

---

# **Status Update Report**

*Microservice 1 (MS1)*



Cordula Eggerth

00750881

Team 301

BSc Wirtschaftsinformatik

VU Distributed Systems Engineering

Wintersemester 2017

---

## Inhaltsverzeichnis

1. Core Design Decisions.....	3
2. Überlegungen zum Service Deployment .....	16
3. Testen & Präsentation .....	17
4. Derzeitiger Status der Implementierung .....	23
5. Anhang .....	24

## 1. Core Design Decisions

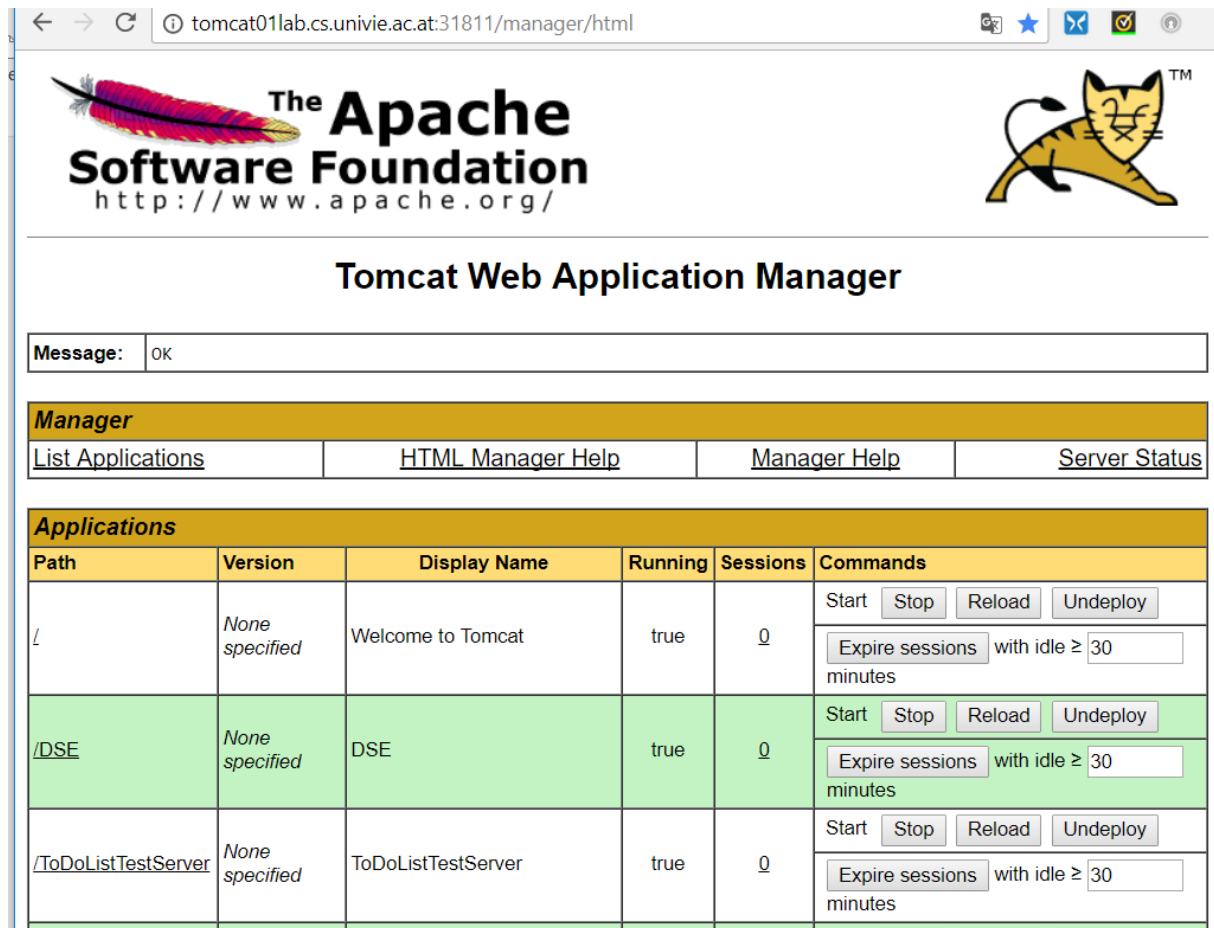
Für die Umsetzung und Planung des Aufgabenteils MS1 und der dazugehörigen Testumgebung habe ich folgende Entscheidungen zum Design bzw. der Architektur des Projekts gemacht. Der Projektname ist BusyBee. BusyBee ist ein Unternehmen, das eine Website für die Verwaltung von To-Do-Listen der User anbietet. Meine bisherige Ausarbeitung des MS1 besteht aus drei verschiedenen Java-Projects (allesamt Dynamic Web Projects, die als Maven-Projekte geführt werden). Diese Teile sind „DSE-Server“ (der die user-bezogenen Services anbietet), „ToDoListTestServer“ (der die Services für eine Testversion der ToDoList anbietet), und „DSETestClient“ (der die Funktion des Clients hat, der dem Nutzer ein User-Interface bietet, in dem er user- und todolist-bezogene Funktionalitäten durchführen kann. Die Kommunikation zwischen den Projekten erfolgt über HTTP unter Verwendung von JAX RS 2.20 (für RESTful Web-Services). Die persistente Speicherung der Daten der User erfolgt über die MySQL-Datenbank der Universität Wien namens a0750881mysql3 und die persistente Speicherung der Daten der (Test-Version der) To-Do-Listen erfolgt in der MySQL-Datenbank der Universität Wien namens a0750881mysql4<sup>1</sup>.

### **DSE-Server:**

Im Java-Projekt „DSE“ werden alle serverseitigen Dienste, die die Registrierung von Usern, den Login von Usern und weitere user-bezogene Funktionalitäten durchführen, angeboten. Der DSE-Server kann von einem Client, der user-bezogene Informationen oder verbundene Funktionen abfragen möchte, angesprochen werden. Der DSE-Server wurde von mir auch am Tomcat-Server der Universität Wien deployed, sodass er von Clients angesprochen werden kann. Der DSE-Server kann erreicht werden unter <http://tomcat01lab.cs.univie.ac.at:31811/DSE/> und kann je nach gewählter Extension des Links (URI) die Requests von der Client-Seite bedienen (siehe auch untenstehender Screenshot in Abbildung 1 über das Deployment der DSE-Services).

---

<sup>1</sup> **Anmerkung:** Der Bereich To-Do-Listen ist von mir zum Testen der user-bezogenen Services und des Gesamtprojekts in einer einfacheren Version implementiert worden, damit alle meine Services von MS1 in ihrer Anwendung vorgezeigt werden können. Auch der Test-Client und das zugehörige User-Interface enthält alle Funktionalitäten, sodass MS1 ordnungsgemäß getestet und vorgezeigt werden kann.



