REST Tabelle

Produkte

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ressource** | **Methode** | **Semantik** | **Content-type(req)** | **Content-type(res)** | **HTTP Statuscodes** |
| /Item | GET | Informationen von allen Items |  | Applikation/JSON | 200, 500 |
| /Item/{id} | GET | Information eines bestimmten Items |  | Applikation/JSON | 200, 404, 500 |
| /Item | POST | Fügt ein neues Item hinzu | Applikation/JSON | Applikation/JSON | 200,201,500 |
| /Item/{id} | PUT | Aktualisiert Item Daten | Applikation/JSON | Applikation/JSON | 200, 201, 204, 500 |
| /Item/{id} | DELETE | Entfernt ein Item | Applikation/JSON |  | 200, 500 |

User

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ressource** | **Methode** | **Semantik** | **Content-type(req)** | **Content-type(res)** | **HTTP Statuscodes** |
| /User | GET | Gibt alle Nutzer aus |  | Applikation/JSON | 200, 500 |
| /User/{id} | GET | Information eines bestimmten Nutzers |  | Applikation/JSON | 200, 404, 500 |
| /User | POST | Fügt einen neuen Nutzer hinzu | Applikation/JSON | Applikation/JSON | 200, 201, 500 |
| /User/{id} | PUT | Aktualisiert Nutzer Daten | Applikation/JSON | Applikation/JSON | 200, 201, 204, 500 |
| /User/{id} | DELETE | Löscht Nutzer | Applikation/JSON |  | 200, 500 |

Bestellung

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ressource** | **Methode** | **Semantik** | **Content-type(req)** | **Content-type(res)** | **HTTP Statuscodes** |
| /Bestellung | GET | Gibt alle Bestellungen aus |  | Applikation/JSON | 200, 500 |
| /Bestellung/{id} | GET | Informationen einer bestimmten Bestellung |  | Applikation/JSON | 200, 404, 500 |
| /Bestellung | POST | Erstellt eine Bestellung | Applikation/JSON | Applikation/JSON | 200, 201, 500 |
| /Bestellung/{id} | PUT | Aktualisiert eine Bestellung | Applikation/JSON | Applikation/JSON | 200, 201, 204, 500 |
| /Bestellung/{id} | DELETE | Löscht eine Bestellung | Applikation/JSON |  | 200, 500 |

*Ressourcen*

Die Ressourcendefinition wurde von unserem überarbeitetem Domeinmodell sowie unserer Projektbeschreibung abgeleitet.  
Da unser Domainmodell einen Kreislauf besaß war es einfacher die einzelnen Ressource herauszuleiten.   
Links oben konnte man bei dem Nutzer/in anfangen (Ressource: Kunde) der eine Bestellung tätigt (Ressource: Bestellung) die daraufhin an einem Lieferanten (Ressource: Lieferant). Nachdem der Lieferant die Bestellung annimmt bekommt er eine Liste an verfügbaren Einkaufsläden, die für die Bestellung am geeignetsten sind (Ressource: Einkaufsladen).  
Des weiteren konnte man die Ressource Produkte aus der Bestellung Ressource ableiten.

* Nutzer
  + Vor/Nachname
  + ID
  + Alter
  + Username
  + Kunde
    - Adresse
    - Bezahlmittel
  + Lieferant
    - Smartphone
    - Fortbewegungsmittel
    - Personal-Score
    - Status (Beschäftigt oder Verfügbar)
* Produkte
  + ID
  + Art
  + Preis
  + Gewicht
  + Standartmenge
* Einkaufsladen
  + ID
  + Öffnungszeiten
  + Produkte
* Bestellung
  + ID
  + Produkt
  + Gesamtpreis
  + Gewicht

Path / Quarry parameter und Primär-/Subressourcen sowie Listenresourcen.

Wir Benutzen Quarry-parameter, um z.b in Items nach Produkten einer Bestimmten Art zu suchen.  
Der Server liefert dann eine Liste aller Produkten, die mit der Art übereinstimmen.

