstgeber soll an das Pub-Sub System angebunden werden, um bei bestimmten Ereignissen Nachrichten publizieren zu können. Mit den mobilen Clients wird Synchron über eine REST-Schnittstelle kommuniziert. Die identifizierten Geschäftsobjekte lassen sich gut als Ressourcen modellieren (Angebote, vorausgeplante Sammelaktionen usw..). Die Eigenschaften eines RESTful-Services, wie Skalierbarkeit und gute Integration in das "Ökosystem von Services" im Web waren ausschlaggebend für eine Konzeption einer REST-konformen Architektur. Außerdem besteht im Projektteam Erfahrung mit der Implementierung von RESTful Services. Jedoch ist bei der Konzeption der Kommunikationsabläufe im System darauf zu achten die beiden Paradigmen ( Nachrichten / Eventbasierte Architektur, PubSub und Client-Server bei REST) zu trennen um die dadurch entstehende Komplexität beherrschen zu können.

## Datenhaltung

Die Datenhaltung auf dem Server erfolgt über das Datenbank Management System Mongo DB. Dies wurde ausgewählt, da Mongo DB[9] integrierte Funktionalitäten aufweist mit geobasierten Inhalten umzugehen. Geobasierte Daten sind für das zu entwickelnde System essentiell wichtig. Außerdem ist Mongo DB eine NoSQL Datenbank, die i.d.R sehr gute Skalierbarkeitseigenschaften aufweisen. Da der Projektgegenstand potentiell eine sehr große Gruppe von Benutzern anspricht wurde diese Eigenschaft als sehr Vorteilhaft bewertet, stellte aber zunächst kein Alleinstellungsmerkmal von MongoDB dar. Die Anfragesprache dieser Datenbanklösung ist der Javascript Syntax sehr ähnlich, was aus Sicht des Projektteams den Einarbeitungsaufwand im Vergleich zu SQL basierten Systemen etwas reduzieren könnte.

Alternative Datenbanksysteme waren der Key-Value Speicher „Redis[10]", da im Projektteam Erfahrung mit diesem System besteht und die Kompatibilität mit den gewählten Dienstgeber-Technologien (Node.js) nicht mehr nachgewiesen werden muss. Da Redis jedoch wie viele andere Alternativlösungen nicht über die als sehr interessant eingestufte Funktionalität von „geospatial queries" verfügt wurde von einer Nutzung dieser Lösungen abgesehen. Da dieses Feature jedoch nicht zwingend benötigt wird stellt die Verwendung von Redis eine Fallbackoption im Rahmen dieses Projektes dar.

Als dritte Alternative wurde OrientDB[11] , ein ebenfalls als open source Implementierung unter Apache Licence 2.0 verfügbares DBMS beleuchtet. Dieses graphbasierte Datenbanksystem wurde Aufgrund seiner Flexibilität bezüglich verfügbarer APIs und seiner Performance als mögliche Alternative eingestuft. Zudem wurden hier Funktionalitäten, wie sie aus dokumentbasierten Lösungen wie MongoDB bekannt sind implementiert. Da zum Zeitpunkt dieser Abwägungen jedoch noch nicht geklärt werden kann inwieweit diese Flexibilität auch benötigt wird wurde auch von der Wahl dieser Alternative abgesehen.

Im Folgenden soll die geplante Anwendungslogik des Dienstgebers beschrieben werden

## Distributionsplattform

Der Dienstgeber verwaltet alle Angebote und Nachfragen. Er soll fähig sein Angebote nach bestimmtem Parametern (Standort, Lebensmitteltyp usw.) zu filtern und Benutzer über den Eintritt bestimmter Events (bspw. Erstellung eines Angebots mit zum Benutzer passenden Vorlieben) zu benachrichtigen. Diese Funktionalität wurde im Hinblick auf die Schaffung von Teilnahmemotivation und zur Unterstützung des Austauschs von Lebensmitteln zwischen Privatpersonen entwickelt.

## Angebots-Aggregator

Diese Komponente des Dienstgebers hat die Funktion zu überprüfen, ob in einem gewissen Bereich angebotene Spenden in Aggregation groß genug wären um von der Tafel abgeholt zu werden. Ist dies der Fall kann der Dienstgeber betroffene Benutzer durch die Pub/Sub Komponente dazu anstoßen diese Spenden zusammenzutragen.

## Transportkettenplaner

Ist eine Spende groß genug um von der Tafel berücksichtigt zu werden wird geprüft, ob eine Abholung durch die Tafel praktikabel erscheint. Im erfolgsfall wird diese benachrichtigt. Ist dies nicht der Fall, prüft der Dienstgeber welcher User den Transport mit möglichst minimalem Aufwand unterstützen könnte. Dabei werden alle Rollen die in der Transportkette vorkommen, also User die Lebensmittel zwischenlagern können und User die transportieren können berücksichtigt.