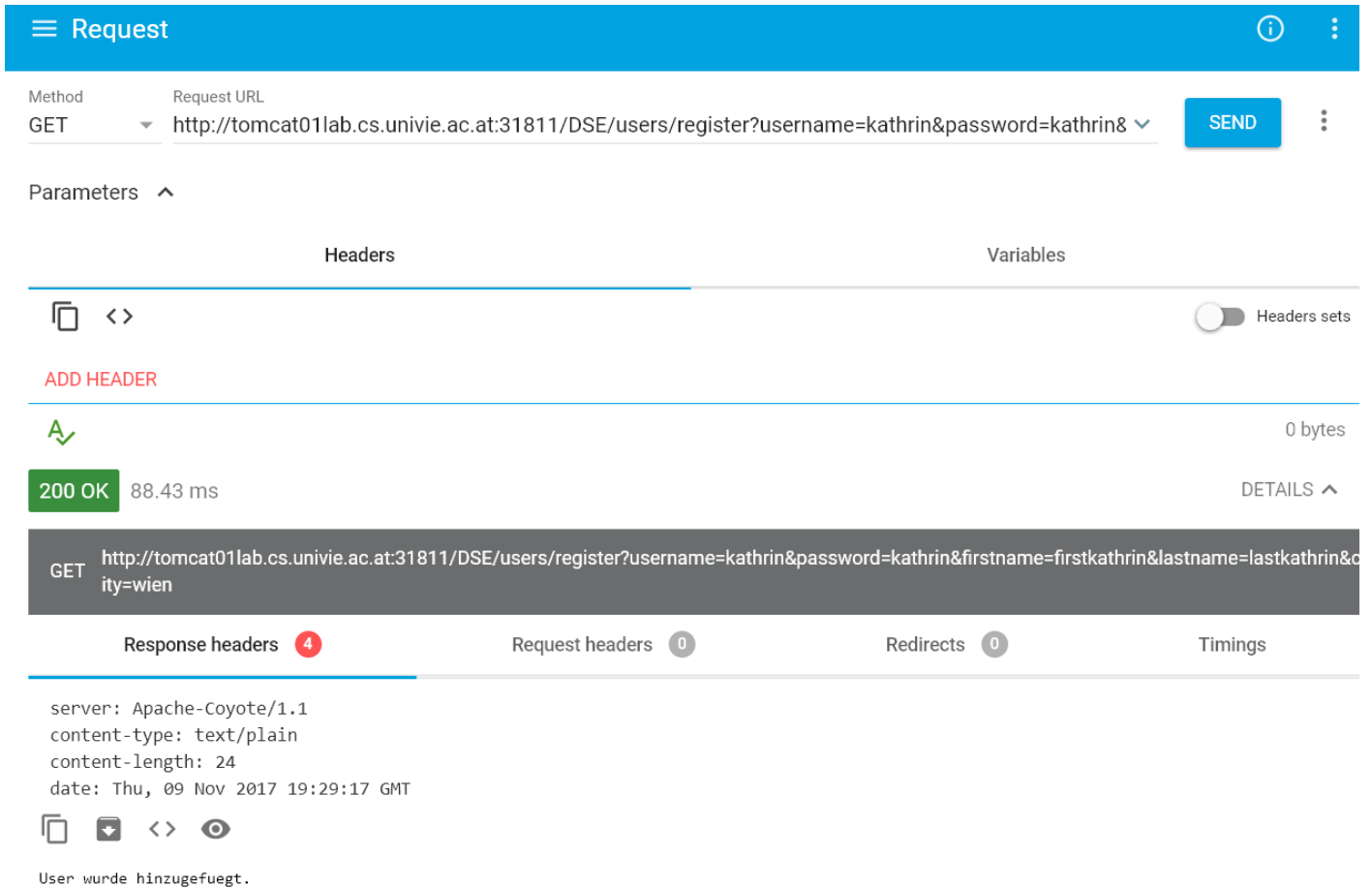
Path	Version	Display Name	Running	Sessions	Commands
/	None specified	Welcome to Tomcat	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/DSE	None specified	DSE	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/ToDoListTestServer	None specified	ToDoListTestServer	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

Abbildung 1 - Deployment der Services am Tomcat-Server

Die Speicherung der User-Daten, die durch den anfragenden Client erzeugt werden, erfolgt in der MySQL-Datenbank a0750881mysql3, die von der Universität Wien zur Verfügung stellt wird. Zu beachten ist hierbei, dass man im UniVPN eingeloggt sein muss, damit Zugang zu der Datenbank gegeben wird. Die Kommunikation erfolgt über HTTP.

Die ordnungsgemäße Durchführung der Services und die Speicherung wurde unter anderem mit der GoogleChrome-App Advanced REST Client getestet und überprüft. Die Parameter werden übergeben und der Service wird unter der Extension .../DSE/users/register?username=... angefragt, und meldet sich mit der HTTP-Response „200 OK“<sup>2</sup> und der Meldung „User wurde hinzugefügt“ zurück. Der untenstehende Screenshot zeigt den Request einer Registrierung eines neuen Users und die Response, die der auf Tomcat deployte Server zurückgeliefert hat, da der User gültig hinzugefügt werden konnte:

<sup>2</sup> Der HTTP-Status-Code steht für „Success“, d.h. der Request war erfolgreich.



The screenshot shows a web browser's developer tools interface. At the top, the 'Request' tab is active. The 'Method' is 'GET' and the 'Request URL' is 'http://tomcat01lab.cs.univie.ac.at:31811/DSE/users/register?username=kathrin&password=kathrin&'. A 'SEND' button is visible. Below the URL bar, the 'Parameters' section is expanded, showing 'Headers' and 'Variables' tabs. The 'Headers' tab is selected, showing a green checkmark and '0 bytes'. The '200 OK' status is displayed with a response time of '88.43 ms'. The 'DETAILS' link is visible. The 'Response headers' section is expanded, showing the following headers: 'server: Apache-Coyote/1.1', 'content-type: text/plain', 'content-length: 24', and 'date: Thu, 09 Nov 2017 19:29:17 GMT'. The 'Request headers' section shows '0' headers. The 'Redirects' section shows '0' redirects. The 'Timings' section is also visible. The 'User wurde hinzugefuegt.' message is displayed at the bottom.

Abbildung 2 - Request von Client an DSE-Server und Erhalt der Response (über HTTP)

Auf der Admin-Seite der MySQL-Datenbank der Universität Wien kann dann eingesehen werden, ob der User ordnungsgemäß angelegt wurde, wie man es für diesen Fall im untenstehenden Screenshot sieht – der User wurde also angelegt und kann sich nun auf der Website, die vom Client gemanagt wird, einloggen:

Server: mitzi02.univie.ac.at » Datenbank: a0750881mysql3 » Tabelle: Users

Anzeigen Struktur SQL Suche Einfügen Exportieren Importieren

☐ Messen [ Inline bearbeiten ] [ Bearbeiten ] [ SQL ]

☐ Alles anzeigen | Anzahl der Datensätze: 25 | Zeilen filtern: Diese Tabelle durchsuchen | Nach

+ Optionen

	userId	username	passw	firstname	lastname	city
<input type="checkbox"/> Bearbeiten <input type="checkbox"/> Kopieren <input type="checkbox"/> Löschen	1	username2	passw2	firstname2	lastname2	city2
<input type="checkbox"/> Bearbeiten <input type="checkbox"/> Kopieren <input type="checkbox"/> Löschen	2	username1	passw1	firstname1	firstname2	city1
<input type="checkbox"/> Bearbeiten <input type="checkbox"/> Kopieren <input type="checkbox"/> Löschen	4	username3	pw3	vn3	nn3	ww3
<input type="checkbox"/> Bearbeiten <input type="checkbox"/> Kopieren <input type="checkbox"/> Löschen	5	testUser	test1	testUser	testUser	Graz
<input type="checkbox"/> Bearbeiten <input type="checkbox"/> Kopieren <input type="checkbox"/> Löschen	6	testUser2	test2	testUser2	testUser2	Wien
<input type="checkbox"/> Bearbeiten <input type="checkbox"/> Kopieren <input type="checkbox"/> Löschen	8	test5	passw5	first5	last5	city5
<input type="checkbox"/> Bearbeiten <input type="checkbox"/> Kopieren <input type="checkbox"/> Löschen	9	albert	albert	firstalbert	firstalbert	wien
<input type="checkbox"/> Bearbeiten <input type="checkbox"/> Kopieren <input type="checkbox"/> Löschen	11	kathrin	kathrin	firstkathrin	lastkathrin	wien

Abbildung 3 - Ansicht der mySQL-Datenbank a0750881mysql3 zur Kontrolle der Registrierung des Users "kathrin"

