Entwurfsbeschreibung „Interaktiver Haushaltsplanrechner“

Projektgruppe „Interaktiver Haushaltsplanrechner Leipzig 2015“
http://www.leipzig-data.de/IHR-15

Version vom 9. Oktober 2015

Inhaltsverzeichnis

Hintergrund

Das Projekt **Interaktiver Haushaltsplanrechner Leipzig 2015** ist ein wesentlicher Baustein des in enger Zusammenarbeit der Koordinierungsstelle für Bürgerbeteiligung der Stadt Leipzig („Leipzig weiter denken“), des Dezernats Finanzen der Stadt Leipzig und des Instituts für Öffentliche Finanzen und Public Management entwickelten Vorhabens **Nachhaltige Stadtfinanzen – Akzeptanzsteigerung der bürgerschaftlichen Beteiligung an der Haushaltsplanung**. Dieses Vorhaben wurde im Rahmen der Initiative „ZukunftsWerkStadt” im Zeitraum von Oktober 2014 bis August 2015 vom Bundesministerium für Bildung und Forschung (BMBF) durch Fördermittel unterstützt.

Als Teil der Strategie „Leipzig weiter denken 2.0“ war das Ziel des Vorhabens, die deliberativen Diskussions- und Beteiligungsstrukturen im Haushaltsplanungsprozess der Stadt Leipzig weiter zu stärken. Neben der im Rahmen von „Leipzig weiter denken“ bereits entwickelten repräsentativen Bürgerwerkstatt sollte deshalb eine noch intensivere bürgerschaftliche Einbindung ermöglicht werden. Für die Haushaltsentwurfsplanung bedeutet dies, das Handeln der Stadt noch transparenter zu gestalten und die Bürgerinnen und Bürger aktiver in Entscheidungsprozesse mit einzubeziehen. In diesem Zusammenhang wurde der von der Stadt Leipzig in den Jahren 2008 bis 2012 bereitgestellte, aber wenig genutzte „interaktive Haushaltsplan“ geprüft. Das Vorhaben wurde von der Koordinierungsstelle „Leipzig weiter denken“ beraten.

Aufbauend auf Ergebnissen aus Umfragen, Workshops und Good-Practiceanalysen wurde im Projektbaustein „Interaktiver Haushaltsplanrechner Leipzig 2015“ ein für Leipzig bedarfs- und zielgruppengerechtes Instrument erstellt. Anknüpfend an vorhandene Erfahrungen auch der Leipziger Agenda21 Gruppe bzw. des Forums Bürgerstadt Leipzig, deren Mitarbeiter die Entwicklung des „Interaktive Haushaltsplan“ unterstützt und begleitet haben, wurden Instrumente entwickelt, um haushaltsrelevante Informationen nutzergruppenfreundlich aufzubereiten und geeignete Partizipationsmöglichkeiten zu schaffen.

Projektpartner bei der Entwicklung seitens der Universität waren das Institut für Öffentliche Finanzen und Public Management (Prof. Lenk, Herr Redlich, Herr Glinka), das in der informationstechnischen Umsetzung durch das Institut für Informatik (Prof. Gräbe) bei der Anforderungsanalyse und der prototypischen technischen Realisierung eines neuen Online-Tools unterstützt wurde.

An der Realisierung des neuen Online-Tools arbeiteten die folgenden Studierenden mit:

> Wolfgang Amann, Janos Borst, Sarah Cujé, Christian Hoffmann, Dennis Kreußel, Fabian Niehoff, Tobias Wieprich, Tamara Winter, Kalle Willi Wollinger, Sebastian Zänker.

Die Arbeiten wurden weiterhin betreut von Prof. Gräbe und Konrad Höffner, Mitarbeiter am Lehrstuhl „Betriebliche Informationssysteme“, und Marius Brunnert als studentischer Tutor sowie durch Philipp Glinka und Matthias Redlich als projektverantwortliche Mitarbeiter am Institut für Öffentliche Finanzen und Public Management.

