# Building effective Restful APIs with Oracle Rest Data Services 19

## Octobre 2019

**AUTHOR:**
Iheb Eddine IMAD

**SUPERVISOR:**
Luis Rodriguez Fernandez

CERN openlab

# PROJECT SPECIFICATION

ORDS is a java component that allows to expose REST APIs directly from data model with a limited amount of configuration and code. Nowadays at CERN we are running 3.1 and the latest version is the 19.2.

The aim of this project is :

- Test the latest ORDS version and explore its latest features.
- Test the compatibility with different application servers : Apache-Tomcat and Oracle WebLogic. And different databases versions : Oracle 12, Oracle 18...
- Investigate how to package and deploy it in a container (docker) environment (kubernetes)

# ABSTRACT

In 2005, the first installation of the Oracle HTML DB came out in production. Very soon the CERN developer community adopted the technology, using it in all the areas of the organization, from administrative applications to accelerators control system. Since then the platform has evolved dramatically (PL/SQL gateway, APEX, ORDS) and nowadays the developers can write complete and secure RESTful API in the database without the help of any other backend technology.

The interest of the community for this technology has not decreased and the developers are asking for new features like OAUTH2, Open API (swagger) or even execute SQL statements on the fly via HTTP.

The goal of this project is to evaluate and test the capabilities of the latest release of the Oracle REST Data Services designing a solution that fulfills these requirements:

Throughout this report, we outline the process we followed during the nine weeks to achieve the main purpose of this project.

# TABLE OF CONTENTS

# INTRODUCTION

## 1. Project context and objectives

At CERN we already have ORDS in production and the version we are working with is 3.1, which is running on WebLogic (12.1.3. ) application server, this one is reaching the End-Of-Support, so we are going to upgrade it giving us a good opportunity of upgrading the ORDS version.

The main goal and intention of this project to test/ evaluate the latest version of ORDS, to be prepared for the future migration from the existing ORDS 3 to ORDS 19, and design a solution that fulfils these requirements:

- The proposed setup has to be compatible with Oracle Database 18c.
- The solution has to be portable among different application servers: Apache-Tomcat and Oracle WebLogic.
- It should be possible to deploy the final product in different cloud vendors like Openstack or Oracle Cloud.
- It has to be integrated in the CERN Single Sign On system.

## 2. Organization of this report

We have organized this report in four parts:

- **Part I, introduction:** project's context.
- **Part II, main technical concepts:** it gives an overview about REST, ORDS, Docker, OAuth 2, Swagger.
- **Part III, technical implementation:** installation, configuration and set up. It summarizes the work done during the nine weeks.
- **Part IV, conclusions and future work**.

# Main technical concepts

## 1. REST Overview

REST stands for 'Representational State Transfer':

- It is an architectural pattern for developing web services as opposed to a specification. Generally, REST web services communicate over the HTTP protocol.

REST uses HTTP vocabulary:

- Methods (GET, POST, PUT, DELETE, etc.,)
- HTTP URI syntax (paths, parameters, etc.,)
- Media types (xml, json, html, plain text, etc.,)
- HTTP Response codes (200, 404, 503 etc.,)

## 2. ORDS Overview

Oracle REST Data Services ORDS enables developers with SQL and database skills to develop REST APIs for the Oracle Database:

- Mid-tier Java application.
- Runs in a Java application server like WebLogic, Glassfish, or Tomcat.
- Maps standard http(s) RESTful requests to database transactions.
- Can declaratively returns results in JSON format.
- Can connect to Oracle NoSQL and Oracle container databases in Cloud.

*Figure 1 ORDS logo*

Services:

- Formally known as Oracle APEX Listener.
- Access to Relational data over HTTP(s) without installing JDBC/ODBC drivers.

- Oracle JSON collection-based schema-less access.
- Supports Swagger based Open API integration.

