

Oracle APEX Kassel- Online Meetup am 24.11.2020

Mit APEX einfach und sicher Sensor Daten in der Cloud speichern

MESSDATENERFASSUNG IN DER ORACLE APEX CLOUD



gunther@pipperr.de

Mein Blog

<https://www.pipperr.de/dokuwiki/>



Oracle Datenbank und APEX Tips
und Tricks

Zuletzt angesehen: • [start](#) • [oracle_dbsat](#)



Bergweg 14 - 37216 Witzenhausen/Roßbach

Freiberuflicher Oracle Datenbank Experte - Ich unterstütze Sie gerne in ihren Projekten.

Die Idee

- Mit einfachen IOT Sensoren sollen Daten wie Temperaturen, Anlagen Zustände , Störungsmeldungen gesammelt werden
- Je nach Meldungstyp soll bei Bedarf eine Alarmierung erfolgen
 - Wie „Licht vergessen auszuschalten“ => Mail an Besitzer
 - Temperatur sinkt unter Schwellwert, Heizung im Gewächshaus ausgefallen => Mail an Hausmeister

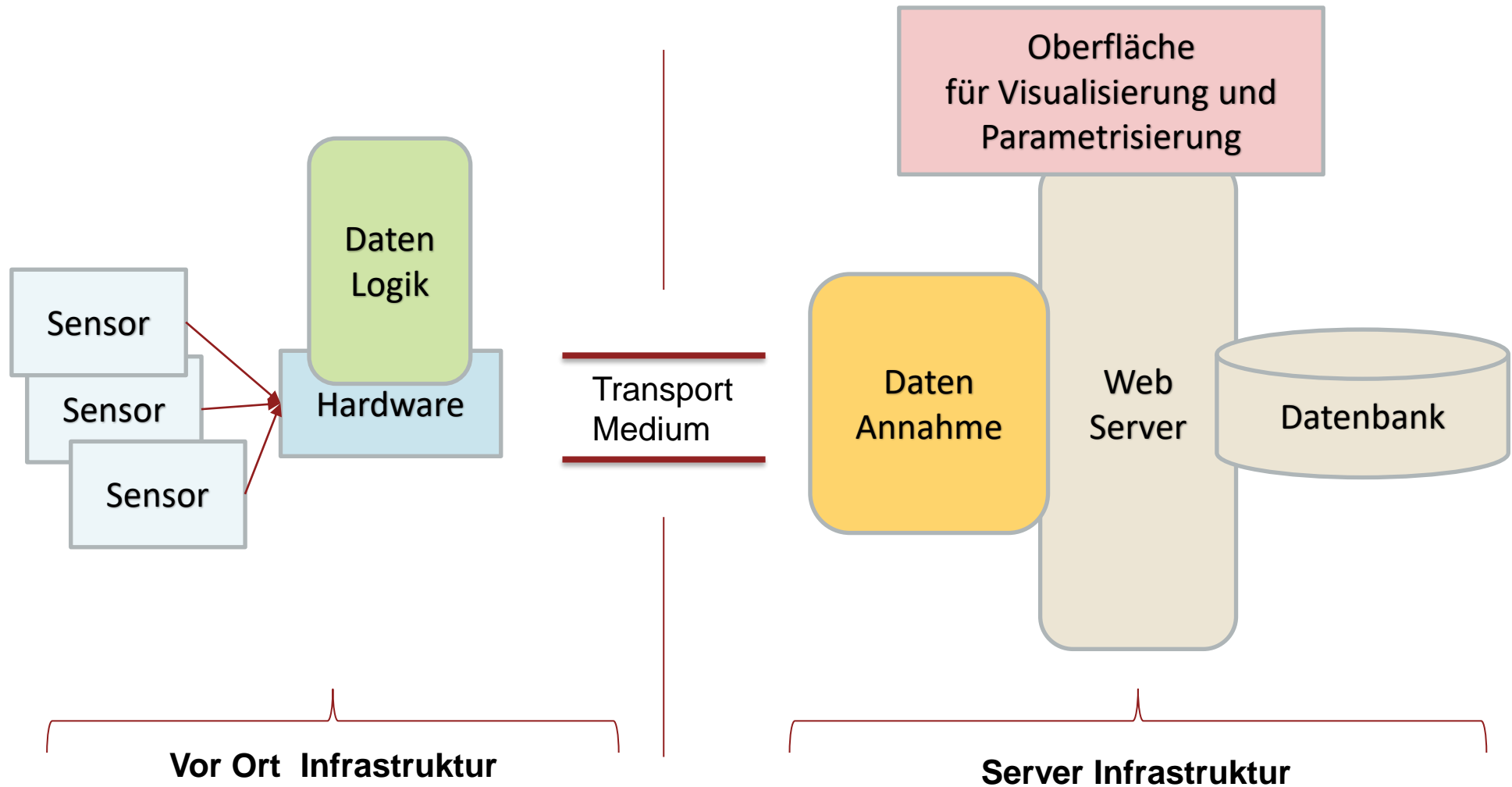
Die Idee

- Die Umsetzung soll möglichst ohne große Kosten durchgeführt werden
- Die Datenauswertung soll auch Remote von Unterwegs möglich sein
- Der Betrieb soll möglichst störungsfrei und ohne Wartung einfach „nur laufen“
- Der Oracle Produkt Stack mit Oracle APEX soll zum Einsatz kommen

Grundlagen

Wie Daten in die Cloud übertragen

- Was benötigen wir?

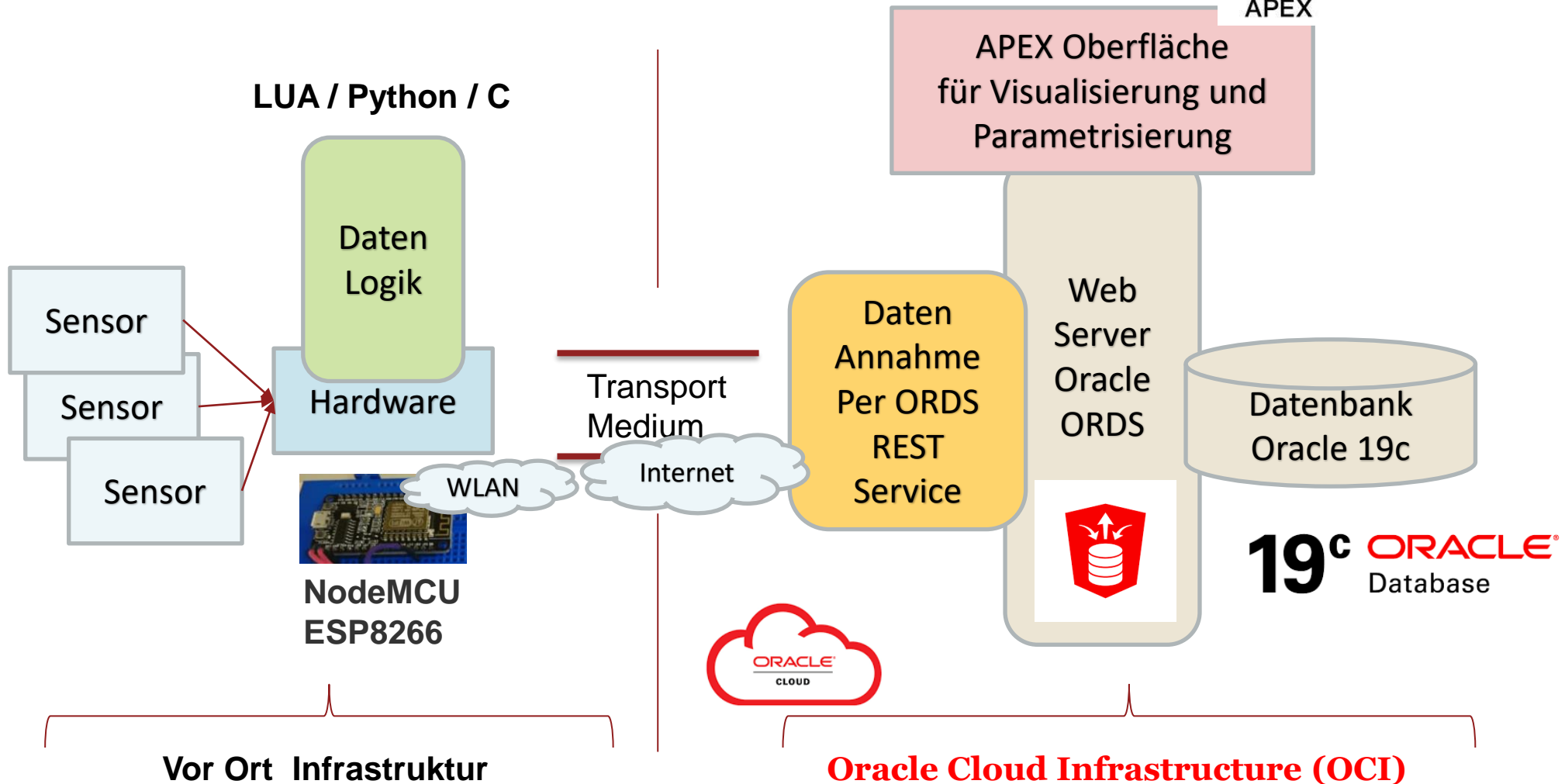


Wie Daten in die Cloud übertragen?

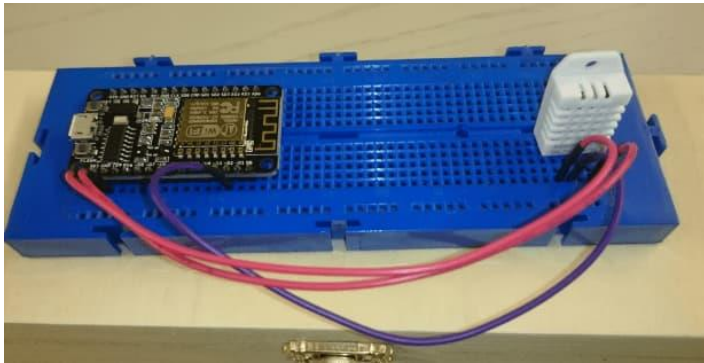
- Was können wir einfach verwenden?



