WBA Modellierungen

Dieses Artefakt beinhaltet Modellierungen der im System ausgetauschten Informationen, Abwägungen bezüglich des verwendeten Datenformates sowie die Beschreibungen der modellierten Ressourcen der REST-Schnittstelle, welche im Artefakt „REST-Schnittstellenspezifikation“ fixiert wurden. Zudem werden modellierte Topics und im System geplante Kommunikationsabläufe aus technischer Perspektive dargestellt.

Inhaltsverzeichnis

WBA MODELLIERUNGEN 1

ATTRIBUTE UND INFORMATIONEN 2
ANGEBOT 2
ORT 2
TERMINE 2
TAFELVEREIN 3
SAMMELAKTION 3
SPENDE 4
TRANSPORTAKTION 4
BENUTZER 4
DISTRIBUTIONSGRUPPE 5
LAGERORT 5
SITUATION 5
VERTEILTE ANWENDUNGSLOGIK 5
PSEUDOCODE FÜR DAS IDENTIFZIEREN VON MÖGLICHEN STANDORTEN FÜR SAMMELAKTIONEN 6
PSEUDOCODE FÜR DIE SITUATIONSANALYSE 8
DATENFORMATE UND SCHEMATA 9

REST-SCHNITTSTELLE 10

STATUSCODES 10
CONTENT TYPES , CONTENT NEGOTIATION 11
HYPERMEDIA 11
RESSOURCEN 11
BENUTZERRESSOURCE 11
DISTRIBUTIONSGRUPPE 12
ANGEBOTSRESSOURCE 12
SPENDENRESSOURCE 12
SAMMELAKTIONSRESSOURCE 13
LAGERORTRESSOURCE 13
TAFELVEREINRESSOURCE 13

WEITERE ABWÄGUNGEN 13

FAT PING VS LIGHT PING 13
SICHERHEIT 13
QUELLEN 14

ANHANG 14

Attribute und Informationen

Angebot

Ein Angebot stellt eine einzelne Lebensmittelspende dar. Die Anwendungslogik wird versuchen Angebote zu aggregieren um Spenden, deren Transport wirtschaftlich ist zusammenzustellen.

ID - Da einzelne Angebote vom System gegeneinander abgegrenzt werden sollen wird jedes Angebot mit einem eindeutigen Identifier versehen
Beschreibung – Eine menschenlesbare und
Bild - Durch ein Bild von dem Artikel lässt sich für den Nutzer ein einfaches detailliertes Bild vom Zustand machen. Ist ein Base64 encoded image, um Bilder als String in die Datenrepräsentation einbinden zu können.
Gewicht - Die Tafeln holen Lebensmittelspenden erst ab einem bestimmten Gewicht ab. Um Spenden sinnvoll aggregieren zu können muss das System also das Gewicht eines Aritkel kennen. Außerdem ist das Gewicht auch für Transporteure und Sammler von Bedeutung. Angabe ist hier in Kg.
Artikelkategorie(n) - Um den Annahmeeinschränkungen der Tafeln gerecht zu werden muss die Kategorie der die im Angebot vorhandenen Artikel angehören geführt werden. Das Fehlen von Molkereiprodukten, Eiern und Fleisch in der Menge möglicher Kategorien soll auf die bestehenden Annahmeeinschränkungen hinweisen. Da das System Angebote aus ggf. mehreren Artikeln führt kann ein Angebot mehreren Lebensmittelkategorien angehören.
Erstellungsdatum - Um eine Dringlichkeit einer Abholung feststellen zu können soll zu jedem Angebot das Erstellungsdatum vom System erfasst werden.
Gültigkeitsablaufdatum - Als Grundlage für die Beurteilung eines "Angebotszustandes" wird eine Gültigkeitsdauer jedes Angebotes erfasst.
Status- Der Abholstatus eines Angebots soll signalisieren, ob das Angebot bereits Teil einer (aktuell durchgeführten, oder in der Verantwortlichkeit bereits zugewiesenen) Sammel;- oder Transportaktion ist.
Abholtermine - Ein Array von ebenfalls modellierten Terminobjekten um die verschiedenen Zeiträume für die Abholung anzugeben. Wie auch in Anforderung #37 formuliert ist hier Minutengenauigkeit ausreichend
Standort- Bildet den Standort für die Abholung des Angebotes ab.
Zuständige Tafel – Zur Berücksichtigung des Gebietsschutzes muss auch für ein einzelnes Angebot die Zuständige Tafel ermittelt werden.

Ort

Die Ortsdarstellung im System dient in erster Linie der Lokalisierung von Angebotsstandorten. Die Angaben einer postalischen Adresse (Stadt,Postleitzahl,Strasse,Hausnummer) erscheint zunächst ausreichend. Da die geplante Anwendungslogik in erster Linie auf Geolocations (Tupel aus Breiten;- und Laengengrad) basiert wird eine redundante Datendarstellung,bestehend aus Adress;- und/oder Geolocationdarstellung modelliert.

Termine

Das System gleicht an vielen Stellen bekannte Termine ab um bspw über die Sinnhaftigkeit einer Sammelaktion zu entscheiden. Es wurden verschiedene Darstellungen für wiederkehrende und einmalige Termine gewählt.

