PLATFORM IMPLEMENTATION AND INTEGRATION – SECOND PROTOTYPE

SmartH2O

Project FP7-ICT-619172

Deliverable D6.4 WP6

Deliverable
Version 2.3 – 29 July 2016

Document. ref.:
D6.4 SETMOB.WP6.V2.3

Programme Name:	ICT
Project Number:	619172
Project Title:	SmartH2O
Partners:	Coordinator: SUPSI Contractors: POLIMI, SETMOB, TWUL, SES, MOONSUB
Document Number:	smarth2o.D6.4.SETMOB.WP6.V2.3
Work-Package:	WP6
Deliverable Type:	Document
Contractual Date of Delivery:	31 March 2016
Actual Date of Delivery:	29 July 2016
Title of Document:	Platform Implementation and Integration
Author(s):	Luigi Caldararu, Piero Fraternali, Chiara Pasini, Giorgia Baroffio, Eleonora Ciceri, Ilio Catallo, Alessandro Facchini, Joan Carles Guardiola, Natalia Bakkalian, Andrea Emilio Rizzoli.
Approval of this report	Under review
Summary of this report:	D6.4 Platform Implementation and Integration- Second Prototype
History:	See document history
Keyword List:	platform, implementation, integration, architecture, component, services, data model
Availability	This report is public

This work is partially funded by the EU under grant ICT-FP7-619172

Document History

Version	Date	Reason	Revised By
1.0	01/12/2015	First draft	Luigi Caldararu
1.1	16/12/2015	SETMOB contribution: Code repository URLs alligned	Luigi Caldararu
1.2	15/01/2016	SETMOB contribution: SMDM update	Luigi Caldararu
1.3	18/03/2016	POLIMI contribution: SES demo case	Chiara Pasini
1.4	18/03/2016	POLIMI contribution: Drop!	Jacopo Mossina
1.5	21/03/2016	POLIMI contribution: Drop! backend, customer portal	Chiara Pasini
1.6	22/03/2016	SETMOB: EMIVASA demo case	Luigi Caldararu
1.7	24/03/2016	SETMOB: state of development	Luigi Caldararu
1.8	29/03/2016	SUPSI: ABM and MUB	Alessandro Facchini
1.9	30/03/2016	SETMOB: Deliverable finalized	Luigi Caldararu
2.0	31/03/2016	Final revision	Andrea Emilio Rizzoli
2.1	26/07/2016	Update of Executive Summary	Luigi Caldararu
2.2	26/07/2016	Adition of supporting flows for SMDM, Customer Portal, DROP!	Luigi Caldararu
2.3	27/07/2016	Adition of information on implementation, integration, smart metering infrastructure described in other deliverables	Luigi Caldararu

Disclaimer

This document contains confidential information in the form of the SmartH2O project findings, work and products and its use is strictly regulated by the SmartH2O Consortium Agreement and by Contract no. FP7- ICT-619172.

Neither the SmartH2O Consortium nor any of its officers, employees or agents shall be responsible or liable in negligence or otherwise howsoever in respect of any inaccuracy or omission herein.

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7-ICT-2013-11) under grant agreement n° 619172.

The contents of this document are the sole responsibility of the SmartH2O consortium and can in no way be taken to reflect the views of the European Union.

Table of Contents

Executive Summary

This is the accompanying document of the Deliverable **D6.4: Platform Implementation and Integration – Second prototype**, which, according to the Description of Work (DoW), is a software deliverable containing the second prototype of the SmartH2O platform. Beyond the features included in the First prototype of the SmartH2O platform (the infrastructure to collect and organize the water consumption data of the consumers, the implementation of the first user behaviour models) the second prototype includes upgrades for the chronologically first demo case (SES, Locarno, Switzerland) and initial installations for the second demo case (EMIVASA, Valencia, Spain):

- upgrade of Customer Portal basic version (SES);
- upgrade of Advanced (gamified) version of the Customer Portal which includes the Gamification Engine (SES);
- upgrade of Enterprise Service Bus and Authentication Gateway (SES);
- initial installation of the Customer Portal basic version (EMIVASA);
- initial installation of the advanced (gamified) Customer Portal (EMIVASA);
- initial installation of Enterprise Service Bus (EMIVASA);
- the initial version of the Games Platform (SES);
- the initial version of the Water Utility Admin portal for the water utilities (SES);
- the initial version of the Social Network Crawler and Data Analyzer (installed on development site);
- Models of User Behaviour and Agent Based Modelling (installed on development site).

At this moment, the users are able to start monitoring their respective performance indicators for water savings as well as comparing their performance indicators with the neighbourhood average.

The deliverable is structured as an installation guide of SmartH2O platform – second prototype, describing specific demo-cases and includes the source code and the documentation for using the platform.

The second prototype of the SmartH2O platform is available in internet for the both demo cases.

For SES demo case, the demo site is available in internet at https://www.smarth2o.ch

Testing credentials:
Basic profile
- username: ses.smarth2o
- password: *smarth2o*

Advanced profile
- username: ses2.smarth2o
- password: *smarth2o*

The administration profile of the SmartH2O platform is available in internet at https://www.smarth2o.ch/community/admin

Testing credentials:
- username: admin
- password: *admin123admin*

For EMIVASA demo case, SmartH2O platform is available in production environment at https://www.emivasa.es/VirtualOffice After logging-in to water utility site, the end-consumer can select SmartH2O from the services that EMIVASA offers.

