Service Engineering

Übung 2a – Spezifikation und Nutzung von Web-APIs (Services)

Aufgabenstellung

Ziele der Übung

- Spezifikation eines eigenen Serviceangebots Web-API´s
- Test eines vorgegebenen Web-API-Clients
- Anpassung des Web-API-Clients auf eigene Zugriffsszenarien
- Recherche und Verwendung von HTTP-Monitoren (API-Gateway)

Aufgabenstellungen - 1

- Erzeugen einer WSDL- oder Swagger-Spezifikation
  - Identifikation der Ziele einer entsprechenden Spezifikation
  - Kriterienbasierte Auswahl eines Werkzeugs zur WSDL/Swagger-Erstellung
    - Analyse verfügbarer kommerzieller Produkte
    - Analyse verfügbarer OpenSource-Produkte
  - Erstellen einer WSDL/Swagger-Spezifikation
    (Beispiel mit mind. 4 Operationen/WSDL bzw. 4 Objekte/Swagger)
- Vergleich der Spezifikation mit WSDL und Swagger (OpenAPI)
  - Beschreibung von Vor- und Nachteilen (Kriterienbasiert)
  - Fehlende Elemente zur Servicebeschreibung?

Aufgabenstellungen - 2

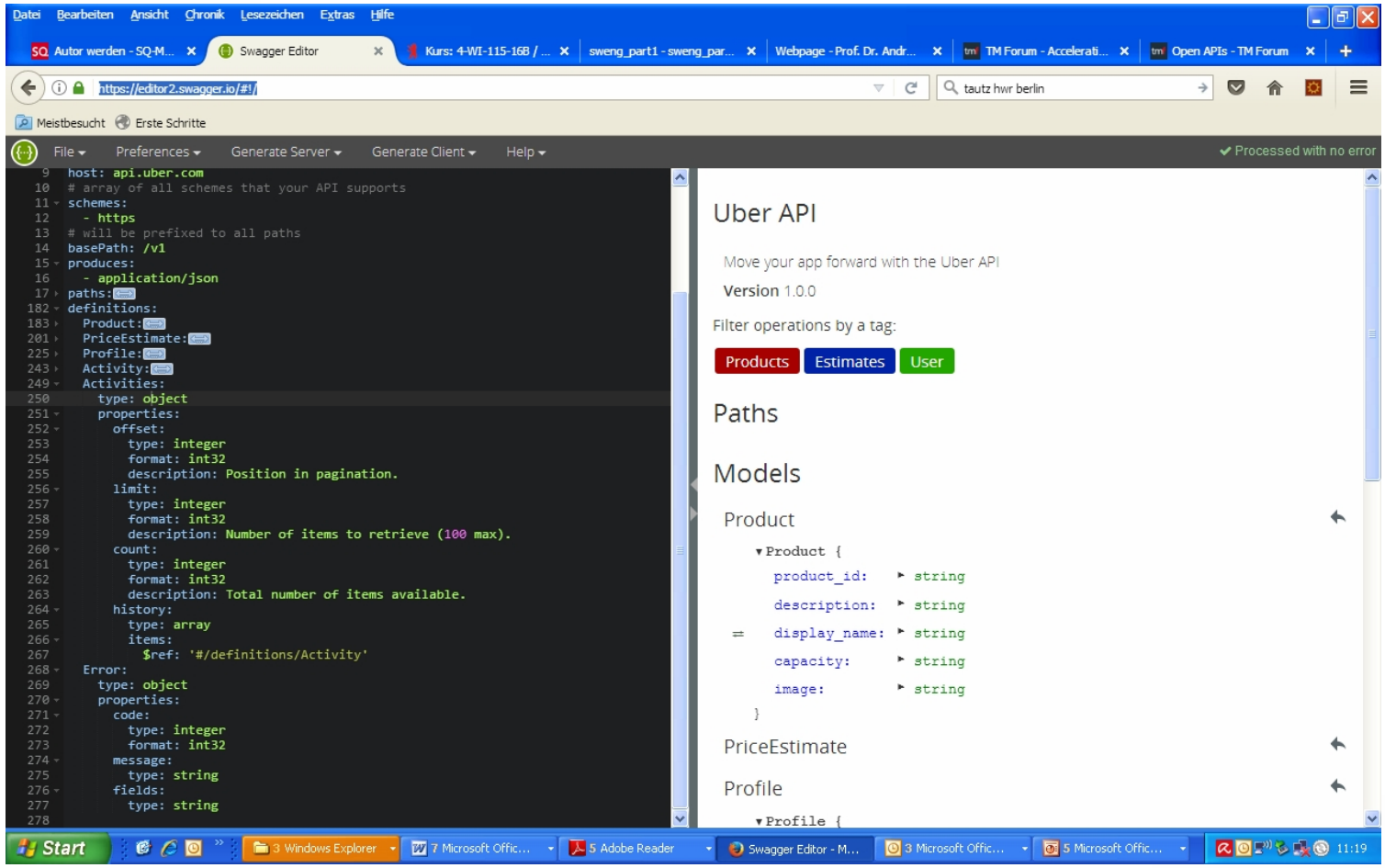
- Testen Sie die vorgegebenen Java-Clients (SOAP und REST)
- Entwickeln Sie einen Java-Client der das SOAP- oder JSON-Protokoll direkt nutzt und auf Funktionen selbst gewählter Web-Services zugreift. → ggf. Java-Bibliotheken verwenden!
  - Auswahl von 2 Web Services (mind. je 2 Operationen testen)
  - SOAP/JSON-Schnittstelle des Serviceangebots muss verfügbar sein!!
  - Anpassung des vorgegeben Lösungsmusters oder eigene Lösung
- Identifikation potentieller Problembereiche