Einmaliger Termin:

von - Uhrzeit mit Datum an dem der Zeitraum beginnt,
bis - Uhrzeit mit Datum an dem der Zeitraum endet.
Ort - Ortobjekt. Der Ort an dem der Termin stattfindet

Wiederkehrender Termin:

Von –Uhrzeit , datum bei Wiederholung wenig sinnvoll
Bis – Uhrzeit
Wochentrag(e) – Die Tage an denen sich der Termin wiederholt
Wiederholung – Das Intervall in dem der Termin wiederholt wird [wöchentlich,monatlich,jährlich]

Tafelverein

Die Tafelvereine als Ziel der Spendentransporte müssen ebenfalls im System erfasst werden. Dies liegt zum einen in ihrer Anbindung an die Messaging Systeme,zum anderen muss das System über ihre Zuständigkeiten bescheid wissen ,um den Gebietsschutz berücksichtigen zu können. Neben bereits erwähnter Adresserfassung werden für die Tafelvereine folgende Attribute erfasst:

Öffnungszeiten – Um festzustellen wann Spenden bei der Tafel angeliefert werden können werden die Öffnungszeiten benötigt. Zu deren Darstellung wird die
Zuständigkeit – Menge von Postleitzahlen die den Zuständigkeitsbereich eines Tafelvereins abdecken
Kontakt – Die Kontaktdaten eines Ansprechpartners bei einem Tafelverein

Sammelaktion

Sammelaktionen stellen ein wesentliches Alleinstellungsmerkmal des zu entwickelnden Systems dar. Zwecks Abgleich mit Daten einer Situationsanalyse potentieller Sammler ähneln sich die Attribute der Sammelaktion mit den Parametern der Situationsanalyse.

id – auch Sammelaktionen müssen eindeutig identifizierbar sein
Sammler – Der für diese Aktion eingetragene verantwortliche Sammler, hier könnten zu späteren Zeitpunkten auch mehrere Sammler in Frage kommen
Streckenlänge - Eine Sammelaktion besteht immer in der aus der mehreren Zielen (Angeboten), deren Standorte über eine Route verbunden werden. Die Streckenlänge ist für den Abgleich mit möglichen Reichweiten von Sammeln interessant und soll auch zur Erstellung von Statistiken, die als Motivationskonzept genutzt werden gespeichert werden.
Teilnehmer – Die von einer Sammelaktion betroffenen Anbieter
betroffeneAngebote – Die Menge aller von der Sammelaktion betroffenen Angebote
Gesamtgewicht – Gewicht der bei der Sammelaktion zusammengetragenen Spenden, dient vor allem der Entscheidung über den Nutzen einer Sammelaktion
Fortbewegungsmittel – Auf Basis des zu transportierenden Gewichts sowie der Länge der zurückzulegenden Strecke soll das System fähig sein ein geeignetes Fortbewegungsmittel für den Transport vorzuschlagen
Route – Eine mögliche Transportroute ist Grundlage für den Vorschlag einer Sammelaktion. Die zu diesem Zeitpunkt modellierte Darstellung von Routen muss ggf. nach einem weiteren Feedbackgespräch, in dem Fragen zur Nutzung externer Dienste geklärt werden angepasst werden.

Eine Route besteht aus mehreren Standorten, deren Darstellung wie folgt modelliert wurde:

Reihenfolgeindex – stellt die Reihenfolge der möglichen Anfahrten dar. Aufgrund unterschiedlicher Präsenzzeiten der Anbieter ist diese Reihenfolge vom System zu erfassen.
Standort – bereits beschriebene Ortsdarstellung , ggf. Längen;- und Breitengrade
Ankunftszeitpunkt – Die Stationen einer Sammelaktion müssen ggf. in einer vorberechneten Reihenfolge angefahren werden. Dieses Attribut dient der Erfassung der zeitlichen Rahmenbedingungen bei der Erreichung einer Teilstation
istAbgeholt – Da bei der Durchführung einer Sammelaktion deren Status an Teilnehmer kommuniziert werden soll wird die Erreichung einzelner Teiletappen vom System erfasst

Ziel – Der Standort an dem die gesammelten Lebensmittel nach der Sammlung abgeliefert werden. Hier kommen Zwischenlager(die Adresse des Sammlers selbst oder eines Zwischenlagers), oder die Tafelvereine in Frage.

Spende

Eine Spende wurde als Bezeichnung für eine Aggregation von Einzelangeboten gewählt. Diese besteht aus mehreren Angeboten und führt

Angebote – Die Einzelangebote aus denen eine Spende zusammengetragen wurde, dient später der Rückverfolgbarkeit der Spendenwege
Gültigkeitsablaufdatum – Zur Erfassung der Dringlichkeit einer Abholung wird das frühste Ablaufdatum eines Einzelangebots einer Spende erfasst
Zuständige Tafel – Wie beim Einzelangebot
Gesamtgewicht – Summe der Teilgewichte der Angebote
Standort – Neuer Standort nach zusammentragen der Einzelangebote

Transportaktion

Eine Transportaktion verkörpert den Transport einer gesammelten Spende zu einem Tafelverein oder einer auf der Strecke liegenden Zwischenlagermöglichkeit.