1 Einführung

Der im Rahmen des Projekts erstellte Prototyp eines neuen Interaktiven Haushaltsplanrechners für die Stadt Leipzig verbindet drei Elemente modernen Designs:

- Die Datenschicht setzt auf moderne semantische Konzepte auf RDF-Basis, was eine leichte Anbindung der Anwendung an die Linked Open Data Welt im Rahmen einer zukünftigen Open Data Strategie der Stadt Leipzig ermöglicht.
- Die Präsentationsschicht setzt mit Drupal und Visualisierungen auf der Basis des verbreiteten Javascript-Frameworks D3 auf nachhaltige Standardlösungen aus dem Open Source Bereich, die perspektivisch von einem kleineren, einschlägig qualifizierten IT-Team betreut und weiterentwickelt werden können.
- Partizipative Elemente sind als „kleine Lösung“ über einen Drupal-Forums-Baustein in die Anwendung integriert, lassen sich aber in eine (noch zu entwickelnde) dezentrale, modular vernetzte „große Lösung“ eines anwendungsübergreifenden Partizipationskonzepts integrieren.

Dieses Konzept und dessen Realisierung wurde in drei Etappen entwickelt und umgesetzt.

In einer *ersten Etappe* (Oktober 2014 bis Januar 2015) analysierte ein interdisziplinär zusammengesetztes studentisches Team Gestaltungsvarianten und Erfahrungen bereits existierender Haushaltsrechner und verdichtete die Recherchen zu einem *Evaluationsbericht*, aus dem eine *Anforderungsanalyse* und ein *Partizipationskonzept* abgeleitet wurden. In letzerem ist bereits festgehalten, dass Beteiligungskonzepte übergreifenden Charakter haben müssen und eine allein auf den Haushaltsplanrechner fokussierte Partizipationslösung zu einer unzulässigen Engführung der Fragestellungen führt. Diese Position bestätigte sich bei den Anforderungsaufnahmen in späteren Workshops und Bürgerbeteiligungsphasen im Rahmen des Projekts. Zugleich wurde in dieser ersten Phase ein genaueres Datenmodell für Haushaltsdaten auf der Basis des RDF Data Cube entwickelt und die vorliegenden Haushaltsdaten für 2014 in dieses Format transformiert.

In einer *zweiten Etappe* (Januar bis Mai 2015) wurde im Rahmen einer Projektgruppenarbeit im Softwaretechnik-Praktikum auf der Basis der vorliegenden Anforderungsanalyse der neue Interaktiven Haushaltsplanrechner als Online-Tool zur Bürgerbeteiligung prototypisch entwickelt. Mit *Drupal* kommt dabei ein weit verbreitetes Open Source CMS zum Einsatz, das einerseits um entsprechende semantische Konzepte für Management und Darstellung von RDF-basierten Daten und andererseits um angemessene Visualisierungskonzepte zu erweitern war. Die technischen Voraussetzungen für die Umsetzung des Partizipationskonzept wurden durch Einbindung und Anpassung eines Drupal-Forums geschaffen. Als RDF Data Store kommt eine Instanz von Virtuoso zum Einsatz, deren SPARQL Endpunkt öffentlich zugänglich ist und damit das Potenzial bietet, die Plattform in einen größeren Open Data Verbund zu integrieren.

In einer *abschließenden dritten Etappe* (Juni bis September 2015) arbeitete ein Teil des zweiten Projektteams an der weiteren Konsolidierung des Prototyps, wobei weitere Anforderungen aus Bürgerbefragungen und Good-Practice-Analysen aufgearbeitet, priorisiert und im Rahmen eines Redesigns vor allem des Informationsteils des Interaktiven Haushaltsplanrechners umgesetzt und integriert wurden. Außerdem wurde die Datenbasis auf den aktuellen Stadt-Haushalt 2015/16 aktualisiert.

In der finalen Version des Haushaltsplanrechners werden die Haushaltsdaten bis zur vierten Ebene der Schlüsselprodukte aggregiert dargestellt, zu denen auch Produktsteckbriefe vorliegen. Entsprechende Informationen wurden im Rahmen der Datenaufbereitung aus den vorliegenden detaillierteren Primärdaten für die Planwerte der Jahre 2014 bis 2019 aggregiert.

2 Grundsätzliche Struktur- und Entwurfsprinzipien

2.1 Barrierefreiheit

Beim Design der Website wurde darauf geachtet, diese gemäß BITV 2.0 [1] barrierearm zu gestalten. Die Entwickler von Drupal und des eingesetzten CSS-Frameworks Bootstrap legen auf diese Fragen bereits großen Wert, so dass allein die Wahl der Softwarebasis des Projekts hohe Standards in Bezug auf Barrierefreiheit sichert. Bei den Anpassungen, die für den Prototyp erforderlich waren, wurden diese Standards ebenfalls berücksichtigt.