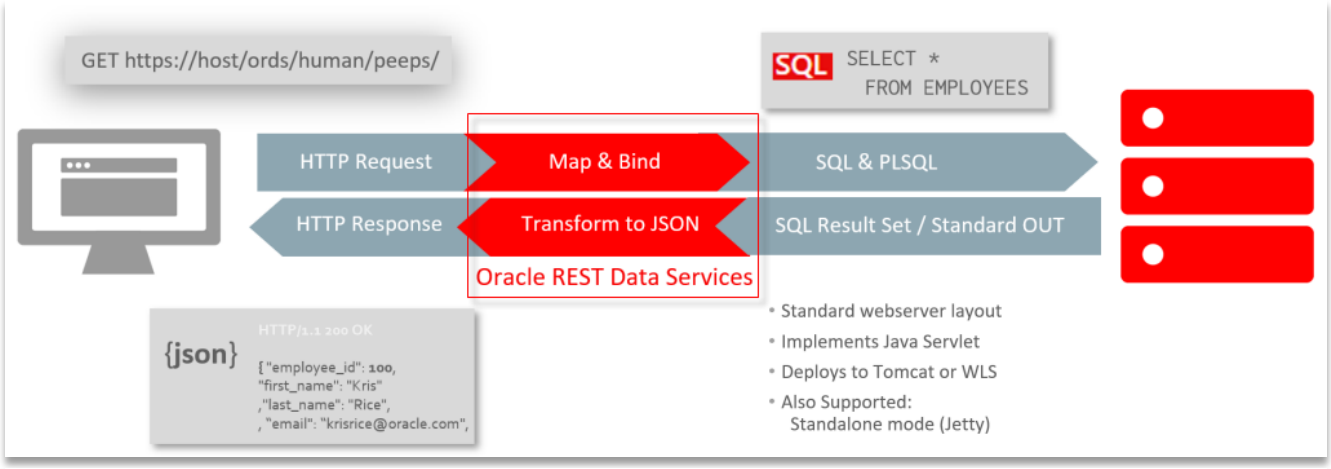


*Figure 2: Relational to JSON with ORDS*

The schema above shows the ORDS architecture and how does it work. It basically acts as a middleman between clients (applications) and the database, mapping incoming HTTP(S) requests to resource handlers for a specific URI pattern. Resource handlers can have different source types, such as query and PL/SQL. With the query source type, ORDS executes the query, converts the results into JSON, and returns the JSON to the client.

## a. ORDS Structure

The following diagram shows the structure of Web Services in ORDS.