Transporteur – Für die Aktion eingetragener Transporteur (Benutzer)
Streckenlänge – Länge der Strecke zwischen Anbieter und Abnehmer
Anbieter – Kann ein Zwischenlager oder der Standort eines Sammlers sein
Abehmer - Kann ein Zwischenlager oder der Zielstandort eines Tafelvereins sein
Spende – Zu transportierende Spende
Zielstandort – Falls ein Teiltransport nicht zu einer Adresse des Abnehmers transportiert werden soll wird der Zielsandort ggf. redundant gespeichert.

Benutzer

Vorname & Nachname – selbsterklärend, wird erfasst um bei Herstellung von Kontakt zwischen Benutzern nicht auf Nicknames o.Ä zurückgreifen zu müssen
wiederkehrendeTermine – Als Basis für die Vorschläge der Situationsanalyse sollen zu einem Benutzer wiederkehrende Termine erfasst werden können. Dies wird keine Voraussetzung für den Erhalt von Vorschlägen sein.
Kontakt – Zur unterstützung des Use Cases „Anbieter Kontaktieren“ soll die Möglichkeit bestehen Kontaktdaten bestehend aus einer Telefonnummer zu spezifizieren
Eingenommene Verantwortlichkeiten – Die Menge aller Aktionen , an denen ein Benutzer teilnimmt.

Distributionsgruppe

Postleitzahlenbereich – Eine oder Mehrere Postleitzahlen, gegliedert nach Zuständigkeitsbereichen der Tafeln
Mitglieder – Alle Mitglieder deren Wohnort innerhalb dieses Bereiches liegt
ZuständigerTafelverein – DIe Tafel an die die Spenden dieser Gruppe geleitet werden sollen

Lagerort

Öffnungszeiten – Um festzustellen wann Spenden bei der Tafel angeliefert werden können werden die Öffnungszeiten benötigt. Ihre Darstellung wird als „wiederkehrender Termin"(siehe JSON-Schemata) erfolgen
Kontakt – Die Kontaktdaten eines Ansprechpartners bei einem Tafelverein
Anmerkung: Lagerort ist der Darstellung des Tafelvereins sehr ähnlich , benötigt aber keine Zuständigkeiten im Sinne mehrerer Postleitzahlen. In Zukunft könnten die Tafeln ihre Mitarbeiter ebenfalls im System identifizieren um so Sammler mit „Sonderrechten" zu ermöglichen.

Situation

Starttermin – Zeitpunkt und Standort ab die Durchführung einer (Sammel;- oder Transport) aktion frühestens möglich ist
Transportmodus – Fortbewegungsmittel mit dem die Aktion durchgeführt wird , eventuell geschätztes Transportvolumen / Gewicht

Verteilte Anwendungslogik

Aus den Qualitätsmerkmalen der WBA-Artefakte geht hervor, dass die Verteiltheit der Anwendungslogik präzise dargestellt werden soll. Um diese Anforderung zu erfüllen soll im Folgenden entwickelter Pseudocode der Anwendungslogik kurz erläutert werden.

Der Android Client eine Analyse der Situation des Benutzers durch. Er approximiert das Fortbewegungsmittel mit dem der Benutzer unterwegs ist, ermittelt aus den dem System bekannten Terminen jenen zu dem der Benutzer unterwegs sein könnte und leitet daraus verfügbaren Zeitraum,maxmimales Transportgewicht sowie Reichweite ab. Die Ergebnisse dieser Berechnungen werden in seine JSON-Darstellung überführt und an den Dienstgeber in einem PUT auf die „Situation" Ressource (Benutzer/{id}/Situation) übertragen. Diese soll die funktionalen Anforderungen #19 und #20 erfüllen.

Der Dienstgeber ließt die Situationsparameter und führt zunächst ein Geocoding durch, um den PLZ-Bereich zu ermitteln in dem der Benutzer sich aufhält.
Anhand dieses Bereichs werden in Frage kommende Angebote abgefragt. Im nächsten Schritt werden Abholtermine, die Anbieter für diese Angebote angegeben haben mit dem verfügbaren Zeitraum abgeglichen und überschneidungen ermittelt. Ist dieser Vorgang abgeschlossen sind alle Angebote die theoretisch vom Benutzer Abholbar wären ermittelt. Im Folgeschritt ermittelt der Dienstgeber mithilfe des Dijkstra-Algorithmus schrittwiese die kürzesten Teilabschnitte zwischen den Angebotsstandorte n und summiert Spendengewichte und Gesamtstreckenlaenge auf. Die Suche kürzester Strecken wird beendet sollte das hinzufügen eines weiteren Angebotes das maximale

Transportgewicht oder die maximale Streckenlaenge (bekannt aus Situationsparametern) übersteigen. Ist auf diese weise ein Set von Angeboten bekannt, das die Situationsconstraints beachtet und theoretisch Abholbar wäre wird diese Route in eine Sammelaktions-Repräsentation überführt und steht von nun an per GET-Request auf die Vorschlagsressource (Benutzer/{id}/Situation/Sammelaktionen) bereit.

Der Benutzer soll gemäß funktionaler Anforderung #13 die Möglichkeit zur Identifikation von Routen für Sammelaktionen erhalten. Der Android Client kann zu diesem Zeitpunkt mehrere mögliche Aktionen abrufen und stellt die Ergebnisse an der Benutzungsschnittstelle dar. Der Benutzer kann nun die Vorschläge (Routen,Zeiträume,Entfernungen, Dauern) sondieren und ggf. eine Aktion auswählen,die er durchführen möchte. Hat er dies getan wird ein POST-Request auf den Endpunkt Sammelaktion durchgeführt. Dieser trägt den Benutzer als Sammler der Angebote ein, blockiert alle enthaltenen Angebote für andere Aktionen und benachrichtigt die betoffenen Angebote.