2.2 MVC-Architekturmodell

Für die Kernfunktionen des Projekts, besonders für die Anbindung der RDF-basierten Datenschicht, kam das MVC-Architekturmodell [6] zum Einsatz. Eine typische HTTP-Anfrage wird dabei zunächst aufbereitet und an die Vermittlungsschicht („Controler“) weitergereicht, welche den Auftrag für die Verarbeitung in der Modellschicht vorbereitet. Die dort zusammengestellte inhaltliche Antwort wird an den „View“ weitergereicht, um die Daten entsprechend zu visualisieren (z. B. in einem Kreisdiagramm).

Zur Verarbeitung der Anfrage in der Modellschicht wird auf die darunter liegende Datenschicht zugegriffen, die in unserem Fall aus einem RDF Data Store für die Haushaltsdaten sowie einer MySQL Datenbank für die Drupal-Funktionalitäten gebildet wird.

Da in Drupal und unserer Plattform insgesamt intensiv Javascript-Funktionen einsetzt werden, die auf dem DOM-Baum der Ein- und Ausgabe operieren, wird das MVC-Muster jedoch oft nicht strikt eingehalten. Eine zentrale Aufgabe des „Views“ ist zum Beispiel auch die Verwaltung von Formularen und Dokumenten zur allgemeinen und übersichtlichen Informationspräsentation für die Benutzer.

2.3 PAC-Architekturmodell

Drupal selbst orientiert sich stärker am PAC-Architekturmodell [7] („Presentation-Abstraction-Control“) als einem spezifischen MVC-Modell, in dem die Informationsverteilung über die Controler auf spezifische Weise organisiert ist – Controler sind verschiedenen Datenabstraktionskonzepten zugeordnet, die als „Agenten“ bezeichnet werden. Dieses Architekturmodell wurde bei den Erweiterungen des Forums-Moduls berücksichtigt, um die Verwaltung des Forums und der Vorschläge so weit wie möglich über Drupal-interne Funktionen abzuwickeln.

2.4 Semantische Technologien

Der Haushaltsrechner verwendet auf der Datenebene die semantische Technologie RDF [8] in der Form von RDF Data Cubes [9]. Das bedeutet, dass die in unserer Datenbank dargestellten

Ressourcen eindeutig bezeichnete Dinge der Welt sind, die durch Darstellung in Tripel in eine Relation mit anderen Ressourcen gebracht werden.

Um diese Daten an die Modellschicht zu binden und mit der RDF Datenbasis über deren SPARQL Endpunkt zu kommunizieren, wurde die *BorderCloud SPARQL 1.1 PHP Bibliothek* [3] eingesetzt. Sämtlicher diesbezüglicher Code einschließlich der zur Verbindung mit dem SPARQL-Endpunkt benötigten BorderCloud Bibliothek ist im *Theme*-Ordner `data` abgelegt.

3 Frontendgestaltung – Informationsteil

3.1 Allgemeiner Aufbau

Abstraktionsebenen

Die Präsentation der Daten erfolgt über verschiedene Abstraktionsebenen, die als verschiedene Ringe in der Diagrammvisualisierung präsentiert werden. Die Abstraktionsebenen haben den Zweck, die Menge an Daten übersichtlich zu präsentieren und ein intuitives Verständnis sowohl der Größenverhältnisse als auch der Verteilung der Ausgaben und Einnahmen zu gewinnen. Die Abstraktionsebenen orientieren sich hierbei an den Produktebenen des Haushaltsplans der Stadt Leipzig. Hierbei sollten möglichst alle Hierarchieebenen bis zum kleinsten Element verfügbar sein.

Designkonzept

Das Aussehen der Website des interaktiven Haushaltsrechners der Stadt Leipzig orientiert sich stark am grundlegenden Design des Bundeshaushaltsrechners und wurde an das Look and Feel der Webseiten der Stadt Leipzig angepasst. Zum Einsatz kam eine leichte Anpassung des CSS-Frameworks *Bootstrap* [2] und dessen Portierung für Drupal 7 sowie die Javascript-Bibliothek D3 für die Diagrammdarstellungen.

Wie beim Bundeshaushaltsrechner werden die Daten in Form von Kreisdiagrammen angezeigt. Zusätzlich werden Produktbereiche, die kleine Beträge aufweisen (weniger als 2 % der Gesamtsumme der jeweiligen Ebene), in den Diagrammen zu einem Segment *Sonstiges* zusammengefasst, das nicht weiter expandiert werden kann. In der tabellarischen Darstellung der Daten der jeweiligen Ebene werden diese kleinen Beträge jedoch einzeln ausgewiesen.

