**OPM Manager - User Guide**

* **Introduction**

This guide explains how to launch, and to use the ‘OPM Manager’ component.

The ‘OPM Manager’ component is a standalone server that manages the operator authorizations. The operator rights are persisted thanks to a relational database. By using this component, we are sure that the operator authorizations are always consistent for every web application servers (Tomcat), and we provide at the same time a unique access for modifying the operator rights.

In order to manage the operator authorizations, the ‘OPM Manager’ component provides REST interfaces which will ease the development of a future OPM edition UI.

* **Architecture**

LDAP

WebServer

(Tomcat)

**mwt-opm-permission.xml**

Authentification

Autorisations(file)

Web Browser

(Chrome)

Existing

PostgreSQL

OPM SCADA schema

OPM Manager

OPM Loader

To do

REST / XML

JDBC

OPM Edition UI

REST / JSON

New

Cluster

Note. The integration of the “OPM Loader” component in an existing web application is explained in another document named “Dynamic loading of the permissions file.docx”.

* **Quick Started**

This section will explain how to run an “OPM Manager” component without installing anything except java 1.8. The idea is to get quickly an overview of the different features.

By default, the delivery is packaged to run with a RAM database (H2), so we don’t need to install a posgresql for the moment.

**Step 1**

Unzip the archive file ‘opmmgt-1.0.0.zip’. You should get the following structure:

opmmgt

|\_ lib

|\_ application.properties

|\_ init\_data.sh

|\_ opmmgt-1.0.0.jar

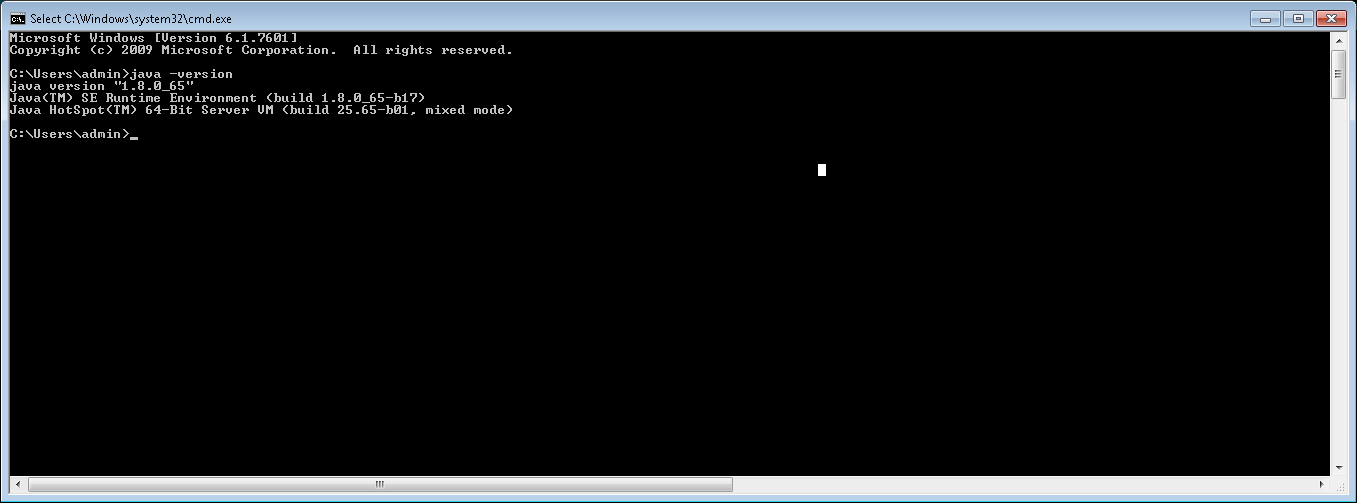
|\_ README.md

|\_ schema.sql

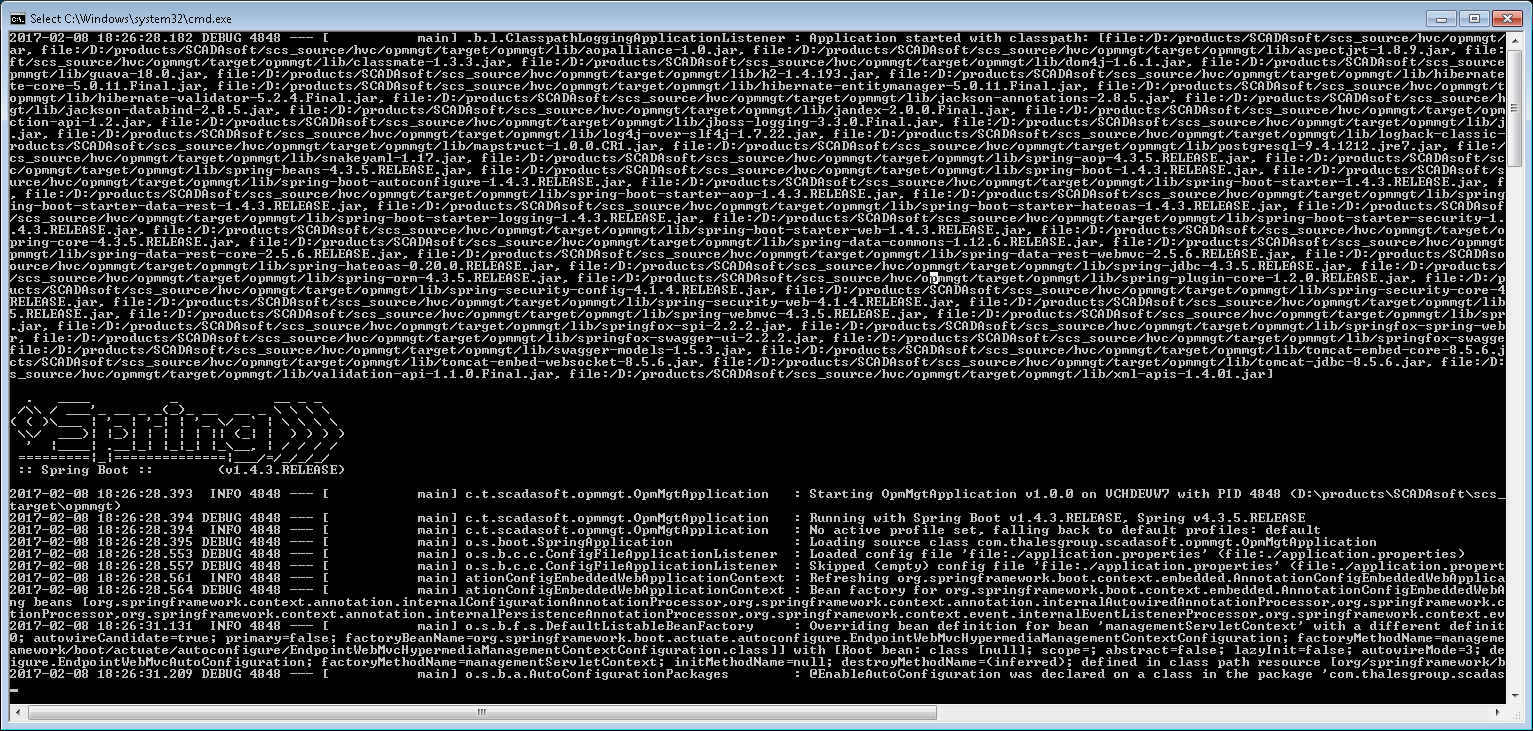
|\_ startup.cmd

**Step 2**

Check that your ‘java’ executable is the expected one (java 1.8):