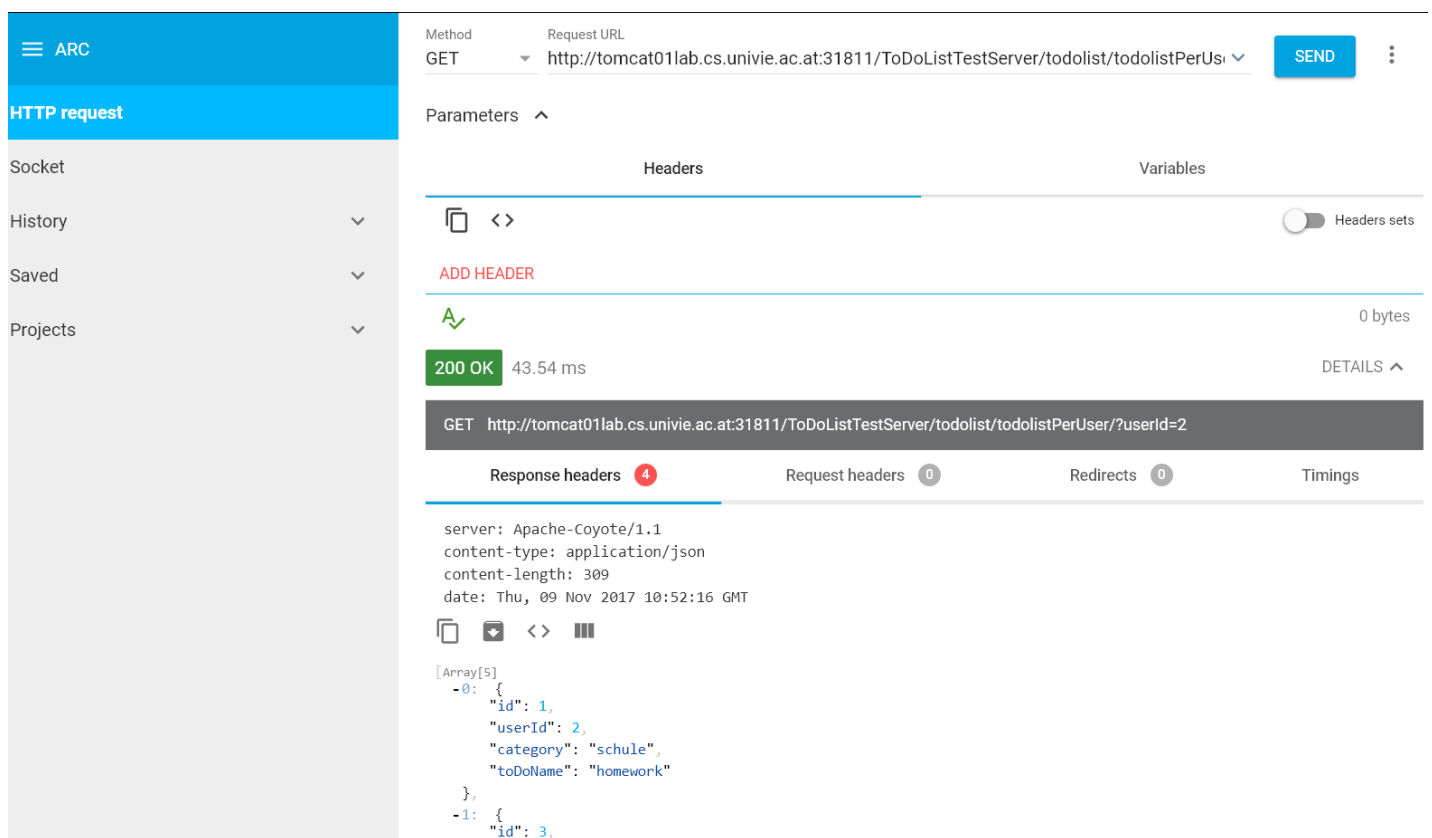
Derzeit umfasst der DSE-Server die user-bezogenen Dienste Login, Registrieren, und Userliste-Anzeigen. Im Sinne einer Erweiterung könnten zukünftig weitere Dienste hinzugefügt werden in der Klasse UserService.java im Package services, die dann über den Client angesprochen werden können. Die Rückgabe der Repräsentation des Users bzw. der Userliste, deren Daten an den Client geliefert werden, erfolgt in Form von JSON, und diese JSON-Daten werden im Client verarbeitet und über das User-Interface angezeigt.

### ToDoListTestServer:

Im Java-Projekt „ToDoListTestServer“ werden exemplarisch serverseitigen Dienste, die auf das Erstellen, Abfragen und Verwalten von To-Do-Listen bezogen sind und gleichzeitig User-Informationen speichern, implementiert. Der ToDoListTestServer kann von Clients – in diesem Fall z.B. DSETestClient angesprochen werden, und die ToDoList-Daten des jeweiligen eingeloggten Users können abgefragt werden. Der ToDoListTestServer wurde von mir ebenfalls auf dem Tomcat-Server der Universität Wien, deployed und ist erreichbar unter: <http://tomcat01lab.cs.univie.ac.at:31811/ToDoListTestServer/>. Ein Client kann, je nach gewählter Link-Extension die entsprechenden Daten aus der Datenbank a0750881mysql4 abfragen und die angebotenen Services nützen. In Abbildung 1 wurde auch gezeigt, dass die Services des ToDoListTestServer auf dem Tomcat-Server der Universität bereitgestellt

wurden. Damit in der Datenbank To-Do-List-Daten gespeichert werden können, ist, wie gesagt, eine Verbindung mit dem UniVPN notwendig. Die Kommunikation mit dem ToDoListTestServer erfolgt über HTTP.

Auch für den ToDoListTestServer kann man die Ausführung der Services und der Datenspeicherungsanfragen unter anderem mittels GoogleChrome-App ARP (Advanced REST Client) testen und überprüfen, ob sowohl die Verbindung ordnungsgemäß aufgebaut wird als auch die Daten korrekt eingetragen bzw. abgefragt werden. Die Parameter werden übergeben und der Service wird unter der Extension .../ToDoListTestServer/todolist/... aufgerufen. Im untenstehenden Beispiel über die Abfrage der To-Do-Liste eines bestimmten Users (hier: User mit userId 2) wird also mittels Get-Request die URL <http://tomcat01lab.cs.univie.ac.at:31811/ToDoListTestServer/todolist/todolistPerUser/?userId=2> abgefragt. Der HTTP-Response-Status-Code ist „200 OK“, was bedeutet, dass der Request in Ordnung war. Der Aufruf war korrekt und so werden für den User mit userId 2 alle To-Do-Listen, die der User bereits angelegt hat, aus der Datenbank abgefragt und an den Client zurückgeliefert. Bezüglich Format der Antwort gibt der ToDoListTestServer ein JSONArray, in dem die ToDo-Objekte in JSONObject-Format erfasst sind, zurück.



The screenshot shows the ARP interface with the following details:

- Method:** GET
- Request URL:** <http://tomcat01lab.cs.univie.ac.at:31811/ToDoListTestServer/todolist/todolistPerUser/?userId=2>
- Parameters:** None
- Headers:** None
- Variables:** None
- Status:** 200 OK (43.54 ms)
- Response headers:**
  - server: Apache-Coyote/1.1
  - content-type: application/json
  - content-length: 309
  - date: Thu, 09 Nov 2017 10:52:16 GMT
- Response body (JSON):**

```
[
  {
    "id": 1,
    "userId": 2,
    "category": "schule",
    "todoName": "homework"
  },
  {
    "id": 3,

```

Abbildung 4 - Request von Client an ToDoListTestServer und Erhalt der Response (über HTTP)

Man kann dann auf der Admin-Seite in der mySQL-Datenbank a0750881mysql4 nachschauen, ob tatsächlich die gespeicherten Daten an den Client weitergegeben wurden (siehe Abbildung 5).

Server: mitzi02.univie.ac.at » Datenbank: a0750881mysql4 » Tabelle: ToDo

Anzeigen Struktur SQL Suche Einfügen Exportieren

☐ Messen [ [Inline bearbeiten](#) ] [ ]

☐ Alles anzeigen | Anzahl der Datensätze: 25 | Zeilen filtern: Diese Tabelle

+ Optionen

				toDold	userId	category	ToDoName
<input type="checkbox"/>	Bearbeiten	Kopieren	Löschen	1	2	schule	homework
<input type="checkbox"/>	Bearbeiten	Kopieren	Löschen	3	2	schule	homework
<input type="checkbox"/>	Bearbeiten	Kopieren	Löschen	4	2	schule	homework
<input type="checkbox"/>	Bearbeiten	Kopieren	Löschen	5	2	sport	laufen gehen
<input type="checkbox"/>	Bearbeiten	Kopieren	Löschen	6	3	einkaufen	milch
<input type="checkbox"/>	Bearbeiten	Kopieren	Löschen	7	3	einkaufen	brot
<input type="checkbox"/>	Bearbeiten	Kopieren	Löschen	8	1	uni	blabla
<input type="checkbox"/>	Bearbeiten	Kopieren	Löschen	9	2	uni	lernen

Abbildung 5 - Ansicht der mySQL-Datenbank a0750881mysql4 bzgl. gespeicherter ToDos von userId 2

Der ToDoListTestServer bietet im Moment die Dienste ToDo-Hinzufügen und ToDoListe-Anzeigen (für den eingeloggten User). Es können zukünftig noch mehr Dienste hinzugefügt werden über eine Erweiterung der Klasse ToDoListService.java im Package services. Für die Rückgabe der Daten wird als Format JSON verwendet, das im User-Interface auf der Client-Seite verarbeitet wird und dann entsprechend auf HTML/JSP-Seiten angezeigt wird.