Da wir in unserer Anwendung mehr als eine Zielgruppe haben besitzt die Ressource Nutzer zum einen die Sub Ressource Kunde sowie die Sub Ressource Lieferant. Es wurde auch noch überlegt ob man bei der Primärressource „Produkte“, Art als Sub Ressource macht jedoch ist das weniger sinnvoll da es nachher nur aus einer ID und einem Schlüsselwort bestehen würde.

REST-Prinzipien in Bezug auf Spacedeliver

„REST APIs must be hypertext-driven“ betont Roy Fielding in einem Blog-Beitragx1 und um das zu schaffen orientierten wir und am Richardson Maturity Modelx2, Dabei haben wir folgende Punkte umgesetzt:

1. Jede Ressource hat seine eigene URL
2. Jede Ressource hat eine oder mehrere Repräsentationen vorgesehen, die über das Internet übertragen werden können
3. In jeder REST- Nachricht sind alle Informationen enthalten, die der Server/Client benötigt, um die Nachricht richtig interpretieren zu können.  
   beim Austausch von Nachricht werden keine Informationen über den Zustand gespeichert
4. Alle Ressourcen haben ein gleichmäßigen Satz von CRUD-Operationen (Create, Read, Update, Delete)
5. Hypermedia Mit der Repräsentation einer Ressource wird auch auf Links anderer Ressourcen-Entitäten referenziert, die für den Ablauf des weiteren Dialoges wichtig sind

# **REST Alternative**

Als alternative zu REST gibt es GraphQL, eine von Facebook entwickelte SQL-ähnliche Laufzeitumgebung, die seit 2017 eine OWFa-1.0 Lizenz besitzt (Open Web Fundation).

Aus mehreren Webseiten konnte man rauslesen, dass GraphQL eine ernstzunehmende Alternative zu Rest sein kann, da man mit einem einzigen Quarry Zugriff auf alle genwünschten Daten zugreifen kann was bei der steigenden Komplexität der heutigen Web Applikationen von großem Vorteil sein kann.

Die GraphQL-Serverinstanz liefert bei einer Abfrage nur die exakt definierten Informationen damit nicht mehr Daten als benötigt über die Schnittstelle versendet werden.

Aus diesem Grund wird GraphQL auch als wesentlich präziser und effizienter als REST betitelt.

Was sich zuerst schön und gut anhört kann im Nachhinein zu Starken Sicherheitsproblemem führen da man bei Offene APIs das Abfrageverhalten nicht kontrollieren kann.

Deshalb kann es passieren, dass durch eine zu hohe Anzahl von Quarry der Server in die Knie geht.

Was auch noch zu erwähnen ist, ist dass die Implementierung eines Caching-Verfahrens für nicht veränderliche abfragen Komplexer als mit REST ist, da REST durch die Chaching-Methoder der http-Spezifikation zwischenspeichert.

Benutzte HTTP Statuscodes

|  |  |  |
| --- | --- | --- |
| Code | Bedeutung | Erläuterung |
| 200 | OK | Die Anfrage wurde erfolgreich verarbeitet, die Antwort enthält weitere Informationen |
| 201 | Created | Die Anfrage wurde erfolgreiche verarbeitet und als Ergebnis wurde eine neue Ressource angelegt, deren URI sich in einem Location-Header befindet |
| 204 | No content | Der Server liefert nur Metadaten, Keine Dateien |
| 404 | Not found | Der Server kennt keine Ressource mit dieser URI |
| 500 | Internal Server Error | Ein nicht näher spezifischer interner Fehler ist bei der Verarbeitung im Server aufgetreten. Dies ist der „harmloseste“ aller Fehlermeldungen – sie erlaubt dem Clienten keinerlei Rückschluss auf die Art des Fehlers. |

x1 <https://roy.gbiv.com/untangled/2008/rest-apis-must-be-hypertext-driven>

x2 <https://blog.restcase.com/4-maturity-levels-of-rest-api-design/>