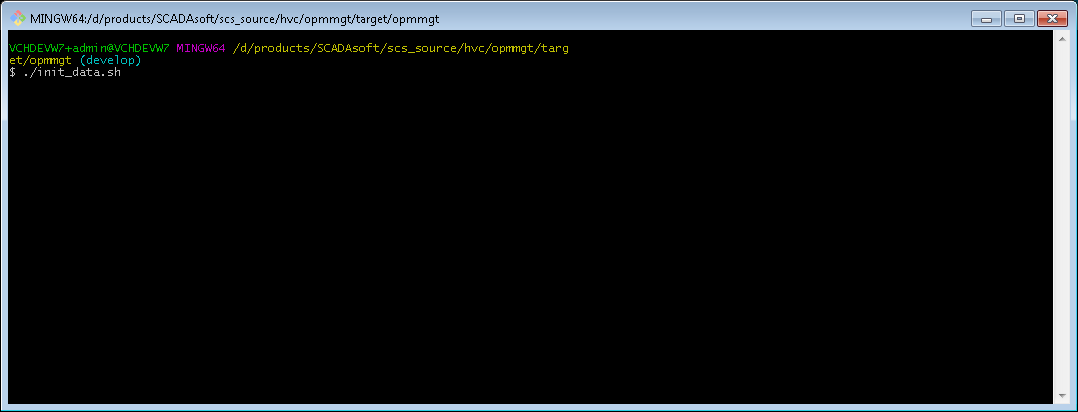


Launch the script ‘startup.cmd’ (double click). You should get the following window:



**Step 3**

Import the data in the RAM database H2 thanks to the script ‘init\_data.sh’, and a unix terminal for windows like git bash.



**Important:**

The script ‘init\_data.sh’ uses the executable ‘curl’, so check that you get the executable, and the last one could be called from anywhere in your unix terminal for windows.

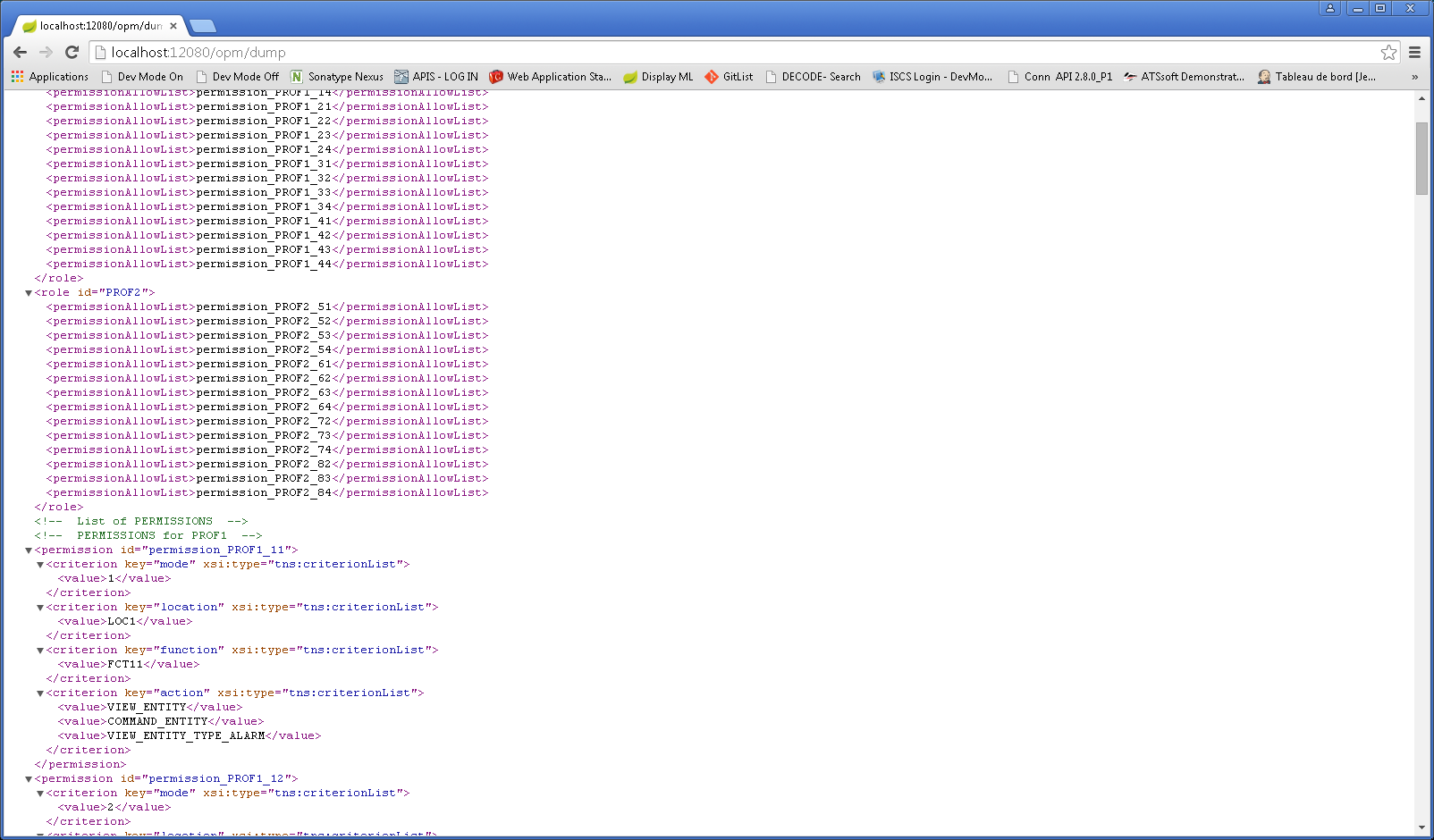
* **How to retrieve the operator authorizations in xml format (hypervisor compliant)?**