3.2 Controller-Struktur

Die zusätzlichen Controller unserer Erweiterung dienen vor allem dazu, SPARQL-Anfragen an das Model zu übergeben, damit das Model die Daten aus dem Datacube auslesen kann. Dabei werden entweder die Ausgaben oder die Einnahmen der Stadt ausgelesen, je nachdem wie der Modus gesetzt ist. Zudem parst der Controller noch die richtigen Namen der Kostenbeschreibung hinzu.

3.2.1 EndpointHandler (= Model)

Im EndpointHandler werden die Klarnamen der Produktnummer hinzugefügt, zudem wird die URI so gegekürzt, dass sie nur die Beschreibung anzeigt. Dies geschieht in der Funktion `sparqling`. Dabei werden die Daten in JSON-Format geparst. Dieses Format wird für das Diagramm benötigt. Im Konstruktor der Klasse wird der Endpunkt zu dem Datacube gelegt, welche über die Variable endpoint festgelegt wird.

3.2.2 QueryWriter (= Controller)

In der Variable `prefix` werden alle benötigten Präfixe für die Abfrage gespeichert. Der Präfix `xsd` wird für die Typkonvertierung benötigt, während `qb` für die Zeitangaben im Cube benötigt wird.

In der Variablen `ihr_uri` wird die URI des Datacube gespeichert. Wird der Name des Datacube geändert, so muss auch diese Variablen geändert werden. Dabei bleibt `PREFIX ihr:` immer gleich, da ansonsten Fehler in der Abfrage entstehen.

Das Array `attribut_array` zeigt alle Kategorien des Datacube in chronologischer Reihenfolge an. Diese werden für die Erstellung der Abfrage benutzt.

Das Array `amount` wird für die Unterscheidung von Ausgaben und Eingaben benutzt. Die Variable `choose_amount` setzt den Modus, ob Einnahmen oder Ausgaben genutzt werden. Diese Variable wird über die Funktion `setAmount` entweder auf 0 (Einnahmen) oder 1 (Ausgaben) gesetzt.

Die Funktion `firstRing($year = null` erstellt den ersten Ring, den der Nutzer sieht, dabei ist das Jahr optional.

Die Funktion `sparqlTransformation(array category, year)` erstellt eine SPARQL-Anfrage dynamisch auf Grundlage des Pfades, welchen der Nutzer ausgewählt hat.

`searchLabel($term)` sucht in der Datenbasis nach der Nutzereingabe. Dabei werden nach Übereinstimmungen gesucht, somit findet er bei der Eingabe „Wesen“ auch „Gesundheitswesen“. Es werden dabei alle Nummer und das Label zurückgegeben. Dabei wird die Funktion durch `searchNumber($term)` unterstützt, welche noch alle Eingaben und Ausgaben gesucht wird.

`getPath($number)` gibt den Pfad zur einer bestimmten Produktnummer zurück, welche für die Weiterleitung auf das Diagramm. Die Funktion `setFrom($year)` ändert die From-Klausel für die Anfrage ab. `getFrom()` gibt die From-Klauseln zurück.

3.3 Suchfunktion

Die Suchfunktion ermöglicht die Suche von Labels etc. in dem RDF-Graphen. Es nutzt JS UI für das Autocomplete und printed in php größtenteils JS-Code, welcher von php-RDF-Anfragen die benötigten Informationen erhält und daraus eine Tabelle erstellt. Unter `Drupal/themes/[Themename]/templates/page.tpl.php` wird eine Form eingefügt, welche zur Suche verlinkt ist, damit sie Drupal oben als Suchleiste anzeigt.

Die Suche selbst ist als normale Seite einzufügen. Als Textformat muss „PHP Code“ gewählt werden.

3.4 Diagramme

Die in der Plattform enthaltenen Kreisdiagramme sind mit der Javascript-Bibliothek D3 in der Version 3.5.6 [4] erstellt. Kenntnisse im Umgang mit D3 sind erforderlich, um die Arbeitsweise der Datendarstellung zu verstehen.