ā'pěks
(#orclapex)



Vor Ort Infrastruktur (1)



NodeMCU

ESP 8266 / 32 Familie

<https://nodemcu.readthedocs.io/en/release/getting-started/>

- 5-10 Euro je nach Qualität und Typ
- Leicht erhältlich

https://www.pipperr.de/dokuwiki/doku.php?id=elektronik:nodemcu_esp8266_erste_schritte



Electric imp

<https://developer.electricimp.com/start-here>



- Ab ~40 Euro
- Nur im Fachhandel

https://www.pipperr.de/dokuwiki/doku.php?id=elektronik:start_electric_imp_iot

Vor Ort Infrastruktur (2)

Eigene Hardware

- Raspberry / Arduino / ESP
“Eigenentwicklungen”
- Kostengünstig – Skills werden
aufgebaut – unabhängig vom
Hersteller
- Tools zur Software
Entwicklung und zum
Deployen müssen eingerichtet
und erlernt werden
- Software muss vom Anwender
auf dem Gerät bei jedem
Update neu installiert /
gebrannt werden

Electronic IMP

- Software Entwicklung in der
Cloud
- Software wird über die Cloud
in das Device deployed
- Ideal für “kommerzielle”
Lösungen
- Hardware nur von diesem
Hersteller
- Beendet der Hersteller sein
Geschäftsmodell ist alles
Elektronik Schrott

Vor Ort Infrastruktur (3)

- Lokales WLAN => Internet als Transport Medium
 - Wie verbinde ich das Device mit meinem lokalen WLAN?
 - Passwort Problematik!
 - Elektronik Imp per „Licht“ konfigurierbar über Smartphone APP
 - Licht Sensor auf dem Chip wertet das „Flackern“ des Smartphone Displays als Datenstrom aus um das Password für das Wlan zu erhalten
 - Datenübertragung über das HTTP Protokoll
 - Standardisiert und einfach zu warten

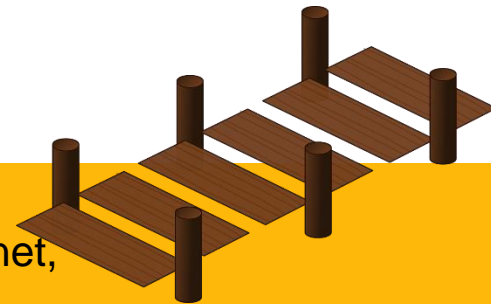
REST – Daten über HTTP übertragen

- REST (REpresentational State Transfer)
 - Verknüpfung aller Komponenten über das HTTP Protokoll
 - Der Standard für die „lockere“ Kopplung von Komponenten aus verschiedenen Welten
 - Mit den „normalen“ HTTP Methoden Daten austauschen

mit dem REpresentational State Transfer (abgekürzt REST) wird ein Programmierparadigma für verteilte Systeme bezeichnet, das insbesondere für Webservices verwendet wird.

Siehe auch

https://www.pipperr.de/dokuwiki/doku.php?id=prog:first_steps_oracle_rest_data_service



REST – Daten über HTTP übertragen

- Möglichkeiten der Daten Übertragung
 - Über die verwendete URL und Aufruf (Request) Typ
 - Über HTTP Parameter beim Aufruf
 - Über Inhalte im HTTP Call bzw. in der Antwort
 - JSON Response Body



Server Infrastruktur

- Der Datensammler in der Cloud
 - Die Sensor Daten werden per REST API zu einem Endpunkt in der Cloud übertragen und in die Datenbank eingefügt
- HTTP Endpunkt => Oracle Rest Data Service – ORDS
 - REST API + Web Applikation Server für Oracle APEX
- Oracle APEX für die Anwender Oberfläche und zur Konfiguration der REST API Calls
- Oracle Datenbank

Was bietet Oracle kostenfrei an?

apex.oracle.com

- Ein Oracle APEX Workspace
 - 25MB Platz für Tabellen

Oracle Autonomous Database

- 2 Compute virtual machines with 1/8 OCPU und 1 GB memory each
- 2 Block Volumes Storage, 100 GB total
- 10 GB Object Storage.
- 10 GB Archive Storage.
- Zwei Oracle Datenbanken
 - 1 OCPU and 20 GB storage
- NoSQL Database
 - 25 GB storage per table, up to 3 tables
- <https://www.oracle.com/cloud/free/#always-free>

Vorteil/Nachteil

apex.oracle.com

- Sofort Verfügbar
- Schnell eingerichtet
- Wenig Platz
- Kein direkter Zugriff auf das Datenbank Schema

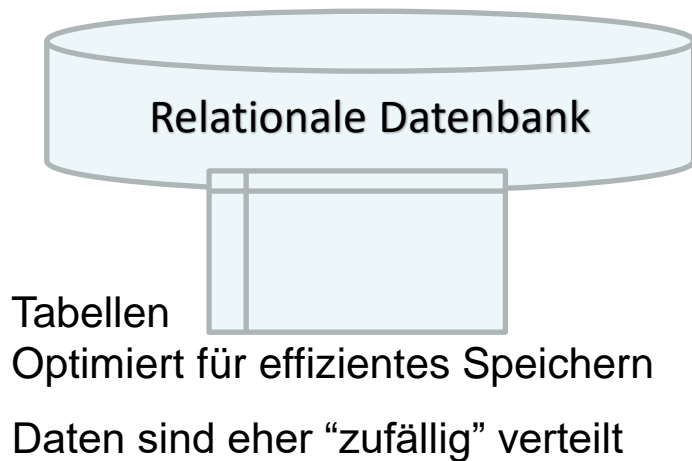
Oracle Autonomous Database

- APEX Umgebung muss erst eingerichtet werden
- URL mit “kryptischer” Subdomain

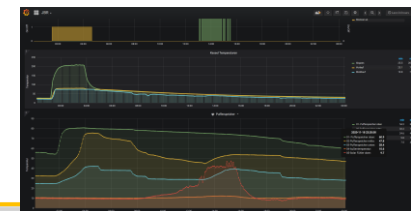
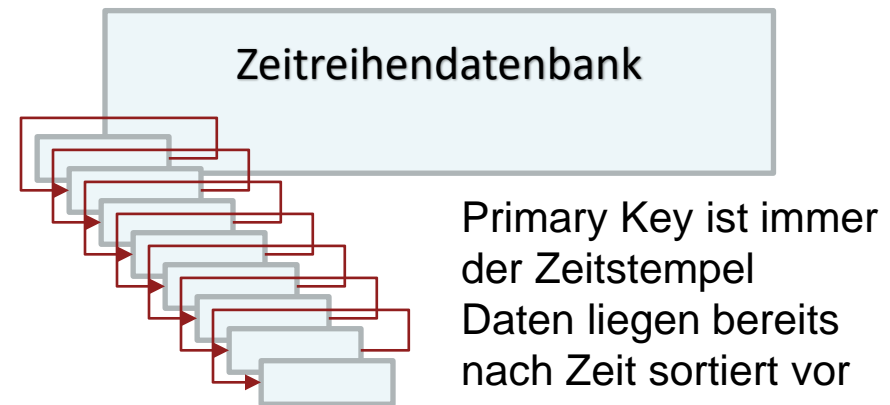
Relationale Datenbanken versus Zeitreihen

- Sensor Daten sind meist in irgendeiner Relation mit dem Zeitpunkt der Messung / Events
 - Sensor Daten können sehr große Mengen an Daten erzeugen

Verwaltung – Reagieren - Workflows



Speichern und Anzeigen



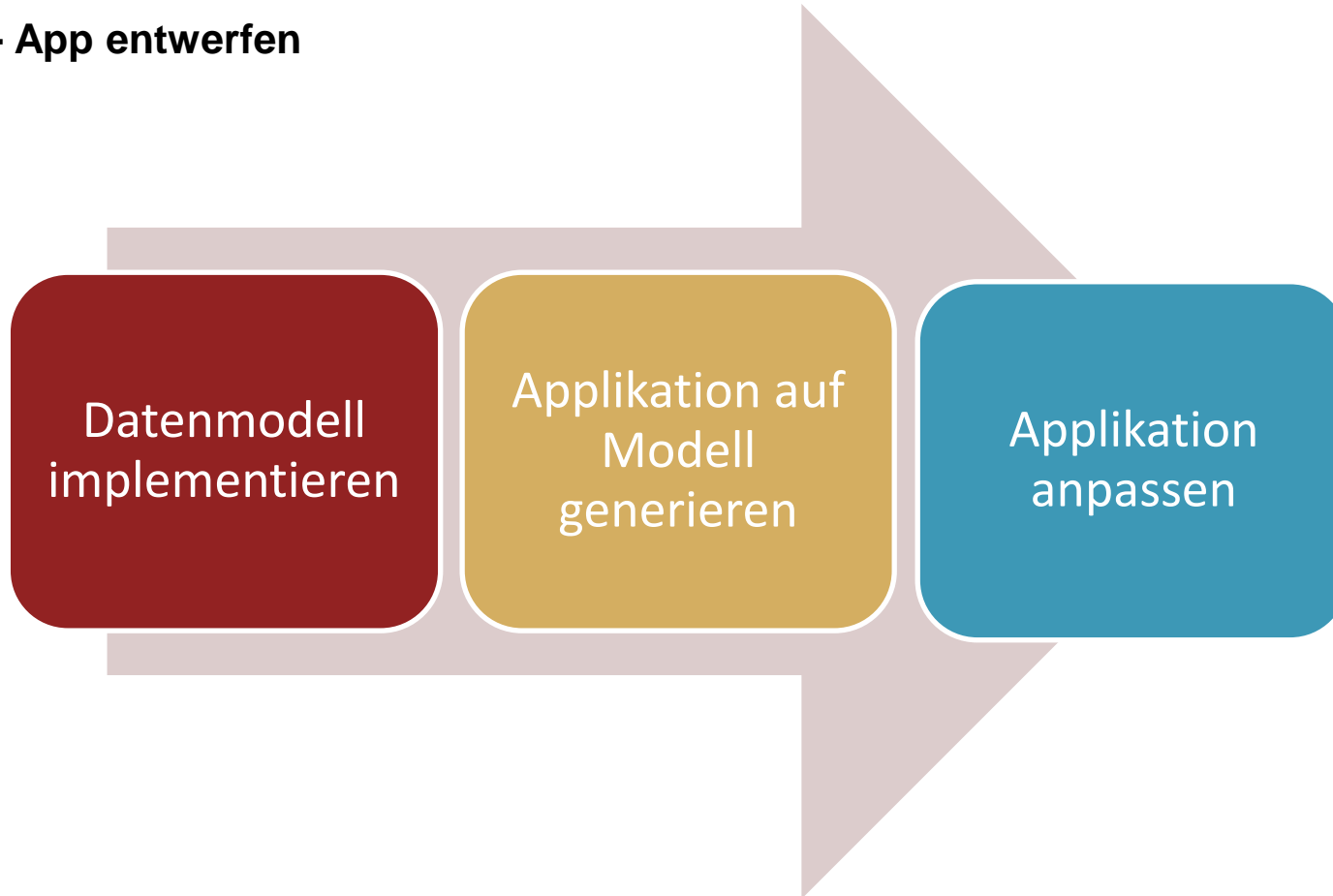
Umsetzung

Wie gehe ich an die Aufgabe heran?