Launch ‘Chrome’ web browser application.

Type the following url:

[**http://localhost:12080/opm/dump**](http://localhost:12080/opm/dump)

You should get something like this:



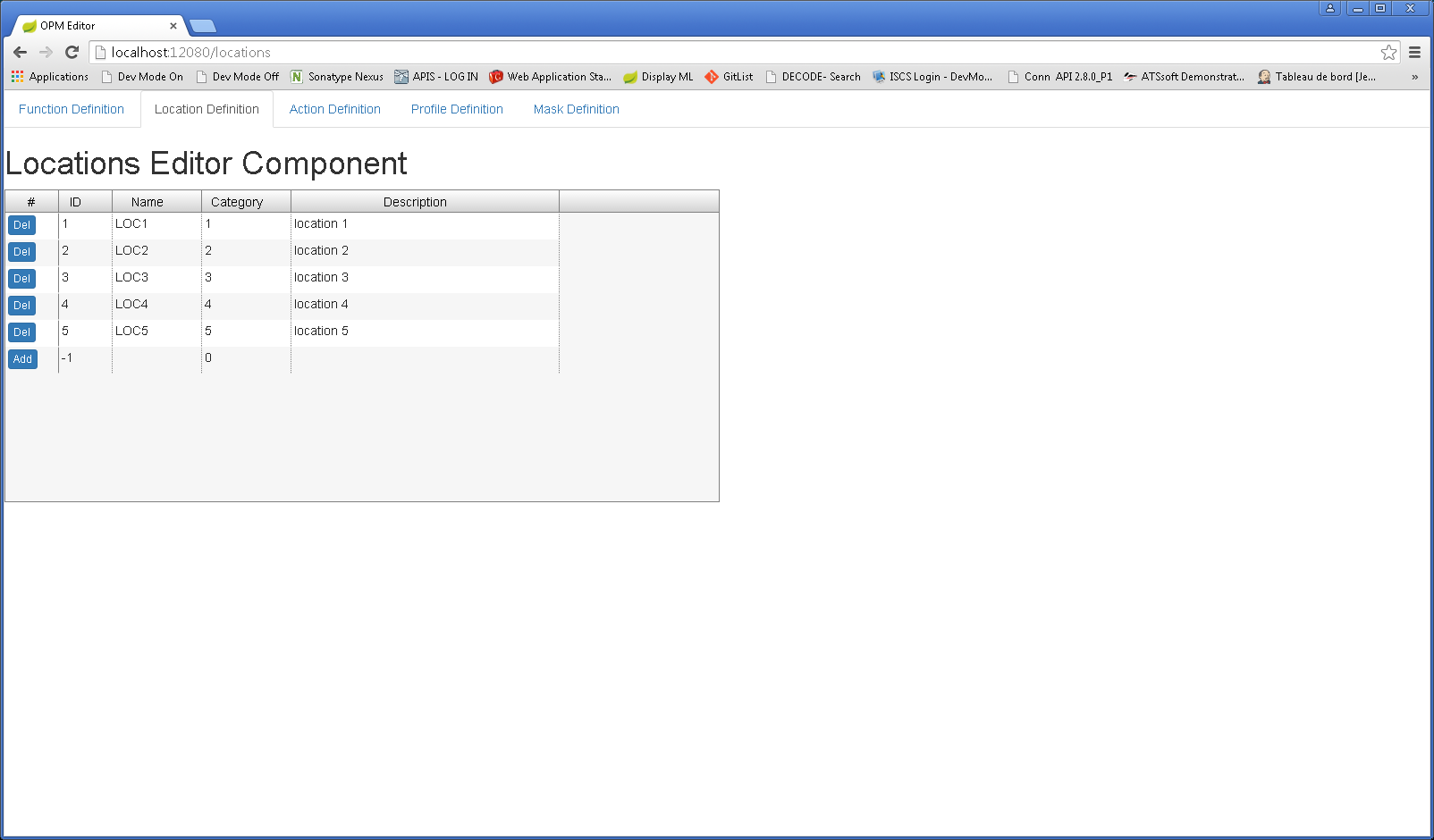
* **How to access a sample OPM edition UI?**