Für die Algorithmen der Situationsanalyse sowie der Ermittlung von Sammelaktionen wurde im Vorfeld Pseudocode entwickelt.

Pseudocode für das identifzieren von möglichen Standorten für Sammelaktionen

//Dies passiert nicht direkt auf eine User Interaktion hin. Nicht zu verlgeichen mit identifzieren von Transportrouten und Sammelaktionen aufgrund von bestimmten Suchparametern

```
Function SpendenAggregator{
  if(NeuesAngebotErstellt){
    SammelaktionroutenVorschlägeBrechnen()
  }
}

Function SammelroutenVorschlägeBerechnen(){

  angeboteInPlz //Datenstruktur für die Angebote in dem Bereich einer Plz

  gesamtgewicht = 0 //Wert des Gesamtgewichtes aller Angebote

  angebotMaxPrioritaet //Angebot mit der höchsten Priorität

  routenZwischenAngeboten // Datenstruktur für die Routen zwischen den Angeboten

  kuerzesteRoute // Var für die Route die am kürzesten ist

  routenAbschnitte //Datenstruktur für die Teilabschnitte einer Sammelatkionsroute

  laengeDerGesamtRoute // Länge der einzelnen Teilrouten

  startpunkt //Var für den Startpunkt einer Route
  zielpunkt //Var für den Endpunkt einer Route
```

```
   gültigeangeboteInPlz = Datenbankabfrage(Alle Angebote mit einer bestimmten Plz, deren Abholtermine in einem bestimmten Zeitraum liegen)

      //Berechne das Gesamtgewicht der einzelnen Angebote
      FOR i=0 to AnzahlgültigeangeboteInPlz{
         gesamtgewicht = gesamtgewicht + gültigeangeboteInPlz[i].gewicht
      }

      //Sind mindestens 5Kg Lebensmittel in dem Gebiet der Plz vorhanden
      if(gesamtgewicht >= 5){

         //Suche das Angebot mit der hoechstenPriorität
         FOR i=0 to anzahlgültigeAngeboteInPlz{
            angebotMaxPrioritaet= AngebotWasAlsNächstesAbläuft // Alternativ könnte Priorität auch über das Gewicht + Restgültigkeit berechnet werden
         }

         //Ab hier Routen berechnen
         //Berechne Routen(oder Distanzen) von aktuellen Standorten zu allen verbleidenden Punkten. Wähle die Route mit dem kürzesten Weg

         //Startpunkt für die Route festlegen
         startpunkt = angebotMaxPrioritaet.standort

         while(laengeDerGesamtRoute <= 2){

            FOR i=0 to anzahlgültigeAngeboteInPlz{
               zielpunkt=gültigeangeboteInPlz[i].standort          //Aufpassen! angebotMaxPrioritaet(startpunkt) ist noch in angeboteInPlz enthalten
               routenZwischenAngeboten[i]=Route(startpunkt, zielpunkt)
            }

            //Speichere die kürzeste Route
            kuerzesteRoute = Route aus routenZwischenAngeboten mit der kürzesten länge

            laengeDerGesamtRoute += kuerzesteRoute.länge

            //Füge Route zu RoutenAbschnitte hinzu
            routenAbschnitte[zaehler] = kuerzesteRoute

            //Wenn Route noch nicht lang genug ist, Setze den Zielpunkt zum neuen Startpunkt
            startpunkt = zielpunkt
            zaehler+=1;
         }

         //Die mögliche Sammelaktionsroute ist in routenAbschnitte gespeichert.
         return routenAbschnitte
      }
```

```
    }
}

Function Route(Standort A, Standort B)
{
return RouteZwischenAundB
}

Function TransportroutenVorschläge{
}
```

Pseudocode für die Situationsanalyse

```
Transportmittel
Zielort
Zeit

Function FortbewegungsmittelErfasser{

    userRoute
    fortbewegungsmittel
    DurschnittsGeschwindigkeit

    if(DurschnittsGeschwindigkeit < 8)
        fortbewegungsmittel = zuFuß;

    else if(DurschnittsGeschwindigkeit >= 8 && (userRoute == BahnRoute)
        fortbewegungsmittel = Zug;

    else if(DurschnittsGeschwindigkeit >= 8)
        fortbewegungsmittel = Auto;

    return fortbewegungsmittel
}


Function TerminErfasser
{

    systemZeit
    userTermine // Array mit allen Terminen des Users
    potentiellerUserTermine
    userStandort


    FOR( i=0 to AnzahlUserTermine){

        //Prüfen ob am Selben Tag
        if(userTermine.Datum == systemZeit.Datum)
            if(Route(userStandort, userTermin[i].standort).dauer <= ZeitBisZumTermin)
```

```
            potentiellerUserTermine=userTermin[i] // Zu Datenstruktur hinzufügen
    }

    //PRÜFEN WELCHER TERMIN ALS NÄCHSTES ANSTEHT
    FOR (i=0 to AnzahlpotentiellerUserTermine){

      //PRÜFEN WELCHER TERMIN ALS NÄCHSTES ANSTEHT UND SORTIEREN
      Sortiere potentiellerUserTermine nach Uhrzeit
    }

    //Durch den Termin sind sowohl Zeit auch als Ort identifzierbar
    return potenttielerUserTermine
}

Function Distance(Standort A, Standort B)
{
return DistanceZwischenAundB
}

Function Route(Standort A, Standort B)
{
return RouteZwischenAundB
}
```