- Was will ich erreichen?
 - Ziel festlegen, wie
 - Gewächshaus Temperatur aufzeichnen
 - Alarmieren falls Temperatur unter 4 Grad fällt, Aktion auslösen

Wie gehe ich an die Aufgabe heran?

Schritt 1 - App entwerfen



Datenmodell - Sensor + Aktion definieren

	Sensoren <ul style="list-style-type: none">• Nr• Sensor Name• Sensor Typ• Standort• Bemerkung

Ein Sensor löst eine oder mehrere Aktionen aus

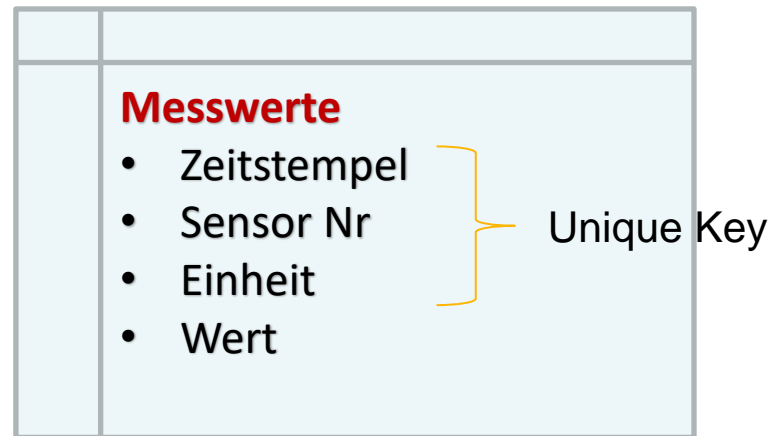
	Aktionen <ul style="list-style-type: none">• ID• ParameterSet• Parameter• Wert

Eine Aktion kann von einem Sensor getriggert werden

	SensorAktionen <ul style="list-style-type: none">• Nr• Sensor Nr• Einheit• Unterer Schwellwert• Oberer Schwellwert• Aktion Nr

Eine oder mehrere Aktionen können von einem Sensor aktiviert werden

Datenmodel – Sensor Daten speichern



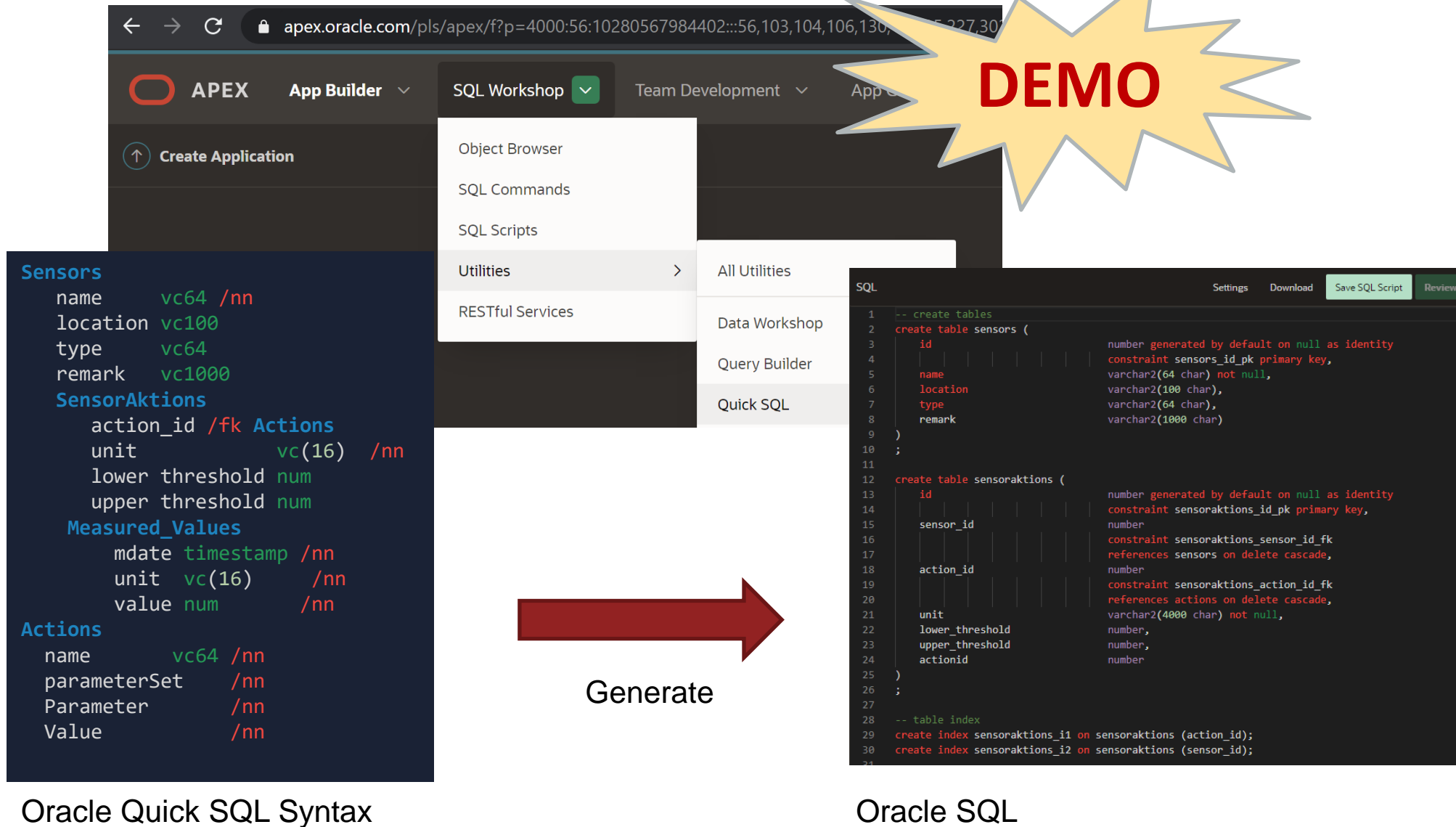
Index Organized Tables (IOT)
verwenden?

DB Modell und APEX APP aufsetzen



- Am Apex Workspace auf <https://apex.oracle.com> anmelden
- Tabellen mit Quick SQL anlegen
- APP auf dem Modell erstellen

Modell in Quick SQL umsetzen



Applikation auf Skript Basis erstellen (1)

Auf Basis des erzeugten Skripts das Modell und die APP erzeugen

DEMO

The screenshot displays the APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'App Gallery'. The 'SQL Scripts' section is active, showing a 'Script Editor' for a script named 'APEX MEETUP 2020'. The script contains two SQL statements: a 'create table sensors' statement and a 'create table sensoraktionen' statement. The 'Create App from Script' dialog is open, showing the 'Schema' as 'GPI_CONSULT' and the 'Script Name' as 'APEX MEETUP 2020'. The dialog lists the tables 'ACTIONS', 'MEASURED_VALUES', 'SENSORAKTIONS', and 'SENSORS' that will be used in the application. A red arrow points from the 'DEMO' starburst to the 'Create App' button in the script editor.

```
1 create table sensors (
2   id                number generated by default on null as identity
3   constraint sensors_id_pk primary key,
4   name              varchar2(64 char) not null,
5   location           varchar2(100 char),
6   type              varchar2(64 char),
7   remark             varchar2(1000 char)
8 )
9 ;
10
11 create table sensoraktionen (
12   id                number generated by default on null as identity
13   constraint sensoraktionen_id_pk primary key,
14   sensor_id          number
15   constraint sensoraktionen_sensor_id_fk
16   references sensors,
17   action_id          number
18   constraint sensoraktionen_action_id_fk
19   references actions,
20   unit               varchar2(16 char) not null,
21   lower_threshold    number,
22   upper_threshold    number
23 )
24 ;
25
```

Create App from Script

Schema

Schema GPI_CONSULT

Script Name APEX MEETUP 2020

The following table(s) in this script will be used to default report and form page(s) in the Create Application wizard.

Table Name

ACTIONS

MEASURED_VALUES

SENSORAKTIONS

SENSORS

Cancel Create Application >

Applikation auf Skript Basis erstellen (2)

Create an Application

Name: APEX_MEETUP_2020 Appearance: Vita, Side Menu

Pages

+ Add Page		
Home	Blank	Edit
Actions	Interactive Report with Form (actions)	Edit
Sensors	Interactive Report with Form (sensors)	Edit
Sensoraktionen	Interactive Report with Form (sensoraktionen)	Edit
Values	Interactive Report with Form (measured_values)	Edit

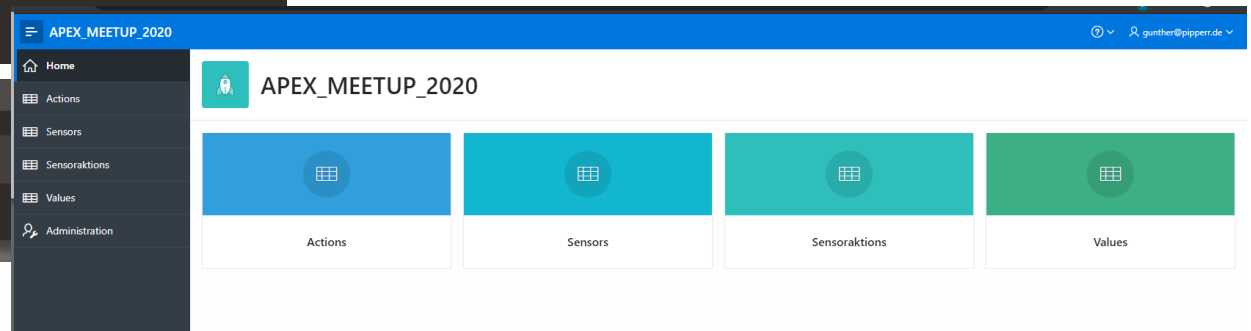
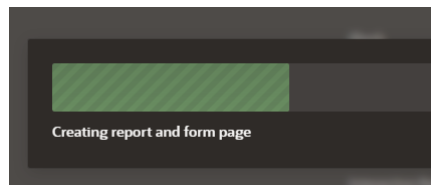
Features Check All