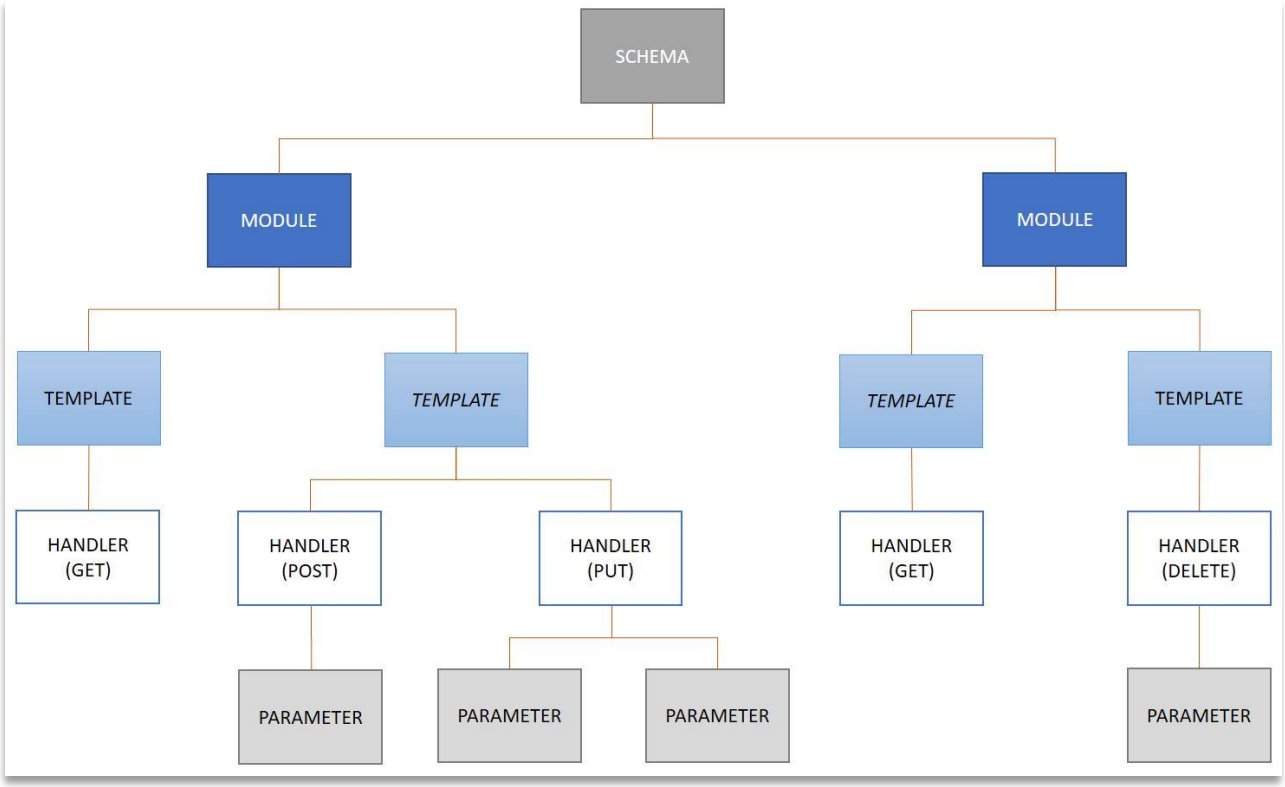


*Figure 3 Web Services structure within ORDS*

An ORDS web service consist on modules, templates, handlers and parameters. The module can have multiple templates, and a template can have a handler for each supported HTTP verbs (GET, POST, PUT and DELETE).

## b. ORDS - URL Structure

The URL for an ORDS Web Service consists of five elements:

https://<host>:<port>/ords/<schema>/<module>/<template>

**Host:** Name of the host.

**Port:** Port Number.

**Schema**: Pattern defined for the schema.

**Module**: Base path for the module.

**Template:** Pattern defined for the template.

## 3. Docker

### a. What is a container?

It is important to remember what a container image is before approaching Docker. It is a software components that includes all the files necessary for running processes: code, runtime, system tools and libraries. They can also be used to run applications for Linux or Windows.

Therefore, the containers are close to the virtual machines, but they have an important advantage. Whereas virtualization includes running several operating systems on a single system, the containers share the same kernel of the operating system and isolate the application processes from the host system. The container virtualizes the operating system instead of virtualizing the hardware as a hypervisor, which is more efficient in terms of system resource consumption. That gives the possibility to run about 4 to 6 times more instances of applications with a container than with virtual machines on the same hardware.

### b. Docker: what is it?

It is an open source software platform on an operating system to create, deploy and manage virtualized application containers. Within the container, the application's services or functions and its various libraries, configuration files, dependencies and other components are grouped. Each executed container shares the services of the operating system.

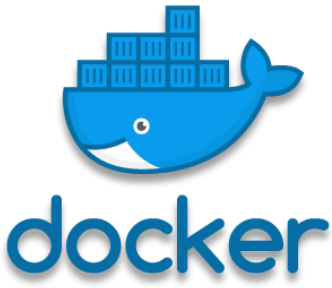


*Figure 4: Docker logo*

Originally created to work with the Linux platform, Docker now works with other operating systems such as Microsoft Windows or Apple macOS. Amazon Web Services and Microsoft Azure versions of the platform are also available.

## 4. OAuth2

OAuth is an open standard protocol for secure API authorization, short for "Open Authorization." In this context, the term API ("Application Programming Interface" abbreviation) refers specifically to an interface that functions as a data transmitter between different applications, user interfaces or web pages. Therefore, in order to avoid any risk of third-party data being intercepted and misused, it is necessary to authorize data transfer between these APIs.

## 5. Swagger

Swagger is an open-source software framework supported by a large tool ecosystem that helps developers design, build, document, and consume RESTful web services. While most users use the Swagger UI tool to identify Swagger, the Swagger tool set includes automated documentation support, code generation and test case generation.

*Figure 5: Swagger logo*

# TECHNICAL IMPLEMENTATION

## 1. Standalone mode and Tomcat deployment

Oracle now supports Oracle REST Data Services (ORDS) running in standalone mode using the built-in Jetty web server, so we don't have to worry about installing WebLogic, Glassfish or Tomcat.

For this test we assume that we have already Tomcat server and Oracle database installed, we start the installation of ORDS by unlocking the SYS user and common public users:

```
CONN / AS SYSDBA
ALTER USER SYS IDENTIFIED BY OraPassword1 ACCOUNT UNLOCK;
ALTER SESSION SET CONTAINER = pdb1;
ALTER USER APEX_PUBLIC_USER IDENTIFIED BY OraPassword1 ACCOUNT UNLOCK;
ALTER USER APEX_REST_PUBLIC_USER IDENTIFIED BY OraPassword1 ACCOUNT UNLOCK;
ALTER USER APEX_LISTENER IDENTIFIED BY OraPassword1 ACCOUNT UNLOCK;
```

We make a directory to hold the configuration.

```
$ mkdir   /ords/conf
```

We can edit the "ords/params/ords_params.properties" file provided with the ORDS software to set the appropriate parameters for the installation.

We install ORDS using the following command:

```
$ $JAVA_HOME/bin/java  -jar ords.war
```

After that we lock the SYS user:

```
ALTER USER SYS ACCOUNT LOCK;
```

## 2. Tomcat deployment

Moving to the deployment on Tomcat is quite easy.

The first step is to copy the APEX images to the Tomcat "webapps" directory.

```
$ mkdir $CATALINA_HOME/webapps/image/
$ cp -R /tmp/apex/images/* $CATALINA_HOME/webapps/image/
```

And then we copy the "ords.war" file to the Tomcat "webapps" directory.

```
$ cd /ords
$ cp ords.war $CATALINA_HOME/webapps/
```

Starting and stopping ORDS Under Tomcat.

```
$ $CATALINA_HOME/bin/startup.sh
$ $CATALINA_HOME/bin/shutdown.sh
```

ORDS should now be accessible using the following type of URL" http://<server-name>:<port>/ords/".

## 3. ORDS deployment on Docker

In this part we assume that we already have a suitable installation of the Docker Engine. The Docker file and scripts that we use can be found here. The build expects the following file system:

- OpenJDK 11
- Tomcat 9.0.x
- Oracle REST Data Services (ORDS) 19.x
- Oracle Application Express (APEX) 18.x
- Oracle SQLcl 18.x

After downloading the software and place it in the "software" directory, we build the image using the following command.

```
$ docker build -t ol7_ords:latest .
```

We notice the build phase doesn't configure ORDS. That is done on the first run of a container where we provide ORDS credentials using environment variables, As explained following.

```
$ docker run -dit --name ol7_ords_con \
        -p 8080:8080 -p 8443:8443 \
        --network= \
        -e "DB_HOSTNAME= " \
        -e "DB_PORT=1521" \
        -e "DB_SERVICE= " \
        -e "APEX_PUBLIC_USER_PASSWORD= " \
        -e "APEX_TABLESPACE= " \
        -e "TEMP_TABLESPACE=TEMP" \
        -e "APEX_LISTENER_PASSWORD= " \
        -e "APEX_REST_PASSWORD= " \
```

Once the container is running, we can connect to a bash shell, stop and start it using the following commands.

```
$ docker exec -it ol7_ords_con bash
$ docker stop  ol7_ords_con
$ docker start ol7_ords_con
```

## 4. Exploring and testing ORDS 19

### a. Creating Web Services

To avoid writing all the PL/SQL code here I preferred to put all the scripts in one file HERE. This file contains procedures to show how to create a simple set of Web Services using ORDS and cover all four supported HTTP verbs (or methods, as they are properly called) POST, GET, PUT, PATCH, and DELETE. These correspond to create, read, update, and delete (or CRUD) operations, respectively.

And the main script blocks this file contains are:

- Create a new user.
- Enable the ORDS schema.
- Define the resource module.
- Create the templates for the module.
- Create a template handler for each supported HTTP method (GET, POST, PUT or DELETE).
- Define parameters for some methods.
- SQL query to review all of the Web Services defined.

## b. Generating a Swagger document through ORDS

### 1- Using the open-api-catalog (Swagger Editor):

The open-api-catalog output is in Open API (Swagger) 2.0 format, which makes it really simple to generate documentation and an example of the calling code in several programming languages.

An overview of the contents of an ORDS enabled schema can be displayed in a browser (GET HTTP method) using the following type of URL. The output is a JSON document:

**-Format :** http://server:port/ords/<connection>/<schema-alias>/open-api-catalog/<object-alias>

**- Example:** http://localhost:8080/ords/api/open-api-catalog/hr/v1/

Then we can go to the online Swagger Editor (https://editor.swagger.io/), we can paste in the output JSON text, which converts to YAML and displays the available endpoints.

### 2- Using the swagger-UI server:

What was not obvious in the previous version of APEX is that there is also a new setting under the REST section. This setting contains an URL which points to a SWAGGER UI 2.0 server. If the instance setting is set, the URL that generates the web

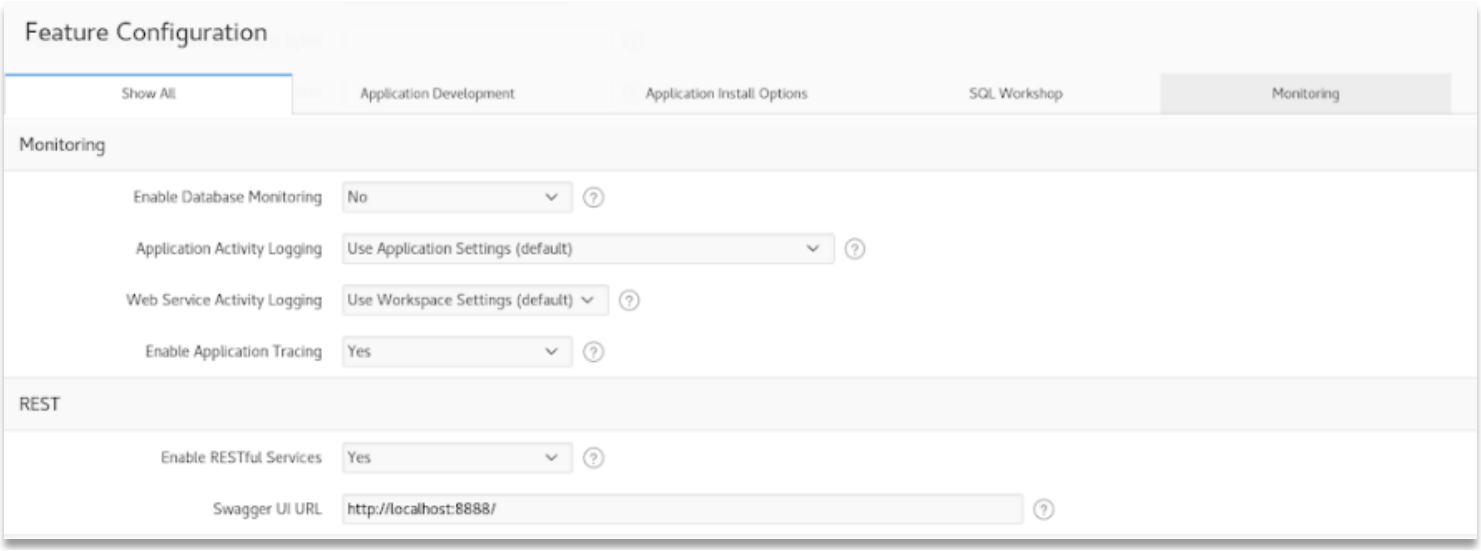


*Figure 6: Setting the Swagger UI URL*

service's swagger JSON document will be passed to the Swagger UI server. If there is no URL specified, raw JSON will be produced.

Setting up the Swagger server is actually quite easy. It is a simple set of HTML, Javascript and CSS files that can be downloaded from here (https://swagger.io/tools/swagger-ui/) and unzipped into a directory into the web root of our local web server. During my tests I pulled a pre-built docker image of the swagger-ui directly from Docker Hub and I ran it in a separate container http://localhost:8888/. Then, from the **ORDS REST Workshop** section, we navigate to the module definition level and click the Generate Swagger Doc button.