Datenformate und Schemata

Zur Repräsentation der benötigten Informationen wurden die Datenformate XML und JSON gegeneinander abgewogen. Maßgebliche Entscheidungsfaktoren waren die Konformität der gewählten Datenrepräsentation mit genutzten Diensten sowie der gewählten Datenbanktechnologie. Zudem sollten Parameter wie Prüfbarkeit, Unterstützung von Hypermediakonzepten und der zu übertragene Overhead der beiden Datenformate auf ihre Relevanz im Projektkontext geprüft werden.

Die Wahl des Netzwerkdatenformates fiel auf die Verwendung von JSON. Dies ist begründet mit geringerem Overhead bei der Übertragung, was gerade in mobilen Kontexten vorteilhaft ist. Die wegen ihrer Unterstützung von Geospatial-Queueries gewählte Datenbank MongoDB arbeitet einem Format, welches JSON sehr ähnlich ist(BSON, Binary JSON). Diese Eigenschaft verspricht gute Kompatibilität mit dem gewählten Format.
Von einer Verwendung von XML wurde abgesehen, da der Umgang mit XML in vorangegangenen Projekten als aufwändiger (im Vergleich zu JSON) empfunden wurde. Aufgrund des knappen Zeitrahmens des Projektes sollte mit der Wahl von JSON das Risiko des ungeübten Umgangs(im Bezug auf Parsing Auffinden von Inhalten über XPath oder Anfragen über Xquery)eliminiert werden. Damit beraubt sich das Projektteam bewusst der Möglichkeit zur Anfrageprüfung über Validation mit einem XML-Schema oder DTD. Die bereits modellierten Ressourcen und ihre Attribute wurden stattdessen mit der Angabe von JSON Schemata formalisiert. (Siehe Verzeichnis „Ressourcenrepraesentationen_Schemata(link zum repository)). Da die verwendete Datenbanktechnologie die formale Angabe eines Datenbankschemas nicht erforderte wurde dies nicht modelliert. Die Angabe der JSON-Schemata wurde zur Repräsentation

der verwendeten Daten als ausreichend empfunden. Erläuterungen zu den modellierten Datenstrukturen finden sich im Artefakt „Datenstrukturen“.

REST-Schnittstelle

Die Entscheidung für einen REST-Webservice wurde schon früh im Projekt (Architekturdiagramm)gefällt und mit bestehender Erfahrung sowie der guten Abbildbarkeit der Geschäftsobjekte als Ressourcen begründet. Da diese Entscheidung schon lang bekannt war konnte im Projektteam in fast jeder Prozessphase über schon im voraus über mögliche Modellierungsalternativen diskutiert werden.

Nachdem zuvor JSON als Netzwerkrepräsentationsformat gewählt wurde hat das Projektteam vor der Spezifikation der Schnittstelle einige Ressourcenübergreifende Konventionen getroffen.

Statuscodes

Bei den Statuscodes wurde sich an den bereits im letzten Projekt(WBAII) als sinnvoll erachteten Konventionen orientiert[2]. Diese waren im Einzelnen:

Event	**Statuscode**	**Spez. Headerinhalte**
Erfolgreiches Anlegen einer Ressource durch POST	201 - Created	Location:URI der neuen Ressource
Erfolgreiches Abrufen einer Repräsentation durch GET	200 - OK	Cache-Control:Für die Ressource angemessene Caching Dauer
Erfolgreiches Ändern einer Ressource durch PUT	200 - OK	---
Erfolgreiches Löschen einer Ressource durch DELETE	204 - No Content	---
Ein Client versucht, ohne sich authentifiziert zu haben, eine PUT, POST oder DELETE Operation auszuführen	401 - Unauthorized	Authenticate mit gewünschtem Authentifizierungsverfahren
Ein Client stellt eine Anfrage mit einem nicht unterstützten Content Type	415 - Unsupported Media Type	Accepts (Unterstützter Content Type)
Ein Client stellt eine Anfrage, Service kann keinen im "Accepts" Header angegebenen Content Type bedienen	406 - Not Acceptable	
Ein Client stellt eine Anfrage auf eine URI, auf die er trotz Authentifizierung keine Zugriffsrechte hat	403 - Forbidden	

Wie im Abschnitt „Anwendungsogik“ beschrieben generiert der Dienstgeber mögliche Aktionen unter dem Endpunkt Benutzer/{id}/Situation/Sammelaktionen bzw. Benutzer/{id}/Situation/Transportaktionen. Eine Aktion wird erst wirksam, wenn ein Benutzer einen Vorschlag akzeptiert. Geschieht dies, so wird ein POST-Request auf die Endpunkte der beiden möglichen Aktionsarten (Sammlung;- Transport) abgesetzt. Da jedoch immer nur ein Benutzer einen Vorschlag zu einer Sammlung auch annehmen kann wird zusätzlich zu den o.g. Statuscodes auch Statuscode 409-Conflict verwendet, sollte sich im Zeitraum zwischen Vorschlagsgenerierung und Annahme bereits ein anderer Benutzer für die Aktion eingetragen haben.

