The “Internet of Services Master Project” is equivalent to the “Internet of Services Lab”
Established in December 2009

– Prof. Dr. Axel Küpper
– Part of TU Berlin and Telekom Innovation Laboratories

Information
– http://www.snet.tu-berlin.de

Research Areas
– Mobile Computing
– Positioning Mechanisms
– Cloud Computing
– Online Social Networks
– Semantic Web & Linked Data

Other Courses
– Mobile Services (ST)
– Geschäftsprozesse und IT-Dienste (ST)
– Electronic Commerce (WT)
– Digital Communities (WT)
Prerequisites
– Basic knowledge of computer science
– Practical experience in object-oriented programming and software design

Good to have
– Profound knowledge and experience in
  • Web Services (Protocols and APIs)
  • Mobile Computing (especially Android)
  • DBMS (e.g. SQL)
– Experience with IDEs (e.g. Eclipse) and Version Control Systems
Project organization

You will be working together in small teams
  – Each team will get a specific topic
  – ca. 3-4 students per topic
  – Teams will have to organize themselves, i.e. come up with a project plan etc.

Working mode
  – The project has 9 ECTS (approx. 6 SWS)
  – Project duration: 15 weeks

Workload per person per week
  – Max: 15 – 18 hours
  – Realistic: 10 – 15 hours
Project organization

Meetings
- Milestone presentation: January 7th, 2014
- Final presentation: February 25th 2014
- Furthermore, there will be regular meetings with the supervisors

Talks
- Milestone presentations: 15 minutes + 5 minutes Q&A
- Final presentations: 20 minutes + 10 minutes Demo / Q&A

Attendance
- Attendance to all appointments is mandatory
Teams are required to define a project plan
  – Define and distribute tasks
  – Estimate each task's time consumption
  – Track who worked on what task for how long

Project documentation
  – To be handed in one week after final presentation

Project focus
  – Implementation (50%)
  – Documentation (30%)
  – Presentation (20%)
Tools and Infrastructure

- **Redmine Project Management System**
  - Repository: SVN/GIT/Mercurial
  - Wiki
  - Ticketing system

- **Docker server**

- **Mobile devices**
Docker

• Container technology
  – Lightweight, self-supporting, build from a text file, starts in ms
  – One container for each component (e.g., DB, Web, MQ, ...)
  – Easy deployment & management

• Remarkable History
  – Open Sourced in 03/13 from internal project
  – Funding: 55 M$ in 17 months
  – Rapid adoption, crazy hype, high-paced development

• Docker Hub Registry: 14.000+ container images
  – docker run <image>

• No virtualization!
Docker

Web A
Web B
DB A
DB B

Web Layer (ro)
DB Layer (ro)
Guest OS Layer (ro)
Docker Engine
Host OS
Hardware
FROM dockerfile/ubuntu

# Copy Proxy files
ADD . /root/tresor-proxy
WORKDIR /root/tresor-proxy

# Install Ruby 2.0 using rvm and install gems of proxy
RUN curl -sSL https://get.rvm.io | bash -s stable --ruby=2.0 &&
   /bin/bash -c "source /usr/local/rvm/scripts/rvm && bundle install" &&
   chmod +x /root/tresor-proxy/bin/proxy_docker.sh

# Run Proxy
CMD ["/bin/bash", "-c"]
ENTRYPOINT ["/root/tresor-proxy/bin/proxy_docker.sh"]

Simple, Reproducible, Comprehensible
Configure, link, run multiple containers

```
web:
  build: .
  command: python app.py
  ports:
    - "5000:5000"
  volumes:
    - .:/code
  links:
    - redis
redis:
  image: redis
```
What we’ll expect: Code structure

- One directory per application component (e.g., web, db, redis, ...)

- Directory contents:
  - Dockerfile + application sources, schemas, etc.

- Root contents
  - fig.yml, README.md, AUTHORS, LICENSE

- fig up should be enough to „run“ your results

- Best practices
  - Write automated tests to test your application
  - Be mindful what you git add (no logs, binary dbs, temp files, etc.)
  - Use git submodules for including external projects
IoSP
TOPICS
Tools and Infrastructure