Aufgabenstellungen - 3

- Recherchieren Sie nach Möglichkeiten zur SOAP/XML und JSON-Überwachung und geben Sie verschiedene Lösungsalternativen an.
- Verwendung des ausgewählten Monitors (z.B. Membrane)
  - Analyse des bereitgestellten Funktionsumfangs
  - Konfiguration für ein eigenes Beispiel
  - Protokollierung der ein- und ausgehenden Nachrichten
  - Bewertung des Einsatzpotentials dieses Werkzeugs

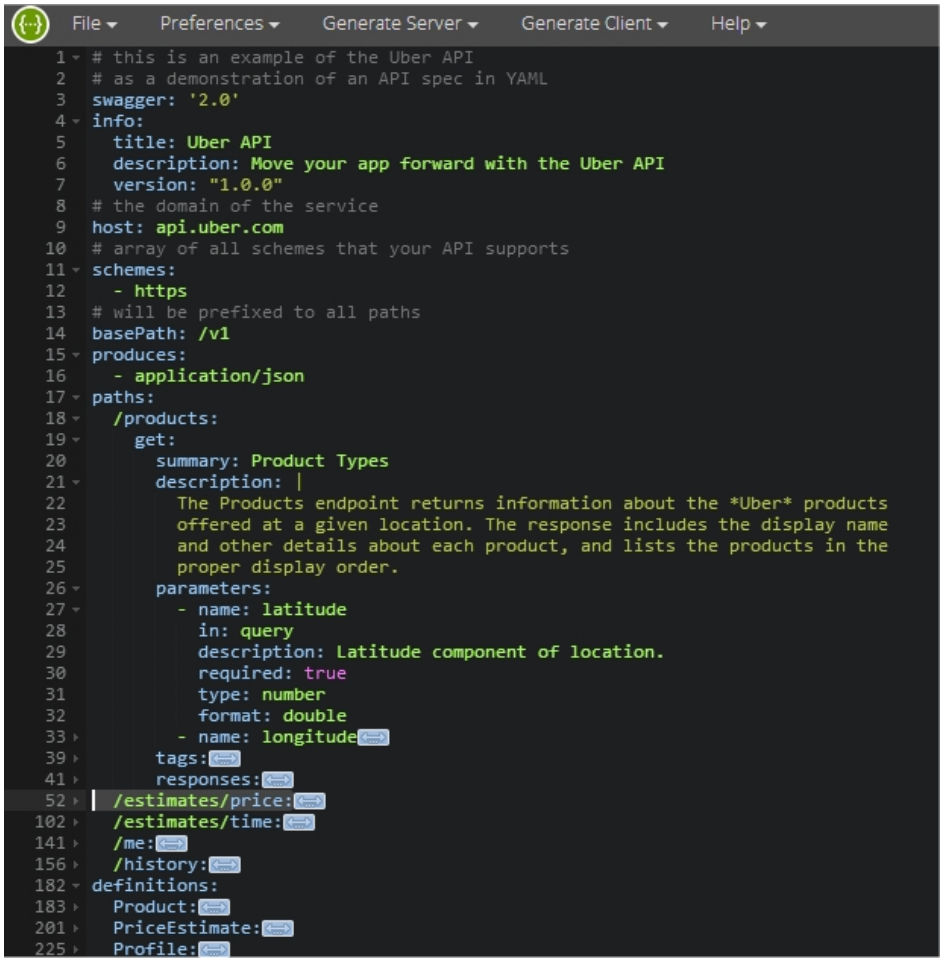
Beispiel Swagger-Editor

Swagger-Editor (Cloud-Lösung)