**DSETestClient:**

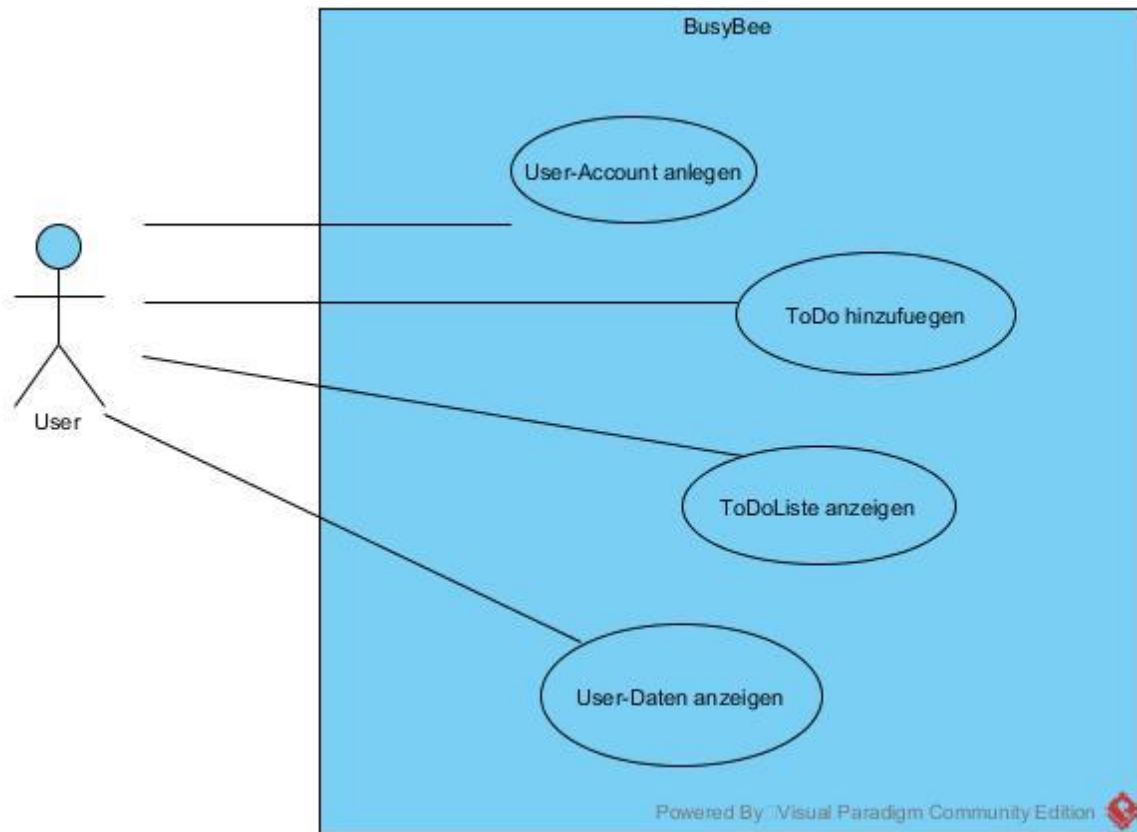
Im DSETestClient befindet sich das User-Interface, über das User sich registrieren können und sich einloggen können, wonach sie dann die ToDoList-Dienste und die allgemeinen User-Datenverwaltungsdienste nutzen können.<sup>3</sup> Das User-Interface besteht aus HTML/JSP-Seiten, über die die jeweiligen Funktionalitäten für den User zugänglich gemacht werden, und in denen bei erfolgreichem Login der User als Session-Attribut gesetzt wird. Im Package controller befinden sich die Servlets, die die Daten aus dem User-Interface entgegennehmen und an den jeweiligen Server leiten, der auf die Datenbank zugreifen kann. Die Servlets nehmen auch die entsprechende Response vom Server entgegen und verarbeiten sie, bzw. leiten die entsprechenden Meldungen weiter, sodass im User-Interface die korrekte Meldung für den User bzw. die entsprechenden Daten angezeigt werden. Falls der DSETestClient in den Servlets als Response vom Server Daten in JSON-Format empfängt (wie z.B. im Falle der User-Liste oder der To-Do-Listen, dann werden die Daten entsprechend umgewandelt und oder sofort an die JSP-Files weitergegeben und dort wird die Information entsprechend verarbeitet und angezeigt. Die Kommunikation findet in Form von HTTP-Requests und HTTP-Responses statt.

---

<sup>3</sup> Anmerkung: Das User-Interface wird im Kapitel „Testen & Präsentation“ gezeigt.

**Use-Case-Diagramm von BusyBee:**

In der derzeitigen Implementierung werden folgende Services über mein Test-User-Interface angeboten:

**Anmerkungen:**

- Der Login wird nicht als Use-Case angeführt, da er laut VO SWE (Prof. Benkner) außerhalb des Use-Case-Spektrums liegt.
- Der Use-Case „Userliste anzeigen“ kann auch als Service aufgerufen werden, allerdings derzeit nicht über das Test-User-Interface.

Die folgenden **Klassendiagramme** dienen zur Übersicht über die Struktur meiner drei Java-Projekte:

**Klassendiagramm des Java-Projekts DSE (Server-Seite):**

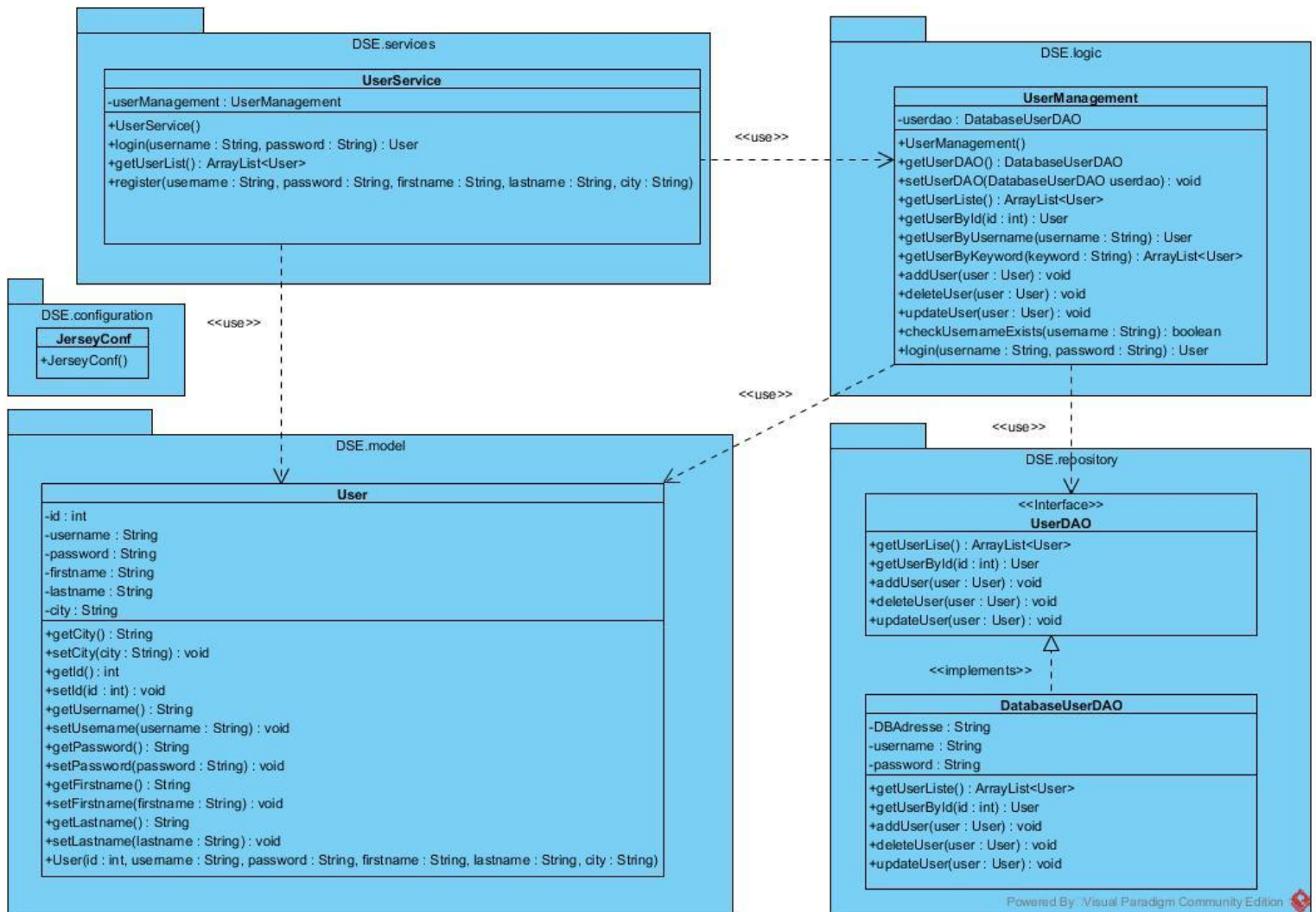


Abbildung 6 - Klassendiagramm des Java-Projekt DSE (user-bezogene Services)

### Klassendiagramm des Java-Projekts ToDoListTestServer (Server-Seite):

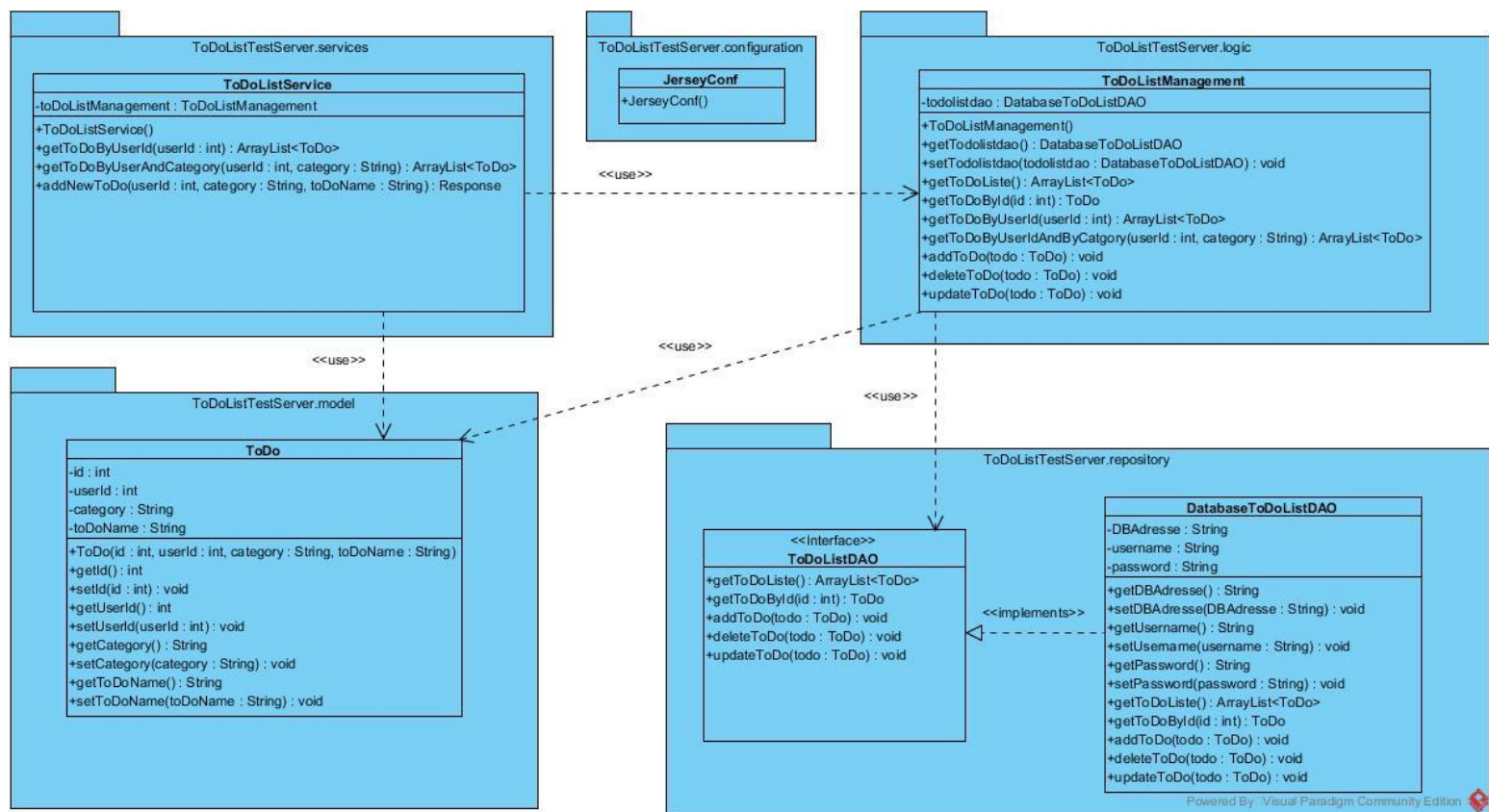


Abbildung 7 - Klassendiagramm des Java-Projekt ToDoListTestServer (todolist-bezogene Dienste)