3.4.1 Grundlegendes

Der Quelltext jedes Diagramms besteht aus drei Teilen: dem HTML-Teil am Anfang, der die Verankerung in der Seite festlegt, dem JavaScript-Teil für das Einnahmendiagramm und dem JavaScript-Teil für das Ausgabendiagramm. Dabei handelt es sich bei dem Ausgabendiagramm um eine fast exakte Kopie des Einnahmendiagramms, lediglich mit verschiedener Variablen- und Funktionenbezeichnern. Deshalb wird im folgenden nur auf das Einnahmendiagramm im Detail eingegangen. Im HTML-Teil werden die Tabs zum Umschalten zwischen den verschiedenen Ansichten (Einnahmen, Ausgaben etc.) angelegt. Weiterhin werden hier verschiedene Behälter für die Diagramme und dem Infopanel neben den Tabs festgelegt.

3.4.2 Datenstruktur

Die Daten für die Diagramme und Tabellen sind in Levels/Layers organisiert. Entsprechend sind Arrays innerhalb des Quelltextes angelegt. So gehören zum Beispiel die Daten des ersten Diagrammrings zum Layer1, die Daten des zweiten zu Layer2 usw.

Das Datenobjekt, in dem die vom Diagramm darzustellenden Informationen gespeichert sind, heißt *dataset*. Wichtige Arrays und Datenobjekte sind:

- `dataset` speichert die Daten des Diagramms in der Form

  ```
  { layer1: [[ProdNr11, Betrag11, ausgeschriebener Name11] ...
  [P1n,B1n,aN1n]]
  layer2: [[P21,B21,aN21] ... [P2m, B2m, aN2m]] ... }
  ```

- `selectedArr`
  - speichert die Produktnummern ab, die durch einen Klick vom User ausgewählt wurden, um deren Unterkategorien einzusehen
  - hält sich an die Levelstruktur, ist aber mit 0 beginnend
  - Form: `[ProdNr1, ProdNr2, ProdNr3 ...]`
- `pathArr`
  - enthält sämtliche im Diagramm angezeigte Diagramm-Stücke als mit D3 ausgewählte Path-Objekte
  - hält sich an die Levelstrucktur, ist aber mit 0 beginnend
  - Form: `[[Path-Objekt11, PO12 ..., PO1n], [PO21,PO22, ...,PO2m] ... ]`
- `selectedIndexArr`
  - ähnlich zu textttselectedArr, enthält statt Produktnummern aber Path-Objekte
  - Form: `[path-Objekt1, PO2, PO3 ...]`

3.4.3 Wichtige Funktionen

- `clickFunction(d,i,j,k)`

  Hierbei handelt es sich um die Hauptfunktion der Diagramme. Sie baut neue Diagrammringe auf und auch überflüssige Datenstrukturen wieder ab und ist oft jeweils an einen bestimmten Ring gebunden, d. h. ein Diagrammring baut den nächsten auf und alle folgenden ab.

  - d (optional) Datensatz des vorhergehenden Rings.
  - i (pflicht) Index des aufgerufen Datenwertes.
  - j (pflicht) Index des aufgerufen Datensatzes.
  - k (abhängig) Wenn dieser Parameter gesetzt ist, wird davon ausgegangen, dass diese Methode nicht von einem Diagrammring ausgeführt wird. Sie muss deshalb eine Produktnummer enthalten.

- `cutOffArray(i)`

  Diese Funktion baut überflüssige Datenstrukturen ab und wird in der Regel von der `clickFunction` aufgerufen.

  - i gibt an, bis zu welcher Stufe Datenstrukturen abgebaut werden.

- `urlPathDia`

  Diese Funktion wird stets beim Aufruf der Seite aufgerufen. Sie prüft, ob in der URL Parameter mitgegeben wurden, und baut diesen Parametern entsprechend die Diagramme auf.

Alle weiteren Funktionen werden nur unterstützend hinzugezogen und dienen vorrangig der Realisierung von Darstellungsdetails.

3.4.4 Generelles Vorgehen

Beim Aufruf der Seite wird zunächst eine Anfrage mit Hilfe von Ajax gestartet, deren Antwort die Daten für das erste Level enthält. Diese werden mit Hilfe von verschieden Funktionen in die entsprechenden Arrays und Variablen eingespeichert.

Dann wird mit Hilfe von D3 der erste Diagrammring erstellt und mit Event-Listener-Funktionen versehen (unter anderem die `clickFunction`). Ab da werden weitere Ringe nur über die `clickFunktion` aufgebaut, welche weitere Daten über Ajax-Anfragen erhält. Für die Erstellung der Ajax-Anfragen werden die in `selectedArr` gespeicherten Daten ausgelesen und mit der Auswahl des Users (über die Parameter i und j ermittelbar) kombiniert, um festzustellen, welche Daten aus der Datenbasis ausgelesen werden müssen. Die Antwort auf die Ajax-Anfrage muss eine bestimmte Form haben, um lesbar zu sein:

```
'[[KopfzeileName1, KopfzeileName2, KopfzeileName2]
["Abteilungsnummer1", Betrag1, "Abteilung1"]
["An2",Betrag2,"B2"]...]'
```

Es handelt sich dabei um einen einzigen String, in dem nur eine Arraystruktur hineingeschrieben wurde.

3.4.5 Modifikationsvariablen

- `SIZE` legt die Größe des Diagramms fest.
- `limit` legt die Anzahl der Diagrammringe fest.
- `DiaURL` legt fest, von welcher externen Datei das Diagramm seine Daten bezieht.
- `steckURL` gibt an, von welcher Datei das Diagramm Produktsteckbriefe beziehen kann.
- `colorRing`: In diesem Array sind die vom Diagramm verwendeten Farben als Hexadezimalzahlen gespeichert.
- `grayFac` bestimmt, wie stark die Farbe eines nicht ausgewählten Segments einem Grauton angenähert wird.
- `whiteFac` bestimmt, wie stark die Farbe eines nicht ausgewählten Segments der Farbe weiß angenähert wird.

3.4.6 Diagramm

Neben den Tabellen Id's, die im oben genannten HTML-Teil vergeben werden, benötigt man nur die beiden Funktionen `drawTable(array,theTable)` und `clearTable(theTable)` (leicht umbenannt für das Ausgabendiagramm) zum Aufbau der Tabelle. `drawtable` übernimmt einen Array mit den zu zeichnenden Daten und erstellt daraus in dem angegebenen html-table Container die Tabelle. `clearTable` leert den angegebenen Container und sollte vor `drawTable` genutzt werden.

4 Frontendgestaltung – Partizipationspaket

4.1 Forum

Die erste zentrale Komponente der partizipativen Ebene des „Interaktiven Haushaltsrechners“ ist das Forum, welches sowohl für allgemeine Diskussionen, als auch zur Ausarbeitung oder Diskussion von Vorschlägen verwendet werden kann.

Das Forum wird durch die Drupal-eigenen Module *Forum* und *Advanced Forum* implementiert. Die Nutzerverwaltung wird so von Drupal selbst übernommen und muss nicht extern verwaltet werden. Weiterhin ist es so einfacher für Informationsebene und Partizipationsebene ein einheitliches Design zu erstellen, da beiden dasselbe Drupal Theme zu Grunde gelegt werden kann.

Es ist wie folgt zu strukturieren:

- Das Forum muss moderiert sein.

  Hierzu werden verschiedene Userrollen im Forum benötigt. Diese werden direkt von Drupal verwaltet und mit entsprechenenden Rechten versehen.

- Es gibt bei der Registrierung einen Klarnamenzwang. Dieser Klarname wird nicht öffentlich angezeigt.

- Es existiert ein Melde-Button zum Melden von Beiträgen.

  Der Melde-Button wird durch die Drupal-Module *Flag* und *Flag Abuse* verwirklicht.

- Diskussionen zu einem konkreten Vorschlag finden sowohl unter dem Vorschlag, als auch im Forum statt. Ein Kommentar unter dem Vorschlag wird jedoch im Forum angezeigt und umgekehrt.

- Die Vorschläge und Kommentare sollen bewertet werden können. Dazu kommt das Drupal-Modul *Rate* zum Einsatz.

- Der Aufbau vom Forum sollte sich an folgenden Bereichen orientieren:

  1. Allgemeine Diskussionen
  2. Vorschläge (Unterteilung in die Kategorien, verknüpft mit den Vorschlägen, Diskussion / Fragen im Forum, liken etc beim Vorschlag)
  3. Konkrete Themen zum Diskutieren (z.B. „Wie gehen wir mit Investitionen in Leipzig in Zukunft um?“)

4.2 Vorschläge

Die zweite zentrale Komponente des Partizipationspakets sollte sowohl bei der jeweiligen Kategorie im Informationsbereich zur Verfügung stehen, als auch in einem extra Formular. Nutzer sollen so in der Lage sein, Vorschläge zur Verbesserung des städtischen Haushalts einreichen zu können. Dazu soll es in der Informationsebene zu jeder Hierarchieebene des Haushalts einen Button geben der direkt auf ein Formular verweist über das ein Thread im Forum erstellt wird. Jeder Vorschlag wird in eine Kategorie eingeteilt, die der höchsten Abstraktionsebene entspricht. Die Kategorien, welche keine Möglichkeit zur Mitbestimmung liefern, sollten auch nicht auswählbar sein. Es sollte die Möglichkeit bestehen, Vorschläge zu bewerten und zu kommentieren. Die Vorschläge und die Diskussion dazu sind mit dem Forum verbunden.