☒ About Page Add about this application page

☒ Access Control Enable role-based user authorization

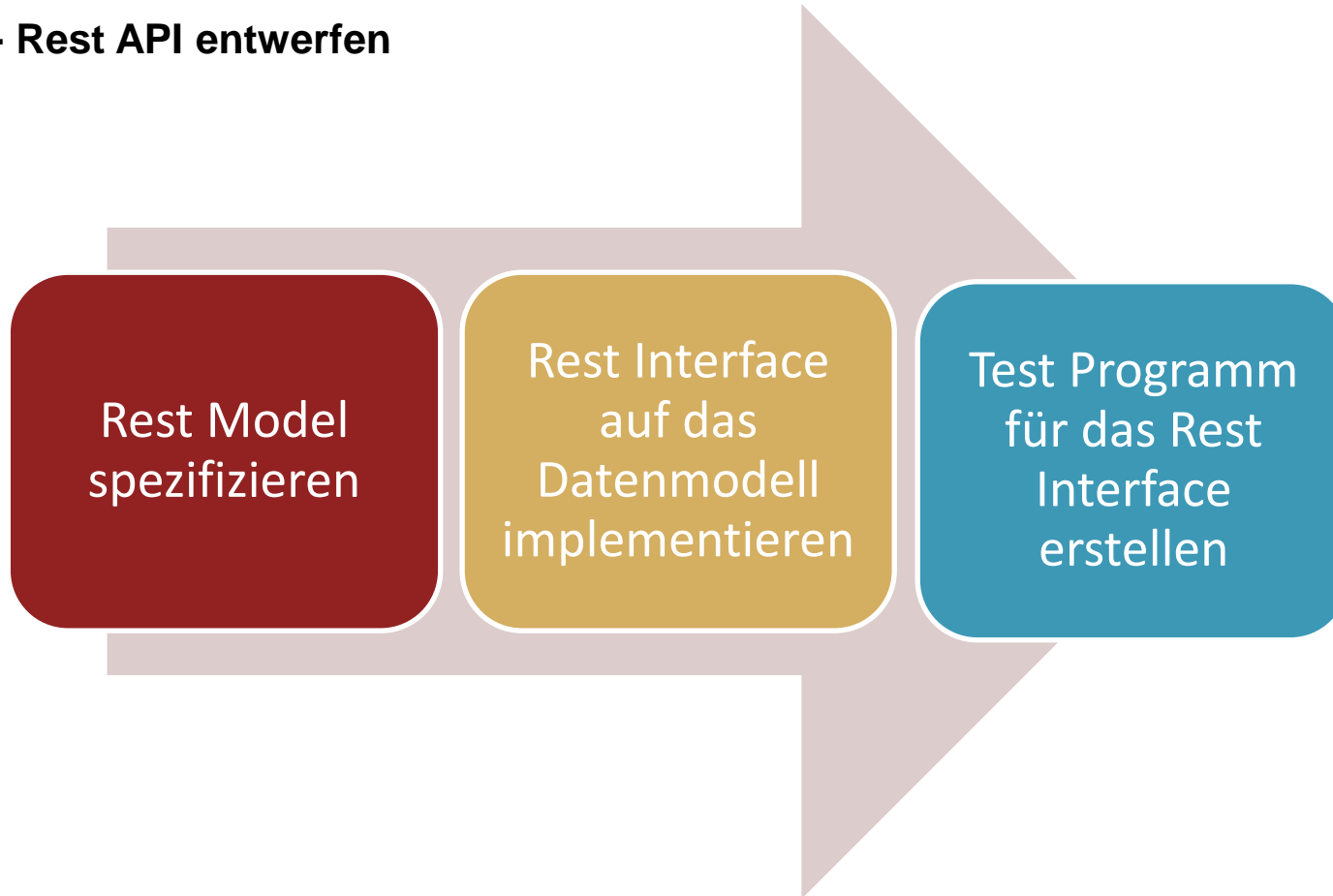
☒ Activity Reporting Include user activity and error reports

Cancel Create Application



Wie gehe ich an die Aufgabe heran? (2)

Schritt 2 - Rest API entwerfen

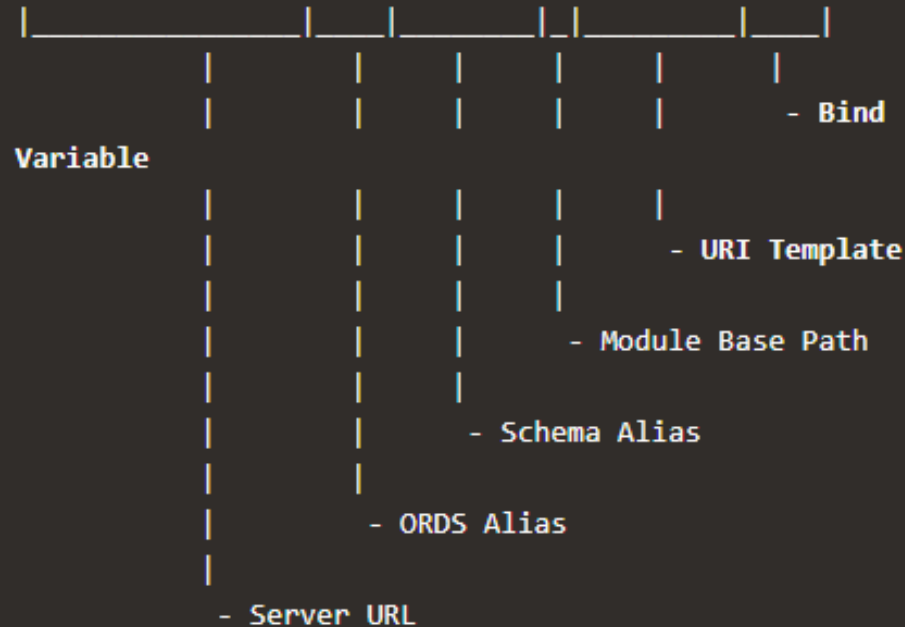


REST URI

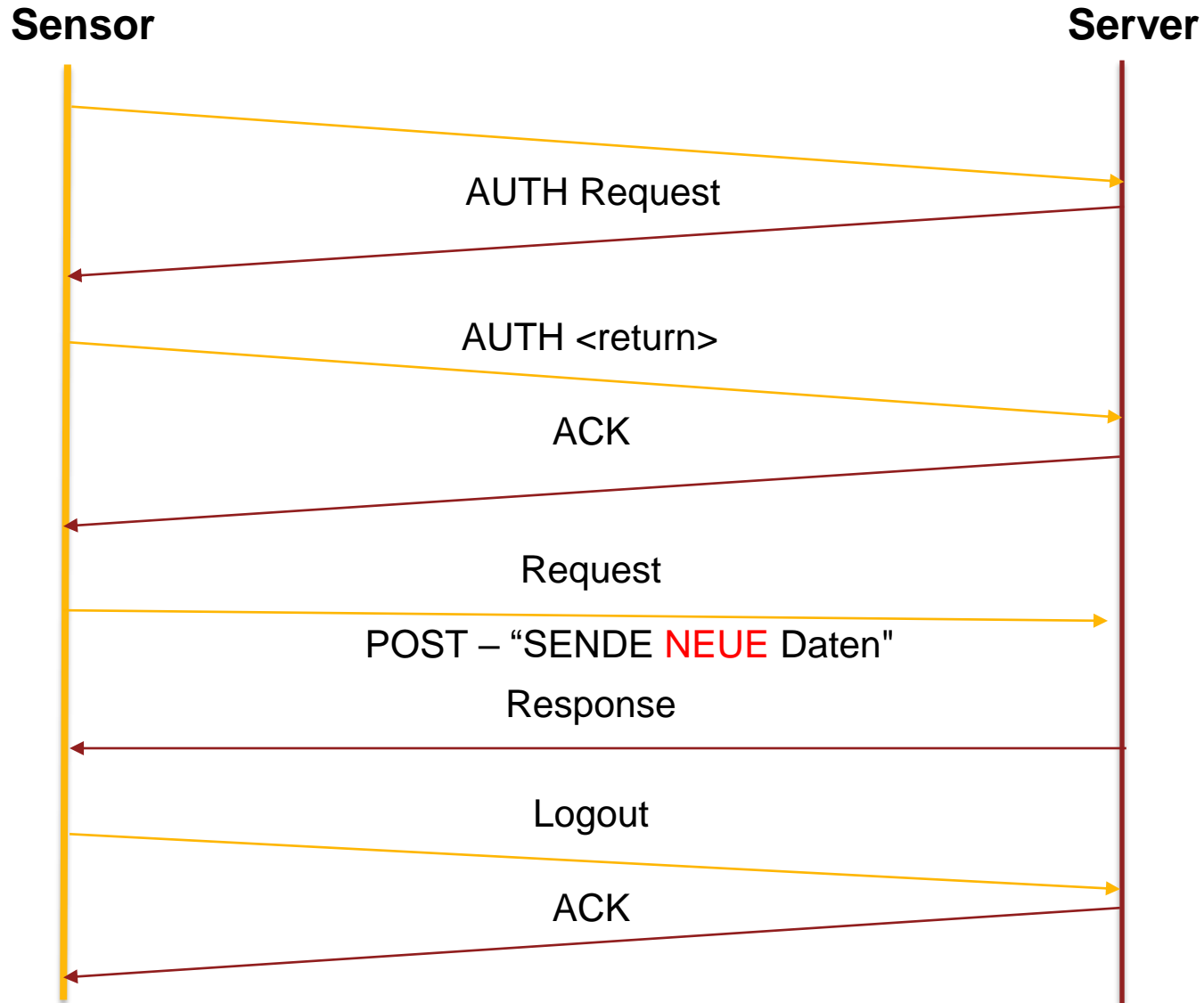
The URI Template may also include a bind variable appended after a forward slash. This allows a value to be passed to the service as part of the URI. The bind variable syntax prefixes the variable name with a colon (:)

Example:

`http://server.com/ords/mySchema/hr/employees/:id`



Rest Interface - Ablauf



Die Response Codes

- Auf korrekte Status Codes achten

1xx Informational „Hold on“	100 Continue
	101 Switching Protocols
	102 Processing
2xx Success „Here you go“	200 OK
	201 Created
	208 Already Reported
3xx Redirection „Go away“	301 Moved Permanently
	304 Not modified
	307 Temporary Redirect
4xx Client Error „You fucked up“	400 Bad Request
	401 Unauthorized
	404 Not Found
5xx Server Error „I fucked up“	500 Internal Server Error
	502 Bad Gateway
	503 Service Unavailable

Aus https://www.doag.org/formes/pubfiles/11276739/2019-SQL-Robert_Marz-RESTful_Services_in_der_Datenbank_mit_ORDS_erstellen-Praesentation.pdf

Eigener Service oder AutoRest Enabled Tables?