**Klassendiagramm des Java-Projekts DSETestClient (Client-Seite):**

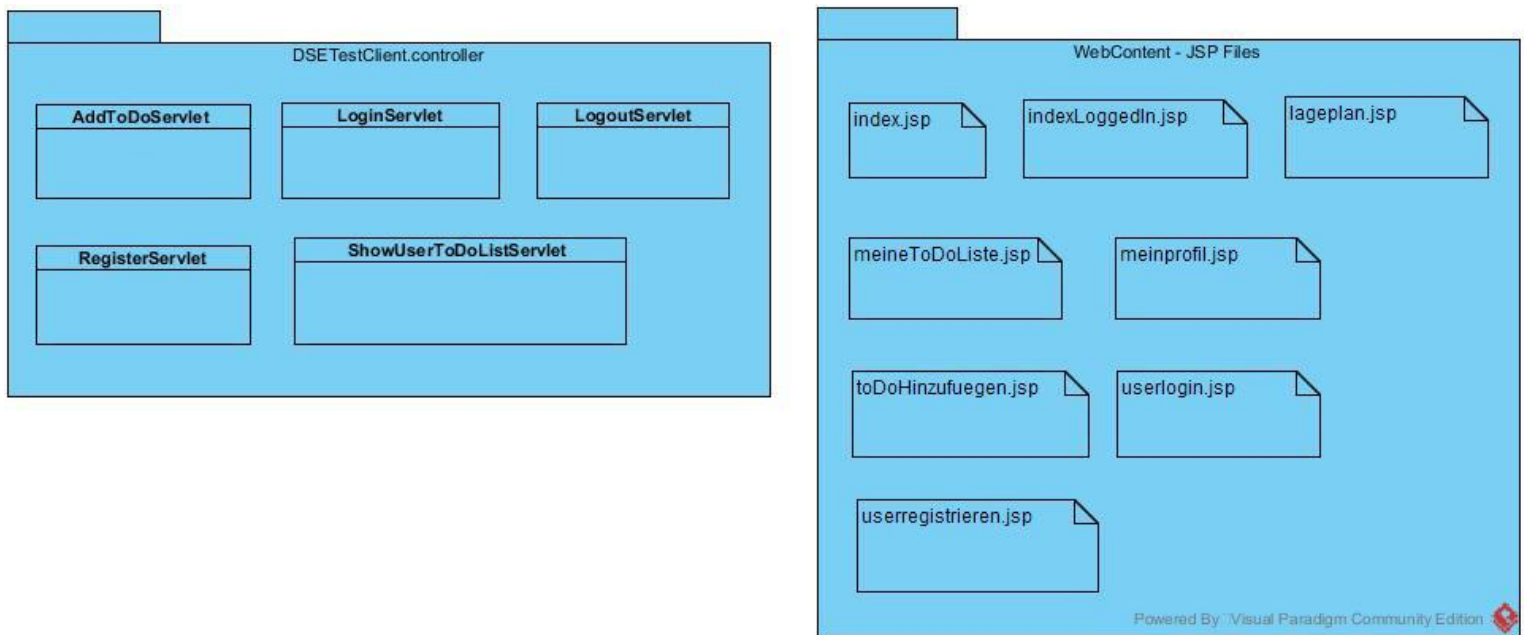
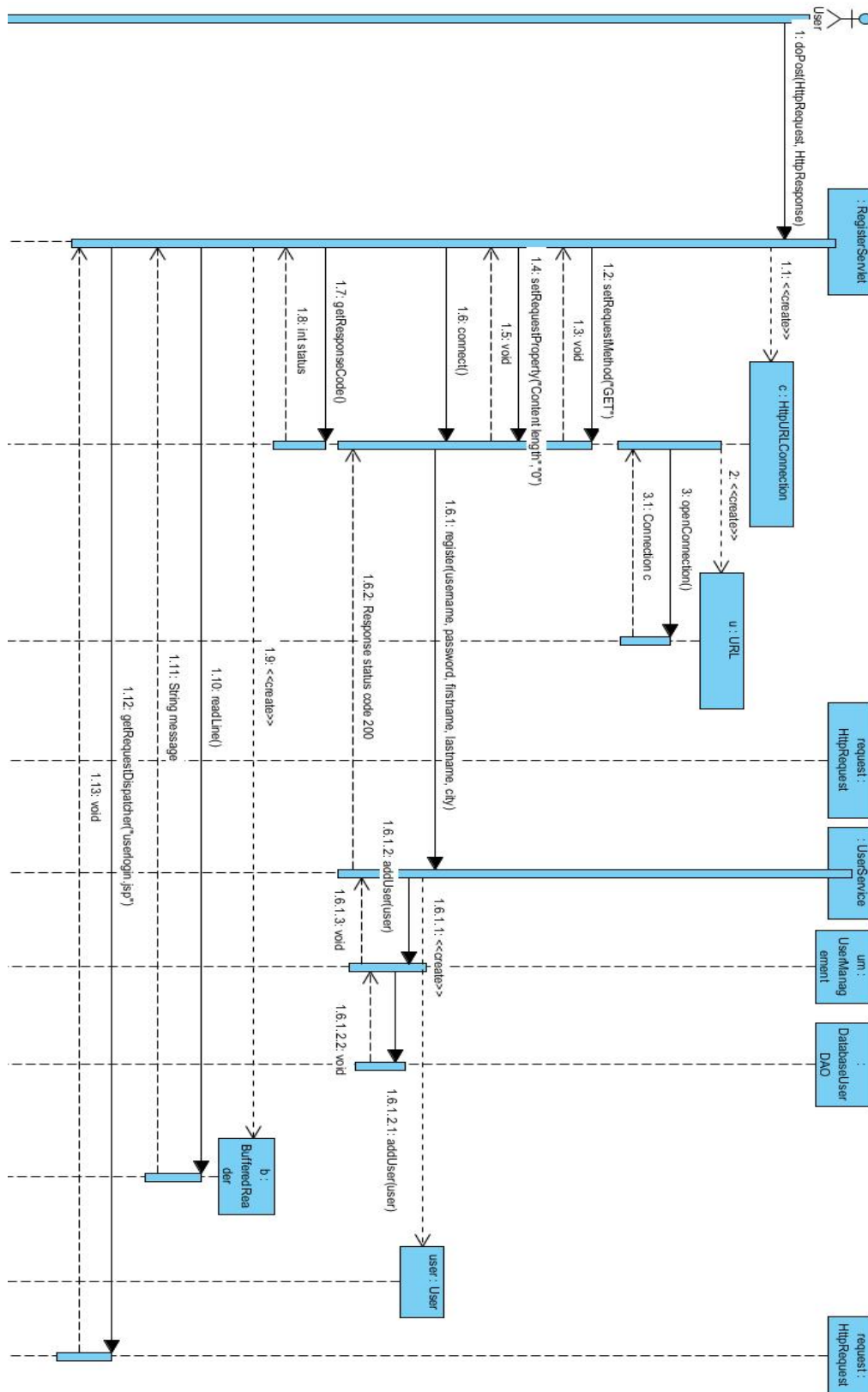


Abbildung 8 - Klassendiagramm des Java-Projekt DSETestClient (client-seitige Aufrufe & User-Interface)

Das untenstehende (aus Platzgründen vereinfachte) **Sequenzdiagramm** dient zur Veranschaulichung eines Beispiels der Abfolge der Aufrufe, die gemacht werden, wenn ein neuer User gültig registriert wird:



**Wie schaut das externe Interface des Service (MS1) aus und wie kann es von meinen Kollegen angesprochen werden?**

Die Services (MS1 und als Dummy-Implementierung MS2) können über die folgenden URLs angefragt werden. Die Kommunikation läuft über HTTP-Requests und -Responses. Zurückgeliefert werden entweder der Status-Code oder Daten im JSON-Format.

Hier eine Liste von Beispielaufrufen, über die die Services, die am Tomcat-Uni-Server deployed wurden, angesprochen werden können<sup>4</sup>:

**Services des DSE-Server (der user-bezogene Dienste anbietet):**

- **User registrieren:**

<http://tomcat01lab.cs.univie.ac.at:31811/DSE/users/register?username=informatikstudent&password=informatikstudent&firstname=michael&lastname=test&city=Salzburg>

(Parameter username, password, firstname, lastname und city sind QueryParam)

- **User einloggen:**

<http://tomcat01lab.cs.univie.ac.at:31811/DSE/users/login>

(Parameter username und password werden über CookieParam mitgeschickt.)

- **User-Liste abfragen:**

<http://tomcat01lab.cs.univie.ac.at:31811/DSE/users/userlist>

**Services des ToDoListTestServer:**

- **Neues To-Do anlegen:**

<http://tomcat01lab.cs.univie.ac.at:31811/ToDoListTestServer/todolist/newToDo?userId=2&category=softwareengineering&toDoName=hausuebung1-machen>

(Parameter userId, category und toDoName sind QueryParam)

- **To-Do-Liste eines registrierten Users anzeigen:**

<http://tomcat01lab.cs.univie.ac.at:31811/ToDoListTestServer/todolist/todolistPerUser?userId=2>

(Parameter userId ist QueryParam)

---

<sup>4</sup> Anmerkung: Es können zur Abfrage auch die statischen IP-Adressen über OpenVPN genutzt werden.

## 2. Überlegungen zum Service Deployment

Wie bereits erwähnt, habe ich die Java-Projekte DSE (das die user-bezogenen Services anbietet) und ToDoListTestServer auf dem Tomcat-Server der Universität Wien deployed.

Die Services sind dort erreichbar unter Portnummer 31811:

<http://tomcat01lab.cs.univie.ac.at:31811/DSE/...>

<http://tomcat01lab.cs.univie.ac.at:31811/ToDoListServer/...>

Die drei Punkte (...) stehen für die jeweiligen Service, den man ansprechen möchte. Die verfügbaren Services wurden bereits auf der vorigen Seite vorgestellt und es wurde beschrieben, was sie genau machen.

