



## REST – Übung 3

Implementierung einer REST-Schnittstelle

---



## Group 46

- Carolin Schwarz, 371802
- Fedor Vitkovskiy, 386458
- Robert Koch, 386471
- Jia Fug Liu, 382333



# Content

## REST-Schnittstellenbeschreibung und –Implementierung

- Beschreibung der Schnittstelle
- Models
- Controllers
- Repositories

## Client

- Beschreibung und Schnittstelle SMeMu
- Implementierung
- Berechnung der Stromstärke für Smartmeter mit ID=...2
- Screenshots



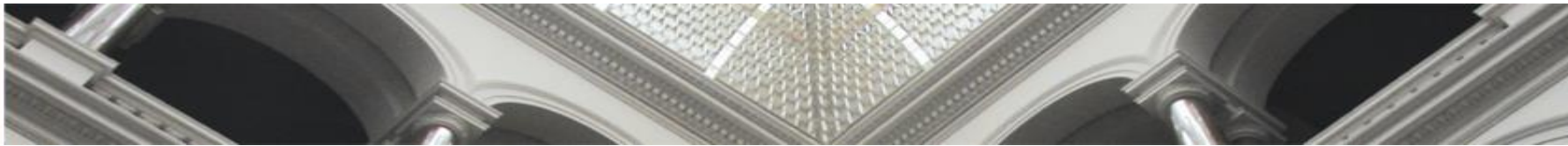
## REST-Schnittstellenbeschreibung

	GET	PUT	POST	DELETE
/smartmeter	✓	✗	✓	✗
/smartmeter/{smartmeter}	✓	✗	✓	✗
/smartmeter/{smartmeter}/record	✓	✗	✓	✗
/smartmeter/{smartmeter}/record/{metric}	✓	✗	✗	✗



## Models

- Smartmeter: Repräsentiert ein Smartmeter. Verfügt über eine Liste von Messgrößen (Measurands) und von Ablesungen (Records). Ein Smartmeter steht in einer Many-to-Many Beziehung zu den Measurands und in einer One-to-Many Beziehung zu den Records.
- Measurand: Repräsentiert eine Messgröße. Verfügt über eine Liste von Smartmeters und steht zu diesen in einer Many-to-Many Beziehung.
- Record: Repräsentiert eine Ablesung. Steht in einer Many-to-One Beziehung sowohl zu den Smartmeters als auch zu den Measurands.



# Smartmeter

```
1 package de.tub.ise.anwsys.models;
2
3 import java.io.Serializable;
4
11
12 @Entity
13 public class SmartMeter implements Serializable {
14
15     private static final long serialVersionUID = -6640481949420444264L;
16
17     private String name;
18     private List<Measurand> measurand = new ArrayList<Measurand>();
19     private List<Record> record = new ArrayList<Record>();
20
21     protected SmartMeter() {
22         // empty constructor required by JPA
23     }
24
25     public SmartMeter(String name) {
26         this.name = name;
27     }
28
29     @Id
30     public String getName() {
31         return this.name;
32     }
33
34     public void setName(String name) {
35         this.name = name;
36     }
```

```
37
38     @ManyToMany
39     public List<Measurand> getMeasurand() {
40         return measurand;
41     }
42
43     public void setMeasurand(List<Measurand> measurand) {
44         this.measurand = measurand;
45     }
46
47     @OneToMany(mappedBy = "smartmeter")
48     public List<Record> getRecord() {
49         return record;
50     }
51
52     public void setRecord(List<Record> record) {
53         this.record = record;
54     }
55
56     @Override
57     public String toString() {
58         return this.name;
59     }
60
61 }
```



# Measurand

```

1 package de.tub.ise.anwsys.models;
2
3 import java.io.Serializable;
4
10
11 @Entity
12 public class Measurand implements Serializable {
13
14     private static final long serialVersionUID = 3501450469684231867L;
15
16     private String metricId;
17     private String metricText;
18     private List<SmartMeter> smartmeter = new ArrayList<SmartMeter>();
19
20     protected Measurand() {
21         // empty constructor required by JPA
22     }
23
24     public Measurand(String metricId, String metricText, SmartMeter smartmeter) {
25         this.metricId = metricId;
26         this.metricText = metricText;
27         this.smartmeter.add(smartmeter);
28     }
29
30     @Id
31     public String getMetricId() {
32         return this.metricId;
33     }