Quelle: https://editor2.swagger.io/#!/

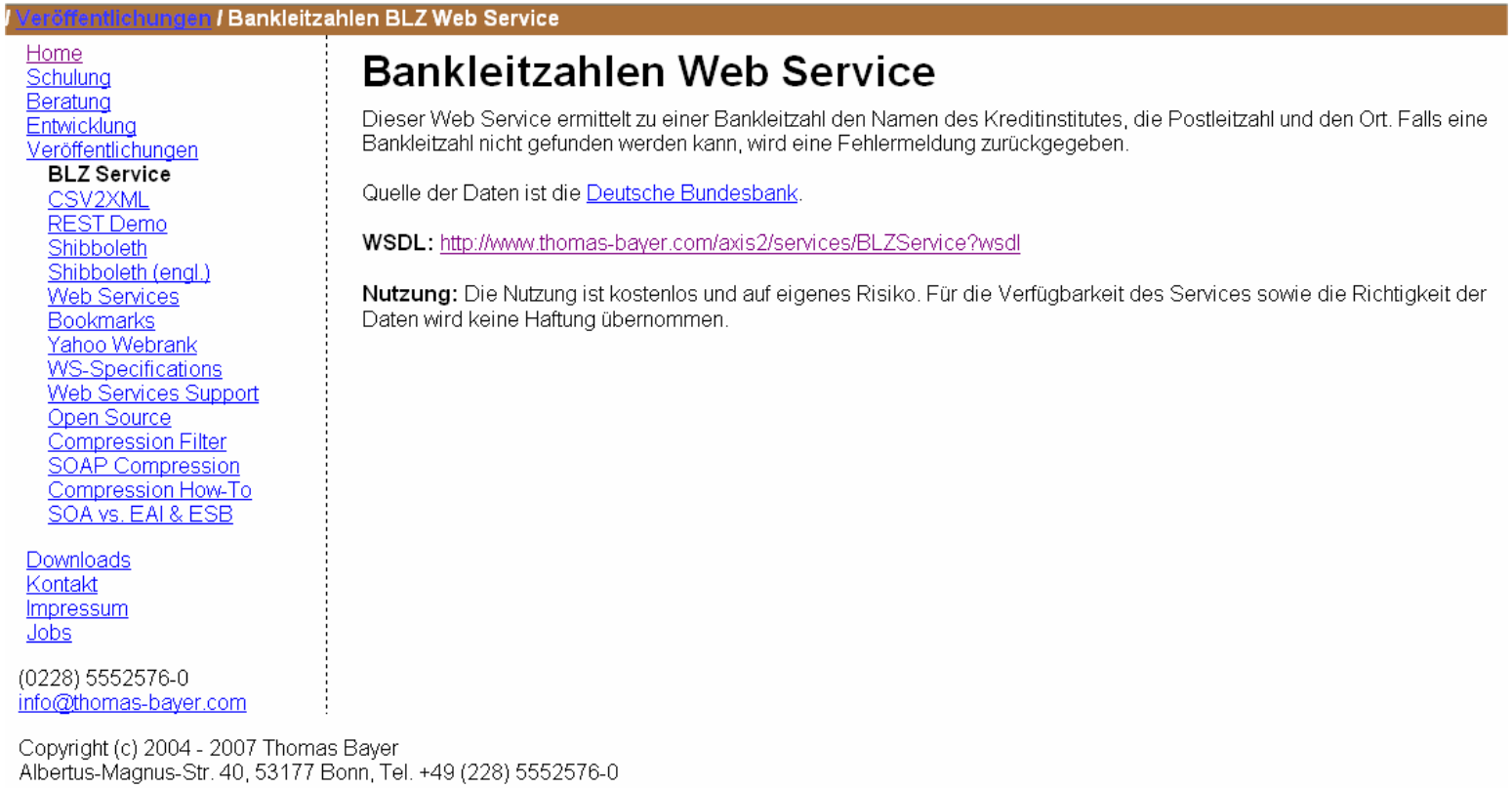
Swagger-Editor (hier Uber-API)



Quelle: https://editor2.swagger.io/#!/

Beispiel eines SOAP-Clients

Serviceangebot im Internet



/ Veröffentlichungen / Bankleitzahlen BLZ Web Service

Home
Schulung
Beratung
Entwicklung
Veröffentlichungen
BLZ Service
CSV2XML
REST Demo
Shibboleth
Shibboleth (engl.)
Web Services
Bookmarks
Yahoo Webrank
WS-Specifications
Web Services Support
Open Source
Compression Filter
SOAP Compression
Compression How-To
SOA vs. EAI & ESB

Downloads
Kontakt
Impressum
Jobs

(0228) 5552576-0
info@thomas-bayer.com

Bankleitzahlen Web Service

Dieser Web Service ermittelt zu einer Bankleitzahl den Namen des Kreditinstitutes, die Postleitzahl und den Ort. Falls eine Bankleitzahl nicht gefunden werden kann, wird eine Fehlermeldung zurückgegeben.

Quelle der Daten ist die Deutsche Bundesbank.

WSDL: http://www.thomas-bayer.com/axis2/services/BLZService?wsdl

Nutzung: Die Nutzung ist kostenlos und auf eigenes Risiko. Für die Verfügbarkeit des Services sowie die Richtigkeit der Daten wird keine Haftung übernommen.

Quelle des Web Service: Bayer, T.: http://www.thomas-bayer.com/soap/blz-web-service.htm

Beispiel eines SOAP-Clients - 1

```
import java.io.*; import java.net.*;

public class TestClient {

    private static final int BUFFER = 50;
    public static void main(String[] argv) throws Exception{
    String request =
        "<?xml version='1.0' encoding='UTF-8'?>"+
        "<soapenv:Envelope " +
        "xmlns:soapenv='http://schemas.xmlsoap.org/soap/envelope/' " +
        "xmlns:xsd='http://www.w3.org/2001/XMLSchema' " +
        "xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'> " +
        "<soapenv:Body> "+
        "<tns:getBank xmlns:tns='http://thomas-bayer.com/blz/'>"
        + "<tns:blz>10090000</tns:blz></tns:getBank>"
        + "</soapenv:Body>"+
         "</soapenv:Envelope>";
```

Beispiel eines SOAP-Clients - 2

```
//   Hier muss die URL des WS-Angebots eingetragen werden
     URL url = new URL("http://www.thomas-
         bayer.com/axis2/services/BLZService?wsdl");
//    Verbindungobjekt erstellen und konfigurieren
     HttpURLConnection con = (HttpURLConnection) url.openConnection();
     con.setDoOutput(true);
     con.setUseCaches(false);
     con.setRequestProperty("Accept", "text/xml");
     con.setRequestProperty("Connection", "keep-alive");
     con.setRequestProperty("Content-Type", "text/xml");
     con.setRequestProperty(
     "Content-length",
     Integer.toString(request.length()));
     con.setRequestProperty("SOAPAction", "\" \"");
```

Beispiel eines SOAP-Clients - 3

```
//   Vorbereitung und Ausgabe der XML-Datei zum Web Service
     OutputStream out = con.getOutputStream();
     out.write(request.getBytes()); out.flush();
//    Verarbeitung und Ausgabe der Antwort XML
     StringBuffer response = new StringBuffer(BUFFER);
     InputStreamReader input = new InputStreamReader(con.getInputStream(),
         "UTF-8");
     char buff[] = new char[BUFFER];
     int n;
     while (( n = input.read(buff, 0, BUFFER - 1)) > 0) {
         response.append(buff, 0, n);
     }
     out.close(); input.close();
     System.out.println( response.toString() );
     }
}
```

Ergebnisausgabe

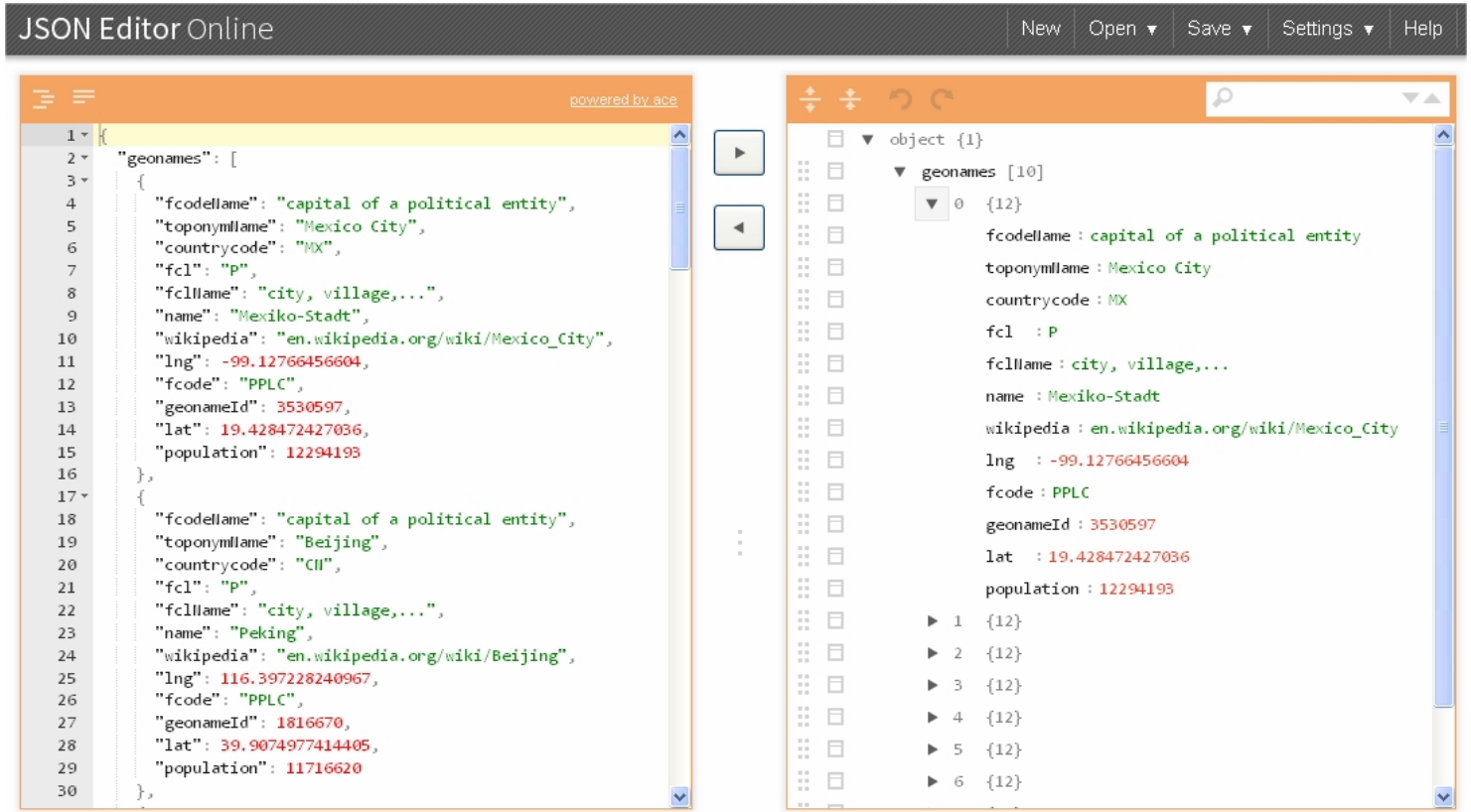
```
<?xml version='1.0' encoding='UTF-8'?>

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">

<soapenv:Body>

    <ns1:getBankResponse xmlns:ns1="http://thomas-bayer.com/blz/">

    <ns1:details>

    <ns1:bezeichnung>Berliner Volksbank</ns1:bezeichnung>

    <ns1:bic>BEVODEBBXXX</ns1:bic><ns1:ort>Berlin</ns1:ort>

    <ns1:plz>10892</ns1:plz></ns1:details></ns1:getBankResponse>

</soapenv:Body>

</soapenv:Envelope>
```