Da eine Sammelaktion erst als Ressource angelegt wird wenn ein Benutzer einen Vorschlag annimmt (und ein POST mit Repräsentation der Aktion im Body absetzt) ist es

Content Types , Content Negotiation

Die Ressourcen sollen in Zukunft in unterschiedlichen Repräsentationen verfügbar sein, bei der Implementation ist daher bereits ein Check für das derzeit einzige unterstützte Format (JSON) einzubauen.

Hypermedia

Um möglichst redundanzfreie Darstellungen der Daten zu gewährleisten soll anstelle vollständiger Repräsentationen wann immer sinnvoll ein Hypermedia-Link zum Verweis auf bereits modellierte Ressourcen verwendet werden.

Ressourcen

Benutzerressource

Das System koordiniert im wesentlichen die Bewegungen einzelner Benutzer und versucht diese für ein größeres Ziel(die Aggregation von Spenden) einzusetzen. Obwohl die Benutzer wie in den User-Roles modelliert in unterschiedlichen Rollen mit unterschiedlichen Nutzungsverhalten auftreten muss ein Benutzer aus technischer Sicht nicht als eine bestimmte Rolle einnehmend modelliert werden. Da die geplante Anwendungslogik basierend auf der Verkehrs;- und Terminsituation des Benutzers Vorschläge zu dessen Teilnahme generiert musste diese „Situation“ ebenfalls mit dem Benutzer assoziiert werden. Diese Situation wurde als Subressource des Benutzers modelliert, da sie voraussichtlich häufig geupdatet wird ohne die Information über die Person (den Benutzer) zu ändern. So sollte entstehender Overhead reduziert und die Situationsinformation von der persönlichen Information entkoppelt werden.

Da die Teilnahmevorschläge vom Dienstgeber basierend auf der Situationsinformation generiert werden erschien es plausibel diese Vorschläge wiederum als Subressource der Situation (Endpoint /Benutzer/{id}/situation/vorschläge) zu modellieren. Es erscheint zudem nicht der Semantik des HTTP-Verbs „PUT“ entsprechend sollte ein Update der Situationsressource bereits die Vorschläge im Response Body enthalten.
Zudem war die Antwortzeit des Dienstgebers, beeinflusst durch die Berechnungszeit der Vorschläge, zu dem Zeitpunkt dieser Modellierung nicht bekannt. Es sollte also durch die Entkopplung von Situationsupdate und erhalt von Teilnahmevorschlag das Risiko von

zuschlagenden Timeouts in der Client-Server Interaktion (bei nicht ausreichend schnellen Antwortzeiten) vermieden werden.

Die Statistiken eines Benutzers beinhalten aggregierte Information über sein Bewegungsverhalten und sollten unter anderem dazu dienen den Beitrag eines Benutzers für ihn zu erfassen (Zielhierarchie: Sensibilisierung für das Thema Umweltschutz). In diesen Statistiken soll beispielsweise eine Berechnung der Kosten enthalten sein, die durch die Weiterverteilung statt die Entsorgung von Lebensmitteln eingespart wurden. Da auch diese Statistikinformation nicht unmittelbar mit den persönlichen Daten des Benutzers assoziiert werden können sollte wurde auch sie als Subressource des Benutzers modelliert.

Distributionsgruppe

Die Domänenrecherche hat u.A ergeben, dass die Tafelvereine bei der Verteilung der Zuständigkeiten für Spenden einen Gebietsschutz pflegen. Dieser teilt jedem Tafelverein einen bestimmten Bereich (gegliedert nach Postleitzahlen) zu. Die Distributionsgruppe soll eine Ressource darstellen, in der die in einem dieser Bereiche wohnhaften Benutzer mit dem Zuständigen Tafelverein assoziiert. Die Alternative zur Einführung dieser Ressource bestand in der Modellierung der zuständigen Tafel in eine Benutzerrepräsentation, in der ebenfalls der Wohnort des Benutzers erfasst wurde. Von dieser Alternative wurde jedoch abgesehen, da auch gegliedert nach diesen Gebieten Statistiken über die Verteilung generiert werden sollten. So erschien es plausibel eine solche Gruppe als eigenes Geschäftsobjekt (und eigene Ressource) abzubilden. Mittels Hypermedia (Link auf die Gruppenressource in der Repräsentation der Benutzerressource) wird ein Benutzer mit der Gruppe der er angehört assoziiert.
Da diese Gruppen vom System zugeordnet werden wurde an diesem Endpunkt lediglich das GET-Verb als gültige Operation definiert.

Angebotsressource

Ein Angebot stellt, wie bereits im Glossar des Projekts definiert, eine Einzelspende eines Anbieters dar. Diese Ressource wurde getrennt von der Ressource „Spende“ um eine Zurückverfolgbarkeit jedes einzelnen Angebots (siehe Anforderung #24) zu ermöglichen. Da für ein Angebot mindestens die CRUD-Operationen verfügbar sein sollten wurden die HTTP Verben GET PUT POST und DELETE für diese Ressource vereinbart.