```

```

35     public void setMetricId(String name) {
36         this.metricId = name;
37     }
38
39     public String getMetricText() {
40         return metricText;
41     }
42
43     public void setMetricText(String metricText) {
44         this.metricText = metricText;
45     }
46
47     @ManyToMany
48     public List<SmartMeter> getSmartmeter() {
49         return smartmeter;
50     }
51
52     public void setSmartmeter(List<SmartMeter> smartmeter) {
53         this.smartmeter = smartmeter;
54     }
55
56     public void addToSmartMeterList(SmartMeter smartmeter) {
57         this.smartmeter.add(smartmeter);
58     }
59
60     @Override
61     public String toString() {
62         return this.metricId;
63     }
64
65 }

```

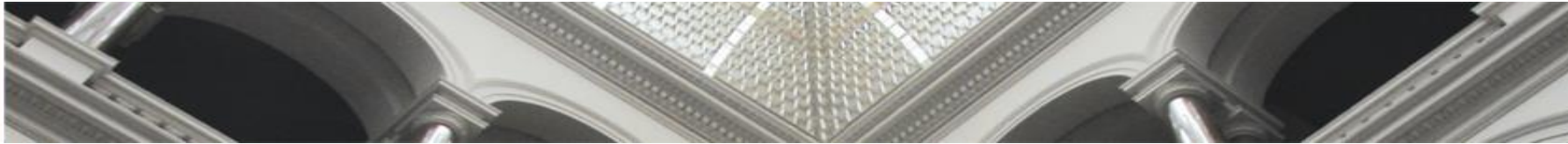


## Record

```
12 @Entity
13 public class Record implements Serializable {
14
15     private static final long serialVersionUID = 2831438593938521629L;
16
17     private int id;
18     private Measurand measurand;
19     private double value;
20     private SmartMeter smartmeter;
21     private int time;
22
23     protected Record() {
24         // empty constructor required by JPA
25     }
26
27     public Record(Measurand measurand, double value, SmartMeter smartmeter, int time) {
28         this.measurand = measurand;
29         this.value = value;
30         this.smartmeter = smartmeter;
31         this.time = time;
32     }
33
34     @Id
35     @GeneratedValue(strategy = GenerationType.IDENTITY)
36     public int getId() {
37         return this.id;
38     }
39
40     public void setId(int id) {
41         this.id = id;
42     }
```

```
44     @ManyToOne
45     @JoinColumn(name = "measurand")
46     public Measurand getMeasurand() {
47         return measurand;
48     }
49
50     public void setMeasurand(Measurand measurand) {
51         this.measurand = measurand;
52     }
53
54     public double getValue() {
55         return value;
56     }
57
58     public void setValue(double value) {
59         this.value = value;
60     }
61
62     @ManyToOne
63     @JoinColumn(name = "smartmeter")
64     public SmartMeter getSmartmeter() {
65         return smartmeter;
66     }
67
68     public void setSmartmeter(SmartMeter smartmeter) {
69         this.smartmeter = smartmeter;
70     }
71
72     public int getTime() {
73         return time;
74     }
75
76     public void setTime(int time) {
77         this.time = time;
78     }
79
80     @Override
81     public String toString() {
82         return this.id + " ";
83     }
```





## Controllers

SmartmeterController: Bietet eine GET-Methode("/smartmeter") zur Ausgabe einer Liste aller Smartmeter an und eine POST-Methode("/smartmeter"), um neue Smartmeter zu erstellen.

MeasurandController: Bietet eine GET-Methode("/smartmeter/{smartmeter}") zur Ausgabe einer Map an, die die Zuordnung von Measurands zu Smartmetern angibt und eine POST-Methode("/smartmeter/{smartmeter}"), die neue Measurands erstellt oder zu einem Smartmeter hinzufügt.

RecordController: Bietet eine GET-Methode("/smartmeter/{smartmeter}/record" ) an, die eine Map zurückgibt, welche die Records eines Smartmeters nach Zeit gruppiert zurückgibt.  
Zudem eine POST-Methode("/smartmeter/{smartmeter}/record"), welche neue Records erstellt.  
Eine weitere GET-Methode("/smartmeter/{smartmeter}/record/{metric}") gibt eine Map der Records eines Smartmeters für einen bestimmten Measurand gruppiert nach Zeit zurück.

## SmartmeterController(1)

```
17 @RestController
18 public class SmartMeterController {
19
20     @Autowired
21     SmartMeterRepository repository;
22
23     /**
24      * gets a list of all smart meters
25      *
26      * @return
27      */
28     @RequestMapping(method = RequestMethod.GET, path = "/smartmeter")
29     public List<String> getAllSmartMeters() {
30         List<SmartMeter> list = repository.findAll();
31         List<String> nameList = new ArrayList<String>();
32         for (SmartMeter sm : list)
33             nameList.add(sm.getName());
34         return nameList;
35     }
}
```

## SmartmeterController(2)

```
37- /**
38  * creates new smart meters
39  *
40  * @param smartmeter
41  * @throws JSONException
42  */
43- @RequestMapping(method = RequestMethod.POST, path = "/smartmeter")
44  public void registerNewSmartMeter(@RequestParam(value = "smartmeter") JSONObject smartmeter) throws JSONException {
45      for (int i = 0; i < smartmeter.getJSONArray("meters").length(); i++) {
46          String name = smartmeter.getJSONArray("meters").get(i).toString();
47          SmartMeter sm = new SmartMeter(name);
48          repository.save(sm);
49      }
50  }
51
52 }
```

## MeasurandController(1)

```
21 @RestController
22 public class MeasurandController {
23
24     @Autowired
25     MeasurandRepository repository;
26     @Autowired
27     SmartMeterRepository smRepository;
28
29     /**
30      * gets a map of all measurands of a smart meter
31      *
32      * @param smartmeter
33      * @return
34      */
35     @RequestMapping(method = RequestMethod.GET, path = "/smartmeter/{smartmeter}")
36     public Map<String, String> getAllMeasurands(@PathVariable String smartmeter) {
37         HashMap<String, String> map = new HashMap<String, String>();
38         List<Measurand> list = repository.findBySmartmeter(smRepository.findByName(smartmeter));
39         for (Measurand m : list) {
40             map.put(m.getMetricId(), m.getMetricText());
41         }
42         return map;
43     }
44 }
```

## MeasurandController(2)

```
45- /**
46   * creates a new measurand or adds a measurand to a smart meter
47   *
48   * @param smartmeter
49   * @param measurand
50   * @throws JSONException
51   */
52- @RequestMapping(method = RequestMethod.POST, path = "/smartmeter/{smartmeter}")
53   public void createMeasurand(@PathVariable String smartmeter, @RequestParam(value = "measurand") JSONArray measurand)
54       throws JSONException {
55       for (int i = 0; i < measurand.length(); i++) {
56           String metricId = measurand.getJSONObject(i).getString("metricId").toString();
57           String metricText = measurand.getJSONObject(i).getString("metricText").toString();
58           Measurand m = repository.findByMetricId(metricId);
59           SmartMeter sm = smRepository.findByName(smartmeter);
60           if (m == null)
61               m = new Measurand(metricId, metricText, sm);
62           else {
63               if (!m.getSmartmeter().contains(sm))
64                   m.addToSmartMeterList(sm);
65           }
66           repository.save(m);
67       }
68   }
```