Beispiel eines REST-Clients

Beispiel eines REST-Clients

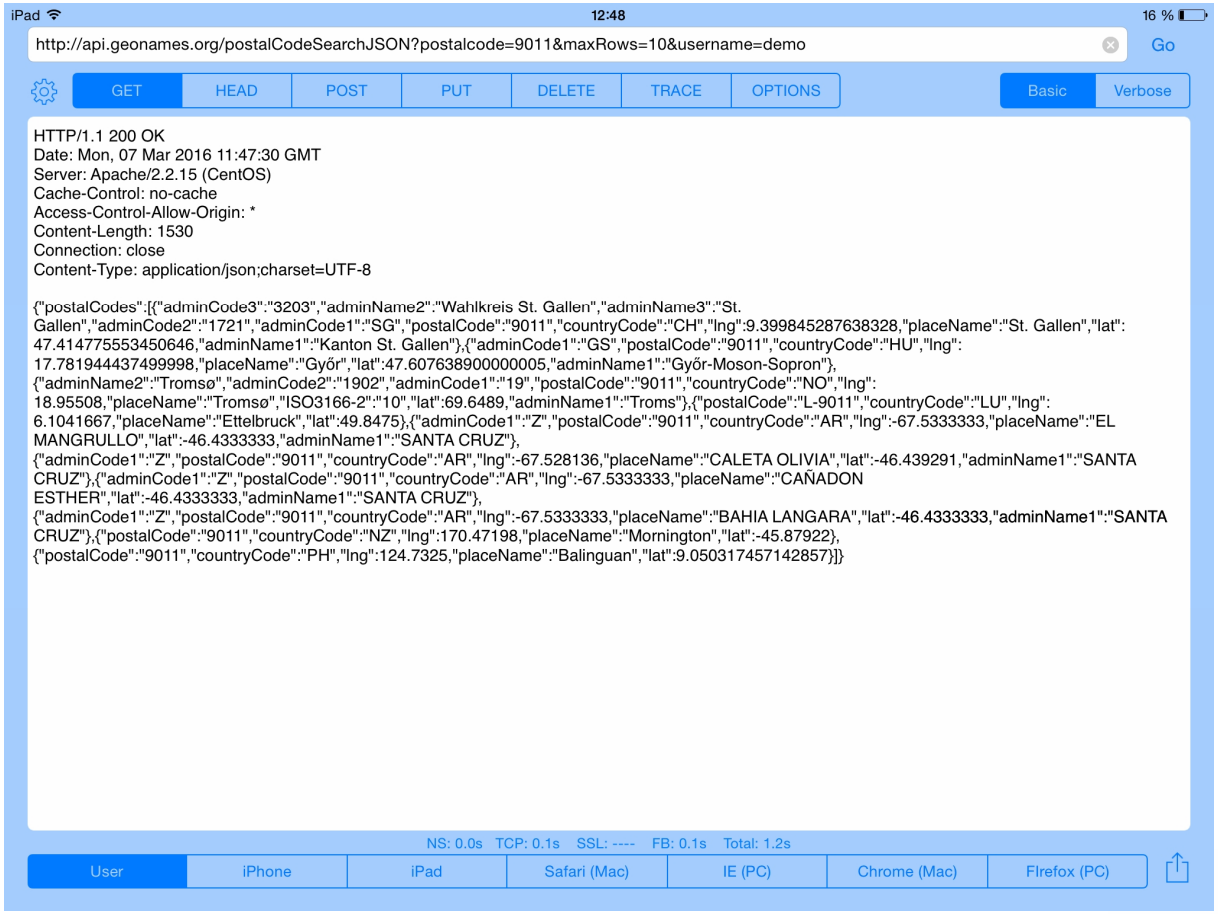
```
import java.io.*;
import java.net.*;

public class RestClient {

    public static void main(String[] args) throws IOException {

        //Adresse der Web-API konfigurieren - JSON
        String urlT1 ="http://api.geonames.org/citiesJSON?north=44.1&south=-9.9&east=-22.4&west=55.2&lang=de&username=demo";
        //Adresse der Web-API konfigurieren - XML
        String urlT2 ="http://api.geonames.org/cities?north=44.1&south=-9.9&east=-22.4&west=55.2&username=demo";
        URL url = new URL(urlT2);
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        conn.setRequestMethod("GET");
        conn.setRequestProperty("accept", "application/json");
        conn.setRequestProperty("accept", "text/xml");

        if (conn.getResponseCode() != 200) {
            throw new RuntimeException("HTTP-Fehler : "
                    + conn.getResponseCode());
        }

        BufferedReader br = new BufferedReader(new InputStreamReader(
                (conn.getInputStream())));

        String ausgabe;
        System.out.println("Ausgabe des Servers ...\n");
        while ((ausgabe = br.readLine()) != null) {
                System.out.println(ausgabe);
        }

        conn.disconnect();
```