Eigener Service

- Logik komplett “im Griff”
- Verschachtelte JSON Records kein Problem, wie z.B. alle Messergebnisse eines Sensors auf einmal übertragen
- Für jeden Handler wie GET / Post etc. muss alles selber umgesetzt werden

AutoREST

- Schnell implementiert
- Alle REST Methoden auf einer Tabelle stehen zur Verfügung
- Etwas wacklig bei zu formatierenden Formaten wie dem Datum
- Master Detail Problematik muss dann im Client gelöst werden

<https://oracle-base.com/articles/misc/oracle-rest-data-services-ords-autorest>

Was für URI's brauchen wir? (1)

■ Sensor Daten übertragen

- URL : **/sensor/:SENSORID/measurement**
- Method : POST
- URL Parameter : Required : SENSORID = Number
- Request Body :

```
{  timestamp      : "DATE TIME"
  timestamp_format : "String"
  measurements :
    [ {
      unit : "String"
      value : "Float"
    } , ...
    ]
}
```

- Query String : None
- Success Response : 200
- Error Response : 400 if Sensor not found

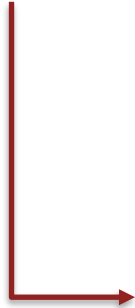
Was für URI's brauchen wir? (2)

- Sensor Aktionsmöglichkeiten abfragen
 - GET **/sensor/:SENSORID/actions**
 - Response: List für alle Threshold + Aktionen
- Sensor löst Aktion(en) aus, falls Werte für die Aktion(en) erreicht
 - POST **/sensor/:SENSORID/action/:ACTIONID**
- Hole die Server Umgebung (debug)
 - GET **/sensor/serverinfo**
 - Response Server Info Daten

Oracle ORDS REST Data Service

Modul

- Logische Gruppierung
- **URI Prefix** wie **/sensor**



Template

- **URI Pattern** wie **/sensor/:SENSORID/measurement**
- Kann Parameter enthalten

Handler

- Http Methoden
GET / POST / PUT / DELETE
- Parameter HTTP-HEADER / URI

```
<code>  
begin ..end;  
</code>
```

REST Interface in der APEX App definieren



REST Interface (1)

https://apex.oracle.com/pls/apex/gpi_consult/sensor

■ Modul anlegen

The screenshot displays the 'Modules' section of the Oracle REST Data Services (ORDS) configuration page, specifically the 'Module Definition' tab. The 'ORDS Module Definition' form is shown with the following fields:

- Module Name:** SENSOR_API
- Base Path:** sensor
- Is Published:** ☒
- Pagination Size:** 25
- Origins Allowed:** (empty)
- Comments:** Sensor API

A green success message 'Module Created' is displayed at the top of the form. The 'Full URL' field is populated with https://apex.oracle.com/pls/apex/gpi_consult/sensor/. The 'Legend' at the bottom left indicates that the template is fully protected by a privilege. The 'Generate Swagger Doc' button is visible at the bottom right.

REST Interface (2)

- Template anlegen

ORDS Template Definition

Cancel

Delete

Apply Changes

RESTful Service Module

SENSOR_API

Module Base Path

/sensor/

* URI Template

serverinfo

?

Full URL

https://apex.oracle.com/pls/apex/gpi_consult/sensor/serverinfo

* Priority

0

?

* HTTP Entity Tag Type

Secure Hash

?

Comments

Server Info

11 of 4000

Resource Handlers

Create Handler >

Q

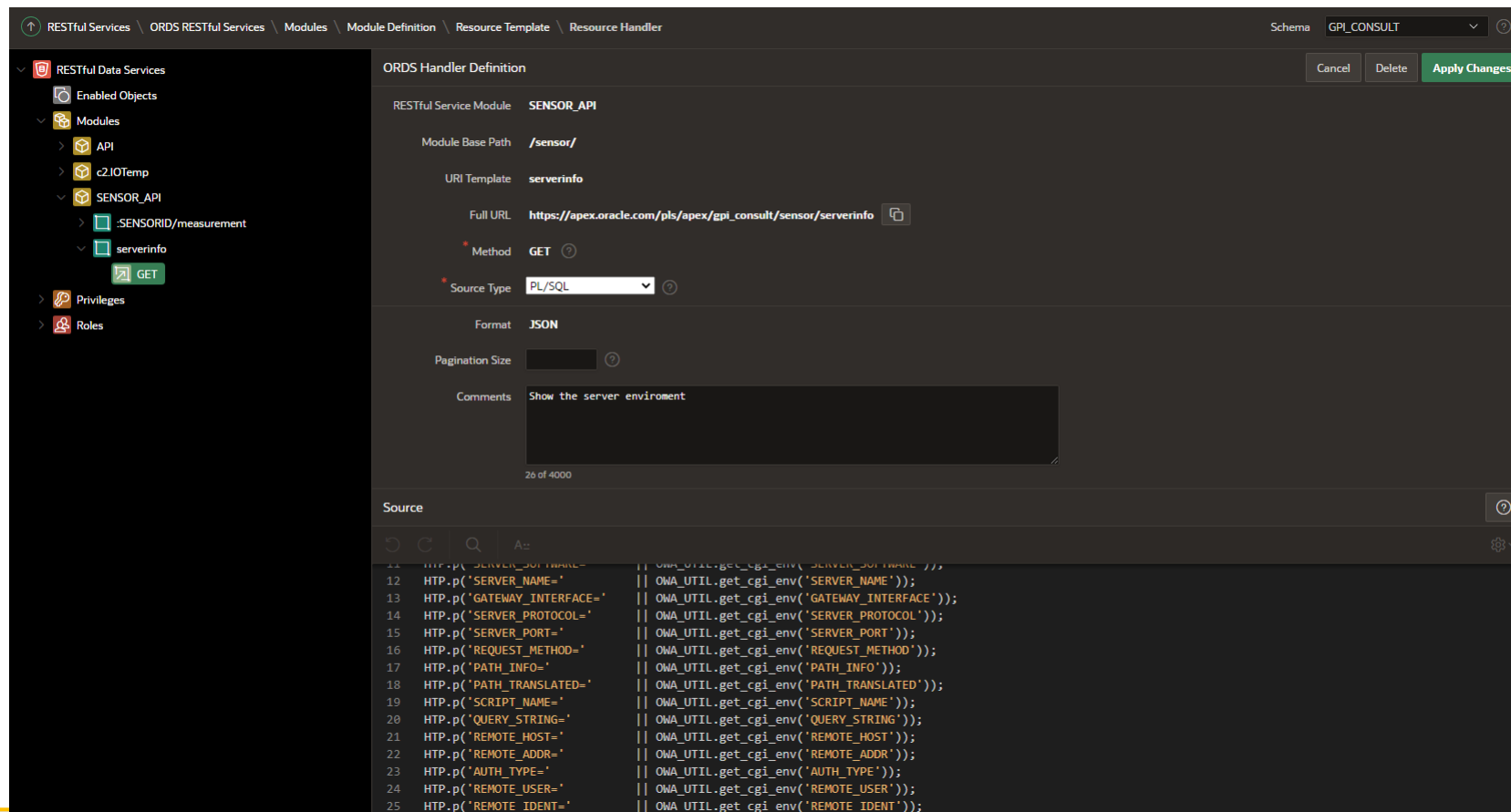
Go

Actions

https://apex.oracle.com/pls/apex/gpi_consult/sensor/serverinfo

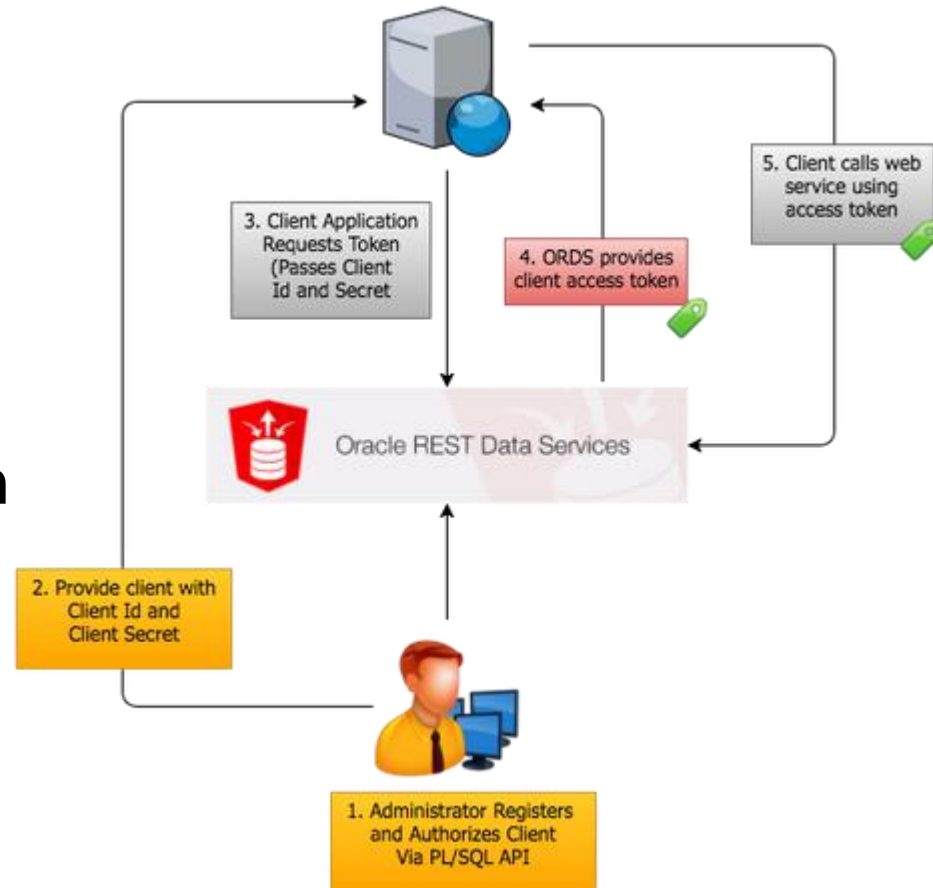
Rest Interface (3)

- Handler anlegen
 - Z.B. im ersten Schritt einen “GET” Handler “serverinfo” um die API einfach zu testen



Sicherheit (1)