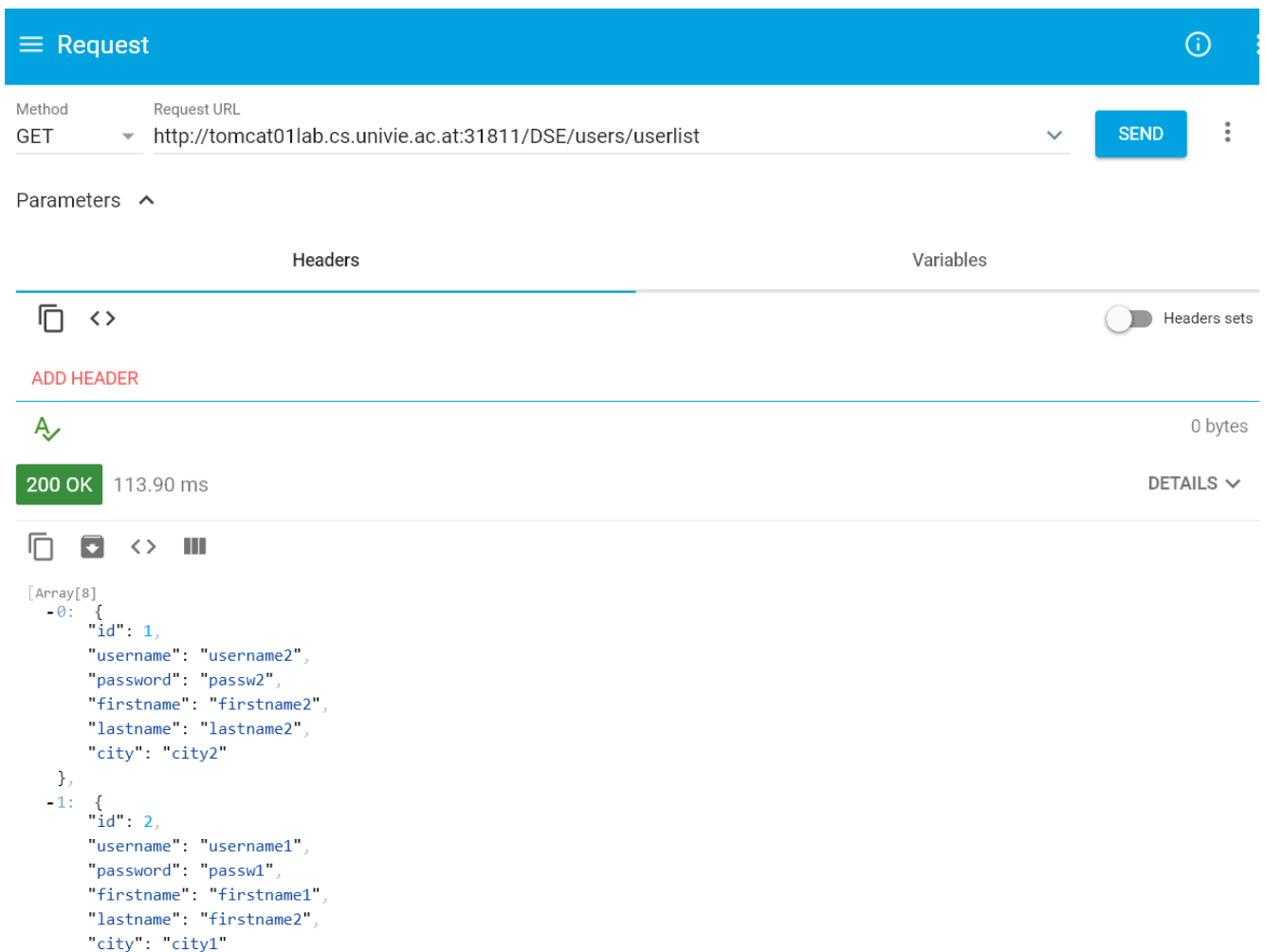
Die Services können auch über meine statische IP-Adresse (10.103.101.5), die von OpenVPN generiert wurde, angesprochen werden.



### 3. Testen & Präsentation

Die Services der bisherigen Implementierung können, wie bereits in den vorherigen Kapiteln erwähnt, beispielsweise mittels ARC-App (Advanced REST Client) von GoogleChrome getestet werden. Damit kann überprüft werden, ob der Request ordnungsgemäß durchgeführt wird, da man den HTTP-Status-Code bzw. weitere Meldungen empfangen kann, und gegebenenfalls werden die angefragten Daten im JSON-Format zurückgegeben. Diese Testmöglichkeit wurde bereits im Rahmen der Abbildung 2 gezeigt, wo der neue User „kathrin“ registriert wurde.

Die Abbildung 9 zeigt beispielsweise den Aufruf des Dienstes „userlist“, der die Liste aller registrierten User übermittelt, was in Form eines JSONArray mit den Usern als JSONObjects beim anfragenden Client ankommt.



The screenshot shows the ARC-App interface with the following details:

- Method:** GET
- Request URL:** http://tomcat01lab.cs.univie.ac.at:31811/DSE/users/userlist
- Parameters:** None
- Headers:** None
- Variables:** None
- Status:** 200 OK
- Response Time:** 113.90 ms
- Response Body:**

```
[Array[8]
  -0: {
    "id": 1,
    "username": "username2",
    "password": "passw2",
    "firstname": "firstname2",
    "lastname": "lastname2",
    "city": "city2"
  },
  -1: {
    "id": 2,
    "username": "username1",
    "password": "passw1",
    "firstname": "firstname1",
    "lastname": "firstname2",
    "city": "city1"
  }
]
```

Abbildung 9 - Request an den Service "userlist" am DSE Server

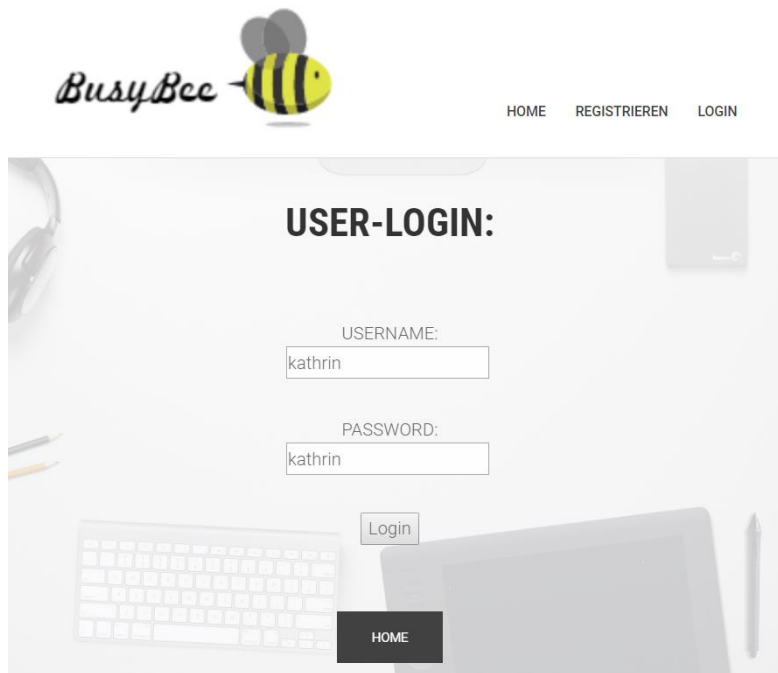
Als weitere Möglichkeit, mit der die Funktionalität der bisher von mir implementierten Services gezeigt werden kann, habe ich einen eigenen DSETestClient angelegt, der ein User-Interface anbietet, über das die von mir implementierten Services aufgerufen werden mittels Request an die DSE (für user-bezogene Dienste) und ToDoListTestClient (für todolist-bezogene Dienste). Die Kommunikation wird mittels HTTP durchgeführt und es werden Nachrichten ausgetauscht. Die ToDoList-Implementierung wurde hier nur in ihrer Basisversion umgesetzt, da MS1 das Ziel war, aber es zum nahtlosen funktionieren auch Dienste, die sich auf die To-Do-Listen beziehen, braucht. Mittels dieser Kombination an implementierten Komponenten kann das Testen des MS1 isoliert und unabhängig von der Implementierung der Team-Kollegen sichergestellt werden. D.h. auch in Situationen, wo die Funktionalitäten der anderen Services (MS2, MS3) nicht vorhanden sind, können meine Services für MS1 getestet werden.