## Statistikerstellung

Der Dienstgeber soll Statistiken zu den Kontexten in denen Abholungen stattfinden sowie den umgesetzten Items generieren. Diese Statistiken können dem User dann bspw. aufzeigen wie viel Geld er gespart hat und in welchem maß er die Umwelt durch seine Teilnahme entlastet hat. Außerdem kann sich der einzelne User so mit den anderen Usern Vergleichen.

## Hotspot Events

Startet ein User eine Sammel- oder Verteilaktion werden alle User in einem bestimmten Bereich darüber Informiert und aufgefordert diese Aktion zu unterstützen. Der Dienstgeber berechnet anhand von Daten der teilnehmenden User den aktuellen Status des Events und kann somit auf verschiedene Ereignisse, wie bspw. das Schwinden der Resttransportkapazität des Sammlers, reagieren.

## Verarbeitungsvorschläge

Durch das Aggregieren von Angebotsdaten in einer bestimmten Umgebung mit den Abgleich einer Rezeptdatenbank sollen den Usern Vorschläge gemacht werden wie sie das bestehende Angebot am besten Nutzen können.

# Tafel Client

Als wichtiger Stakeholder des zu entwickelnden Systems wurden die Tafeln und ihre Mitarbeiter in diversen Rollen identifiziert .Aus ihren Nutzungskontexten werden aller Voraussicht nach noch einmal völlig neue Anforderungen entstehen, die mangels Information, die zu diesem Zeitpunkt vorliegt, noch nicht ermittelt werden können. Aus diesem Grund wurden die Tafeln zwar mit einem gesonderten Client aufgeführt, die Auswahl an Technologien ist zu diesem Projektzeitpunkt jedoch noch nicht sinnvoll.
Aus dem Interview mit einem Mitarbeiter der Tafel ging hervor, dass der Großteil der Mitarbeiter der Tafel Rentner sind und kaum Erfahrung im Umgang mit "neuen" Medien wie Smartphones haben. Die Kunden;- und Personalbetreuung wird momentan an einem Desktop Computer mit Windows abgewickelt, also scheint in diesem Bereich ein gewisses Maß an Erfahrung vorhanden zu sein. Um aus technischer Perspektive zu gewährleisten, dass die Mitarbeiter der Tafel ihren Client benutzen können wird dieser über einen Desktop Computer verwendbar sein. Da jede Tafel allerdings eine unterschiedliche interne Organisationsstruktur aufweist nicht sichergestellt, dass auf den von Ihnen verwendeten Computern ein identisches Betriebssystem zum Einsatz kommt. Der Client wird also in form einer Web Applikation realisiert. Da die meisten Tafeln in Ballungsgebieten aktiv sind, kann davon ausgegangen werden, dass eine benötigte Internetverbindung verfügbar ist.

## Externe Dienste

Da zu diesem Zeitpunkt im Projektverlauf noch nicht alle Anforderungen erfasst wurden kann die Nutzung externer Dienste, die voraussichtlich zur Unterstützung von Routenberechnungen notwendig ist noch nicht genau auf Technologien oder bestimmte Anbieter festgelegt werden.

# Nachtrag

Nach der "Überprüfung der projektspezifischen Fachkompetenz(09.11.2015)" wurde eine Fokussierung auf einige Kernfunktionalitäten vorgenommen, da der zuvor angedachte Funktionsumfang den leistbaren Aufwand der Rahmenveranstaltung übersteigen würde. Da der als Ergebnis zu erstellende vertikale Prototyp in erster Linie die verteilte Anwendungslogik sowie die Alleinstellungsmerkmale des Systems beinhalten sollte wurden Anhand dieser Kriterien nicht relevante Bestandteile gestrichen. Der fokussierte Teil der hier vorgestellten Anwendungslogik wird im Artefakt „funktionaler Prototyp“ noch einmal erläutert.

Da die zu MS 4 durchgeführten PoCs bezogen auf die Wahl externer Dienste (Mapbox, GoogleMaps) als zu eindimensional eingestuft wurden und eher als zeitlicher Not gewählt waren soll im Folgenden noch eine umfangreichere Abwägung stattfinden.

Das zu entwickelnde System und seine Anwendungslogik muss an vielen Stellen mit Standort;- und Routendaten umgehen. Routenfindung, Geocoding (Konvertierung von geographischen Koordinaten in Adressen) und andere auf Kartendaten basierende Berechnungen sollen nicht vom zu entwickelnden System durchgeführt werden. Aus diesem Grund wurden bei der Konzeption der Architektur des Systems einige externe Dienste beleuchtet und ihre Nutzung abgewogen. Die von diesen Diensten zu erfüllenden Kriterien waren im Einzelnen :

- Fähigkeit zu Geocoding (muss)
- Fähigkeit zur Bildung von Routen bestehend aus mehreren Standorten (soll)
- Fähigkeit zur Berücksichtigung des Fortbewegungsmittels bei der Berechnung von Routen (soll)
- Fähigkeit Positionen dem Straßen;- oder Nahverkehrsnetz zuzuordnen (soll)