JSON Editor Online



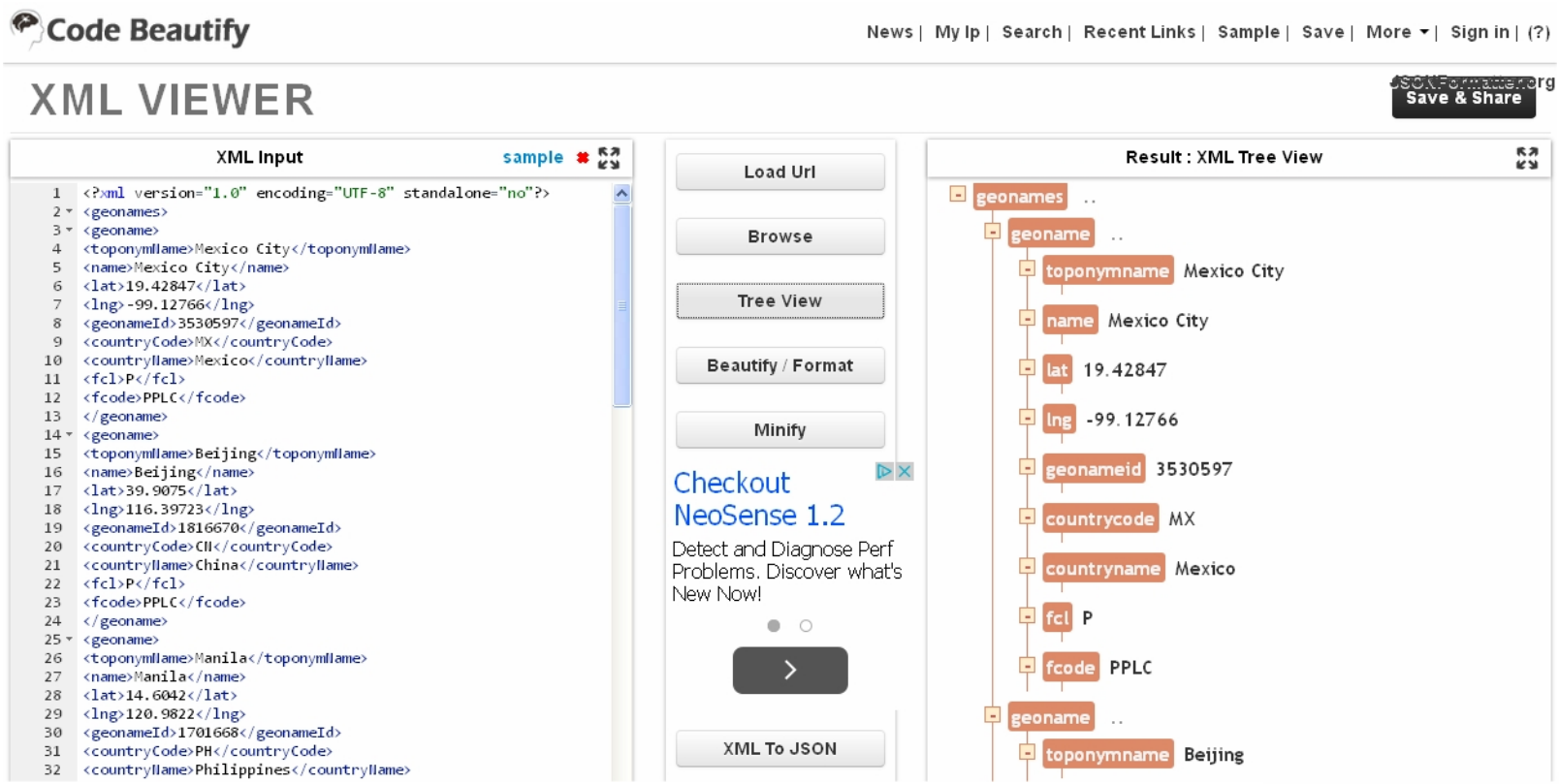
Quelle: http://www.jsoneditoronline.org/ (letzte Verwendung: März 2016)

iCurl – HTTP-Operationen



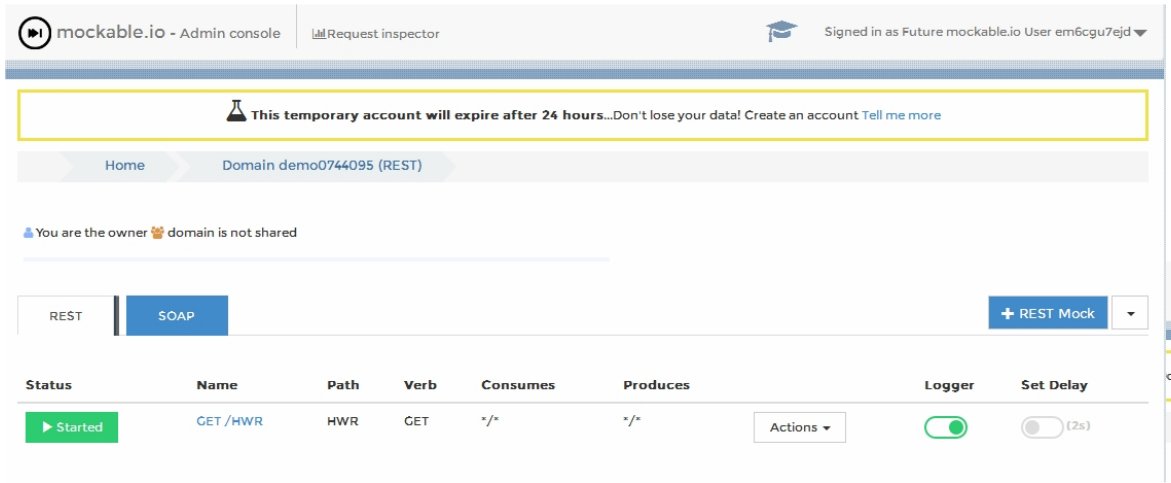
Quelle: https://itunes.apple.com/de/app/icurlhttp/id611943891?mt=8

