h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

Linked Data – Anwendungsentwicklung

Kalorienrechner durch EAN Pflege auf Basis von OpenFoodFacts

Sommersemester 2021

Eine Ausarbeitung von:
Alexander Ceynowa (729668) und Timm Reymendt (750640)

Hochschule Darmstadt, University of Applied Sciences

FB Media, Studiengang Informationswissenschaften

Veranstaltung: Linked Data - Anwendungsentwicklung

Fertigstellung: 08.08.2021

Inhaltsverzeichnis

1. Erwartungen an die Veranstaltung (Timm)

Die beiden Autoren hatten ähnliche Erwartungen an die Veranstaltung. Sowohl für Timm als auch für Alexander war die Praxis der Linked Data Neuland - beide hatten zwar schon mehrfach Internetapplikationen in Form von Websites erstellt, aber noch nicht mit Linked Data gearbeitet. Daher waren gute Grundlagen für die Ausarbeitung einer Anwendung im Linked Data Bereich gegeben. Beide Studenten waren sich darüber bewusst, dass das Seminar eine interdisziplinäre Projektarbeit verlangt, da dies bei einem solchen Thema die Praxis am besten vermitteln kann. Da Alexander schon einmal mit einer Applikation, die die Themen weitestgehend abdeckt in Berührung kam, war der technische Input für ihn nicht so wichtig, wie für Timm. Beide Studenten erwarteten von der Veranstaltung einen guten, praktischen Einblick in die Anwendungsentwicklung für Linked Data zu bekommen.
Zusätzlich zu dem “reinen” praktischen Einblick war es für beide interessant, zu sehen wie weit sich die theoretischen Vorstellungen einer Umsetzung von dem unterscheiden, was sich dann praktisch im finalen Projekt abspielt.

2. Motivation (Alexander)

Als Grundlage für das Projekt diente ein Export der Daten des OpenFoodFact -Projektes (OFF). Einer der Gründe, warum speziell diese Daten genutzt werden sollten, war, dass aus früheren Projekten, unter der Nutzung von MySQL, gewisse Leistungseinbußen bei größeren Datenquellen bekannt waren. Also bot sich mit dieser praktischen Hausarbeit die Gelegenheit Leistungsunterschiede zwischen relationalen Daten und Linked Data festzustellen. Dabei spielte es für die konkrete Umsetzung dieses Projektes keine Rolle, ob die vorherigen Einbußen durch das technische Setup, die Anfragen oder anderen Gründen hervorgerufen würden.
Der Datenexport von OFF beinhaltet zum jetzigen Zeitpunkt rund 63 Millionen Statements, konnte also gut zum Vergleich mit einer bereits vorhandenen MySQL-Datenbank genutzt werden. Die konkrete Nutzung der Daten, nämlich den Betrieb eines Kalorien- und Nährwertrechners, ergab sich aus der Natur der vorhandenen Werte.
Da diese Informationen über ein Community-System erfasst werden (jeder kann eintragen, ergänzen und verändern), mussten ebenfalls Situationen bedacht werden, in denen „Datensätze“ (Nährwertangaben) nicht vollständig waren, also etwa der Vitamin B Gehalt fehlt. Hier gab es die Möglichkeit sich näher mit dem Verhalten von GraphDB / Linked Data / SPARQL auseinander zu setzen, welches sich in solchen speziellen Fällen erheblich von MySQL unterscheidet.
Ein weiterer Grund für die Nutzung von GraphDB und Linked Data war die Tatsache, dass viele Fächer und Themen im Laufe des Studiums nur theoretisch betrachtet werden, im Verlauf des Seminars “Linked Data Anwendungsentwicklung” gab es aber die Möglichkeit das Wissen eines Kurses auch praktisch “in der wahren Welt” anzuwenden, ohne Konsequenzen durch Unwissenheit zu erleiden.

3. Anforderungen (Timm und Alexander)

Anhand des Projekts, welches im Rahmen des Linked Data Anwendungsentwicklung-Seminars entstand, sollten die Vor- und Nachteile von Linked Data, sowie die Unterschiede