### User-Interface des DSETestClient:

1. Ansicht des index.jsp (Hauptseite, wenn User nicht eingeloggt ist):



2. User-Login-Ansicht (Eingabe von username „kathrin“ und password „kathrin“):



The screenshot shows the BusyBee login interface. At the top left is the BusyBee logo. To its right are navigation links: HOME, REGISTRIEREN, and LOGIN. The main content area has a background image of a desk with a keyboard, mouse, and headphones. It features the heading "USER-LOGIN:" followed by two input fields. The first field is labeled "USERNAME:" and contains the text "kathrin". The second field is labeled "PASSWORD:" and also contains "kathrin". Below these fields is a "Login" button. At the bottom center of the main area is a dark "HOME" button.

3. Ansicht des Hauptmenüs, wenn ein registrierter User sich eingeloggt hat:



The screenshot shows the BusyBee main menu after a successful login. The BusyBee logo is at the top left. Navigation links at the top include HOME, MEIN PROFIL, MEINE TODOLISTE, TODO HINZUFUEGEN, and LOGOUT. The main content area has a background image of a desk with a keyboard, mouse, and headphones. It displays the heading "WILLKOMMEN BEI BUSYBEE TO DO LIST PLANNER" in large blue letters. Below this, it says "LOGIN ERFOLGREICH!". Further down, a welcome message reads: "WIR FREUEN UNS UEBER IHREN BESUCH BEI BUSYBEE! BEI UNS KOENNEN SIE TO-DO-LISTEN SCHNELL UND EINFACH ANLEGEN UND VERWALTEN."

4. Anzeige von „Mein Profil“:



**BusyBee**

HOME MEIN PROFIL MEINE TODOLISTE TODO HINZUFUEGEN LOGOUT

## MEINE PROFILDATEN: (ID 11)

USERNAME:  
KATHRIN

PASSWORT:  
KATHRIN

VORNAME:  
FIRSTKATHRIN

NACHNAME:  
LASTKATHRIN

5. Erfassen eines neuen To-Do („ToDo Hinzufuegen“):



## NEUES TODO ERFASSEN:

KATEGORIE:  
WorksheetDSE

TO DO NAME:  
Task4

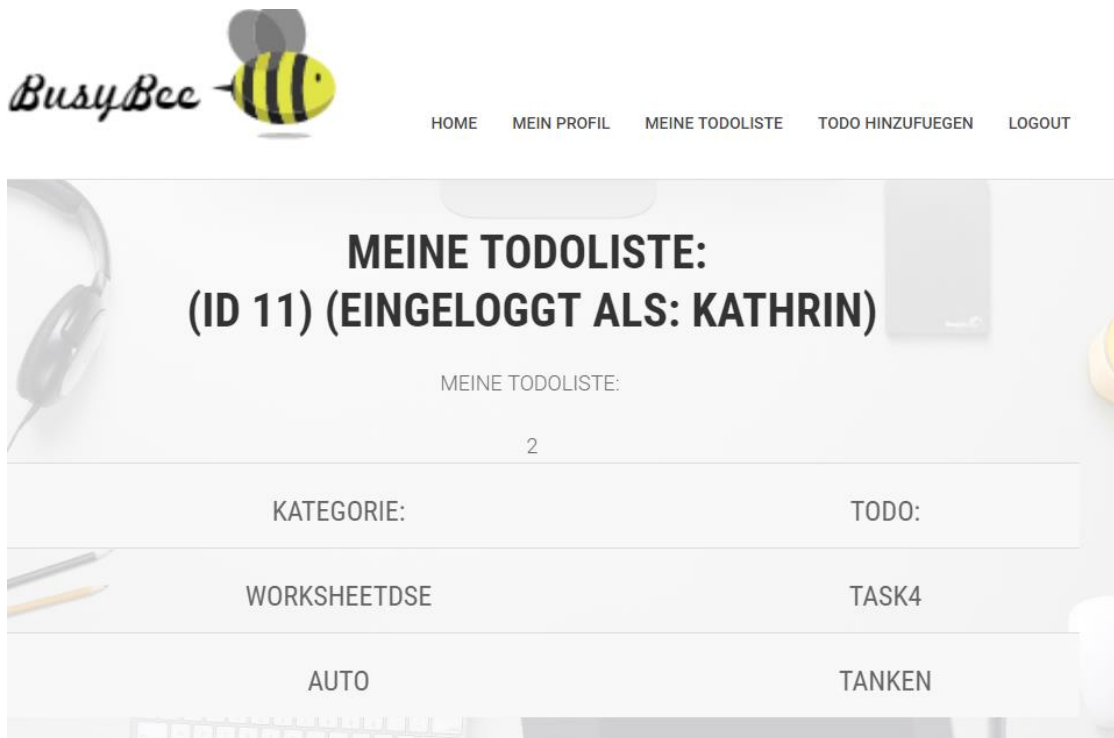
ToDo anlegen

6. Rückmeldung, wenn To-Do korrekt erfasst wurde („ToDo Hinzufuegen“):



The screenshot shows the 'NEUES TODO ERFASSEN' (New To-Do Create) form. At the top left is the 'BusyBee' logo. The navigation bar includes 'HOME', 'MEIN PROFIL', 'MEINE TODOLISTE', 'TODO HINZUFUEGEN', and 'LOGOUT'. The form has two input fields: 'KATEGORIE:' and 'TO DO NAME:'. Below these is a 'ToDo anlegen' button. A message at the bottom states 'TODO WURDE HINZUGEUEGT'.

7. Alle To-Do's des eingeloggten Users anzeigen („Meine ToDoListe“):



The screenshot shows the 'MEINE TODOLISTE' (My To-Do List) page. At the top left is the 'BusyBee' logo. The navigation bar includes 'HOME', 'MEIN PROFIL', 'MEINE TODOLISTE', 'TODO HINZUFUEGEN', and 'LOGOUT'. The main heading is 'MEINE TODOLISTE: (ID 11) (EINGELOGGT ALS: KATHRIN)'. Below this is a sub-heading 'MEINE TODOLISTE:' followed by the number '2'. The list contains three items:

KATEGORIE:	TODO:
WORKSHEETDSE	TASK4
AUTO	TANKEN

8. Zustand nach Logout des Users (wieder bei index.jsp):



## 4. Derzeitiger Status der Implementierung

Der derzeitige Status der Implementierung wurde bereits in den vorherigen Kapiteln dieses Status Update beschrieben. Weiters ist der Code im Ordner ms1 abrufbar und ausführbar (siehe gitlab submission folders). Alle Java-Source-Files sind auch in Form von Javadoc-Kommentaren dokumentiert worden.

Für das Projekt wurde JAX RS 2.20 gewählt und es wurden/werden RESTful Webservices implementiert. Für das User-Interface im Client wurden JSP (JavaServerPages) Files verwendet und die dazugehörigen Servlets implementiert.

Bisher wurden die für MS1 geforderten Services implementiert und mit ARC (Advanced REST Client) und mithilfe des selbst entwickelten DSETestClient inkl. User-Interface getestet. Damit wird gewährleistet, dass die Funktionalitäten meines Projektteils auch unabhängig von den Implementierungen der MS2 und MS3 präsentiert und getestet werden können.

Was noch gemacht werden muss, ist das Zusammenschließen meines Projektteils mit der tatsächlichen Implementierung von MS2 und dem User-Interface von MS3, das die Team-Kolleginnen implementieren. Hierbei muss auch noch entschieden werden, wie das gemeinsame User-Interface schließlich ausschauen sollte, und wie beim Zugriff auf die To-Do-Listen die Authentifizierung des zugreifenden Users erfolgt. Außerdem muss diesbezüglich das Deployment der Services entweder über den Tomcat-Server der Universität oder über das OpenVPN getestet werden.

Weiters können noch Unit-Tests für die Klassen in den Packages repository, model und logic in den Java-Projekten DSE (für die user-bezogenen Dienste) und ToDoListTestServer implementiert werden.

## 5. Anhang

### Nachweis des durchgeführten Verbindungsaufbaues mit OpenVPN:

