Übung RESTful Webservices

am Beispiel von Bahmni OpenMRS

In der letzten Übung wurde Bahmni als OpenMRS Distribution vorgestellt. In dieser Übung betrachten wir die Schnittstellen, die die Kommunikation zwischen dem Bahmni-Web Frontend und dem OpenMRS Backend ermöglichen. Am praktischen Beispiel sollen Daten über die Schnittstellen ausgetauscht werden. Wir verwenden einen API-Client und ein 3LGM² Modell (MI-Lab, inkl. Teilmodell Bahmni).

Ein **Application Programming Interface** (API) ist ein Satz von Regeln, die es Anwendungen ermöglichen miteinander zu kommunizieren. Über Web-APIs können Anwendungen in einem Netzwerk miteinander kommunizieren. Die Hauptaufgabe einer Web-API besteht darin, Repräsentationen von Daten mit einer entsprechenden Client-Anwendung auszutauschen[1]. Serverseitig bestehen Web-APIs aus einem oder mehreren öffentlich zugänglichen Endpunkten für ein definiertes Anfrage-Antwort Protokoll[2].

Representational State Transfer (REST) APIs besitzen alle Merkmale einer Client-Server Architektur und nutzen zusätzlich einheitliche Schnittstellen[3].
Für die Übung relevant sind folgende Eigenschaften:

- Ressourcen – Client-Anfragen beziehen sich auf Ressourcen. Diese stehen für eine Informationseinheit, z.B. Person, Patient, Encounter.
- Adressierung – Ressourcen sind eindeutig adressierbar mit einen Uniform Resource Identifier (URI)
- Universelle Repräsentation – Ein Client kann eine Ressource z.B. im XML-, HTML- oder JSON-Format anfordern.
- Stateless Protokoll – Anfragen werden unabhängig voneinander behandelt.
- HTTP Methoden – REST verwendet einen einheitlichen Satz von HTTP Methoden, GET, POST, DELETE, u.A.

OpenMRS stellt über seine Plattform Backend Operationen bereit, einschließlich der APIs und der erforderlichen Speicher/Datenbanken, zur Unterstützung der OpenMRS-Anwendungen. Verschiedene Distributionen nutzen die OpenMRS Plattform als Backend (z. B. Bahmni, KenyaEMR, UgandaEMR, NigeriaMRS)[4] mit angepasstem Frontend. OpenMRS legt Datenobjekte (persons, patients, encounters, observations, …) in seiner Datenbank ab[5].
Die REST Web-API exponiert Datenobjekte aus OpenMRS als Ressourcen. Diese können über HTTP Methoden erstellt, bearbeitet oder gelöscht werden[6,7]. Bahmni Web agiert als Frontend und kommuniziert über die REST Web-API mit der OpenMRS Plattform als Backend.
Für die Übung stellen wir ein 3LGM² Modell mit detaillierter Darstellung von Bahmni zur Verfügung. Hier finden Sie zudem Informationen zu den Endpunkten (siehe Schnittstellen im Modell) und Ressourcen (siehe Services im Modell).

1 https://education.launchcode.org/csharp-web-dev-curriculum/web-api-rest/reading/web-api/index.html
2 https://en.wikipedia.org/wiki/Web_API#Server_side
3 https://de.wikipedia.org/wiki/Representational_State_Transfer
4 https://openmrs.atlassian.net/wiki/spaces/RES/pages/26273961/Platform+Team
5 OpenMRS Datenmodell: https://docs.openmrs.org/datamodel/
6 Dokumentation zur OpenMRS REST API: https://rest.openmrs.org/#openmrs-rest-api
7 https://demo.mybahmni.org/openmrs/module/webservices/rest/apiDocs.htm

1. Vorbereitung

1.1. Aufgabe – API-Client Bruno

Starten Sie den API-Client Bruno und machen Sie sich mit der Umgebung vertraut.
Download: https://github.com/usebruno/bruno/releases
Dokumentation: https://docs.usebruno.com/

Erstellen Sie eine neue Collection „MI-Lab“. Fahren Sie den Mauszeiger über die Collection, klicken Sie auf das Dreipunkt Menü (Meatball Menu) und im Menü auf Settings.

Bei jeder Anfrage müssen die Authentifizierungsdaten mitgesendet werden.
Tragen Sie im Reiter „Auth“ die Authentifizierungsdaten des OpenMRS Admin ein und bestätigen Sie mit „Save“:
Dropdownliste (Type): Basic Auth
User: superman
Password: Admin123

Tragen Sie im Reiter „Presets“ folgende Base URL ein und bestätigen Sie mit „Save“.
Base URL https://192.168.48.77/openmrs/ws/

Klicken Sie in der oberen Menüleiste (oben links) auf „Collection“ auf „Preferences“. Entfernen Sie dort im Reiter „General“ den Haken bei „SSL/TLS Certificate Verification“. Bestätigen Sie mit „Save“.

1.2. Aufgabe – Session

Legen Sie über das Dreipunkt Menü der Collection „MI-Lab“ eine neue Anfrage an→ „New Request“. Geben Sie ihr den Namen „1-2 Session“. Im Reiter Auth wählen Sie „Inherit“ aus, um die Zugangsdaten der Collection zu übernehmen.
Eine Ressource wird über die Base-URL des Webservice und den Pfad der Ressource angefragt.
Gehen Sie in den Reiter „Vars“. Fügen Sie zwei Variablen bei „Pre Request“ hinzu. Geben Sie einer Variable den Namen „endpoint-prefix“ mit dem Wert „rest/v1/“. Geben Sie der anderen Variable den Namen "resource" mit dem Wert "session". Verwenden Sie die Variablen in der Adresszeile in doppelt geschweiften Klammern, hier mit {{endpoint-prefix}} und {{resource}}.

Die Adresszeile sollte dann wie folgt aussehen:
GET https://192.168.48.77/openmrs/{{endpoint-prefix}}{{resource}}
stellvertretend für https://192.168.48.77/openmrs/ws/rest/v1/session

Diese Client-Anfrage gibt das Session-Token aus, über das der Benutzer „superman“ angemeldet ist. Senden Sie die Anfrage ab mit Klick auf den Pfeil „→“ rechts der Adresszeile.
War die Anfrage erfolgreich, wird als Antwort der Status „200 OK“ angegeben und die Payload im JSON Format angezeigt. Hier können Sie die aktuelle Session des angemeldeten Nutzers begutachten.
Lassen Sie den API-Client geöffnet und schauen Sie in das 3LGM² Modell!

1.3. Aufgabe - 3LGM² Modell MI-Lab

Öffnen Sie das 3LGM² Modell „MI-Lab_v6.z3lgm“. Gehen Sie im Modell Browser auf „02a - BAHMNI (detailed)“ oder gehen Sie alternativ in die logische Ebene halten Sie alt gedrückt und klicken Sie doppelt auf „EMR & Hospital System (Bahmni)“. In der logischen Ebene sehen Sie nun Anwendungssysteme, die in Bahmni-Standard[8] integriert sind und zugehörige Schnittstellen.

Gehen Sie in die fachliche Ebene. Hier sehen Sie eine Auswahl der in OpenMRS Bahmni verwendeten Objekttypen.[9]
a) Öffnen Sie den Objekttypen „Person“ und gehen Sie auf den Reiter „Repräsentationsformen“. Wodurch wird der Objekttyp „Person“ repräsentiert? Wodurch der Objekttyp „Patient“?

8 https://demo.mybahmni.org/
9 OpenMRS Datenmodell: https://docs.openmrs.org/datamodel/

Wir wollen uns die Ressourcen betrachten, die über die REST API kommuniziert werden.
b) Klicken Sie doppelt auf „REST person Ressource“ und gehen Sie anschließend auf den Reiter „Services“. Welche REST Services haben als Ergebnis die Ressource „REST person Ressource“?

c) Öffnen Sie einen dieser Services, z.B. „OpenMRS Create a person“. Schauen Sie sich im Reiter „REST Properties“ die entsprechenden Eigenschaften der Anfrage an.
Alle Services lassen sich auch in der logischen Werkzeugebene unter Services öffnen.