zwischen Linked Data und traditionellen relationalen Datenbanken (wie etwa MySQL) erkundet werden.
Hierzu fiel die Entscheidung, eine Applikation zu entwickeln, welche eine unbestimmte Anzahl Barcodes beliebiger Produkte einliest und anschließend die zusammengestellten Nährwerte ausgibt. Hierfür gibt es eine riesige Datenbank im Internet, die als Datenquelle genutzt wurde: OpenFoodFacts (https://openfoodfacts.org). In dieser Datenbank sind über 1,8 Millionen Produkte zusammen[1] mit ihren Nährwerten eingetragen und jedem Nutzer werden diverse APIs und Datenexporte kostenlos zur Verfügung gestellt.
Für den regulären Nutzer wäre es vermutlich einfacher, wenn anstelle von Barcodes direkt Produkte (Mehl, Milch, Sahne) angegeben werden könnten, aber da OpenFoodFacts speziell Daten über spezifische Nahrungsmittel (Mehl 550 von Ja! Milch von Demeter oder Sahne von Schwälbchen) speichert und anbietet, wurde eine Entscheidung für die Nutzung der Barcodes und gegen beispielsweise Autocomplete mit Ajax-Lookup oder einfach die Nutzung des nächst-passenden Produktes getroffen.
Zu Anfang bestand die Überlegung diese Applikation zusätzlich mit einer API einer Rezeptwebsite zu verbinden - sodass automatisch anhand eines Rezeptes und dessen Inhalten und Mengenangaben der gesuchte Nährwert angegeben wird. Leider war das aufgrund der wenig, bis nicht vorhandenen freien Rezept-APIs und des Arbeitsumfangs im Rahmen dieses Seminars jedoch nicht realisierbar.
Somit fiel die Entscheidung, bei der einfachen Applikation ohne Rezept-Anbindung zu bleiben.
Die Anforderungen waren also wie folgt:

1. Barcode eingeben (oder einscannen)
    a. Die Applikation muss einen dynamischen Input erzeugen, bei dem man Felder beliebig hinzufügen und entfernen kann, damit jedes Rezept ohne weitere Probleme eingepflegt werden kann.
2. Menge (pro Zutat) eingeben
3. Bei Abschicken der Zutaten -> Verbindung zu den OpenFoodFact-Daten und Abruf der benötigten Daten
4. *Bei Nicht-Vorhandensein eines bestimmten Nährwertes soll auf das entsprechende "parent product" zugegriffen werden, um einen Durchschnittswert zu erlangen (vgl. Mehlsorte Typ 550 von JA! - Mehlsorte Typ 550 allgemein)**
5. Benötigte Daten werden auf die entsprechende Mengenangabe heruntergerechnet
6. Alle Nährwerte der eingepflegten Zutaten (heruntergerechnet auf die Menge) zusammenrechnen und auf der Website ausgeben
7. Angabe eines durchschnittlichen Nova Scores

***** Leider war die Implementierung dieser Funktion nicht ohne weiteres möglich, da die Datensätze der einzelnen Produkte nur schwer nachvollziehbar mit der übergeordneten Gruppe verknüpft sind. Daher wurde dieser Schritt aus den Anforderungen entfernt.

[1] https://world.openfoodfacts.org/

4. Implementierung (Timm)

Bevor mit der Implementierung begonnen werden konnten, musste noch einiges an Vorarbeit geleistet werden. Hierzu wurde die gesamte Datenbank von OpenFoodFacts heruntergeladen und anschließend in einen eigenen GraphDB-Server importiert. Glücklicherweise bietet OFF die gespeicherten Daten in diversen Formaten zum Download unter https://world.openfoodfacts.org/data an.
Bevor mit der Entwicklung des Backendes begonnen wurde, musste das Frontend vorbereitet werden. Die Wahl eines geeigneten Frontend-Frameworks fiel hier auf Bootstrap, da dieses Framework aus der Programmierwerkstatt und anderen Veranstaltungen bereits bekannt war.
Nachdem die "initiale" Arbeit (Header, Footer etc.) im Frontend erledigt war, wurden zwei Input Fields zusammen mit einem Button zum Hinzufügen einer Zutat und deren Gewicht (+) und einem Button für das Löschen eines Feldes (-) erzeugt. Diese konnten dann genutzt werden, um ein JavaScript zu implementieren, welches beim Klick auf den (+) Button ein neues Feld erzeugt (einfaches OnClick Script). Mit einer RemoveChild Funktion konnten anschließend auch der Löschen Button (-) funktional gemacht werden.
Nachdem die Felder für die Inputs erledigt waren, wurde ein Output Feld vorbereitet, welches die Daten, die aus GraphDB mithilfe der Inputs exportiert wurden, darstellen kann. Dieses wurde mit einer IF-Funktion entwickelt, damit die Outputs nicht angezeigt werden, falls noch keine Daten vorhanden sind.
Nun waren Input, sowie Output vorbereitet, und das Backend konnte entwickelt werden. Das Backend ist dafür zuständig mittels der Barcodes alle verfügbaren Informationen aus den Datensätzen von OFF zu extrahieren. Hierfür werden alle in das Formular eingegebenen Daten (Barcodes, Mengen und Anzahl der Portionen) an das Backend gesendet. Dort werden die Mengen- und Portionsangaben genutzt, um zu berechnen welches Gewicht eine resultierende Portion hat. Als nächstes werden über eine SPARQL-Abfrage alle verfügbaren Informationen über die Nahrungsmittel aus der Datenbank abgerufen und von "pro 100g"-Werten zu der der Zutat entsprechenden Mengenangabe umgerechnet. Diese Mengen werden für alle Zutaten zusammengezählt und am Schluss durch die Anzahl der Portionen geteilt. Im Backend finden einige Tests statt, ob alle Zutaten beispielsweise Angaben über den Vitamin C Gehalt haben, dieses Feedback wird aber aktuell noch nicht genutzt. Abhängig von den angefragten Zutaten werden von OpenFoodFacts auch Allergene wie Soja, Erbsen oder Laktose erfasst, welche zu einem späteren Zeitpunkt ebenfalls Anwendung finden könnten.
Der NOVA-Score, ein Wert, der sich auf den Grad der Verarbeitung von Produkten bezieht[2] und sich im Rahmen zwischen 0 (Unbekannt) und 1 ("Gesund") bis 4 ("Ungesund") bewegt, wird einfach als ein Durchschnitt der gefundenen Scores berechnet. In der aktuellen Version der Anwendung werden fehlende Barcodes und die Mengenanteile von Zutaten nicht berücksichtigt.