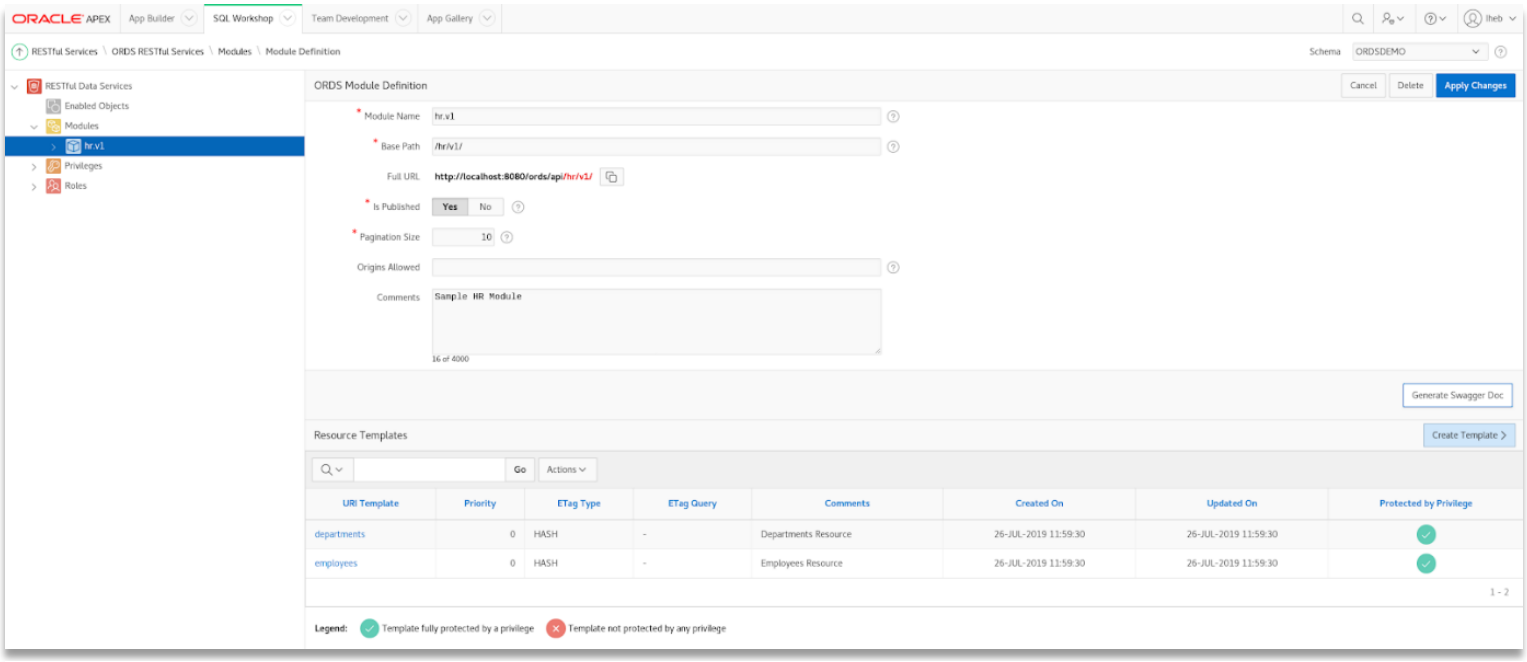


*Figure 7: REST workshop where we can generate Swagger documents*

If the Swagger UI URL is set correctly at the instance level, APEX will forward the URL of the documentation to the Swagger UI Server. The server must be able to reach back to the documentation URL. As long as it can, we will not only get documentation but be able to test the services as well.