2. REST Webservice

2.1. Aufgabe – Schnittstellen und Services

Gehen Sie noch einmal in die fachliche Ebene. Starten Sie eine Analyse zum Objekttypen „Patient“ (Rechtsklick auf den Objekttypen) und schauen Sie zur Beantwortung der Fragen in der logische Ebene nach.

a) Wo wird er kommuniziert? Zwischen welchen Anwendungssystemen wird er kommuniziert?

b) Welche Bausteinschnittstellen können ihn kommunizieren? Was für Schnittstellen stellen die Webservices von OpenMRS bereit? Welche Anwendungssysteme rufen das REST Interface (JSON) auf?

2.2. Aufgabe – REST Anfragen, Personen suchen

Das Bahmni Web Frontend nutzt die REST API von OpenMRS u.A. um Patienten zu suchen oder neu anzulegen. In dieser Aufgabe soll eine eine Person in OpenMRS gesucht werden. Mithilfe des 3LGM² Modells sollen Sie REST Anfragen mit dem API-Client Bruno verschicken.

a) Suchen Sie sich unter Services den REST Service „OpenMRS Search person by name“ heraus. Lesen Sie die notwendigen Attribute für die Anfrage aus dem Modell ab. Öffnen Sie den API-Client wieder. Klonen Sie Ihre letzte Anfrage. Dafür klicken Sie im Dreipunkt-Menü von „1-2 Session“ auf Clone. Geben Sie der neuen Anfrage den Namen „2-2 Search person“. Ändern Sie in der neuen Anfrage im Reiter „Vars“ die angefragte Ressource auf „person“. Im Reiter „Params“ spezifizieren Sie die Anfrage. Klicken sie auf „Add Param“, geben Sie dem Parameter den Namen „q“ mit einem beliebigen Wert für den Namen der gesuchten Person.
Die Adresszeile sollte beispielsweise so aussehen:
GET https://192.168.48.77/openmrs/{{endpoint-prefix}}{{resource}}?q=Martha
Senden Sie die Anfrage. Bei erfolgreicher Anfrage, wird der Status „200 OK“ angegeben und gefundene Personen in der Payload aufgelistet.

b) Als URI verwendet openMRS den Universally Unique Identifier (UUID). Der UUID macht die Ressource vom Typ „person“ eindeutig adressierbar. Welchen UUID hat Ihre gesuchte Person? Kopieren Sie den UUID in die Zwischenablage (Beispiel "uuid": "b877bfd4-9fc0-4b7c-8b25-1b64ef099041").

c) Suchen Sie im Modell den Service „OpenMRS Retrieve person by uuid“. Klonen Sie die letzte Anfrage und nennen Sie die neue „2-2 Retrieve person by uuid“. Deaktivieren Sie die Query „q“ im Reiter „Params“. Tragen Sie im Reiter „Vars“ hinter „person“ ein Slash „/“ und den kopierten UUID ein.

2.3. Aufgabe – Person und Patient erstellen

In dieser Aufgabe soll eine Person und Patient erstellt werden. Die erstellte Person soll dabei eine Sub-Ressource von Patient werden. Nutzen Sie das 3LGM² Modell und den API-Client zur Bearbeitung der Aufgabe.

a) Klonen Sie im API-Client die letzte Anfrage und nennen Sie die neue „2-3 Create a person“. Suchen Sie im Modell unter Services den REST Service „OpenMRS Create a person“ heraus. Die notwendigen Attribute für die Anfrage entnehmen Sie dem Modell (Hinweis: Body(JSON) wird nicht komplett angezeigt). Tragen Sie unter dem Reiter „Body“ die Attribute der Person im Format JSON ein. Ändern Sie die Werte für „names“, „gender“, „birthdate“ und „addresses“ nach Belieben. Ändern Sie unter „Vars“ den Pfad auf „person“. Zum Erstellen der Ressource wird die HTTP Methode „POST“ verwendet. Wählen sie diese in der Dropdownliste links der Addresszeile aus. Senden Sie die Anfrage ab.