5. Ergebnis (Alexander)

Im Verlauf des Projekts stellte sich zunächst heraus, dass Linked Data dem Konzept von MySQL nicht unähnlich ist. Die Ähnlichkeiten zwischen MySQL und SPARQL sind allerdings so

[2] https://world.openfoodfacts.org/nova

groß wie auch die Unterschiede. Die Art der Abfrage ist bei beiden ähnlich, hat aber nicht zu viel miteinander zu tun - etwa wie beim Vergleich von Englisch und Deutsch: Beide Sprachen benutzen lateinische Buchstaben und einige Grundregeln der Grammatik, manche Wörter lassen sich zwischen den Sprachen inferieren, aber sonst sind die Sprachen nicht vergleichbar. Bei MySQL und SPARQL verhält es sich ähnlich. Eine Anfrage besteht aus dem, was angefragt wird, also “ich möchte X angezeigt bekommen”, und (in diesem Fall) der Kondition ”wo Y”, doch die Formulierung der Anfrage sowie die Handhabung der Antwort des Servers unterscheidet sich bei “korrekter” Umsetzung in beiden Systemen. Im Falle dieses Projektes machte es im Nachhinein keinen Unterschied, ob MySQL oder SPARQL genutzt wurde - dies hing jedoch davon ab, was erreicht werden sollte und wie es umgesetzt wurde.
Im Verlauf des Projekts zeigte sich, dass Linked Data zwar ein mächtiges Konzept ist, aber wie jedes Werkzeug dem Zweck angemessen sein muss. Für dieses Projekt wäre die Arbeit mit “konventionellen” Mitteln, also einer MySQL Datenbank oder einer Anbindung an einer API direkt von Open Food Facts einfacher und schneller gewesen, als mit GraphDB, SPARQL und einer PHP-Library, wie es jetzt läuft.
Einer der ursprünglichen Gründe, um eine Linked Data Datenbank zu nutzen, war der Aspekt der Hierarchie von Produkten. Um noch einmal zusammenzufassen: Ziel war es, eine Art Kalorienrechner zu programmieren. Eine Liste von Barcodes sollte zusammen mit Mengenangaben und resultierender Portionen Anzahl in ein Formular eingegeben werden, eine durchschnittliche Menge von Inhaltsstoffen pro Portion käme als Ausgabe zurück. Der hierarchische Aspekt war hier interessant, da Produkte von OFF in Kategorien gegliedert werden. Die Produktgruppe Mehl besteht beispielsweise aus allen eingetragenen Sorten Mehl, die Untergruppe Weizenmehl nur aus allen eingetragenen Sorten Weizenmehl und so weiter. Für jede “Gruppe” wird dann aus den eingetragenen Produkten für jeden eingetragenen Inhaltsstoff ein “Gruppendurchschnitt” (von OFF) berechnet. Der Plan war es, die Durchschnittsdaten zu nutzen, sollte ein Produkt etwa namentlich in der Datenbank vorhanden sein, aber keine Nährwertdaten hinterlegt haben. Die Dokumentation der OpenFoodFacts-Datenbank ließ allerdings zu wünschen übrig und auch das Datensatz-Schema war nicht einsehbar - aus diesen Gründen wurde dieser Aspekt außen vor gelassen, weshalb von den Stärken und Vorteilen von Linked Data nur kaum bis wenig profitiert werden konnte. Basierend darauf konnte das Fazit gezogen werden, dass der Einsatz von Linked Data (und natürlich auch von relationalen Datenbanken) dem Zweck absolut angemessen sein sollte.
Nachdem das Projekt fertig war, kam noch eine weitere Möglichkeit für die Anwendung auf - während das finale Produkt nicht oder nur wenig auf die wesentlichen Fähigkeiten von Linked Data zurückgreifen konnte, wäre ein alternativer Ansatz gewesen, die Liste an Zutaten einzulesen und nach “gesünderen” Alternativen in der Datenbank zu suchen. Dies hätte andere Probleme mit sich gebracht, wie etwa die lokale Verfügbarkeit von Erzeugnissen, da es sich um eine (theoretisch) internationale Datenbank von Produkten handelt, wäre aber definitiv durch die Struktur der Daten (übergeordnete Gruppen von Produkten) eine gute Möglichkeit gewesen, um das Potential von Linked Data weiter zu erkunden.