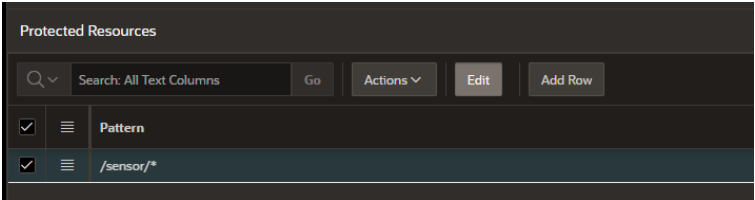
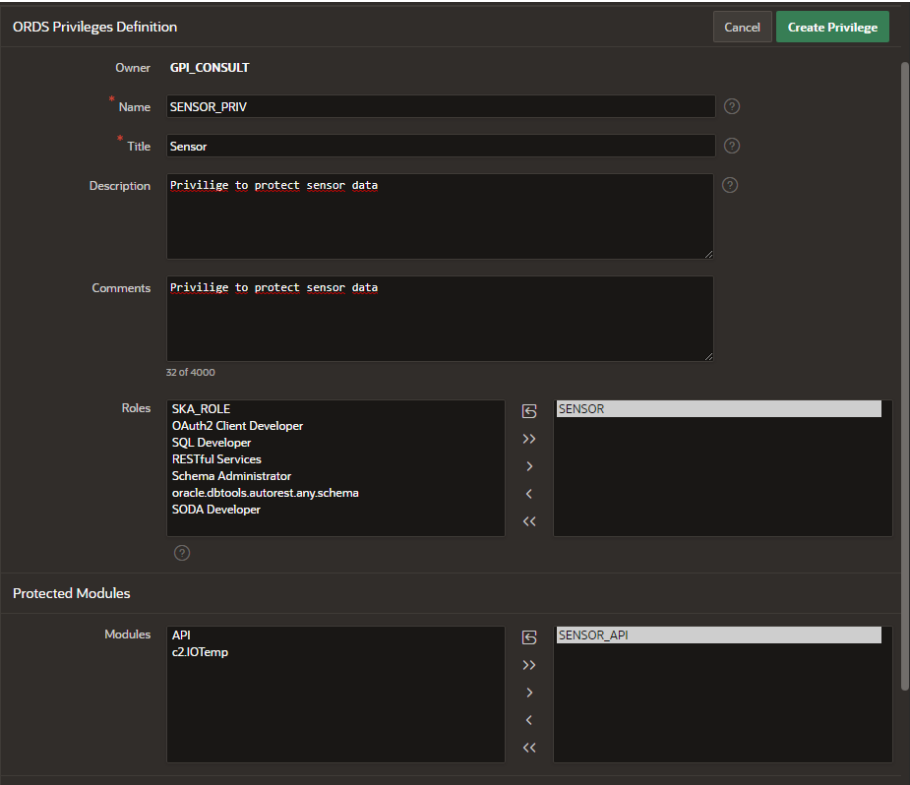
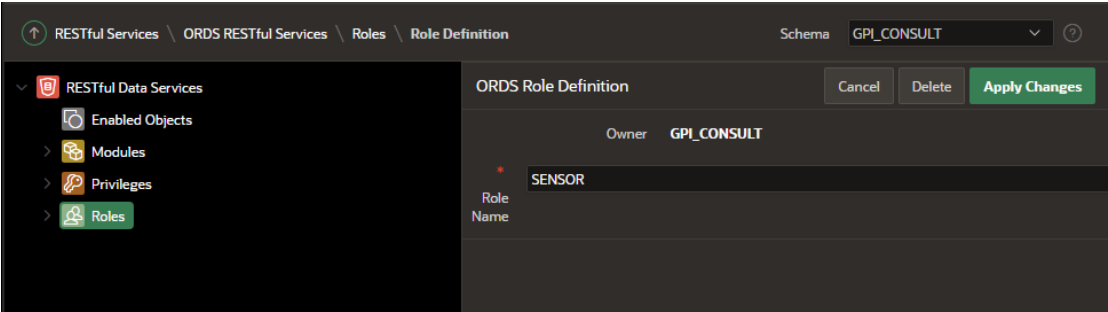
- Security sicherstellen!
- Z.B. über OAUTH Mechanismus
 - Token basierende Authentifizierung nützen



<https://www.jmcloud.com/blog/ords-securing-services-using-oauth2-2-legged>

Sicherheit (2)

- Rolle anlegen
- Privilege anlegen



OAuth Client anlegen

■ Client anlegen und berechtigen

```
BEGIN

OAuth.delete_client(p_name => 'SENSOR_HARDWARE');

-- create the client
OAuth.create_client
( p_name           => 'SENSOR_HARDWARE',
  p_description     => 'Sensor API Hardware Client',
  p_grant_type      => 'client_credentials',
  p_privilege_names => 'SENSOR_PRIV',
  p_support_email   => 'gunther@pipperr.de');

-- Grant the new client access to the SENSOR Rolle
OAuth.grant_client_role
( p_client_name => 'SENSOR_HARDWARE',
  p_role_name   => 'SENSOR'
);

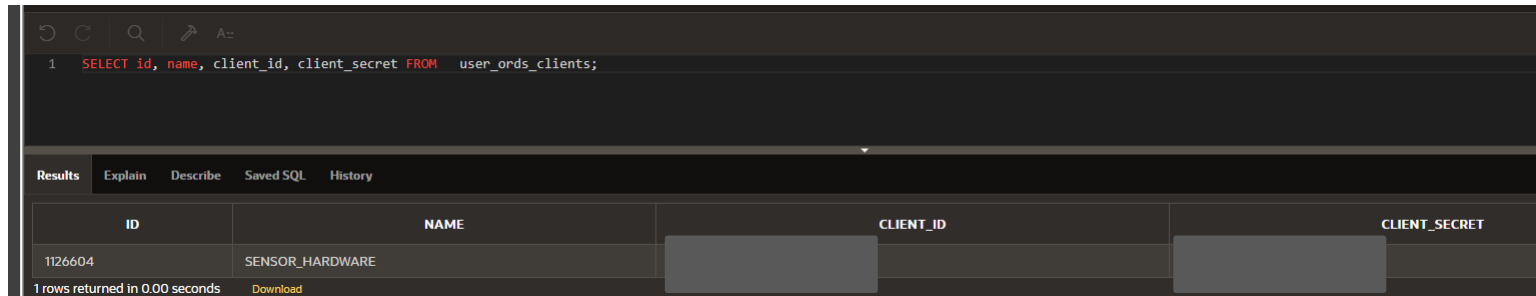
COMMIT;

END;
/
```


OAuth client User / PWD abfragen

- Credentials abfragen

```
SELECT id
      , name
      , client_id, client_secret
FROM   user_ords_clients;
```



The screenshot shows a database query interface. At the top, a SQL query is entered: `1 SELECT id, name, client_id, client_secret FROM user_ords_clients;`. Below the query, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is active, displaying a table with four columns: 'ID', 'NAME', 'CLIENT_ID', and 'CLIENT_SECRET'. The table contains one row of data: '1126604', 'SENSOR_HARDWARE', and two redacted values for 'CLIENT_ID' and 'CLIENT_SECRET'. At the bottom left, it says '1 rows returned in 0.00 seconds' and there is a 'Download' link.

ID	NAME	CLIENT_ID	CLIENT_SECRET
1126604	SENSOR_HARDWARE	[REDACTED]	[REDACTED]

1 rows returned in 0.00 seconds [Download](#)

Secrets per REST abfragen

- Token holen mit:

```
curl -i -k --user 1lOATtaRzxxxxxxxxx1Gt1KX7w...:NsBA3_lxxxxxlPZ564fXzGXw..  
--data "grant_type=client_credentials"  
https://apex.oracle.com/pls/apex/gpi_consult/oauth/token
```

HTTP/1.1 200 OK

Content-Type: application/json

...

```
{"access_token":"WrSRDJFfgxxxxqwGAIWY6g","token_type":"bearer","expires_in":36000}
```

Rest Interface mit dem Token anfragen

Token für die Autorisierung nützen

```
curl -i -k -H "Authorization: Bearer WrSRDJFgxxxxqwGAIWY6g "  
      "https://apex.oracle.com/pls/apex/gpi_consult/sensor/serverinfo"
```

HTTP/1.1 200 OK

Content-Type: text/plain; charset=UTF-8

...

REMOTE_USER=1IOATtaRz29N4L1Gt1KX7w..

....

Ablauf in der späteren Applikation

- Credentials in der APP hinterlegen
- Access Token holen mit **.../oauth/token**
 - Ablaufzeitpunkt merken (nur 3600 Sekunden gültig!)
- Daten mit dem Token anfordern solange dieses noch gültig ist, ansonsten neu anfordern

PL/SQL Routine für die Datenauswertung

- Rest Interface für die Daten Übertragung
 - URI /**sensor**/:**SENSORID**/measurement
- Übermittelt diese Struktur

```
{  "timestamp": 1606153909.336712
  , "timestamp_format": "epoch",
  , "measurements": [
    { "unit": "temp",
      , "value": 11.478
    },
    { "unit": "hum",
      , "value" : 66.61 }
  ]
}
```



	Messwerte
	<ul style="list-style-type: none">• Zeitstempel• Sensor Nr• Einheit• Wert

REST URL /sensor/:SENSORID/measurement

- Oracle JSON Object verwenden

```
v_payload      blob;  
v_json_data    clob;  
v_body_json    json_object_t;  
  
v_measurement_arr  json_array_t;  
v_measurement_obj  json_object_t;
```

- Body auslesen über die „:“ Bind Variablen Notation und in JSON wandeln

```
v_payload      :=:body;  
v_json_data := wwv_flow_utilities.blob_to_clob(v_payload);  
v_body_json := json_object_t.parse( v_json_data );
```

Implizite Parameters wie :BODY (in) und :STATUS_CODE (OUT) etc:
<https://docs.oracle.com/en/database/oracle/oracle-rest-data-services/18.3/aelig/implicit-parameters.html>

Parameter prüfen

- Über Bind Variablen Notation auf dem URL Parameter zugreifen
 - Mit `CAST(:SENSORID AS NUMBER DEFAULT -1 ON CONVERSION ERROR)` auf Fehler prüfen!

```
v_sensor_id := CAST(:SENSORID AS NUMBER DEFAULT -1 ON CONVERSION ERROR);

if v_sensor_id=-1 then

    :status_code := 400; -- Bad Request

    apex_json.open_object;
    apex_json.write( p_name => 'Error Message '
                    , p_value => 'Sensor ' || v_sensor_id || ' invaild format -
have to be a number'
                    , p_write_null => true);
    apex_json.close_object;
    apex_json.close_all;

else
...end if
```