XML Editor Online

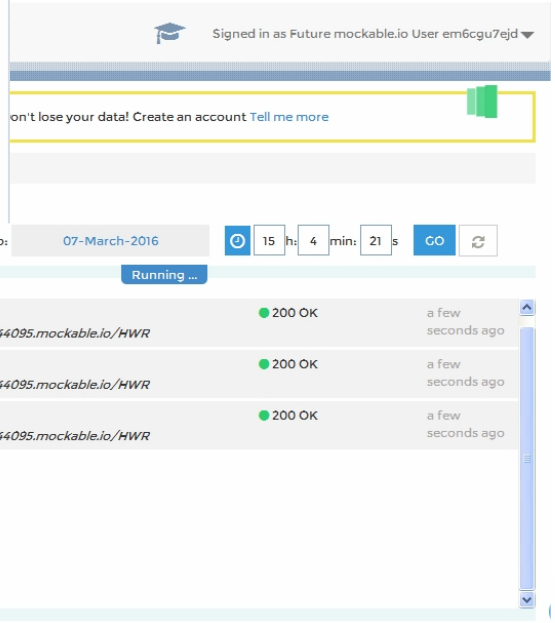


Quelle: http://codebeautify.org/xmlviewer (letzte Verwendung: März 2016)

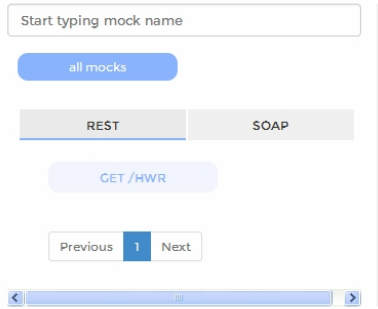
Einsatz Mock-Service (REST/XML)



Mock-Service einrichten



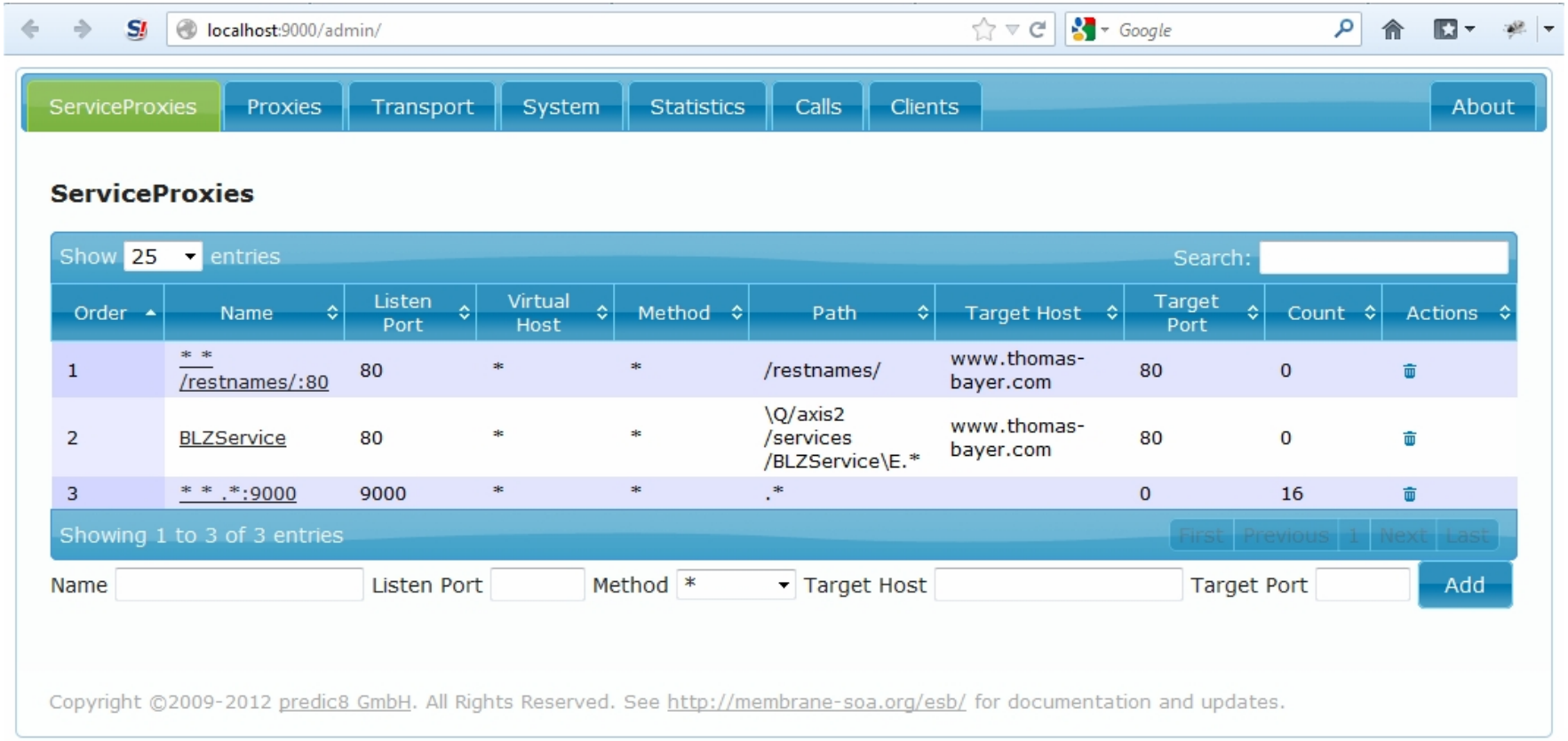
Requests überwachen



Quelle: https://www.mockable.io (letzte Verwendung: März 2016)