Launch ‘Chrome’ web browser application.

Type the following url:

**http://localhost:12080**

You should get something like this:



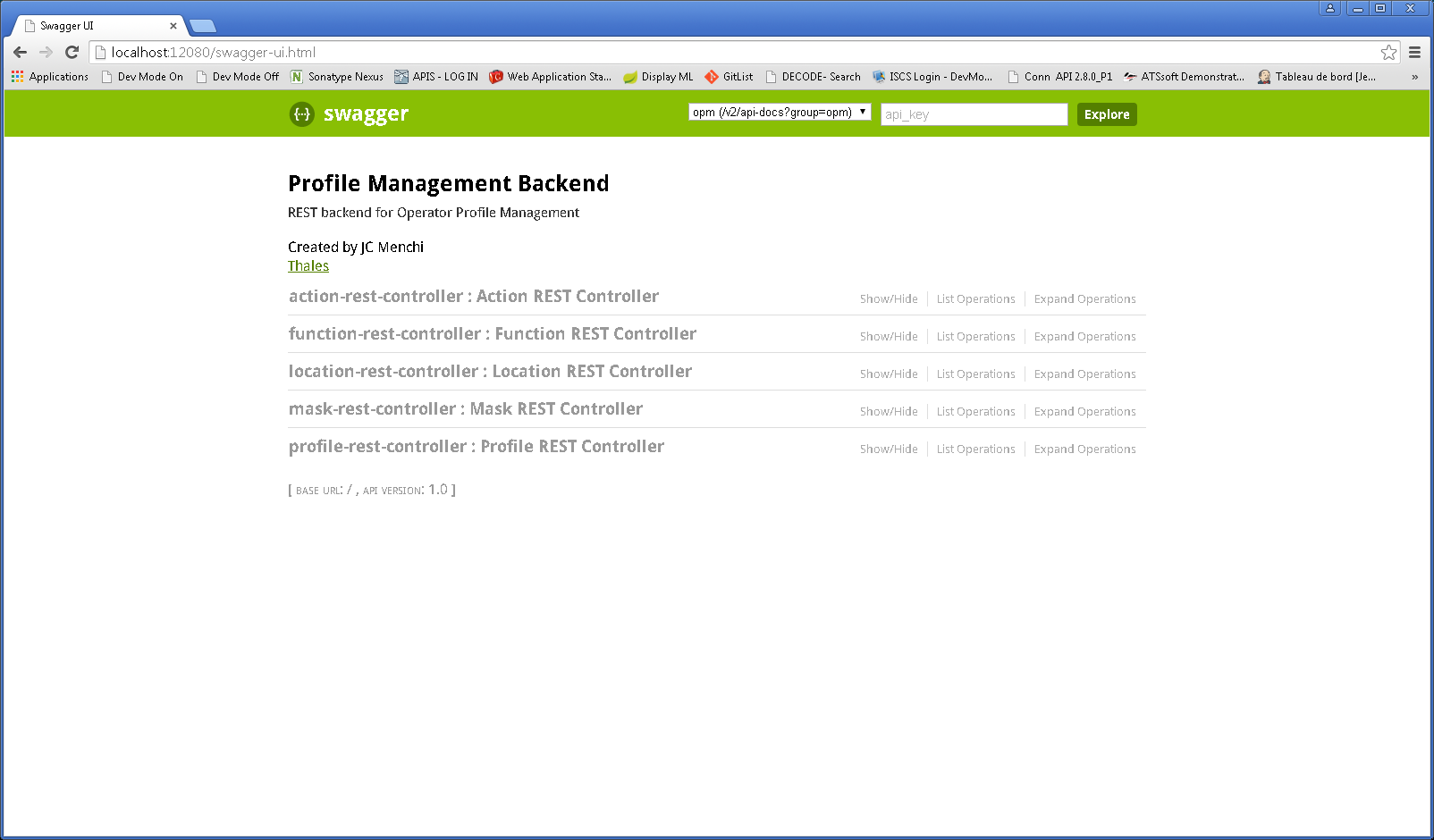
* **How to access to REST APIs documentation?**