REST URL /sensor/:**SENSORID**/measurement

```
v_time_stamp:=v_body_json.get_String('timestamp');
v_time_stamp_type:=v_body_json.get_String('timestamp_format');

v_measurement_arr := v_body_json.get_array('measurements');

<<mess_loop>>
for i in 0 .. v_measurement_arr.get_size - 1
loop
  IF v_measurement_arr.get(i).is_object
  THEN
    v_measurement_obj := TREAT(v_measurement_arr.get(i) AS JSON_OBJECT_T);

    v_unit :=v_measurement_obj.get_String('unit');
    v_value :=v_measurement_obj.get_String('value');

    insert into MEASURED_VALUES (
      SENSOR_ID
      , MDATE
      , UNIT
      , VALUE
    )
    values (
      v_sensor_id
      , v_epoch_to_date
      , v_unit
      , v_value
    );

    commit;

  end if;

end loop mess_loop;
```

```
{  "timestamp": 1606153909.336712
  , "timestamp_format": "epoch",
  , "measurements": [
    { "unit": "temp",
      , "value": 11.478
    },
    { "unit": "hum",
      , "value" : 66.61 }
  ]
}
```


Zeitangaben übertragen – C nach PL/SQL

- Kontroller liefert EPOCH
- Python

23-NOV-20 08.36.44.878090000 PM
EUROPE/BERLIN

```
now = datetime.now()
date_string = time.mktime(now.timetuple()) + now.microsecond * 1e-6

# => 1606160204.87809
```

- Oracle Timestamp

```
SELECT
  (   TIMESTAMP '1970-01-01 00:00:00 GMT'
    + numtodsinterval(1606160204.87809, 'SECOND' )
  ) at TIME zone 'Europe/Berlin'
FROM dual;
```

Error Handling - ORDS

- Fehler im ORDS anzeigen
 - Default Server Error 500 => Keine Fehlermeldungen wird bis zum Client durchgereicht
 - Für Entwicklungsumgebungen aktivieren mit:

```
<entry key="debug.printDebugToScreen">true</entry>
```

Ein Response Objekt füllen

- APEX_JSON einsetzen

```
exception
when others then

    apex_json.open_object;
    apex_json.write( p_name => 'Error Message : '
                    , p_value => 'Error in Call to ' || v_script_name
                    || ' SQL Error : ' || SQLERRM
                    , p_write_null => true);

    apex_json.write( p_name => 'Get this data : '
                    , p_value => v_json_data
                    , p_write_null => true);

    apex_json.close_object;
    apex_json.close_all;

    :status_code := 400; -- Bad Request
end;
```

Test-Programm für das REST Interface

- Möglichst in ähnlicher Programmiersprache wie die später Umsetzung
- Siehe Python Beispiel



DEMO

```

import requests
import json
import time
import random
from datetime import datetime

# https://apex.oracle.com/pls/apex/gpi_consult/sensor/:SENSORID/measurement

base_url = 'https://apex.oracle.com/pls/apex/gpi_consult/'
token_url = 'oauth/token'
serverinfo_url='sensor/serverinfo'
data_url = '/sensor/1/measurement'

client_id = 'xxxxxxxxxxxxxxxxxw..'
client_secret = ',NxxxxxxxxxxxxxGXw..'

# Sensor Client API Test
# define the payload
now = datetime.now()
date_string = time.mktime(now.timetuple()) + now.microsecond * 1e-6
temperatur = random.random() * 20 + 1
Luftfeuchtigkeit = random.random() * 100 + 1

data_record = {
    "timestamp" : date_string , "timestamp_format" : "epoch"
    , "measurements" : [ { "unit" : "temp" , "value" : temperatur } , { "unit": "hum" , "value": Luftfeuchtigkeit } ] }

# request token
response = requests.post( base_url+token_url , auth=(client_id, client_secret), data={'grant_type': 'client_credentials'} )

json_response = response.json()
access_token = json_response['access_token']

# Send the data

header_data = {"Authorization" : "Bearer "+access_token}

response = requests.post( base_url+data_url , headers=header_data, data=json.dumps(data_record) )

# test a get request
# response = requests.get(base_url+serverinfo_url, headers=header_data )

print ( json.dumps(data_record))

print ( "\n-----start the call -----")
print( str(response.headers).replace("%20"," ") )

print ( "\n-----Response -----")
print(response.content)

```

Swagger Dokumentation generieren

- Automatisch die Dokumentation über Swagger erzeugen

The screenshot displays the Swagger Editor interface. On the left, a code editor shows the OpenAPI specification in JSON format. The specification includes the title "ORDS generated API for SENSOR_API", version "1.0.0", and a base path of "/pls/apex/gpi_consult/sensor". It defines a "Sensor API" with a "serverinfo" endpoint (GET) and a "measurement" endpoint (POST). The "measurement" endpoint has a body parameter and several query parameters including "row_count", "row_offset", and "status_code".

On the right, the rendered Swagger UI is shown. It features the title "ORDS generated API for SENSOR_API 1.0.0" and the base URL "https://apex.oracle.com/pls/apex/gpi_consult/open-api-catalog/sensor/". Below the title, there is a "Schemes" dropdown menu set to "HTTPS". The "default" section is expanded, showing two endpoints:

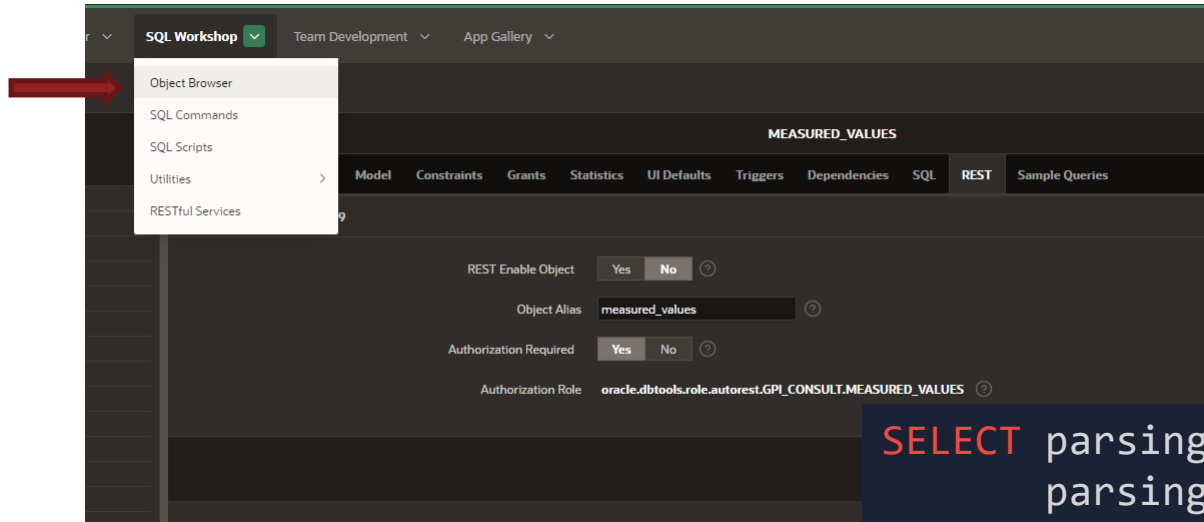
- GET /serverinfo**
- POST /{SENSORID}/measurement**

The "POST /{SENSORID}/measurement" endpoint is selected, and its details are shown below. It includes a "Try it out" button and a table for parameters:

Name	Description

REST ENABLED Tables

- Tabellen direkt per REST Bearbeiten

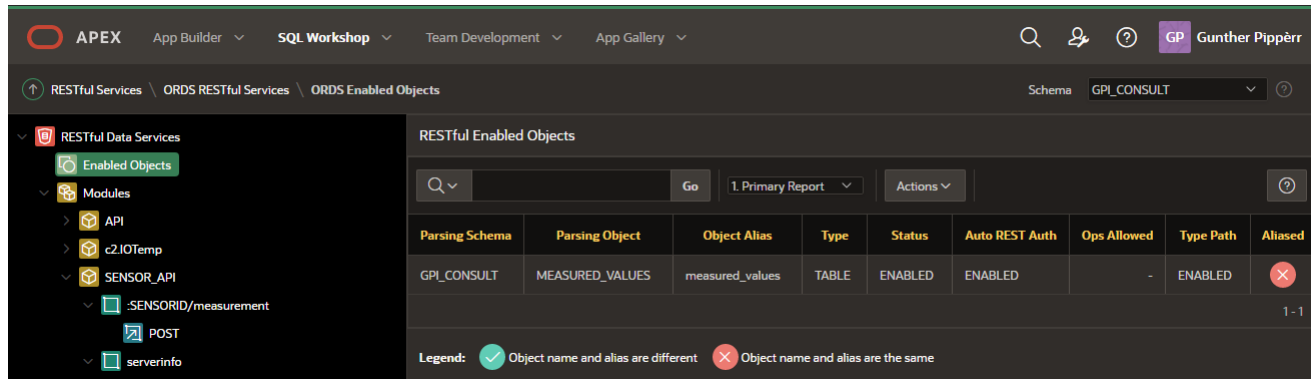


```
SELECT parsing_schema,  
       parsing_object,  
       object_alias,  
       type,  
       status  
FROM   user_ords_enabled_objects  
ORDER BY 1, 2;
```

Results				
Explain Describe Saved SQL History				
PARSING_SCHEMA	PARSING_OBJECT	OBJECT_ALIAS	TYPE	STATUS
GPI_CONSULT	MEASURED_VALUES	measured_values	TABLE	ENABLED
1 rows returned in 0.02 seconds Download				

Übersicht in APEX

■ Übersicht in APEX



The screenshot shows the APEX interface with the following components:

- Top Bar:** APEX, App Builder, SQL Workshop, Team Development, App Gallery, Search, User (GP), Gunther Pippèr.
- Breadcrumbs:** RESTful Services > ORDS RESTful Services > ORDS Enabled Objects.
- Schema:** GPI_CONSULT.
- Left Panel:** RESTful Data Services > Enabled Objects > Modules > API > c2.IOTemp > SENSOR_API > .SENSORID/measurement > POST > serverinfo.
- Main Panel:** RESTful Enabled Objects.
- Table:**

Parsing Schema	Parsing Object	Object Alias	Type	Status	Auto REST Auth	Ops Allowed	Type Path	Aliased
GPI_CONSULT	MEASURED_VALUES	measured_values	TABLE	ENABLED	ENABLED	-	ENABLED	✗

Legend: ✓ Object name and alias are different ✗ Object name and alias are the same

Auto Rest Rechte

- Über "Privilege" und Rolle wie ein "normaler" Service

ORDS Privileges Definition Cancel Delete Apply Changes

Owner: **GPI_CONSULT**

* Name: ⓘ

* Title: ⓘ

Description: ⓘ

Comments:

37 of 4000

Roles

SKA_ROLE	oracle.dbtools.autorest.any.schema
OAuth2 Client Developer	SENSOR
SQL Developer	oracle.dbtools.role.autorest.GPI_CONSULT.MEASURED_VALUES
RESTful Services	
Schema Administrator	
SODA Developer	

Protected Modules

Modules	API
	SENSOR_API
	c2IOTemp

Protected Resources

Go Actions Edit Add Row Reset

<input type="checkbox"/>	<input type="checkbox"/>	Pattern
<input checked="" type="checkbox"/>	<input type="checkbox"/>	/measured_values/*
<input type="checkbox"/>	<input type="checkbox"/>	/metadata-catalog/measured_values/*

Per URL und Meta Katalog abfragen

- Meta Katalog über URL Abfragen
 - Meta Katalog /**metadata-catalog**

https://apex.oracle.com/pls/apex/gpi_consult/metadata-catalog

```
. .  
, {"name": "MEASURED_VALUES", "links": [{"rel": "describes", "href": "https://ap  
ex.oracle.com/pls/apex/gpi_consult/measured_values/"}  
, {"rel": "canonical", "href": "https://apex.oracle.com/pls/apex/gpi_consult/  
metadata-catalog/measured_values", "mediaType": "application/json"}  
, {"rel": "alternate", "href": "https://apex.oracle.com/pls/apex/gpi_consult/  
open-api-  
catalog/measured_values/", "mediaType": "application/openapi+json"}]]}  
...
```

Per URL und Meta Katalog abfragen

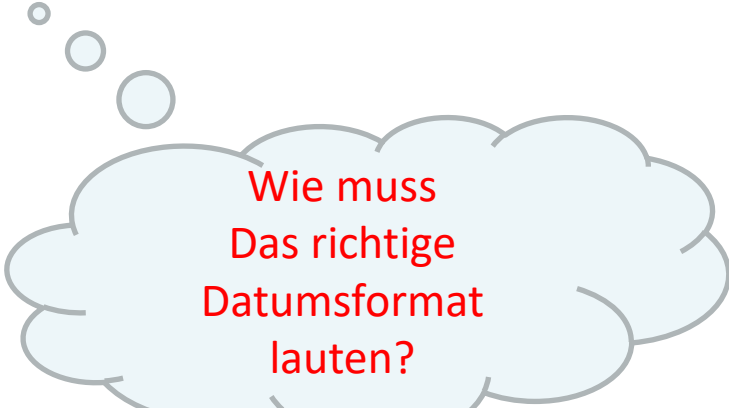
- Per URL Details Abfragen – Geschützt!
 - Abfrage wieder mit OAUTH Token!

```
{ "name": "MEASURED_VALUES", "primarykey": ["id"]
, "members": [
{ "name": "id", "type": "NUMBER" }
, { "name": "sensor_id", "type": "NUMBER" }
, { "name": "mdate", "type": "TIMESTAMP (6) " }
, { "name": "unit", "type": "VARCHAR2" }
, { "name": "value", "type": "NUMBER" } ]
, "links": [ { "rel": "collection", "href": "https://apex.oracle.com/pls/apex/
gpi_consult/metadata-catalog/", "mediaType": "application/json" }
, { "rel": "canonical", "href": "https://apex.oracle.com/pls/apex/gpi_consul
t/metadata-catalog/measured_values/", "mediaType": "application/json" }
, { "rel": "alternate", "href": "https://apex.oracle.com/pls/apex/gpi_consul
t/open-api-
catalog/measured_values/", "mediaType": "application/openapi+json" }
, { "rel": "describes", "href": "https://apex.oracle.com/pls/apex/gpi_consul
t/measured_values/" } ] }
```

Per URL Daten einfügen

- JSON Record mit gleicher Struktur wie die Tabelle
 - Per Post Request eintragen

```
[{  
  "sensor_id" : 1  
, "mdate" : "2020-11-23T19:38:47.381Z"  
, "unit" : "temp"  
, "value" : 13.5  
}]
```



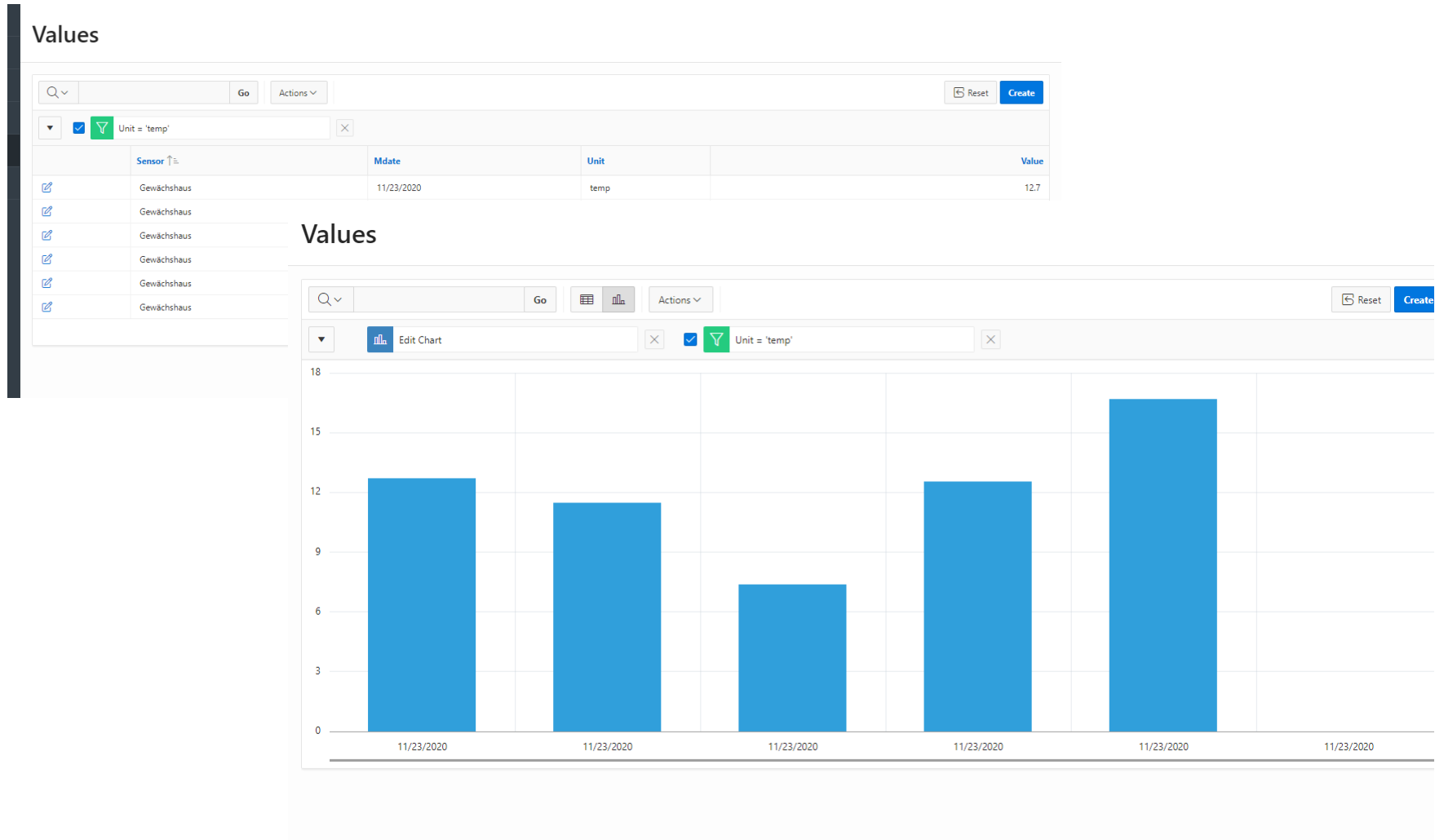
Wie muss
Das richtige
Datumsformat
lauten?

Test führt zu Fehler "ORA-01400 cannot insert NULL into MDATE"

Abfragen mit GET liefert aber das Format! =>

{"id":1,"sensor_id":1,"mdate":"2020-11-23T17:48:39.693Z","unit":"temp","value":12.7" ...

Daten in der APEX Anwendung anzeigen



Wie geht es auf dem Server weiter?

- Per REST Interface Aktions-Parameter regelmäßig vom Client abholen lassen
- Per Rest Interface Aktion auslösen
 - Per PL/SQL die entsprechen Logik zum Versand der Benachrichtigungen erstellen

Wie geht es mit dem Client weiter?

- REST Aufruf aus Python Demo in die Controller Sprache übersetzen

Summary

Fazit „Sensor Daten in der APEX Cloud“

- “Einfacher” Einstieg in die Datenhaltung in der Cloud mit Oracle APEX
 - Deklarative Programmierung
 - „Fertige“ Umgebungen steht zur Verfügung

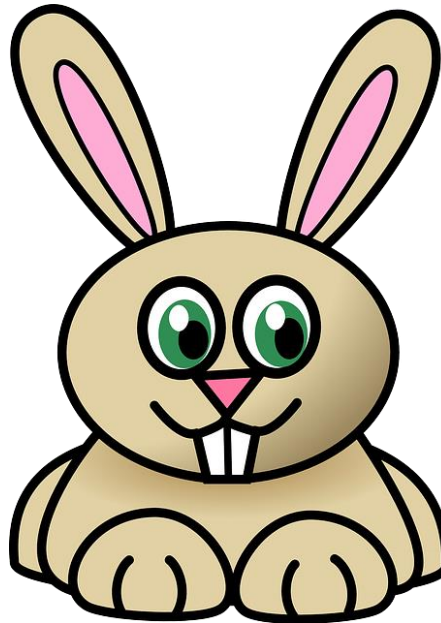
F&A

Fragen



Sensor Daten in der APEX Cloud

Fragen ?



Mehr

- Oracle REST Data Services
 - <https://docs.oracle.com/en/database/oracle/oracle-rest-data-services/index.html>
 - <https://github.com/oracle/oracle-db-tools/tree/master/ords>

- Blog Gunther Pippèrr
<https://www.pipperr.de/dokuwiki/doku.php>
 - Wieder mal eine andere Skript Library
 - <https://github.com/gpipperr/OraPowerShell>

- Bildmaterial : <https://pixabay.com>