4.3 Bürgereinwände

Mit Blick auf die bis zuletzt unklare Anforderungslage zum Thema „Bürgereinwände“ sowie die generellen Unwägbarkeiten, ob überhaupt ein Forum wie im Prototyp vorgeschlagen zum Einsatz kommt, wurde nur die folgende Variante eines *einfachen Bürgereinwands* umgesetzt:

- Angemeldete Benutzer können einzelne Vorschläge über einen Knopf als (einfachen) Bürgereinwand einreichen.

  Ein spezifischer Registrierungsprozess, wie für *qualifizierte Bürgereinwände* gefordert, findet dabei nicht statt. Moderatoren haben Zugriff auf die für sie durch die Administration freigeschalteten persönlichen Daten der Einreicher, welche diese zu ihrem Account übermittelt haben, und können auf diesem Weg Benutzerinformationen einsehen und ggf. weitere Informationen über die Kontaktfunktionen der Plattform anfordern.

- So markierte Vorschläge werden zusammen mit dem Verweis auf den Einreicher in einer drupalinternen Liste aggregiert, die nur in der Moderatorensicht zugänglich ist.

Damit kann erfasst werden, welche Vorschläge wie oft als Bürgereinwände eingereicht wurden.

- Reaktionen der BEBS auf einzelne Vorschläge (Verwaltungsstandpunkt) können als Beitrag im entsprechenden Vorschlagsthread veröffentlicht werden.

Ausführungen zu möglichen Erweiterungen dieses Konzepts sind im Dokument „Anforderungserhebungen“ zu finden.

5 Datenmodell

Die vom Auftraggeber zur Verfügung gestellten Daten für unser Projekt bestehen aus mehreren Excel-Dateien, die per dynamischer Listenausgabe aus SAP exportierte relevante Daten zum Doppelhaushalt 2015/16 enthalten. In Zuge des Redesigns wurden diese Daten in RDF Cube Serien transformiert. Weiterhin wurden aus den Daten der Produktbaum, das Produktmodell sowie die Produktbeschreibungen der Schlüsselprodukte extrahiert. Ergänzend enthalten die Daten einen RDF Graphen *Config*, der die Jahreszuordnung der eingesetzte RDF Data Cube Serie näher beschreibt, sowie eine Beschreibung des Dataset-Formats der RDF Data Cube Serie.

Details hierzu sind im Dokument *Designprinzipien des IHR-15 RDF Data Stores* zu finden.

6 Dokumentationskonzept

6.1 Interne Dokumentation

Als internes Dokumentationsformat wird einerseits für in PHP programmierte Elemente das Software-Dokumentationswerkzeug *PHPDoc* verwendet. Diese Dokumentation kann mit entsprechenden Tools wie z. B. *phpDocumentor* aus dem Quelltext generiert werden. Andererseits wird für die Dokumentation des Diagramm-Quellcodes *JSDoc* verwendet.

7 Glossar

Drupal. Drupal 7 ist ein freies (GPL) und kostenloses Content-Management-Framework auf der Basis von PHP. Es besteht aus einem Core, der aus anderen Content-Management-Systemen bekannte Funktionen liefert, und Modulen, mit denen man zusätzliche Funktionen implementieren kann.

Forum. Ein Internetforum dient als virtueller Platz zum Austausch von Gedanken, Meinungen und Erfahrungen. Ein großer Vorteil dieser Kommunikationsform ist deren Asynchronität, d. h. dass Diskussionsteilnehmern ermöglicht wird, zeitversetzt zu antworten. Üblicherweise werden zur Unterhaltung sogenannte *Threads* erstellt (= Themen), in denen einzelne Diskussionsstränge zu einem übergeordnetem Thema gebündelt sind.