Wenn Status „201 Created“ rückgemeldet wird, war die Erstellung erfolgreich. Die neu angelegte Ressource wird in der Payload zusammen mit neu erstelltem UUID angezeigt. Notieren Sie sich den UUID.

b) Klonen Sie im API-Client die letzte Anfrage und nennen Sie die neue „2-3 Create a patient“.
c) Um einen Patienten zu erstellen, wird die UUID einer Person (siehe a)) und eine noch nicht verwendete Patienten-ID (patient.identifiers.identifier) benötigt. Diese Informationen werden mitttels POST in einem Nachrichten-Body übermittelt.
Ein Template für den Nachrichten-Body finden Sie im Modell unter Services den REST Service „OpenMRS Create a patient“. Gehen Sie in den Reiter „REST Properties in das Feld „Body (JSON)“. Kopieren Sie das JSON-Template in Bruno in den Reiter „Body“ (ändern Sie ggf. den Typ des Body von „No Body“ auf „JSON“. Fügen Sie nun die in a) notierte UUID der „person“ ein.
Weisen sie dem Patienten einen zufälligen Identifier zu, beginnend mit ABC21xxxx. Denken Sie daran die HTTP Methode „POST“ auszuwählen. Senden Sie die Anfrage.
Wenn der Status 400 „Bad Request“ ist, schauen Sie unter error.globalErrors nach der Fehlermeldung. Ist der von Ihnen erzeugte Identifier bereits in Verwendung, denken Sie sich einen neuen aus und wiederholen Sie die Anfrage.
Wenn Status 201 Created rückgemeldet wird, war die Erstellung der Ressource erfolgreich.

Wenn alles funktioniert hat, kann man den neu erstellten Patienten über die Bahmni Web Oberfläche aufrufen: https://192.168.48.77/bahmni/home/ → Anmelden → Clinical → All → Namen oder Identifier eingeben → Search.

2.4. Aufgabe – Personendaten anzeigen, ändern und löschen

Zeigen Sie Personendaten an, ändern und löschen sie diese. Nutzen Sie das 3LGM² Modell zur Bearbeitung der Aufgabe.

a) Personendaten können als Subressourcen angefragt werden. Hierfür sendet man den UUID der Ressource in der Anfrage und die gesuchte Subressource. Suchen Sie im Modell unter Services den REST Service „OpenMRS Retrieve person name subresource by uuid“ heraus.
Klonen Sie die Anfrage „2-2 Retrieve person by uuid“ und nennen Sie diese „2-4 Retrieve person name subresource“. Gehen Sie in den Reiter „Vars“. Verändern Sie die Variable „ressource“ indem Sie die vorhandene UUID durch die UUID der gesuchten Person ersetzen. Nehmen Sie die UUID der von Ihnen neu erstellten Person aus 2.3 a). Ergänzen Sie anschließen dahinter „/name“, um die Subressource „name“ anzufragen. Senden Sie die Anfrage ab.

b) Personendaten können direkt in der Ressource geändert werden. Klonen Sie die letzte Anfrage und nennen Sie diese „2-4 Update person Attributes“. Gehen Sie in den Reiter „Vars“ und entfernen Sie „/name“ der Variable „resource“.
Suchen Sie im Model den entsprechenden Service. Entnehmen Sie dem Modell die notwendigen Attribute (Hinweis: Body(JSON) wird nicht komplett angezeigt). Tragen Sie unter dem Reiter „Body“ die Attribute der Person im Format JSON ein. Tragen Sie beliebige neue Werte für Namen, Geschlecht und Geburtsdatum ein.
Nutzen Sie die HTTP Methode „POST“ und senden Sie die Anfrage ab.

c) Lassen Sie sich nochmal die Subressource „/name“ ausgeben. Nutzen Sie unter dem Reiter „Params“ die Query „v“ mit dem Wert „full“.
   - Was hat sich geändert?
   - Wie viele Namen hat der Patient? Schauen Sie sich das Attribut „names“ an.