6. Lessons Learned (Alexander und Timm)

Das Einrichten einer GraphDB, eines Webservers für die resultierende Website und auch der Import der Daten in die GraphDB funktionierten ohne Probleme. Der Umgang mit GraphDB

und auch SPARQL war intuitiver als erwartet und schnell durch “learning by doing” gelernt. Schwieriger wurde es bei der Formulierung von SPARQL-Anfragen. Die fehlende Dokumentation der Daten und der Datenschemas brachte viele Experimente mit sich, auch anwendungsspezifische SPARQL-Beispiele an sich waren nicht so leicht auffindbar wie erwartet. Wirklich “kompliziert” war nur die Dokumentation der PHP-Library, die für die Kommunikation mit dem SPARQL-Endpunkt genutzt wurde. Hier wurde viel Zeit darauf verwendet, grundlegende Funktionalität zu erreichen.
Auch fiel während des Implementierungsprozesses (wie bereits im Kapitel “Anforderungen” erwähnt) auf, dass die Datenbank von OpenFoodFacts scheinbar keine guten Verlinkungen zu parent-Produkten hat. Daher konnte Schritt 4 aus den wohlüberlegten Anforderungen nicht implementieren werden. Dieser hätte nochmal sehr schön das Thema “Linked Data” aufgegriffen. Learning hieraus ist, das Anforderungsmanagement erst zu leisten, sobald die genauen Funktionalitäten einer Datenbank analysiert wurden. Die selbstverständliche Annahme, dass die Datensätze dieser Anforderung gewachsen waren, war falsch.
Insgesamt konnte sehr viel aus der Veranstaltung mitgenommen werden. Die vorab gestellten Erwartungen, einen guten, praktischen Einblick in die Anwendungsentwicklung von Linked Data zu bekommen, wurden zu einhundert Prozent erfüllt und das Seminar lässt sich sehr gut als Aufbauseminar zum Grundkurs “Linked Data” empfehlen. Vor allem das theoretisch in Linked Data Gelernte in der Praxis direkt an der Anwendung zu implementieren und zu “erleben” hat sehr geholfen, das Wissen in dem Bereich zu festigen und das Gelernte noch besser zu verstehen.

7. Konstruktive Kritik der Veranstaltung

7.1. Alexander

Meine Erwartungen an die Veranstaltung sind erfüllt worden. Ich hatte mir erhofft, weiter in das Thema Linked Data einzutauchen und Inhalte aus der Vorlesung quasi unter “Anleitung” auch in der Praxis umzusetzen. Wirklich angenehm war der Aufbau mit festen Meetings und den zwei oder drei Terminen dazwischen, wo nach Bedarf ein Treffen vereinbart werden konnte. Dadurch wurde gefühlt genug Zeit gegeben, um mit der Technik zu experimentieren und dann bei Bedarf “sofort” oder “später” Rückfragen zu stellen.
Einen aktiven Vorschlag zur Verbesserung der Veranstaltung kann ich nicht bieten, ich war sehr zufrieden mit den Inhalten und der Umsetzung.
Zuletzt: Ich persönlich werde mich soweit möglich weiter mit dem Thema auseinandersetzen. Das Projekt aus LDA in Kombination mit den Ideen, die während der Durchführung entstanden sind, haben für mich Potenzial zu einem kleinen Hobbyprojekt.