## RecordController(1)

```
26 @RestController
27 public class RecordController {
28
29     @Autowired
30     RecordRepository repository;
31
32     @Autowired
33     MeasurandRepository measurand_repository;
34
35     @Autowired
36     SmartMeterRepository smartmeter_repository;
37
38     /**
39      * returns a map of a specific smart meter which is grouped by time
40      *
41      * @param smartmeter
42      * @return
43      * @throws JSONException
44      */
45     @RequestMapping(method = RequestMethod.GET, path = "/smartmeter/{smartmeter}/record")
46     public Map<Integer, List<Map<String, Double>>> getMapOfRecord(@PathVariable String smartmeter)
47         throws JSONException {
48         // gets a list of all records of this smart meter
49         List<Record> list = repository.findBySmartmeter(smartmeter_repository.findByName(smartmeter));
50     }
51 }
```

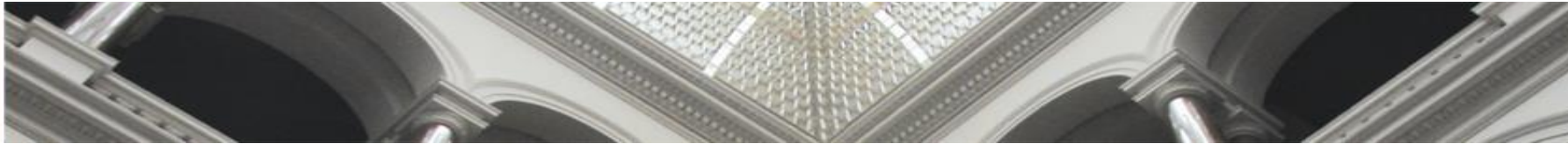


## RecordController(2)

```
48 // gets the latest record
49 Optional<Record> latestRecord = list.stream()
50   .max((r1, r2) -> Integer.compare(r1.getTime(), r2.getTime()));
51 int latestTime = latestRecord.get().getTime();
52 // gets all records with the same time stamp
53 Map<Integer, List<Record>> newMap = list.stream()
54   .filter(r -> r.getTime() == latestTime)
55   .collect(Collectors.groupingBy(Record::getTime));
56 // initializes the map that is going to be returned with a time as key
57 Map<Integer, List<Map<String, Double>>> map = new HashMap<Integer, List<Map<String, Double>>>();
58 // collects all records that have the same time
59 List<Record> allRecordsOfSameTime = newMap.get(latestTime);
60 // initializes a value list of the map that is going to be returned
61 List<Map<String, Double>> toAdd = new ArrayList<Map<String, Double>>();
62 // iterates over the list of all records that have the same time
63 for (Record r2 : allRecordsOfSameTime) {
64     // creates a new value for the list above
65     Map<String, Double> mapValue = new HashMap<String, Double>();
66     mapValue.put(r2.getMeasurand().getMetricId(), r2.getValue());
67     // adds the map to the list above
68     toAdd.add(mapValue);
69 }
70 // puts the list toAdd into the map that is going to be returned
71 map.put(latestTime, toAdd);
72 // returns a map
73 return map;
74 }
```

## RecordController(3)

```
76- /**
77-  * creates a new record
78-  *
79-  * @param smartmeter
80-  * @param record
81-  * @throws JSONException
82-  */
83- @RequestMapping(method = RequestMethod.POST, path = "/smartmeter/{smartmeter}/record")
84- public void createNewRecord(@PathVariable String smartmeter, @RequestParam(value = "record") JSONArray record)
85-     throws JSONException {
86-     // finds the smart meter by name
87-     SmartMeter sm = smartmeter_repository.findByName(smartmeter);
88-     // gets time
89-     int time = (int) record.getJSONObject(0).get("unixTimestamp");
90-     // creates a new record
91-     for (int i = 1; i < record.length(); i++) {
92-         String metricId = record.getJSONObject(i).getString("metricId");
93-         Measurand measurand = measurand_repository.findByMetricId(metricId);
94-         double value = record.getJSONObject(i).getDouble("value");
95-         Record r = new Record(measurand, value, sm, time);
96-         repository.save(r);
97-     }
98- }
```