Spendenressource

Eine Spende bündelt mehrere Angebote. In dieser Ressource sollte der Sachverhalt abgebildet werden, dass Lebensmittel(-spenden) erst ab einem gewissen Spendenvolumen wirtschaftlich transportiert werden sollten. Die einer Spende angehörigen Angebote wurden nicht als Subressource modelliert um die Angehörigkeit eines Angebots zu einer Spende mit einer einzigen PUT-Operation auf die Spendenressource ändern zu können.

UPDATE 20.12.2015

Die Trennung von Angebots;- und Spendenressource wurde aufgehoben. Stattdessen wird die Angebotsressource über ein Attribut „beinhalteteAngebote“ verfügen, um eine Gruppierung mehrerer Angebote zu ermöglichen. Auf diese Weise sieht das Projektteam den gleichen Sachverhalt wie oben beschrieben mit weniger Komplexität abgebildet.

Sammelaktionsressource

Eine Sammelaktion stellt einen im System sehr zentralen Sachverhalt ab. Eine solche Aktion soll dazu dienen kleinstspenden an geographisch eng beieinander liegenden (bestenfalls fußläufig erreichbaren) Standorten zusammenzutragen. Die einem Benutzer unterbreiteten Vorschläge zur Teilnahme werden im wesentlichen Links auf diese Ressource sein. Die wohl wichtigste Designentscheidung bei der Modellierung dieser Ressource bestand darin, sie gemeinsam mit der in den Use Cases modellierten „Transportaktion“ in eine Ressource zu modellieren. Eine Sammelaktion ist im Wesentlichen eine Reihe von Transporten von Lebensmitteln hin zu einem Zwischenlager oder dem endgültigen Ziel. Der Unterschied besteht hauptsächlich in der Menge von betroffenen Anbietern und der damit betroffenen Menge von Lebensmitteln. Da diese Unterscheidung jedoch nicht als ausreichend empfunden wurde um Transport;- und Sammelaktion als separate Ressourcen abzubilden wurden sie in einer gemeinsamen Ressource verankert.

Lagerortressource

Die Zwischenlager Ressource bildet Orte ab, die entweder als Zwischenlager oder als endgültiges Ziel von Spenden zur Verfügung stehen. In dieser Ressource sollen private Benutzer und kommerzielle Betriebe ihre Lagerkapazitäten zur Verfügung stellen können um Spendentransporte nicht in einem einzigen zusammenhängenden Zeitraum abwickeln zu müssen.

Tafelvereinressource

Die Tafelvereine als Ziel der Spendentransporte werden trotz ähnlicher Attribute nicht gemeinsam mit Lagerorten (siehe Ressource „Lager“) in eine Ressource modelliert. Dies hängt damit zusammen, dass ein Tafelverein als Institution die Zuordnung des Vereins zum Zuständigkeitsgebiet herstellen können soll ,was bei privaten Lagerorten nicht notwendig ist.

Weitere Abwägungen

Fat Ping vs Light Ping

In der Kommunikation zwischen Dienstgeber und den Dienstnutzern wird die Kommunikation nach dem Message Passing Paradigma zur Benachrichtigung über Ressourcenveränderungen verwendet. Für diese Benachrichtigung ist es nicht notwendig Nutzdaten der Ressourcen zu übertragen. Demnach wird in diesen Kommunikationsvorgängen ein Light Ping Verfahren eingesetzt, die Nachrichten werden lediglich verweise auf die veränderten Ressourcen enthalten.

Sicherheit

Wie bereits in den Anforderungen festgehalten ist sowohl bei Systeminterner Datenhaltung als auch bei der Übertragung von Information über ein Netzwerk für entsprechenden Schutz der Daten vor Fremdzugriff zu sorgen. Da aus den Informationen einer Spende (Standort, Abholzeiten) sehr leicht Rückschlüsse auf Anwesenheitszeiten von Bewohnern geschlossen werden könnte und ein personenbezogenes Bewegungsprofil (bspw. der Benutzer Subressource „Situation“) für eine Vielzahl von Dritten sehr interessant wäre müssen speziell diese Daten geschützt werden. Da der Veranstaltungsrahmen jedoch keinen Fokus auf IT-Sicherheit legt wird die Ausarbeitung konkreter Konzepte zunächst verschoben. Auch in diesem Zusammenhang relevante Authentifizierungskonzepte und die Modellierung von Datenstrukturen für Login-Funktionalitäten wurden zunächst nicht betrachtet.

Quellen

[2] https://github.com/TobiGe/WBA2SS15MichelsWalgerGerstenberg/wiki/1.4-Statuscodes

[3] http://json-schema.org/documentation.html
(Zuletzt eingesehen 20.12.2015)(siehe Unterpunkt „standard schemas“

JSON-Schema:
[4]http://json-schema.org/latest/json-schema-core.html
(Zuletzt eingesehen 20.12.2015)

[5]http://json-schema.org/latest/json-schema-validation.html
(Zuletzt eingesehen 20.12.2015)

[6]http://json-schema.org/latest/json-schema-hypermedia.html
(Zuletzt eingesehen 20.12.2015)

[7] http://spacetelescope.github.io/understanding-json-schema/UnderstandingJSONSchema.pdf
(Zuletzt eingesehen 20.12.2015)