   Schauen Sie sich den Patienten in BAHMNI an und prüfen Sie, ob die neuen Werte vorhanden sind.

d) Hat der Patient aus c) mehr als einen Namen, löschen Sie den letzten Namen. Klonen Sie die Anfrage „2-4 Update person Attributes“ und nennen Sie diese „2-4 Delete person name subresource“. Suchen Sie im Modell den entsprechenden Service. Hinweise: Für die Anfrage werden der UUID der Ressource „person“ und der UUID der Subressource „name“ benötigt. Zur Löschung eines Eintrags wird die HTTP Methode „DELETE“ verwendet. Um die Subressource dauerhaft zu löschen nutzen Sie unter „Params“

die Query „purge“ mit dem Wert „true“.
Erhalten Sie den HTTP Status Code 204, war die Anfrage erfolgreich.

3. Atom Feed

Um andere Anwendungssysteme zu integrieren müssen diese über Änderungen in OpenMRS informiert werden. Atom Feed ermöglicht es Ereignisse, wie die Erstellung und Bearbeitung von Webressourcen, zu veröffentlichen. Bei einem Ereignis wird über einen Publisher ein Ereignisfeed (Patienten-, Encounter-, Labor- und Medikamenten-Feed) veröffentlicht. Andere Anwendungssysteme abonnieren den Feed, um über Ereignisse benachrichtigt zu werden. Im Feed ist die Adresse (URL) der entsprechenden Ressource enthalten. Die Systeme, die den Feed abonniert haben, können relevante Daten über die REST Schnittstelle anfragen und ihrer Datenbank hinzufügen. [10]

3.1. Aufgabe – Patientenliste über Atom Feed

Nutzen Sie das Modell um die Client Anfragen zu ermitteln. Lassen Sie sich die neuesten Patienten des openMRS Patient Feeds anzeigen. Klonen Sie die erste Anfrage „… Session“ und nennen Sie diese „3-1 Atom Feed Patient“. Geben Sie unter „Vars“ dem „endpoint-prefix“ den neuen Wert „atomfeed/“ und „resource“ den Wert „/patient/recent“. Senden Sie eine „GET“ Anfrage.

- Was ist in den Entries enthalten?
- Wo liegt die URL zur entsprechenden REST-Ressource?
- Rufen Sie die REST-Ressource über die URL ab. Erstellen Sie dafür eine neue Anfrage und kopieren Sie die URL in die Adresszeile.

3.2. Aufgabe – OPD Visit (Konsultation) einfügen

Schauen Sie sich Ihren Patienten auf der Bahmni Weboberfläche im Browser an und fügen Sie beliebige Notizen zur Konsultation ein.
Gehen Sie auf die Bahmni Hauptseite: https://192.168.48.77, klicken Sie auf „Clinical Service“ und loggen Sie sich, wenn notwendig, ein.

User:	superman
Password:	Admin123

Gehen Sie auf Registration und suchen sie Ihren Patienten. Öffnen Sie den Patienten und klicken Sie auf „Start OPD Visit“. Der Patient ist nun für die ärztliche Konsultation aktiv. Gehen Sie über das Home Symbol zurück zum Bahmni Dashboard. Dann gehen Sie auf „Clinical“, wählen Ihren Patienten aus, gehen auf „Consultation“ und auf „Consultation Notes“. Tragen Sie eine Arztnotiz ein und schließen Sie ab mit Save.

3.3. Aufgabe – OPD Visit abrufen,

Nutzen Sie das Modell um die Client Anfragen zu ermitteln. Lassen Sie sich die neuesten Encounter des openMRS Encounter Feeds anzeigen. Klonen Sie die letzte Anfrage und nennen Sie diese „3-3 Atom Feed Encounter“.

a) Fragen Sie den letzten Kontakt (Encounter) über Atom Feed an. Ein Eintrag zur ambulanten Konsultation (OPD Visit) Ihres Patienten sollte darin enthalten sein. Kann man Ihren Encounter anhand des Feeds finden? Wenn ja, wie?
b) Fragen Sie über die bereitgestellten URLs Ressourcen ab.