Launch ‘Chrome’ web browser application.

Type the following url:

**http://localhost:12080/swagger-ui.html**

You should get something like this:



* **How to use PostgreSQL instead of H2 (RAM database)?**

By default, the ‘OPM Manager’ component does not connect to PostgreSQL, but uses a RAM database H2. Each time you shut down the application all data are lost.

**Step 1**

Install PostgreSQL.

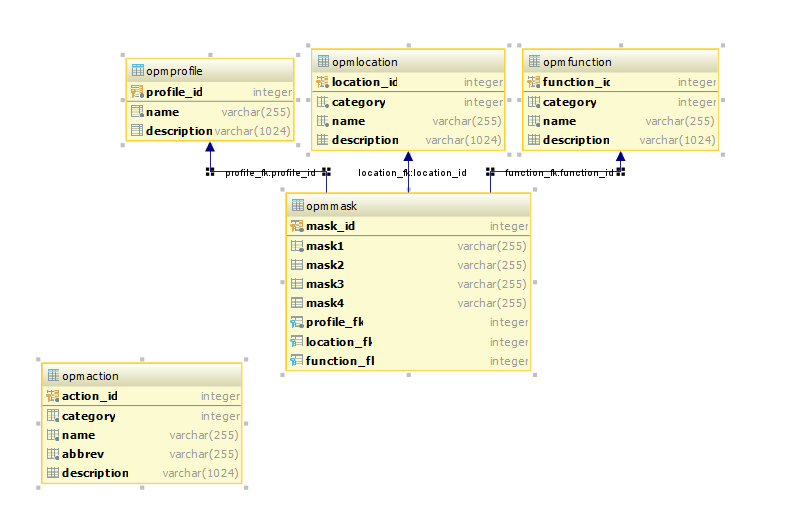
For information, the version tested here is ‘postgresql-9.4.1-3-windows-x64’.

**Step 2**

Create the database user, schema, tables by typing the following command in a ‘cmd’ window:

psql.exe -U postgres -f schema.sq

For information, this is the used database schema:



**Step 4**

In the file ‘application.properties’, uncomment the following lines:

spring.jpa.database=POSTGRESQL

spring.datasource.platform=postgres

spring.jpa.show-sql=true

spring.jpa.hibernate.ddl-auto=validate

spring.datasource.url=jdbc:postgresql://localhost:5432/opmdb

spring.datasource.username=opmuser

spring.datasource.password=opmuser

spring.jpa.properties.hibernate.default\_schema = public

spring.jpa.database-platform=org.hibernate.dialect.PostgreSQLDialect

spring.jackson.serialization.fail-on-empty-beans=false

**Step 5**

Launch the script ‘startup.cmd’ (double click).

**Step 6**

Import the data thanks to the script ‘init\_data.sh’.