## RecordController(4)

```
100- /**
101  * returns a map of a specific smart meter and measurand which is grouped by
102  * time
103  *
104  * @param smartmeter
105  * @param metric
106  * @return
107  * @throws JSONException
108  */
109- @RequestMapping(method = RequestMethod.GET, path = "/smartmeter/{smartmeter}/record/{metric}")
110  public Map<Integer, List<Map<String, Double>>> getRecordOfSpecificMetric(@PathVariable String smartmeter,
111      @PathVariable String metric) throws JSONException {
112      // gets a list of all records of this smart meter
113      List<Record> list = repository.findBySmartmeter(smartmeter_repository.findByName(smartmeter));
114      // gets the latest record
115      Optional<Record> latestRecord = list.stream()
116          .max((r1, r2) -> Integer.compare(r1.getTime(), r2.getTime()));
117      int latestTime = latestRecord.get().getTime();
```

## RecordController(5)

```
118 // gets all records with the same stamp
119 List<Record> newList = list.stream()
120     .filter(r -> r.getMeasurand().getMetricId().equals(metric))
121     .filter(r -> r.getTime() == latestTime)
122     .collect(Collectors.toList());
123 Map<Integer, List<Map<String, Double>>> map = new HashMap<Integer, List<Map<String, Double>>>();
124 // iterates over the list of all records that have the same time
125 List<Map<String, Double>> toAdd = new ArrayList<Map<String, Double>>();
126 for (Record r : newList) {
127     // creates a new value for the list above
128     Map<String, Double> mapValue = new HashMap<String, Double>();
129     mapValue.put(r.getMeasurand().getMetricId(), r.getValue());
130     // adds the map to the list above
131     toAdd.add(mapValue);
132 }
133 // puts the list toAdd into the map that is going to be returned
134 map.put(latestTime, toAdd);
135 // returns a map
136 return map;
137 }
```



# Repositories

SmartmeterRepository: Bietet Methoden zur Rückgabe aller Smartmeter oder eines spezifischen Smartmeters an.

MeasurandRepository: Bietet Methoden an um alle Measurands, einen bestimmten Measurand, oder alle einem spezifischen Smartmeter zugeordneten Measurands zurückzugeben.

RecordRepository: Bietet Methoden an um alle Records oder alle einem spezifischen Smartmeter zugeordneten Records zurückzugeben.

## SmartmeterRepository

```
1 package de.tub.ise.anwsys.repos;
2
3 import java.util.List;
4
5
6
7
8
9 public interface SmartMeterRepository extends CrudRepository<SmartMeter, String> {
10     List<SmartMeter> findAll();
11
12     SmartMeter findByName(String smartmeter);
13 }
14
```



## MeasurandRepository

```
1 package de.tub.ise.anwsys.repos;
2
3 import java.util.List;
4
5
6
7
8
9
10 public interface MeasurandRepository extends CrudRepository<Measurand, String> {
11     List<Measurand> findBySmartmeter(SmartMeter smartmeter);
12
13     List<Measurand> findAll();
14
15     Measurand findById(String metricId);
16 }
```

## RecordRepository

```
1 package de.tub.ise.anwsys.repos;
2
3+ import java.util.List;
9
10 public interface RecordRepository extends CrudRepository<Record, String> {
11     public List<Record> findAll();
12     public List<Record> findBySmartmeter(SmartMeter smartmeter);
13 }
```



## Client

Erstellt Smartmeter und Measurands und nimmt die Testmessungen vor.

SMEmu Restful?