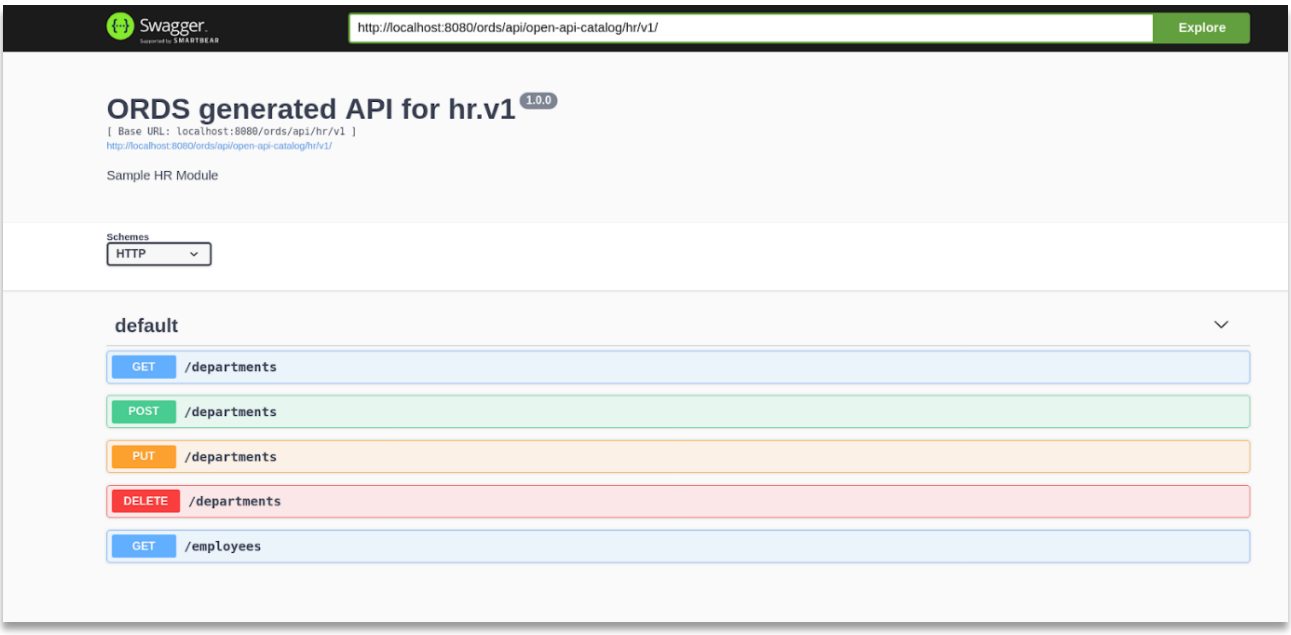


*Figure 8: Documentation using Swagger UI*

## c. Protecting Web Services with OAuth2

The next step was protecting these Web services to ensure they can only be accessed by the users we specify. To ensure the Web Services are only accessible by the appropriate users (clients) we use OAuth2. This involves creating roles and privileges to protect the Web Services and the creation of clients, which can be for individuals or applications. Each client can then be granted access to one or more roles in order to access the associated Web Services.

### OAuth Methods:

ORDS implements three OAuth2 methods:

- **Client Credentials:** Two-stage messaging system for server-to-server where there is no human interaction. To create an access token a client secret and client id are needed.
- **Authorisation Code:** Three-stage process when human interaction takes place. Once the user enters its credentials and allows the client to use the

API, an authorization code is generated for the creation of the access token for Web Service calls authentication.

- **Implicit:** Double-stage phase where human interaction happens. It produces a key for Web Service calls authentication.

In my tests I have used the Client Credentials method to protect the previously created Web Services.

Projecting a Web Service is a four-step process:

- Create the ORDS roles that will be granted to the clients.
- Create the privileges that will protect the Web Services and link to one or more roles.
- Map the privileges on to the required URL patterns.
- Creating a Client to generate an access token.

Also, to avoid putting the whole script in this document I preferred to put it in same file where I put all PL/SQL procedures to create Web Services.

# CONCLUSION AND FUTURE WORK

## 1. Conclusion

The main aim of this project was preparing for the future migration of ORDS. However, making and succeeding a migration when switching from a version to another is an important step and is often a source of stress.

But, when a migration is well prepared, there is very little reason to worry. For this purpose, were the main tasks in my project that I summarized below:

- I tested the compatibility of the latest ORDS with the CERN databases.
- The examples that I have developed are being reuse and are part of CERN documentation.
- Tests can assist ORDS service managers in switching from VM to container environment.

## 2. What's next:

About the future work that might or should be done:

- Further tests are needed (more specifically test and secure web service with CERN single sign on).
- Upgrade of the WebLogic infrastructure.
- Deployment of the latest ORDS version.

## ACKNOWLEDGMENT

I would not exaggerate when I say that this summer was one of the best I have ever had, the internship opportunity I had with CERN was a great chance for learning and professional development. I am also grateful for having a chance to meet so many wonderful people and professionals who led me though this internship period.

My greatest thanks go to my supervisor **Luis Rodriguez Fernandez** who has been guiding me throughout the program. I think him for all the time devoted to me, for his total availability, for his kindness and especially for his remarks and constructive criticism. I am also thankful to all the IT-DB group members for their warm welcome, their help during this project and for making this experience very pleasant and rewarding.

My thanks also go to the Open lab team for the excellent work they have done: lectures, visits to external organizations (IBM, OpenSystems and ETH Zurich) and to CERN experiments (CMS and the Antimatter Factory).

# REFERENCE and WEBOGRAPHY

[1] https://www.oracle.com/database/technologies/appdev/rest.html

[2] https://www.oracle.com/middleware/technologies/weblogic.html

[3] https://cern.service-now.com/service-portal/sls_grid.do

[1] https://docs.oracle.com/en/database/oracle/oracle-rest-data-services/19.1/index.html

[2] https://oracle-base.com/articles/linux/docker-oracle-rest-data-services-ords-on-docker

[3]https://oracle-base.com/articles/misc/oracle-rest-data-services-ords-installation-on-tomcat.

[4] https://docs.oracle.com/javame/8.0/api/oauth2/api/com/oracle/oauth2/OAuth2.html

[5]https://oracle-base.com/articles/misc/oracle-rest-data-services-ords-open-api-swagger-support