10 https://bahmni.atlassian.net/wiki/spaces/BAH/pages/3506200/Atom+Feed+Based+Synchronization+in+Bahmni

4. FHIR Webservice

FHIR (Fast Healthcare Interoperability Resources) ist ein HL7-Standard um Gesundheitsinformationen elektronisch zu repräsentieren. OpenMRS implementiert seit FHIR, um eine bessere Interoperabilität zwischen Gesundheits-Informationssystemen zu gewährleisten.[11]

4.1. Aufgabe – FHIR Anfragen

Der FHIR Webservice von OpenMRS unterstützt die „GET“ Methode.[12] [13] Hiermit lassen sich Ressourcen vom RESTful Server suchen, abfragen und auflisten. Klonen Sie die letzte Anfrage und nennen Sie diese „4-1 FHIR Search Patient“. Geben Sie unter „Vars“ dem „endpoint-prefix“ den neuen Wert „fhir2/R4/“.

Nutzen Sie erneut das Modell um die Aufgaben zu lösen. Suchen Sie nach FHIR Services.

a) Lassen Sie sich eine Liste der Patienten ausgeben.
b) Suchen Sie alle Patienten mit dem Namen „Bilbo“. Nutzen Sie den Parameter „?name=Bilbo“.
   - Wie viele finden Sie?
c) Suchen Sie alle Ressourcen vom Typ „Observation“ mit dem Patienten „Bilbo“.
   - Wie lautet die Anfrage? Wie viele Treffer gibt es?

11 https://fhir.openmrs.org/
12 Unterstützte Suchanfragen: https://openmrs.atlassian.net/wiki/spaces/projects/pages/26938492/ARCHIVED+FHIR+Support+in+FHIR2+Module+version+1.3.0
13 Unterstützte Suchparameter: https://fhir.openmrs.org/artifacts.html

5. Lösungen

5.1. Lösung 2.2 Aufgabe – REST Anfragen, Personen suchen

GET https://192.168.48.77/openmrs/ws/rest/v1/person?q=Martha
GET https://192.168.48.77/openmrs/ws/rest/v1/person/{person_uuid}
POST https://192.168.48.77/openmrs/ws/rest/v1/person
Body: JSON

```
{
   "names": [
      {
      "givenName": "Test",
      "familyName": "Person"
      }
   ],
   "gender": "M",
   "birthdate": "1990-09-02",
   "addresses": [
      {
      "address1": "Musterstrasse 1",
      "cityVillage": "Halle",
      "country": "Germany",
      "postalCode": "06112"
      }
   ]
}
```

5.2. Lösung 2.3 Aufgabe – Person und Patient erstellen

GET https://192.168.48.77/openmrs/ws/rest/v1/patient?q=Martha
GET https://192.168.48.77/openmrs/ws/rest/v1/patientidentifiertype
POST https://192.168.48.77/openmrs/ws/rest/v1/patient
Body: JSON

```
{
   "person": "{person_uuid}",
   "identifiers": [
      {
         "identifier": "ABC200001",
         "identifierType": "b9a9e100-f496-11ed-b02c-0242ac150003",
         "preferred": false
      }
   ]
}
```

5.3. Lösung 2.4 Aufgabe – Personendaten anzeigen, ändern und löschen

GET https://192.168.48.77/openmrs/ws/rest/v1/person/{person_uuid}/name
DELETE https://192.168.48.77/openmrs/ws/rest/v1/person/{person_uuid}/name/{name_uuid}?purge=true

5.4. Lösung 3.1 Aufgabe – Patientenliste über Atom Feed

In den Entries des Open MRS Feed Publishers ist u.a. der Name der Ressource, die Atom Feed ID, das Veröffentlichungsdatum und der Content enthalten:

```
<content type="application/vnd.atomfeed+xml">
<![CDATA[/openmrs/ws/rest/v1/patient/80bb5371-1173-486d-b969-30c7815bcc74?v=full]]>
</content>
```

GET https://192.168.48.77/openmrs/ws/rest/v1/patient/{patient_uuid}?v=full

Im Feed ist der Name oder der UUID eines Patienten nicht hinterlegt. Man kann die Kontakte eigentlich nur anhand des Zeitstempels finden.