7.1. Timm

Auch meine Erwartungen sind vollständig erfüllt worden. Im Gegensatz zu anderen Veranstaltungen, welche in diesem Semester online angeboten wurden, hat Herr Thull einen sehr guten Mittelweg zwischen Input in Form von Theorie und der eigentlichen Praxis gefunden. Auch wurde der Input gut vermittelt, während der Vorlesung wurde auf jedes Feedback und jede Frage sehr gut eingegangen und alles geklärt.

Besonders gefallen hat mir, dass während der Implementierungsphase die Termine optional gesetzt wurden, sodass man sich selbst entscheiden konnte, ob man während der Vorlesungszeit in einer Besprechung Rückfragen beantwortet, oder ob man die Zeit nutzt, um weiter an der Applikation zu entwickeln.

Einen konstruktiven Kritikpunkt muss ich allerdings loswerden: Die Technik hat während der Vorlesung nicht immer mitgespielt, oft war Herr Thull für mich schwer zu verstehen - das kann aber auch an meiner eigenen Internetverbindung gelegen haben.
Wenn ich hier einen Tipp geben dürfte, wäre das, sich für zukünftige Veranstaltungen ein ordentliches Headset zu besorgen (am besten ein Gaming Headset), welches einfach viel bessere Tonqualität liefert.

Die Veranstaltungen online zu besuchen wird (hoffentlich) bald wieder der Vergangenheit angehören, weshalb dieser Kritikpunkt sich in den nächsten Semestern wieder erledigt haben könnte. Alles in allem muss ich aber auch sagen, dass Herr Thull mit der Gesamtsituation sehr gut umgegangen ist und, wie bereits erwähnt, sehr schöne Wege gefunden hat, die Veranstaltung online abzuhalten.

8. Leistungstagebuch

8.1. Alexander

Die folgenden Einträge sind eine Kombination aus handschriftlichen Notizen und den Git-Commits von Alexander. Zeiten sind auf die nächsten 5 Minuten gerundet.

15. Mai 2021
Einrichten eines Git-Repositories (5 Minuten)
Einrichten von GraphDB (10 Minuten) und Import der Daten (5 Minuten Arbeit, 40 Minuten warten)

29. Mai 2021
Einrichten von Composer, “Installation” von easyrdf-lib (30 Minuten)
Erste Experimente mit SPARQL in GraphDB (60 Minuten)

04. Juni 2021
Einrichten von automatischem Deployment des Git-Repositories auf eine Subdomain (5 Minuten)

13. Juni 2021
Lesen der easyrdf-lib Dokumentation (40 Minuten)
Erste Experimente mit easyrdf-lib in Kombination mit eigenem SPARQL-Endpunkt (105 Minuten)

21. Juni. 2021
Grundlegendes Formular für Kalkulation, funktionaler Prototyp der Kalkulation im Backend (125 Minuten)

28. Juni 2021
Einfügen von JQuery, JQuery-UI und Bootstrap für das Frontend (5 Minuten)

07. Juli 2021
Finalisierung des Backendes, anpassen der Berechnung für durchschnittliche Nährwerte (70 Minuten)
Bug fixes und Änderungen am Styling für das Frontend (20 Minuten)

12. Juli 2021
Finale Änderungen an den Formeln (runden zum Beispiel), UI finalisiert, einfügen von benötigten Grafiken (20 Minuten)

Finale Arbeitszeit für Alexander: 500 Minuten bzw. 8 Stunden und 20 Minuten

8.2. Timm

Die folgenden Einträge sind eine Kombination aus handschriftlichen Notizen und den Git-Commits von Timm. Zeiten sind auf die nächsten 5 Minuten gerundet.

04. Juni 2021
Erstellung index.php - erste Testings für dynamische Inputs (120 min)

07. Juni 2021
Get-in-Touch und erste Testings mit SPARQL in GraphDB (60 min)

28. Juni 2021
Designanpassungen für index (NavBar, FormFields, Cols, Containers, FormControls) - Javascript für wiederholende Inputs / JQuery Abfragen / Background Anpassung / Button Implementierung / Post Scripts (100 min)

07. Juli 2021
Testimonials für Inputs / Testing für dynamical JS Fields (50 min)

09. Juli 2021
Bugfixing (30 min)

11. Juli 2021
Creation of Output Field (Container) / Insertion of Output values / Testing for Styling (60 min)

12. Juli 2021
End User Testing (30 min)

Finale Arbeitszeit für Timm: 450 min bzw. 7 Stunden und 30 Minuten