## Client Implementierung(1)

```
13 public class TestClient {
14
15     public static void main(String[] args) throws IOException, UnirestException {
16
17         // creates all smart meters
18         HttpResponse<JsonNode> response = Unirest.get("http://localhost:7878/meters").asJson();
19         Unirest.post("http://localhost:8080/smartmeter").field("smartmeter", response.getBody().getObject()).asJson();
20
21         // creates measurands
22         for (int i = 0; i < 3; i++) {
23             HttpResponse<JsonNode> metric = Unirest.get("http://localhost:7878/meters/ise1224hi563" + i).asJson();
24             Unirest.post("http://localhost:8080/smartmeter/ise1224hi563" + i)
25                 .field("measurand", metric.getBody().getArray()).asJson();
26         }
27     }
28 }
```

## Client Implementierung(2)

```
28 // calculates the average value
29 int k = 0;
30 while (k < 1000) {
31     double avgCurr = 0;
32     double avgVolt = 0;
33     for (int j = 0; j < 3; j++) {
34         int time = 0;
35         for (int i = 0; i < 900; i++) {
36             HttpResponse<JsonNode> record = Unirest
37                 .get("http://localhost:7878/meters/ise1224hi563" + j + "/data").asJson();
38             avgCurr += (Double) record.getBody().getObject().getJSONArray("measurements").getJSONObject(0)
39                 .get("value");
40             avgVolt += (Double) record.getBody().getObject().getJSONArray("measurements").getJSONObject(1)
41                 .get("value");
42             time = (int) record.getBody().getObject().get("unixTimestamp");
43         }
44         double resultCurr = avgCurr / 900;
45         double resultVolt = avgVolt / 900;
46         // average current of the smart meter which id ends with 2
47         if (j == 2) {
48             System.out.println("average current (15 min interval): " + resultCurr);
49         }
50     }
51     k++;
52 }
```

## Client Implementierung(3)

```
50      // creates a JSON array
51      JSONArray array = new JSONArray();
52      JSONObject metric1 = new JSONObject();
53      metric1.put("metricId", "MX-11460-01");
54      metric1.put("value", resultCurr);
55      JSONObject metric2 = new JSONObject();
56      metric2.put("metricId", "MX-11463-01");
57      metric2.put("value", resultVolt);
58      JSONObject timeObject = new JSONObject();
59      timeObject.put("unixTimestamp", time);
60      array.put(timeObject);
61      array.put(metric1);
62      array.put(metric2);
63      Unirest.post("http://localhost:8080/smartmeter/ise1224hi563" + j + "/record").field("record", array)
64          .asJson();
65      }
66      k++;
67  }
68 }
```



## Stromstärke Smartmeter mit ID=...2

GET `http://localhost:8080/smartmeter/ise1224hi5632/record/MX-11460-01` Params Send Save

Authorization Headers Body Pre-request Script Tests Cookies Code

Type No Auth

Body Cookies Headers (3) Tests Status: 200 OK Time: 23 ms Size: 180 B

Pretty Raw Preview JSON

```
1 {
2   "1484458199": [
3     {
4       "MX-11460-01": 7163.605690008332
5     }
6   ]
7 }
```



## /smartmeter

GET  Params

Authorization Headers Body Pre-request Script Tests [Cookies](#) [Code](#)

Type

Body Cookies Headers (3) Tests Status: 200 OK Time: 17 ms Size: 179 B

Pretty Raw Preview JSON

```
1 [
2   "ise1224hi5630",
3   "ise1224hi5631",
4   "ise1224hi5632"
5 ]
```





/smartmeter/{smartmeter}

GET ▼ http://localhost:8080/smartmeter/ise1224hi5632 Params Send ▼ Save ▼

Authorization Headers Body Pre-request Script Tests Cookies Code

Type No Auth ▼

Body Cookies Headers (3) Tests Status: 200 OK Time: 71 ms Size: 186 B

Pretty Raw Preview JSON ▼  

```
1 {  
2   "MX-11460-01": "Current(mA)",  
3   "MX-11463-01": "Voltage(V)"  
4 }
```

## /smartmeter/{smartmeter}/record

GET  Params

Authorization Headers Body Pre-request Script Tests [Cookies](#) [Code](#)

Type

Body Cookies Headers (3) Tests Status: 200 OK Time: 52 ms Size: 214 B

Pretty Raw Preview JSON

```
1 {
2   "1484420399": [
3     {
4       "MX-11460-01": 7137.517185514227
5     },
6     {
7       "MX-11463-01": 694.4962971757826
8     }
9   ]
10 }
```