**Topic 1: Open Service Compendium**
- Supervisor: Mathias Slawik (ca. 4 Places)

**Topic 2: 2D/3D Visualization of Geofences**
- Supervisor: Bersant Deva, Sandro Rodriguez (3-4 Places)

**Topic 3: TU-B Here App**
- Supervisor: Sebastian Zickau, Bersant Deva (3-4 Places)

**Topic 4: Peer2Peer Directory Service**
- Supervisor: Sebastian Göndör, Felix Beierle (3-4 Places)

**Topic 5: eID-card enabled Web-Service**
- Supervisor: Dirk Thatmann (3-4 Places)
Topic #1: Mathias Slawik

OPEN SERVICE COMPRENDIUM
Agenda

- Background
- Your working mode
- Possible contributions
The TRESOR Open Service Broker

BACKGROUND
• **Trusted Ecosystem for Standardized** and **Open cloud-based Resources**
  – Funded by BMWi, runs from 03.2012 until 03.2015

• **Building a Secure Cloud Ecosystem for the German Healthcare sector**
• As an SME cloud provider, I want to describe my SaaS offerings
  – My “Health Record” service saves its data in Germany, can import Word and PDF files, and can be used by Internet Explorer and Chrome

• As an SME cloud consumer, I want to search for services matching my selection criteria
  – Show me all services, which save their data in the EU, can import PDFs and can be used by Chrome

• As an SME marketplace operator, I want to define a vocabulary for service description
  – Let marketplace stakeholders describe their data location, importable formats, compatible browsers, etc.
Components

1. SDL-NG framework
2. Pertinent business vocabulary
3. TRESOR broker

Open Source: https://github.com/TU-Berlin-SNET
SDL-NG framework examples

Vocabulary definition

Service Description
dynamic do
  fetched_features = fetch_from_url 'http://www.salesforce.com/sales-cloud/overview',
  '.slide h3 + *'
  feature 'Mobile', fetched_features[0]
  feature 'Contact Management', fetched_features[1]
  feature 'Chatter', fetched_features[3]
  feature 'Email Integration', fetched_features[4]
end

<table>
<thead>
<tr>
<th>Features</th>
<th>Name</th>
<th>Mobile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Now, Salesforce, combined with your custom apps and AppExchange mobile apps, turns your mobile device into a portable sales office. You can log calls, respond to hot leads, work opportunities, or check dashboards no matter where you are. And collaborate across teams, from anywhere.</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Contact Management</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>Bring social intelligence to your sales process. Gain insights from popular social media sites — like Facebook, Twitter, LinkedIn, YouTube, and Klout — right within Salesforce to help you increase productivity.</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Opportunity Management</td>
<td></td>
</tr>
<tr>
<td>Area</td>
<td>Example properties</td>
<td></td>
</tr>
<tr>
<td>--------------------</td>
<td>-------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Characteristics</td>
<td>Cloud service model, service categories</td>
<td></td>
</tr>
<tr>
<td>Charging</td>
<td>Charge unit (user account, floating license)</td>
<td></td>
</tr>
<tr>
<td>Compliance</td>
<td>Data location, audit options (e.g. audit logging)</td>
<td></td>
</tr>
<tr>
<td>Delivery</td>
<td>Billing and payment options</td>
<td></td>
</tr>
<tr>
<td>Dynamics</td>
<td>Duration for provisioning an end user</td>
<td></td>
</tr>
<tr>
<td>Interop</td>
<td>Features, interfaces, and compatible browsers</td>
<td></td>
</tr>
<tr>
<td>Optimizing</td>
<td>Maintenance windows and future roadmaps</td>
<td></td>
</tr>
<tr>
<td>Portability</td>
<td>Exportable and importable data formats</td>
<td></td>
</tr>
<tr>
<td>Protection</td>
<td>Communication protection (HTTPS, VPN)</td>
<td></td>
</tr>
<tr>
<td>Provider management</td>
<td>Support availability</td>
<td></td>
</tr>
<tr>
<td>Reliability</td>
<td>Offline capabilities</td>
<td></td>
</tr>
<tr>
<td>Reputation</td>
<td>Year of service establishment</td>
<td></td>
</tr>
<tr>
<td>Trust</td>
<td>Providers' financial statement, reference customers</td>
<td></td>
</tr>
</tbody>
</table>

= 37 Type classes, 31 Service properties and 52 Type instances
Outcome: TRESOR Broker Service editor

Services > Edit 'Google Drive for Business'

Service description (Syntax)

```plaintext
# base
service_name 'Google Drive for Business'

is_billed monthly, in_advance

# characteristics
cloud_service_model saas

in_advance local
```
# Service