Anhang

REST-Spezifikation :

https://github.com/TobiGe/EISWS1516MichelsGerstenberg/blob/master/MS%204/WBA_Modellierungen/referenzierteArtefakte/REST-Schnittstellenspezifikation%20.pdf

JSON-SCHEMATA :
https://github.com/TobiGe/EISWS1516MichelsGerstenberg/tree/master/MS%204/WBA_Modellierungen/referenzierteArtefakte/Ressourcenrepraesentationen_schemata

ressourcenname	Endpoint	Verb	Semantik	Content Type Request	Content Type Response
enutzer	/Benutzer/{id}	GET	Liefert eine Repräsentation eines Benutzers mit {id}	-	application/json
	/Benutzer/{id}	PATCH	Ändert die Information des Benutzers mit {id}	application/json	application/json
	/Benutzer	POST	Fügt der Benutzercollection einen Benutzer hinzu	application/json	application/json
	/Benutzer/{id}	DELETE	Entfernt den Benutzer mit {id} aus dem System	-	-
ubressource Situation	/Benutzer/{id}/Situation	PUT	Ändert die Situationsinformation des Bennutzers mit {id} , Antwort beinhaltet Hypermedia Link auf Teilnahmevorschläge passend zu dieser Situation	application/json	application/json
	/Benutzer/{id}/Situation/Transportaktionen	GET	Liefert eine Collection von Transportaktionen passend zur Situation mit {id}	-	application/json
	/Benutzer/{id}/Situation/Sammelaktionen	GET	Liefert eine Collection von Sammelaktionen passend zur Situation mit {id}	-	application/json
ubressource Statistiken	/Benutzer/{id}/Statistiken	GET	Liefert Information über umgesetzte Lebensmittel, zurückgelegte Stecken u.Ä des Benutzers mit {id}	-	application/json
istributionsgruppe	/Distributionsgruppe/{id}	GET	Liefert eine Repräsentation einer Gruppe, beinhaltet Links auf ihre Teilnehmer und die Zuständige Tafel, Teilnehmer werden nach ihrem Wohnort einer Gruppe zugeteilt	-	application/json
	/Distributionsgruppe/{id}/Statisiken	GET	Liefert Information über in einer Distributionsgruppe umgesetzte Lebensmittel, zurückgelegte Stecken u.Ä	-	application/json
ngebot	/Angebot/{id}	GET	Liefert eine Repräsentation eines Angebots mit {id}	-	application/json
	/Angebot/{id}	PATCH	Ändert die Information des Angebots mit {id}	application/json	application/json
	/Angebote	POST	Fügt der Collection aller Angebote ein Angebot hinzu	application/json	application/json
	/Angebot/{id}	DELETE	Entfernt das Angebot mit {id} aus dem System	-	application/json
ammelaktion	/Sammelaktionen	GET	Liefert die Collection aller Sammelaktionen	-	application/json
	/Sammelaktion/id	GET	Liefert eine Repräsentation der Sammelaktion mit {id}	-	application/json
	/Sammelaktion/{id}	PATCH	Ändert die Information einer Sammelaktion	application/json	application/json
	/Sammelaktionen	POST	Fügt der Collection aller Sammelaktionen eine weitere hinzu, blockiert alle enthaltenen Angebote	application/json	application/json
	/Sammelaktion/{id}	DELETE	Entfernt eine Sammelaktion aus dem System, gibt darin enthaltene Angebote wieder frei	-	-
ager	/Lager	GET	Liefert die Collection aller Lagerorte im System	-	application/json
	/Lager/{id}	GET	Liefert eine Repräsentation des Lagerorts mit <{id}>	-	application/json
	/Lager/{id}	PATCH	Ändert die Information eines Lagerortes	application/json	application/json
	/Lager	POST	Fügt der Collection aller Lagerorte eine weitere hinzu	application/json	application/json
	/Lager/{id}	DELETE	Entfernt einen Lagerort aus dem System	-	-
afelverein	/Tafelvereine	GET	Liefert die Collection aller im System registierten Tafelvereine	-	application/json
	/Tafelverein/{id}	GET	Liefert eine Repräsentation des Tafelvereins mit {id}	-	application/json
	/Tafelverein/{id}	PUT	Ändert die Information einer Tafelvereins	application/json	application/json
	/Tafelvereine	POST	Fügt der Collection aller Tafelveriene einne weiteren hinzu	application/json	application/json
	/Tafelverein/{id}	DELETE	Entfernt einen Tafelverein aus dem System	-	-
opics	**potentielle Subscriber**	**potentielle Publisher**	**Semantik**		
Teilnahmevorschlaege/{PLZ}	Benutzer mit Standort im {PLZ} Bereich	Dienstgeber	Topic für Teilnahmevorschläge , gegliedert nach Postleitzahlen		
Sammelaktion/{id}	Alle Teilnehmer der Aktion	Sammler(Android Dienstnutzer)	Topic für Statusupdates einer Sammelaktion		
Distributionsgruppe/{id}	Angehörige der Distributionsgruppe	Zuständiger Tafelverein(-client), Dienstgeber	Topic für Benachrichtigungen über Statistiken einer Distributionsgruppe und Spendenaufrufen		
Spende/{id}	Benutzer deren Angebot in der Spende enthalten ist	Dienstgeber	Topic für Neuigkeiten über den Weg einer Spende		

bbildung 1 - REST-Schnittstellenspezifikation