Service Engineering

Spezifikation, Implementierung und Management von Web-APIs

Die Inhalte der Vorlesung wurden primär auf Basis der angegebenen Literatur erstellt.

Darüber hinaus finden sich vielfältige Beispiele aus dem industriellen Umfeld.

Agenda

- Begriff der API-Economy
- Ansätze zur Spezifikation (hier: WSDL und OpenAPI)
- Möglichkeiten zur Entwicklung
- Aspekte des API-Management

Begriff der API-economy

Möglichkeit einer generischen Definition

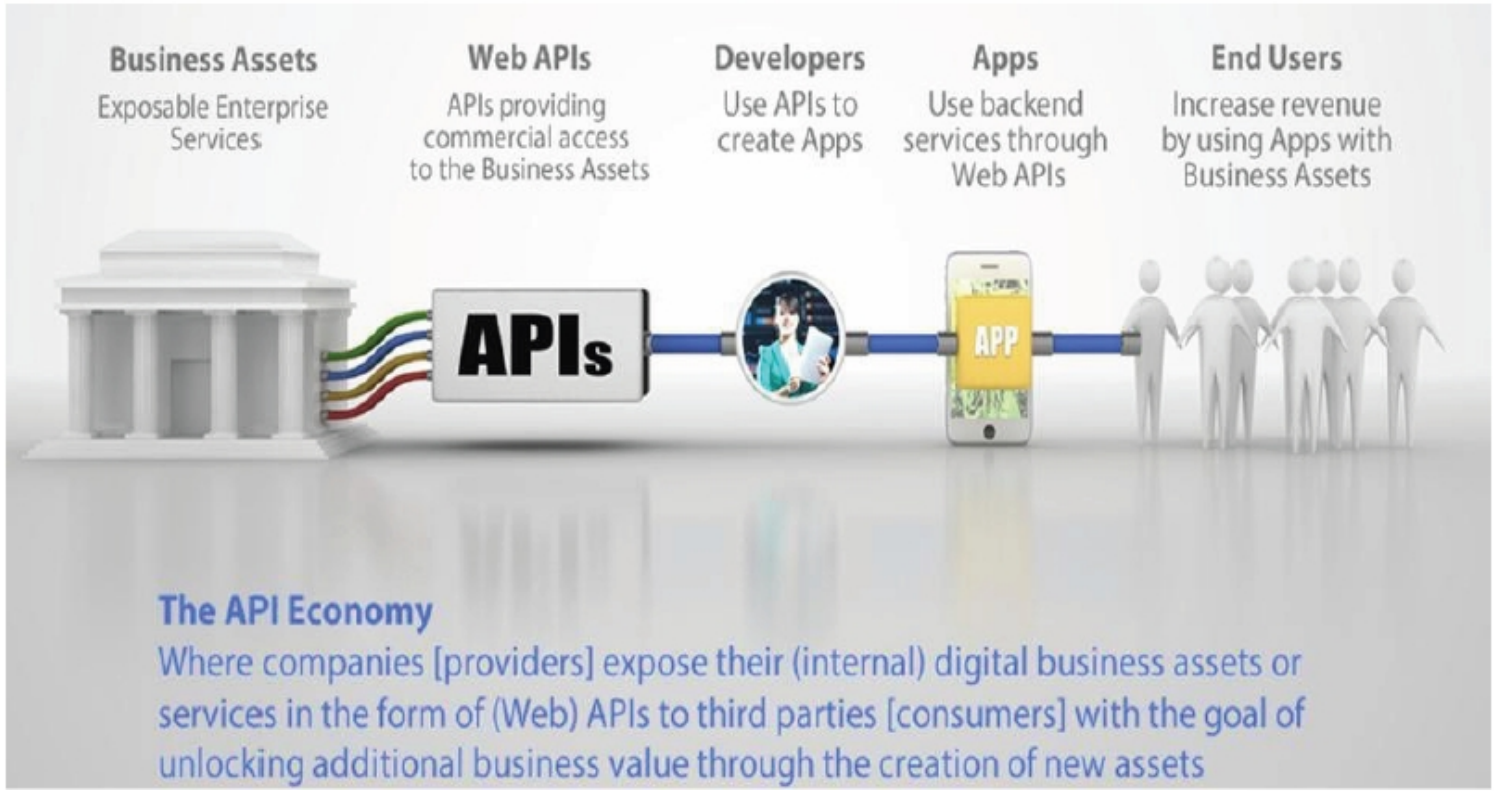
„Geschäftliches Handeln, das auf automatisierten, multilateralen,

dynamischen und vergleichsweise anonymen Kompetenznetzwerken

hoch spezialisierter Partner beruht.“

Quelle des Zitats: Resch, O.: API-Economy − eine Situationsbestimmung,
in Tagungsband. BSOA/BCloud2015, Shaker-Verlag Aachen

Sicht entsprechend IBM



Quelle der Abbildung: Alan, G.: API Economy Drivers, https://developer.ibm.com/apiconnect/2014/08/12/api-economy-drivers, Abruf: Feb. 2018

Sicht der OpenAPI-Initiative

"APIs form the connecting glue between modern applications. Nearly every application uses APIs to connect with corporate data sources, third party data services or other applications. Creating an open description format for API services that is vendor neutral, portable and open is critical to accelerating the vision of a truly connected world."

frei nutzbar

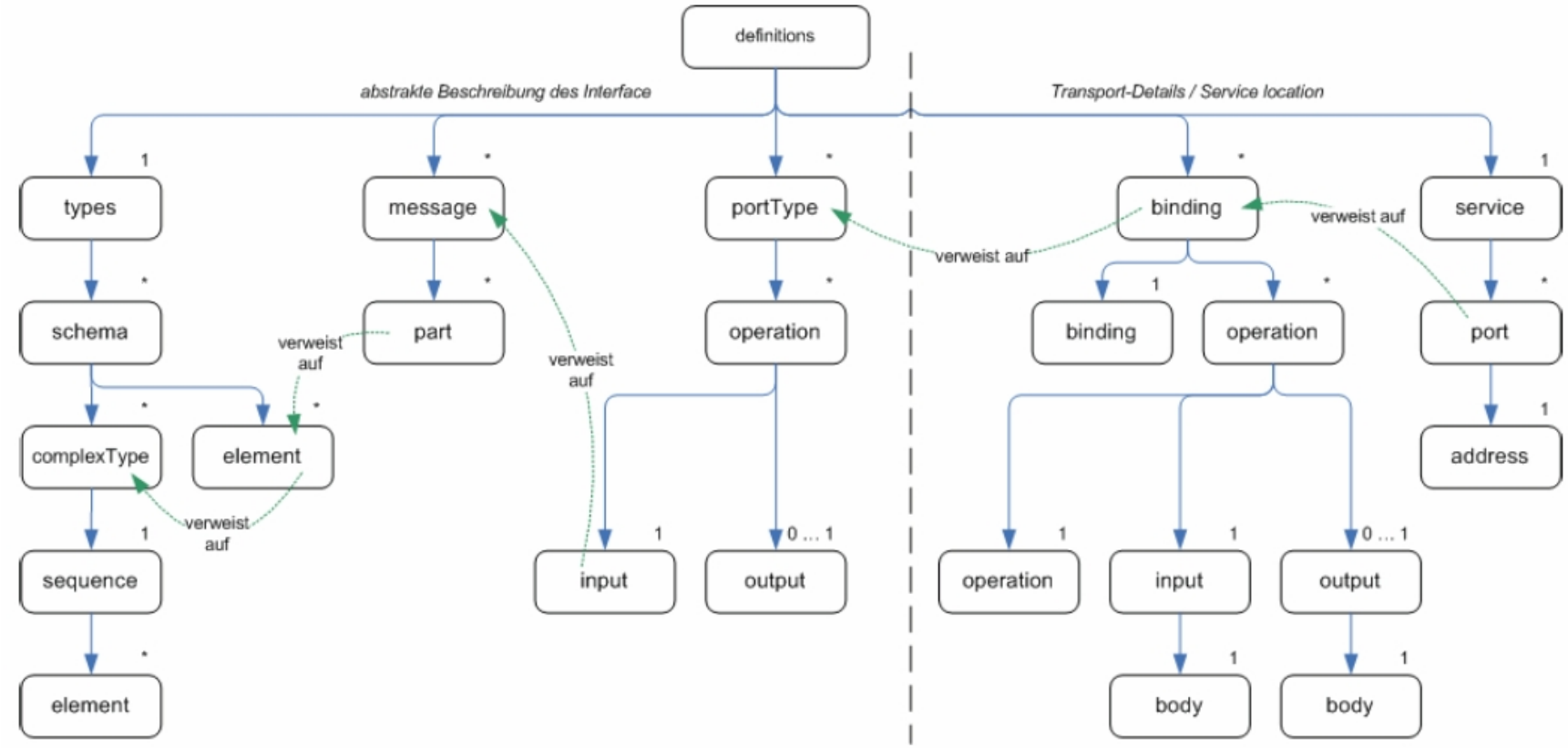
liefern Daten
und Algorithmen

offener Standard

Source: Open API Initiative, The Linux Foundation, URL: https://www.openapis.org, (last accessed 10 September 2017)

Ansätze zur Spezifikation

Web Service Desciption Language



Quelle: https://de.wikipedia.org/wiki/Web_Services_Description_Language, letzter Abruf Feb. 2018

WSDL Beispiel

```
− <wsdl:definitions targetNamespace="http://pegelonline.wsv.de/webservices/version2_3/2007/10/31">
  + <wsdl:documentation></wsdl:documentation>
  + <wsdl:types></wsdl:types>
  + <wsdl:message name="getParameterListRequest"></wsdl:message>
  + <wsdl:message name="getParameterListResponse"></wsdl:message>
  + <wsdl:message name="getGewaesserListRequest"></wsdl:message>
  + <wsdl:message name="getGewaesserListResponse"></wsdl:message>
  + <wsdl:message name="getGewaesserRequest"></wsdl:message>
  + <wsdl:message name="getGewaesserResponse"></wsdl:message>
  + <wsdl:message name="getMessstellenListRequest"></wsdl:message>
  + <wsdl:message name="getMessstellenListResponse"></wsdl:message>
  + <wsdl:message name="getMessstelleRequest"></wsdl:message>
  + <wsdl:message name="getMessstelleResponse"></wsdl:message>
  + <wsdl:message name="getMessstellenParameterRequest"></wsdl:message>
  + <wsdl:message name="getMessstellenParameterResponse"></wsdl:message>
  + <wsdl:message name="getDatenverfuegbarkeitRequest"></wsdl:message>
  + <wsdl:message name="getDatenverfuegbarkeitResponse"></wsdl:message>
  + <wsdl:message name="getPegelinformationenRequest"></wsdl:message>
  + <wsdl:message name="getPegelinformationenResponse"></wsdl:message>
  + <wsdl:message name="getGanglinienUrlRequest"></wsdl:message>
  + <wsdl:message name="getGanglinienUrlResponse"></wsdl:message>
  + <wsdl:message name="getGanglinienImageRequest"></wsdl:message>
  + <wsdl:message name="getGanglinienImageResponse"></wsdl:message>
  + <wsdl:message name="getMessungenDateiRequest"></wsdl:message>
  + <wsdl:message name="getMessungenDateiResponse"></wsdl:message>
  + <wsdl:message name="getMessungenAktuellRequest"></wsdl:message>
  + <wsdl:message name="getMessungenAktuellResponse"></wsdl:message>
  + <wsdl:portType name="PegelonlineWebservicePortType"></wsdl:portType>
  + <wsdl:binding name="PegelonlineWebserviceSoapBinding" type="tns:PegelonlineWebservicePortType"></wsdl:binding>
  − <wsdl:service name="PegelonlineWebservice">
    − <wsdl:port binding="tns:PegelonlineWebserviceSoapBinding" name="PegelonlineWebservicePort">
        <wsdlsoap:address location="http://www.pegelonline.wsv.de/webservices/version2_3/2007/10/31/PegelonlineWebservice"/>
```

http://www.pegelonline.wsv.de/webservices/version2_3/2007/10/31/PegelonlineWebservice?WSDL, letzter Abruf Feb. 2018

OpenAPI-Spezfikation (Swagger)

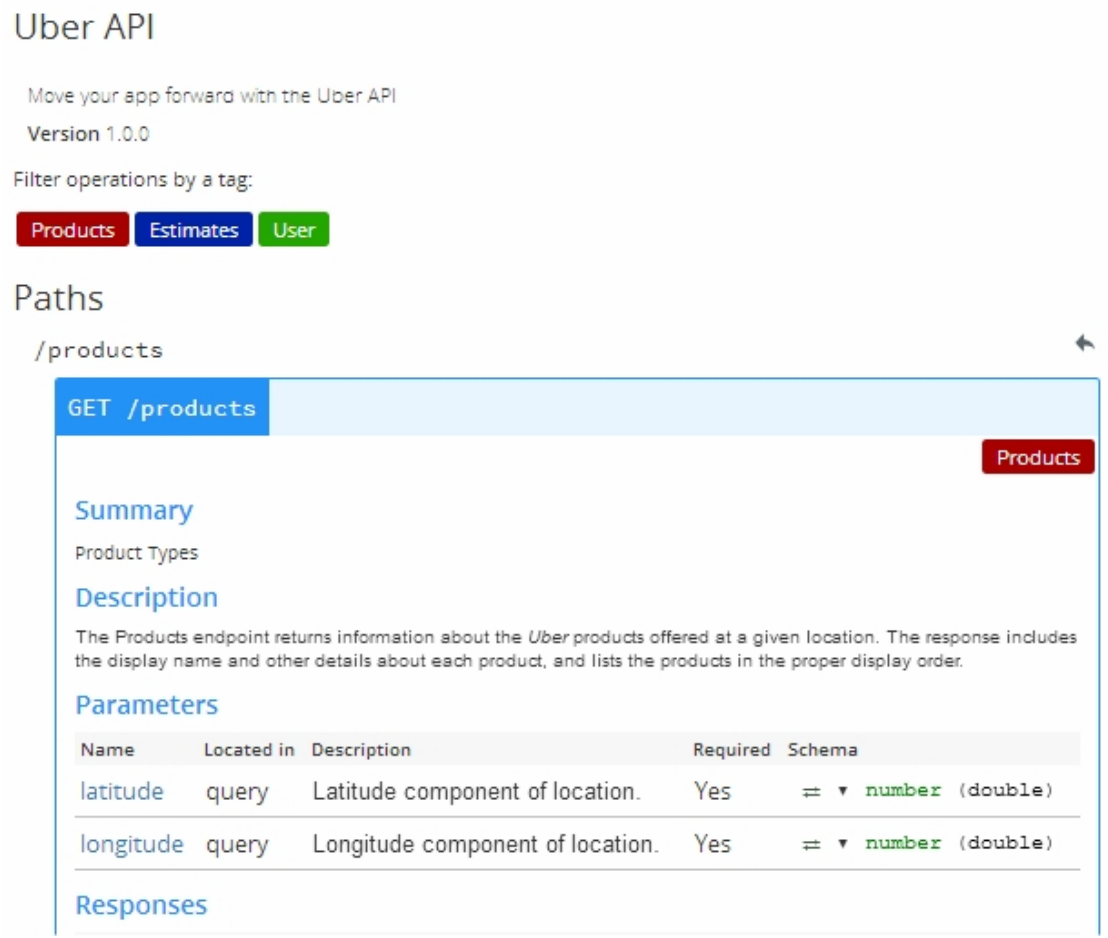
- Vertrag zwischen dem Nutzer und Anbieter einer Open-API.
- Annotative Hinterlegung im Quellcode
- Automatische Erzeugung von Quellcode-Stub.
- Erzeugung von „mock services“ als virt. Serviceendpunkt.
- Unterstützung einer (dynamischen) “späten Bindung”.
- Generative Bereitstellung einer Servicebeschreibung (z.B. HTML)
- Verwendung im Versions- und Konfigurationsmanagement.

OpenAPI (Swagger/YAML) Beispiel

```yaml
# this is an example of the Uber API
# as a demonstration of an API spec in YAML
swagger: '2.0'
info:
  title: Uber API
  description: Move your app forward with the Uber API
  version: "1.0.0"
# the domain of the service
host: api.uber.com
# array of all schemes that your API supports
schemes:⟺
# will be prefixed to all paths
basePath: /v1
produces:
  - application/json
paths:
  /products:
    get:
      summary: Product Types
      description: |
        The Products endpoint returns information about the *Uber* products
        offered at a given location. The response includes the display name
        and other details about each product, and lists the products in the
        proper display order.
      parameters:⟺
      tags:⟺
      responses:⟺
  /estimates/price:⟺
  /estimates/time:⟺
  /me:⟺
  /history:⟺
definitions:⟺
```

Quelle der Abbildung: https://editor2.swagger.io/#!/, Abruf: Feb. 2018

OpenAPI (Swagger) Beispiel



Quelle der Abbildung: https://editor2.swagger.io/#!/, Abruf: Feb. 2018

Vergleich von Open API Angeboten

	NASA Open-API	Island Open-API	World Bank	Deutsche Bahn	Lufthansa
Anzahl API's	12	20	3	10	2
Technik	REST/HTTP	REST/HTTP	REST/HTTP	REST/HTTP	REST/HTTP
Format	JSON, XML	JSON	JSON, Atom, RDF	JSON	JSON
Spezifikation	Swagger, extra Dokumentation	Nachrichten-Beispiele	Nachrichten-Beispiele	Swagger	Swagger
Test	Frei	Frei	Frei	Registrierung (Key)	Registrierung (Key)
Nutzung	Registrierung (Key)	Frei	Frei	Registrierung (Key)	Registrierung (Key)
Verfügbarkeit	keine Angabe	"jederzeit"	Caching empfohlen	unbekannt (Beta-Version)	unbekannt (Vertrag erf.)
Messaspekte	Zugriffsrate je Stunde/Tag	Keine	Keine	Zugriffsrate	Zugriffsrate je Sek./Stunde
API Publikation	Ja	Nein	Nein	Nein	Nein
Bemerkungen		Open Source	interaktiver Query-Builder	alle in Vers. 1.0 (eine Ausn.)	Umsatzbeteiligung

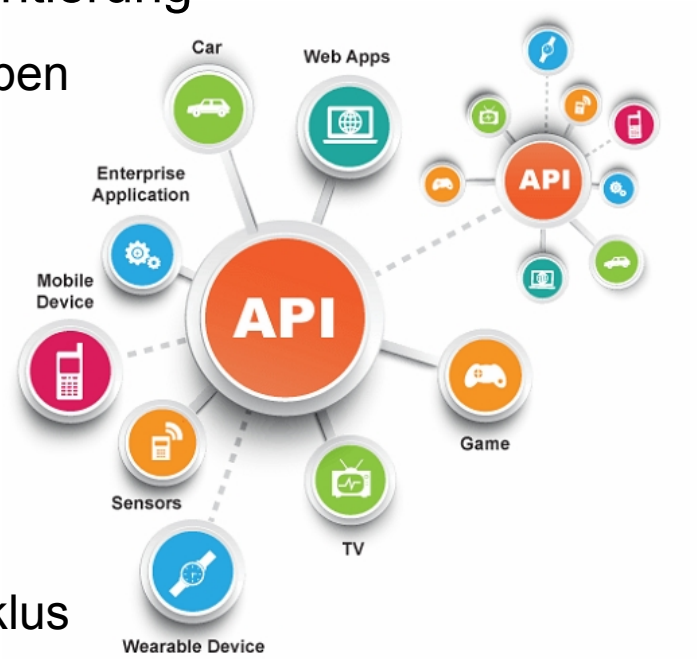
Vergleich von Open API Angeboten

	NASA Open-API	Island Open-API	World Bank	Deutsche Bahn	Lufthansa	
Anzahl API's	12	20	3	10	2	
Technik	REST/HTTP	REST/HTTP	REST/HTTP	REST/HTTP	REST/HTTP	**REST/HTTP und JSON sind als Standard gesetzt**
Format	JSON, XML	JSON	JSON, Atom, RDF	JSON	JSON	
Spezifikation	Swagger, extra Dokumentation	Nachrichten-Beispiele	Nachrichten-Beispiele	Swagger	Swagger	
Test	Frei	Frei	Frei	Registrierung (Key)	Registrierung (Key)	**Registrierung auch für Testzwecke im europäischen Raum üblich**
Nutzung	Registrierung (Key)	Frei	Frei	Registrierung (Key)	Registrierung (Key)	
Verfügbarkeit	keine Angabe	"jederzeit"	Caching empfohlen	unbekannt (Beta-Version)	unbekannt (Vertrag erf.)	**Durchgängig sehr dürftige Aussagen zum Qualitätsverhalten**
Messaspekte	Zugriffsrate je Stunde/Tag	Keine	Keine	Zugriffsrate	Zugriffsrate je Sek./Stunde	
API Publikation	Ja	Nein	Nein	Nein	Nein	
Bemerkungen		Open Source	interaktiver Query-Builder	alle in Vers. 1.0 (eine Ausn.)	Umsatz-beteiligung	

Möglichkeiten zur Entwicklung

API-First vs. API-Design-First, Contract-First

- Design der API ohne Fokus auf Implementierung
  - Optimale Unterstützung fachlicher Aufgaben
  - Existierende Systeme auf API anpassen
- APIs für Kunden (Entwickler)
  - Qualitätsgesicherte API
  - Vertragliche Zusicherungen
- Professionelle Dokumentation
  - Unterstützung des kompletten Lebenszyklus
  - Selbstbeschreibende Schnittstellen anstreben



Quelle der Abbildung: https://dzone.com/articles/an-api-first-development-approach-1 (Abruf: April 2018)

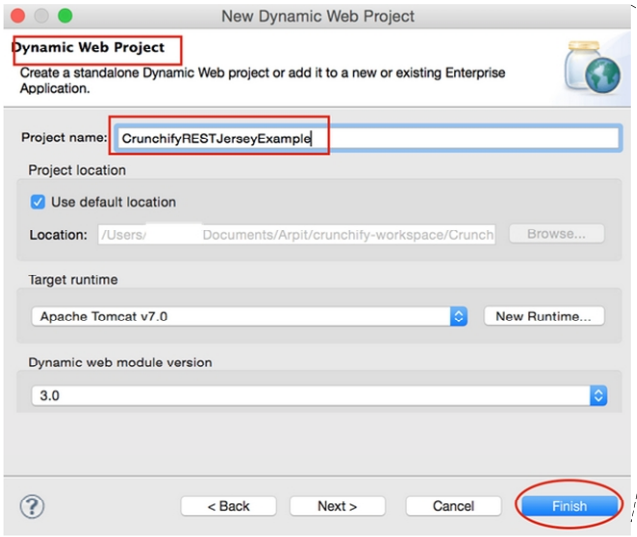
Technologien und Frameworks

- JAX-WS, JAX-RS
- Axis2
- Spark
- Dropwizard
- Springboot
- WSO2 MSF4J
- RabbitMQ

- OWIN
- ASP.NET Web-API
- Service Fabric

- Laravel
- Symfony
- Slim Framework

Vorgehen bei Java (Jersey)



Vorraussetzungen sichern	Java SDK bzw. JEE (Jersey enthalten) JAX RS Referenzimplementeirung (Jersey) Web Server (Tomcat, Grizzly, jetty)
Web API Projekt anlegen	Verzeichnisstruktur erzeugen `bin\ lib\ src\` Projektabhängigkeiten `z.B. POM.xml`
Service-implementierung	Java-Klassen - Serviceimplementierung Modell-Klassen (z.B. als POJOs) Controller-Klassen (z.B. HTTPServlet)
Projekt übersetzen und linken	Java-Compiler → `javac ...`
Deployment in der Laufzeitumgebung	Webserver in IDE einbinden Port für HTTP festlegen – z.B. 8080 Deployment des Service
Start/Test der API	Starten und Beenden einer Web-API Tests der Funktionen – Browser, cURL API-Management/-Monitoring

Quelle des Dialogs (links): http://crunchify.com/how-to-build-restful-service-with-java-using-jax-rs-and-jersey, Abruf: Feb. 2018

Beispiel Serviceimplementierung (Jersey)

Service Implementierung

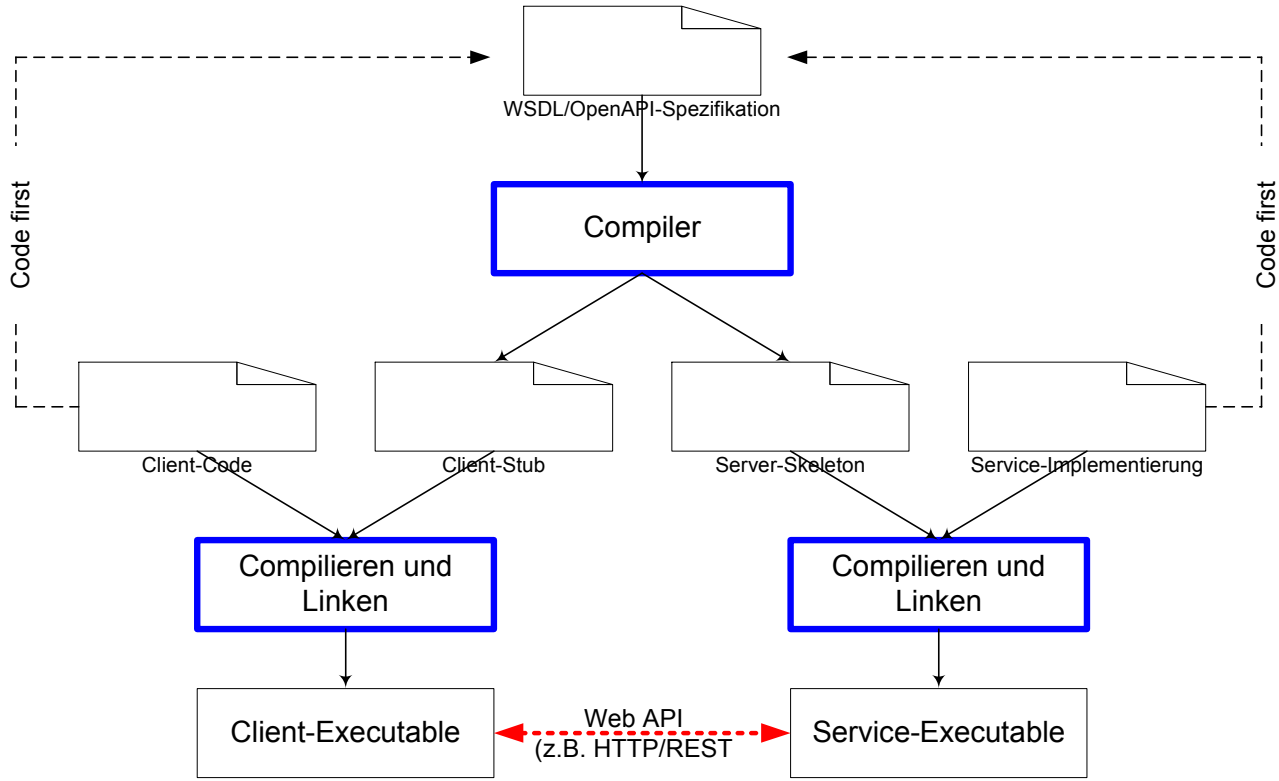
```
package minirestwebservice;

import javax.ws.rs.*;
import javax.ws.rs.core.MediaType;

@Path( HalloWeltService.webContextPath )
public class HalloWeltService
{
   static final String webContextPath = "/helloworld";

   @GET @Produces( MediaType.TEXT_PLAIN )
   public String halloPlainText( @QueryParam("name") String name )
   {
      return "Plain-Text: Hallo " + name;
   }

   @GET @Produces( MediaType.TEXT_HTML )
   public String halloHtml( @QueryParam("name") String name )
   {
      return "<html><title>HelloWorld</title><body><h2>Html: Hallo " + name + "</h2></body></html>";
   }
}
```

```
package minirestwebservice;

import java.io.IOException;
import java.net.URI;
import org.glassfish.grizzly.http.server.HttpServer;
import org.glassfish.jersey.grizzly2.httpserver.GrizzlyHttpServerFactory;
import org.glassfish.jersey.server.ResourceConfig;

public class HalloWeltTestServer
{
   public static void main( String[] args ) throws IOException, InterruptedException
   {
      String baseUrl = ( args.length > 0 ) ? args[0] : "http://localhost:4434";

      final HttpServer server = GrizzlyHttpServerFactory.createHttpServer(
            URI.create( baseUrl ), new ResourceConfig( HalloWeltService.class ), false );
      Runtime.getRuntime().addShutdownHook( new Thread( new Runnable() {
         @Override
         public void run() {
            server.shutdownNow();
         }
      } ) );
      server.start();

      System.out.println( String.format( "\nGrizzly-HTTP-Server gestartet mit der URL: %s\n"
                                         + "Stoppen des Grizzly-HTTP-Servers mit:      Strg+C\n",
                                         baseUrl + HalloWeltService.webContextPath ) );

      Thread.currentThread().join();
   }
}
```

Grizzly Web-Server

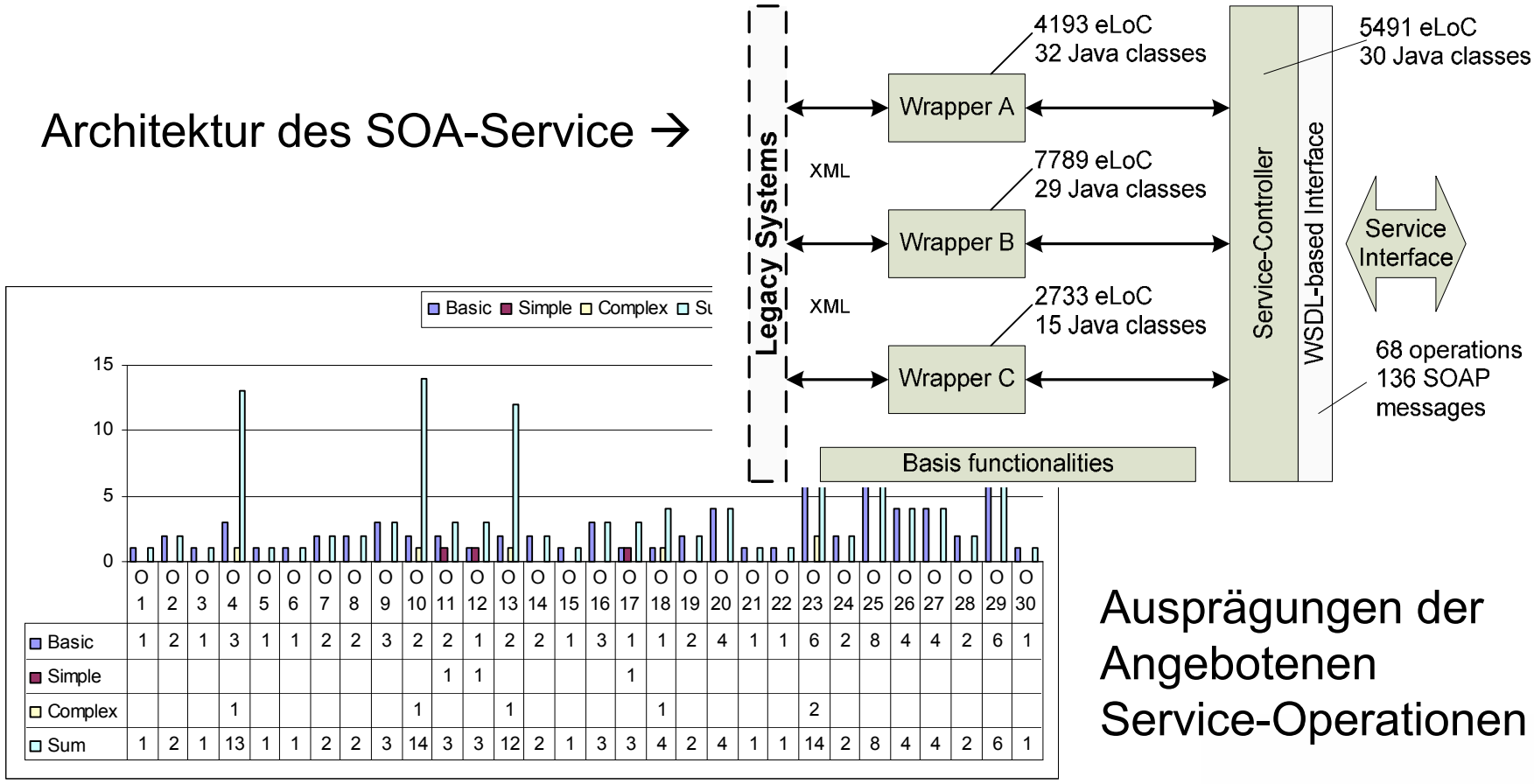
http://www.torsten-horn.de/techdocs/jee-rest.htm, letzter Abruf: Feb 2018

Contract first vs. Code first



Beispiel einer Wrapper-Applikation

Architektur des SOA-Service →



	O 1	O 2	O 3	O 4	O 5	O 6	O 7	O 8	O 9	O 10	O 11	O 12	O 13	O 14	O 15	O 16	O 17	O 18	O 19	O 20	O 21	O 22	O 23	O 24	O 25	O 26	O 27	O 28	O 29	O 30
Basic	1	2	1	3	1	1	2	2	3	2	2	1	2	2	1	3	1	1	2	4	1	1	6	2	8	4	4	2	6	1
Simple											1	1					1													
Complex				1						1			1					1					2							
Sum	1	2	1	13	1	1	2	2	3	14	3	3	12	2	1	3	3	4	2	4	1	1	14	2	8	4	4	2	6	1

Ausprägungen der Angebotenen Service-Operationen

Vielfältige Tutorials

- RESTful Web Services mit JAX-RS 2.1 und Jersey 2.26 (inkl. Maven)
  - → http://www.torsten-horn.de/techdocs/jee-rest.htm
- Web Services mit der Apache Axis2 Engine
  - → http://axis.apache.org/axis2/java/core
- How to build RESTful Service with Java using JAX-RS and Jersey
  - → http://crunchify.com/how-to-build-restful-service-with-java-using-jax-rs-and-jersey
- Erstellen einer Web-API mit ASP.NET Core und Visual Studio für Windows
  - → https://docs.microsoft.com/de-de/aspnet/core/tutorials/first-web-api

API Management

Bezugsbereich

„Als API-Management wird das Veröffentlichen von Application Programming Interfaces (API) in einer sicheren und skalierbaren Umgebung bezeichnet. Ergänzt wird die Veröffentlichung durch den Support und die Dokumentation der API. Ziel des API-Managements ist es, den Lebenszyklus einer API verfolgen zu können, um sie so anbieten zu können, dass sie Nutzern und Programmierern den bestmöglichen Nutzen stiftet.“

Quelle: »From SOA2WOA« - Leitfaden, bitkom 2016, abrufbar unter: https://www.bitkom.org

Funktionen im API-Management I

- Management des Lebenszyklus einer Web API
  - Publizieren einer API
  - Kaufen einer API
- Zugriffskontrolle auf konkrete APIs
  - Autorisierung via z.B. OAuth 2.0
  - AccesKey – konfigurierbare Gültigkeit
  - Ist je HTTP-Request hinzufügen
- Identity Repository
  - Datenbanken für Credentials
  - Zugriff auf externe Verzeichnisse (z.B. LDAP)

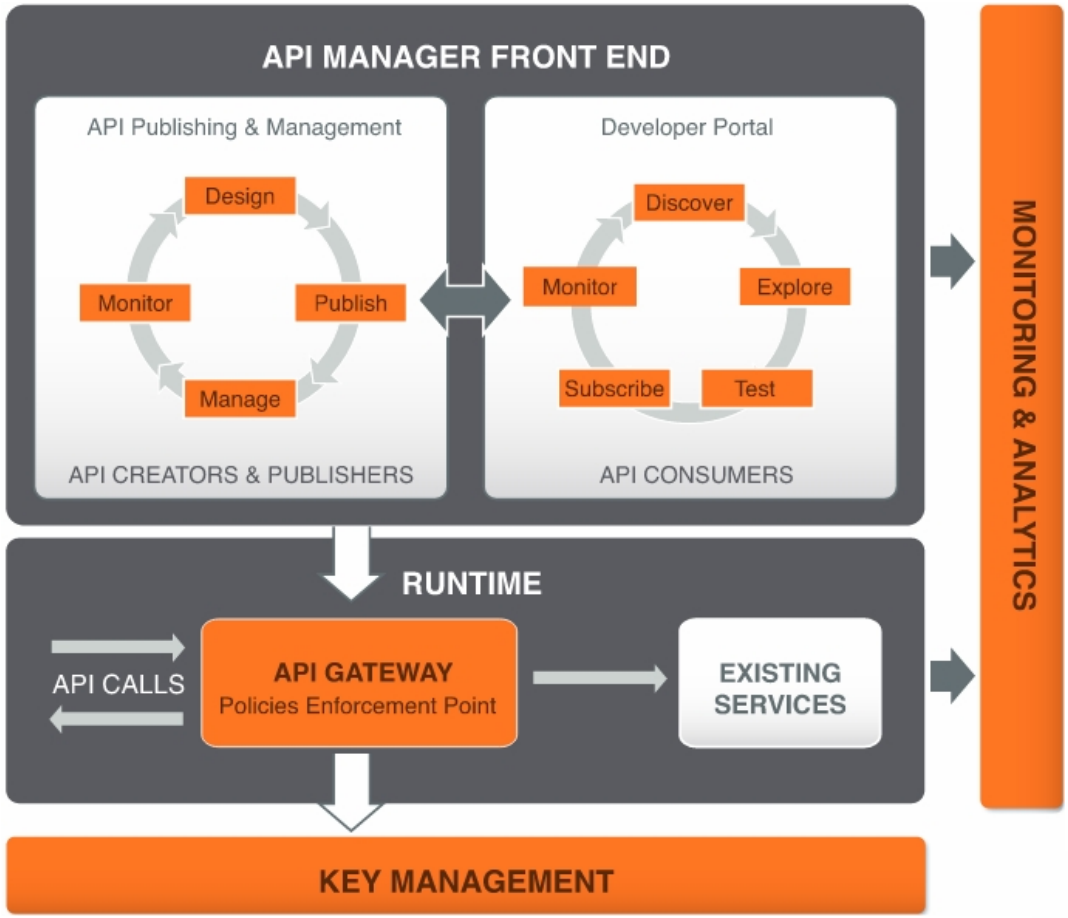
In Anlehnung: Knuth, M.: Kann die Nutzung des API-Managements die Bereitstellung von geschäftsrelevanten Schnittstellen verbessern?, in Proc. BSOA/BCloud 2016, Shaker-Verlag Aachen

Funktionen im API-Management II

- Monitoring und Billing
  - Nutzungshäufigkeiten im Geschäftskontext
  - Bereitstellung von Preismodellen
- Publisher und Store
  - Informationen über die API (Versionierung)
  - Lebenszyklus-Management (verschiedene Stati)
  - Subsciption eines API-Nutzers (API-Abo)
- Container für speziellen Nutzer
  - Verwaltung der abonnierten APIs
  - Zugriff entsprechend der eingeräumten Rechte

In Anlehnung: Knuth, M.: Kann die Nutzung des API-Managements die Bereitstellung von geschäftsrelevanten Schnittstellen verbessern?, in Proc. BSOA/BCloud 2016, Shaker-Verlag Aachen

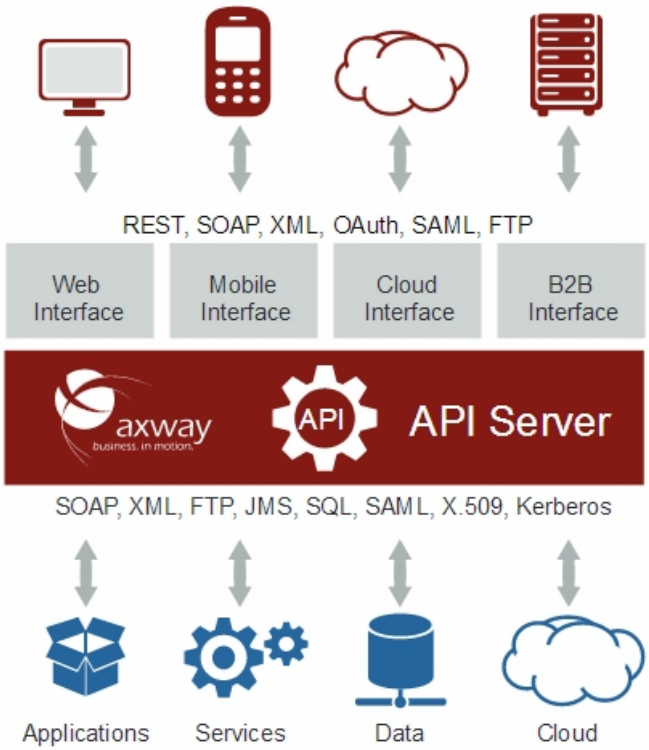
Open Source API-Management WSO2



Quelle: API-Management, WSO2, http://cdn.wso2.com/wso2/files/wso2-apim-datasheet.pdf

Axway API Gateway

- End-to-end lifecycle management (Virtual.)
- Security and authentication (u.a. OAuth)
- Control & Governance
- Monitoring & Reporting (u.a. API Nutzung)
- Asynchrones Messaging (u.a. JMS)
- Administration (u.a. Verfügbarkeit)



Quelle: Axway API Gateway, https://www.axway.com/sites/default/files/datasheet_files/axway_datasheet_api-gateway_en.pdf, Abruf: 10/2016