A service

## Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>service_name</td>
<td>Service name (SDLString)</td>
</tr>
<tr>
<td>cloud_service_model</td>
<td>Cloud service model (CloudServiceModel)</td>
</tr>
<tr>
<td>service_categories[]</td>
<td>Categories (ServiceCategory)</td>
</tr>
<tr>
<td>service_tags[]</td>
<td>Tags (SDLString)</td>
</tr>
<tr>
<td>add_on_repository</td>
<td>Service Add-On Repository (AddOnRepository)</td>
</tr>
<tr>
<td>is_charged_by</td>
<td>Charging unit (ChargeUnit)</td>
</tr>
<tr>
<td>data_location</td>
<td>Data location (Location)</td>
</tr>
<tr>
<td>data_deletion_policy</td>
<td>Data deletion policy (SDLUrl)</td>
</tr>
<tr>
<td>status_page</td>
<td>The status page URL (SDLUrl)</td>
</tr>
</tbody>
</table>
**CloudServiceModel**

A cloud service model

**Predefined instances**

- `:saas` Software as a Service (SaaS)
- `:paas` Platform as a Service (PaaS)
- `:iaas` Infrastructure as a Service (IaaS)
- `:haas` Hardware as a Service (HaaS)

**CommunicationProtection**

A communication protection means

**Predefined instances**

- `:https` HTTPS encryption
- `:vpn` VPN connection
## Service

A service

### Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>service_name</td>
<td>Service name (SDLString)</td>
</tr>
<tr>
<td>cloud_service_model</td>
<td>Cloud service model (CloudServiceModel)</td>
</tr>
<tr>
<td>service_categories[]</td>
<td>Categories (ServiceCategory)</td>
</tr>
<tr>
<td>service_tags[]</td>
<td>Tags (SDLString)</td>
</tr>
<tr>
<td>add_on_repository</td>
<td>Service Add-On Repository (AddOnRepository)</td>
</tr>
<tr>
<td>is_charged_by</td>
<td>Charging unit (ChargeUnit)</td>
</tr>
<tr>
<td>data_location</td>
<td>Data location (Location)</td>
</tr>
<tr>
<td>data_deletion_policy</td>
<td>Data deletion policy (SDLUrl)</td>
</tr>
<tr>
<td>status_page</td>
<td>The status page URL (SDLUrl)</td>
</tr>
</tbody>
</table>
TRESOR Open Service Broker

Main functionality

The TRESOR Open Service Broker is an information system realizing the following main functions:

- Managing service descriptions, clients, and providers
- Performing booking of services

Format of parameters

All described parameters can be sent using these methods:

- Managing service descriptions, clients, and providers
- Performing booking of services

Format of parameters

All described parameters can be sent using these methods:

- Sending parameters within the HTTP body, \texttt{multipart/form-data} encoded (on POST & PUT
Independent, crowd-sourced “service Wikipedia“
WORKING MODE
Your working mode

1. Defining your contribution
   - Based on team composition, skill set, preferences

2. Doing research
   - state-of-the-art, existing contribution, related technologies

3. Iterative-incremental application development
   - user story definition, implementation, automated testing, deployment
Your new skills

• Modern web application development
  – Ruby, Rails, MongoDB, testing, deployment

• Cloud Computing business aspects
  – Market overview, service description & elicitation

• Modelling and language design
  – Models and meta-models, language concepts
What you can do

POSSIBLE CONTRIBUTIONS
Your possible contributions

- **Broker**
  - Containerized and secure SDL execution
  - SDL editor enhancements
  - Service comparison
  - Faceted search
  - Descriptions, descriptions, descriptions,

- **Vocabulary**
  - Analytic evaluation
  - Vocabulary refinement
  - Category-specific properties

- **SDL-NG**
  - Dynamic vocabulary
  - Price models and calculation
  - Property brokering information
  - Service variants
Topic #2: Sandro Rodriguez Garzon, Bersant Deva

2D/3D VISUALIZATION OF GEOFENCES
Topic 2: 2D/3D Visualization of Geofences

Idea
Topic 2: 2D/3D Visualization of Geofences

Tasks

• Get familiar with Geofencing
• Get familiar with a Geofencing System (Interfaces)
• Develop a 2D visualization of geofences based on the Google Maps API.
• Develop a 3D visualization of geofences based on the Google Earth API.
• Enable a stereoscopic view of the 3D visualization using the Google Cardboard Framework for Android
Topic 2: 2D/3D Visualization of Geofences

Tasks

• Interest in location-based services
• Good experience in scripting (JavaScript)
• Experience with RESTful Web Services, XML, JSON
• Skills in Java Programming (Android Development)
• Beneficial: Knowledge in using Google Maps API, Google Earth API, KML
• Beneficial: Google Cardboard and Google Cardboard Framework
• Beneficial: 3D Visualization
Topic #3: Sebastian Zickau, Bersant Deva

TU-B HERE APP
Topic 3: TU-B Here App
Idea: Harry Potter Maurader’s Map
Topic 3: TU-B Here App
Hogwarts University of W&W
Topic 3: TU-B Here App
What’s already there?

• Done by IoSP Group in Summer Term 2014
• API by tubIT using Cisco MSE
• Server Component
• Client Component
• Gitlab with Code
• Documentation
• Contact tu TUB privacy officer
Topic 3: TU-B Here App Architecture

tuBiT Infrastruktur

eduroam

TU-B Here Server

eduroam
Topic 3: TU-B Here App
App Screenshots
Topic 3: TU-B Here App

New topic goals

• Integrate Your Ideas

• See Ideas by SummerTerm Group / Discussion
  – Table Tennis Example
  – Messaging aspects
  – Publish/Subscribe aspects

• Integrate Indoor-Maps into the application

• Update location management

• Add additional context-based aspects
  – Now: Cafeterias are included
Topic 3: TU-B Here App
New topic goals

• Think about security and privacy aspects
• Work together with tubIT
• Integration in Moccha-App (QU-chair) or tub2go
• Talk to TUB data privacy officer
• Publish App in Play Store
• Nice to have:
  – iOS Version
  – Android Wear Interface
Topic 3: TU-B Here App
Android Wear Integration
Topic 3: TU-B Here App Requirements

- Java programming
- Android / iOS development experience
- Interest in indoor positioning techniques
- Extensive Test/Evaluation of Solution/App
- English communication
Topic #4: Sebastian Göndör, Felix Beierle

PEER2PEER DIRECTORY SERVICE
Peer2Peer (P2P) Networks are innovative self-organizing architectures
- Decentralization
- High resilience
- Fast, efficient routing ($O(\log(n))$)

Structured P2P networks based on Distributed Hash Tables (DHT)
- All data is mapped to one huge address space $[0, ..., n-1]$, organized as a ring
- All participating nodes of the network are responsible for a certain part of that ring
- i.e. information is stored and managed by the node responsible for this part
Topic 4: Peer2Peer Directory Service

Introduction
Topic 4: Peer2Peer Directory Service

Introduction
Problem: Data is not stored by the owner

- Confidentiality and Integrity can be ensured using asynchronous cryptography
- Availability cannot be guaranteed. Anyhow, mapping a file to multiple addresses creates replica
Project “SONIC”
   - Distributed Online Social Network allowing users to migrate between servers

GSLS
   - Decentralized Directory Service for OSN profiles
   - Similar to the (hierarchical) DNS
   - Built with TomP2P/Jetty (Java)
Topic 4: Peer2Peer Directory Service

Objectives

- Dig into the source code of TomP2P / Jetty
- Add functionality to support monitoring of nodes and data and security checks
  - Integration of integrity checks
- Built a web service that works as an “Administration Panel” for the GSLS
  - Monitoring nodes
  - Configuring nodes
  - Tracing replica of data
Topic #5: Dirk Thatmann

eID-CARD ENABLED WEB-SERVICE
Topic 5: eID-card enabled Web-Service

Objectives

Entrance Project

Social Networks
Access Policies
Data sharing

Data owner centricity

Data privacy
regaining control of your Data.

Attribute-based encryption
Topic 5: eID-card enabled Web-Service
The playground
Topic 5: eID-card enabled Web-Service
Open eCard client – authentication
Topic 5: eID-card enabled Web-Service
Objectives

• Tasks:
  – Dive into the nPA and its authentication functions
  – Define three authentication scenarios
  – Setup the Clients
    • Android mobile phone, Open Ecard, Browser/App + NFC
    • Windows/Mac Open eCard + Browser + NFC Card reader
  – Develop a RESTful service: Play Framework and/or Jetty+Servlet on embedded Devices (e.g. FritzBox (MIPS), Raspberry Pi (ARM), Netgear NAS (x86))
    • Evaluate Dogger on embedded devices.
  – Integrate all components + create a documentation

• Requirements:
  – Ability to work independently, good organizational discipline
  – Good knowledge in Java, Servlet, HTTP, JavaScript, TLS,…
  – Experience in embedded Linux
  – Interests in understanding communication flows and function of the german ID-Card (nPA)