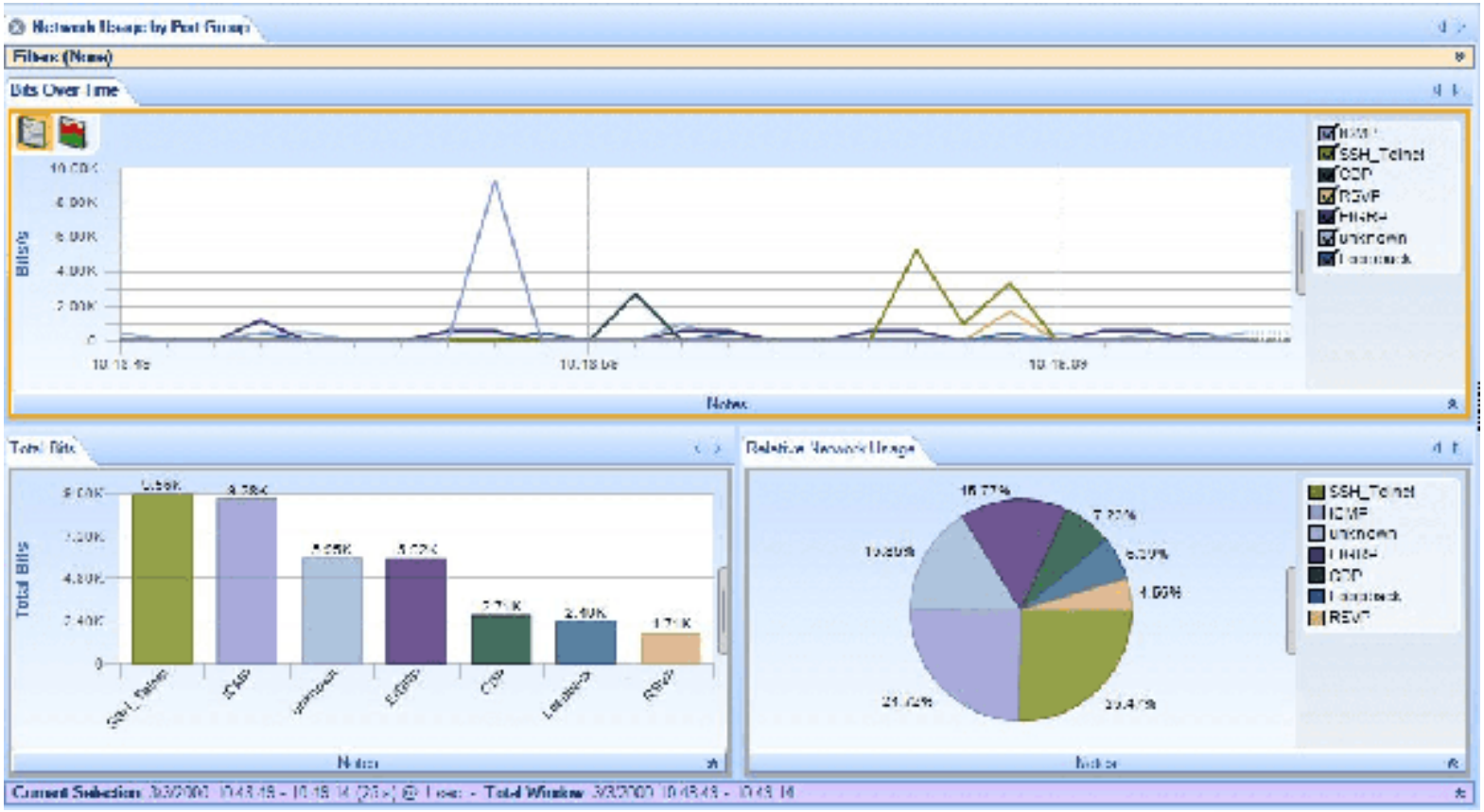
HTTP-Monitoring

HTTP-Monitoring



Quelle: https://www.membrane-soa.org/

Netzwerkmonitoring



Quelle: www.wireshark.org

Organisation der Übung

Organisation der Übung

Bitte berücksichtigen Sie die folgende Vorgehensweise:

- Die Übung wird in max. 4er Gruppen durchgeführt!
- Vorgeschlagene Methode zur Aufgabenlösung
  - Abstimmung einer geeigneten Vorgehensweise (20 min)
  - Ausführen der Aufgabenstellungen (150 min)
  - Erstellung eines Protokolls (30 min)
- Gesamtzeit für die Durchführung: ca. 200 min
- Bereitstellung eines entsprechenden Protokolls je Gruppe
- Abgabe des Protokolls am Ende des Semester

Aufbau des Protokolls

Verwenden Sie bitte des folgende Muster für das Protokoll:

- Allgemeine Angaben
  - Versuch, Beteiligte Studenten, Datum
  - Rahmenbedingungen (Software, …)
  - Methodisches Vorgehen zu Bearbeitung
- Aufgaben des Laborversuchs
  - Aufgabenstellung
  - Textliche Ausführungen zu den Lösungen
  - Verwendung von Grafiken und Screenshots
- Zusammenfassung (Bewertung der erreichten Ergebnisse)
- Genutzte Quellen (z.B. Literatur, Internet, …)