Model-View-Controller. Als eines der beliebtesten Architekturmodelle der objektorientierten Programmierung ermöglicht das MVC-Muster [6] eine klare Rollenteilung im Programm in die drei Einheiten Datenmodell (*model*), Präsentation (*view*) und Programmsteuerung (*controller*). Der „Observer“ (controller) nimmt hierbei eine zentrale Rolle ein – er ermöglicht es, auf Veränderungen in den zu beobachten Objekten zu reagieren. Diese werden durch das „Model“ und den „View“ beschrieben.

Das „Model“ bietet Zugang zu zentralen Daten des Programms und Funktionen, um diese abzurufen oder zu verändern. Diese Daten werden dann in dieser Architektur vom „View“ dem Benutzer zugänglich gemacht. Auf Benutzereingaben reagiert von hier an wieder der „Observer“ und kann so die Daten abhängig vom Programmverlauf ändern.

Presentation-Abstraction-Control. ist ein von Drupal verwendetes Architekturmodell [7], das MVC ähnelt. Es teilt ein System in die Komponenten „Presentation“, „Abstraction“ und „Control“ ein. „Presentation“ ist dabei für die Darstellung der Daten zuständig, „Abstraction“ für die Verwaltung der Daten und „Control“ für Flusskontrolle und Kommunikation zwischen den beiden anderen Komponenten.

RDF. Das Resource Description Framework [8] als Konzept erweitert die Möglichkeit des Webs, Inhalte miteinander zu verbinden. Hierbei wird versucht, nach semantischen Zusammenhängen zu ordnen. Die Aussagen die die eigentlichen Daten in RDF sind, betreffen sogenannte Ressourcen. Diese Ressourcen sind eindeutig bezeichnete Dinge der Welt, die durch Tripel oder Graphen in eine Relation mit anderen Ressourcen gebracht werden. Nicht nur durch das mit RDF im Zusammenhang benutzten Vokabular ähnelt dieses Modell stark an die klassischen Modelle wie UML-Diagramme oder Entitiy-Relationship-Modelle, auch die Modellierungsweisen sind stark miteinander verknüpft, wodurch RDF als Grundansatz des Semantischen Webs an Bedeutung gewinnt.

RDF Data Cube. RDF Data Cube [9] ist ein spezielles Vokabular zur Beschreibung von Daten im RDF-Standard. Durch das Bereitstellen von einer beliebigen Anzahl von Dimensionen ist das Format vor allem zur Beschreibung von statistischen Daten oder Messdaten geeignet, die nach einem festen Schema mehrfach erhoben werden.

SPARQL. SPARQL als rekursives Akronym für *SPARQL Protocol And RDF Query Language*) [10] ist eine Abfragesprache für RDF, d. h. eine semantische Abfragesprache für Datenbanken, mit der man im RDF-Format gespeicherte Daten aufrufen, durchsuchen und manipulieren kann.

Triplestore. Ein Triplestore ist eine spezielle Datenbank zur Speicherung und Abfrage von Tripeln durch semantische Abfragen. Ein Daten-Tripel besteht aus Subjekt, Prädikat und Objekt, z. B. „Franz kennt Fritz“.

BorderClound SPARQL 1.1 PHP Bibliothek. BorderCloud [3] ist eine Programmierbibliothek für PHP, die Funktionen zum Senden von Anfragen und Verwalten von Antworten an einen Tripelstore-Endpunkt zur Verfügung stellt.

Bootstrap-Framework. Bootstrap [2] ist ein Open-Source Framework für HTML, CSS und JavaScript zur Erstellung von Webseiten.

Literatur

[1] BITV 2.0. `https://github.com/BorderCloud/SPARQL`.

[2] Bootstrap-Framework. `http://getbootstrap.com/`, `https://www.drupal.org/project/bootstrap`.

[3] BorderCloud SPARQL 1.1 PHP Bibliothek. `https://github.com/BorderCloud/SPARQL`.

[4] Javascript-Bibliothek D3. `http://d3js.org/`.

[5] Drupal 7. `http://de.wikipedia.org/wiki/Drupal`.

[6] MVC Konzept. `http://de.wikipedia.org/wiki/Model_View_Controller`.

[7] PAC Konzept. `http://de.wikipedia.org/wiki/Presentation-abstraction-control`.

[8] RDF. `http://en.wikipedia.org/wiki/Resource_Description_Framework`.

[9] RDF Data Cube. `http://www.w3.org/TR/vocab-data-cube/`.

[10] SPARQL. `http://en.wikipedia.org/wiki/SPARQL`.