- Fähigkeit zur Berücksichtigung aktueller Verkehrsinformationen bei der Routenberechnung (soll)
- Fähigkeit zur Step-by-Step Navigation (soll)
- Im Kontext mobiler Nutzung und voraussichtlich kurzen Nutzungsphasen angemessene Antwortzeiten (muss)
- Möglichst open source, ggf. sogar Teil des OpenStreetMap Projektes (soll)
- Falls kommerziell dem Projektrahmen angemessenes Lizenzmodell (muss)

Die beleuchteten Dienste waren:

**GoogleMaps** [12]

| **Vorteile** | **Nachteile** |
|---|---|
| Interoperabilität mit der Android Plattform | Entscheidungsmacht über dargestellte Information liegt bei Google |
| Sehr gut dokumentierter Setup, große Community | |
| Alle benötigten Funktionalitäten enthalten | Limitierte kostenfreie Nutzung der APIs |
| Riesiger finanzieller Aufwand zur Aktualisierung des Kartenmaterials | Kartenmetarial wird nicht von der „crowd“(wie bei OSM) aktualisiert |
| | |

**Mapbox** [13]

| **Vorteile** | **Nachteile** |
|---|---|
| Für einen kommerziellen Dienst attraktives Lizenzmodell | Kommerzieller Dienst , Limitierte kostenfreie Nutzung der APIs |
| Gute Dokumentation , möglicher Kontakt zu Helpdesk | |
| Kartendaten stammen von OSM | |
| SDKs für iOS,Android, Web | |
| Ansprechende visuelle Aufbereitung des Kartenmaterials | |

**OpenRouteService** [14] und diverse andere Projekte im Rahmen des OpenStreetMap-Projektes [15]

| **Vorteile** | **Nachteile** |
|---|---|
| Basiert auf Openstreetmap-Data, in Deutschland viele Verfügbare Einträge | Outputformate nicht Einstellbar , auf XML festgelegt |
| Freie Nutzung in non-profit Projekten und akademischem Umfeld | Verlangt Kontaktaufnahme vor Nutzung , kostet etwas Zeit |
| Bietet RouteService, Directory Service, Network Analysis Service | |
| Geocoding und Reverse Geocoding Serivce , Step by Step Navigation | |
| Inhaber bei der Uni Heidelberg , kulturelle Nähe zum Entwicklerteam | |
| Visualisierung der Kartendaten könnte auf weiteren Plugins (bspw. Leaflet) basieren | |
| Basiert auf der Open Source Engine „Graphhopper“ | |
| Konsumierbar über eine REST-Schnittstelle | |

Nach Abwägung dieser Alternativen wurde die Nutzung des OpenRouteService zunächst als die Vorteilhafteste bewertet.

**OSRM (**Routing und Geocoding Engine **)**

| **Vorteile** | **Nachteile** |
|---|---|
| Schnittstelle für Node existent | Management der OSM Datasets wird vom Projektteam übernommen |
| Verwendet JSON als Out;- und Inputformat | |
| Ist kein Dienst, kein Overhead durch Netzwerkübertragung , volle Kontrolle über Daten und Fähigkeiten der Engine | |
| | |

**Quellen:**


[1] - IDC Worldwide Quarterly Mobile Phone Tracker, February 24, 2015 (Zuletzt eingesehen 22.10.2015 )

[2] Faye
http://faye.jcoglan.com/
(Zuletzt eingesehen 07.11.2015 )

[3] PubNub

[4] XMPP
Saint-André, Peter; Smith, Kevin; Troncon, Remko (2009): *XMPP – the definitive Guide*. Sebastopol, Calif.: O'Reilly Media, Inc.

[5] AMQP
https://www.amqp.org/
(Zuletzt eingesehen 30.11.2015 )

[6] MQTT
http://mqtt.org/
(Zuletzt eingesehen 30.11.2015 )

[7] JMS
http://www.oracle.com/technetwork/java/jms/index.html
(zuletzt eingesehen 30.11.2015 )

[8] GCM
https://developers.google.com/cloud-messaging/
(Zuletzt eingesehen 25.10.2015 )

[9] MongoDB
https://docs.mongodb.org/manual/?_ga=1.218277785.1849592950.1446135213
(Zuletzt eingesehen 06.11.2015)

[10] Redis
http://redis.io/
(Zuletzt eingesehen 7.11.2015 )

[11] OrientDB
http://orientdb.com/orientdb/
(Zuletzt eingesehen 25.10.2015 )

[12] GoogleMaps
https://developers.google.com/maps/
(Zuletzt eingesehen 25.10.2015 )

[13] Mapbox
https://www.mapbox.com/products/
(Zuletzt eingesehen 7.11.2015 )

[14] OpenRouteService(ORS)
http://openrouteservice.org/contact.html#services
(Zuletzt eingesehen 25.10.2015 )
http://wiki.openstreetmap.org/wiki/OpenRouteService
(Zuletzt eingesehen 25.10.2015 )
[15] Openstreetmap
http://www.openstreetmap.de/
(Zuletzt eingesehen 25.10.2015 )
[16] RabbitMQ
https://www.rabbitmq.com/
(Zuletzt eingesehen 7.11.2015 )