Since all EMIVSA user accounts are real, testing credentials cannot be made public at the moment of realesing of this deliverable, for security reasons.

As specified in the Description of Work, Workplan Tables section, page 3, D6.4 is a software package. It is important to note that the present document has an accompanying role that in the following describes the actual platform components and the steps needed for performing installation and deployment by a third party. The actual software packages have been deployed at demo case premises in Locarno, Switzerland for SES and in Valencia, Spain for Emivasa while a development and testing infrastructure is being under continuous update and maintenance at SETMOB premises.

1. SmartH2O platform – Second prototype

1.1 State of the development process

In the context of the first platform prototype, the software production has been performed in a development and testing environment, on a server provided and managed by SETMOB while using smart metering data provided by SES through a FTP connection using an automated nightly job.

The second SmartH2O platform prototype has been deployed in two production software environments hosted by SES in Locarno, Switzerland and EMIVASA in Valencia, Spain. Prior to the launch into production, both second prototype deployments have passed standard incremental tests by undergoing: alpha tests using pools of selected users known to the developers; beta tests with real world users that accepted to provide feedback for the SmartH2O project.

At the same time, the development environment is continually used for developing, deploying and testing new features, security configurations and SmartH2O platform demos for various prospects or industry events. In addition of using the development server, software development deployments have been performed using the partners' own infrastructure, as for example in the cases of POLIMI's Social Network Crawler and Data Analyzer and SUPSI's Models of User Behaviour and Agent Based Modelling.

Figure 1 shows the UML component representation of the SmartH2O platform. The representation depicts the state of development of the software components at the moment of releasing the second prototype of the SmartH2O platform. The components' background colors have the following meaning:

- Green – completed
- Yellow – in progress, in various degree of execution
- Red – development not started

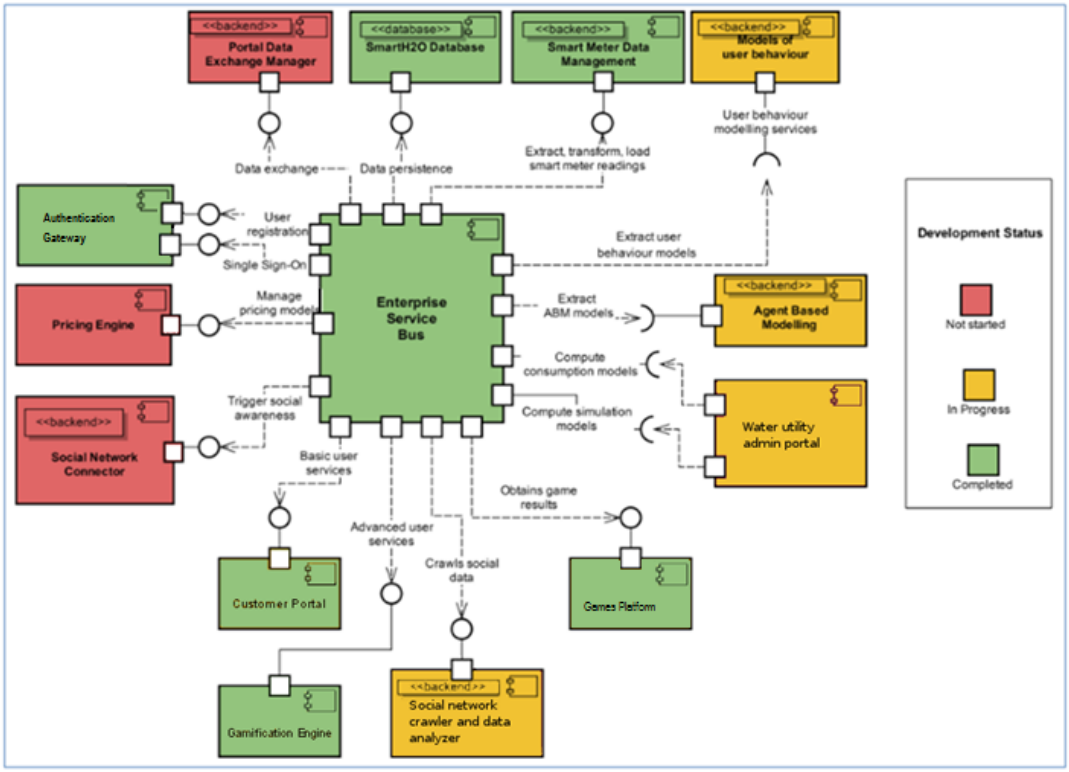

Figure 1. Overview of the main components of the SmartH2O architecture and their

current state of development within the second prototype.

The main components of the integrated SmartH2O platform developed and installed during the initial iteration of the first platform prototype were:

- SmartH2O platform database.
- Smart Meter Data Management component - having the role of meter reading data provider for the platform.
- Enterprise Service Bus which is the centralizing component acting as:
  - Single Point of Access.
  - Transaction Manager.
  - Security Manager.
- Customer portal – first version.
- Gamification Engine –first version.

With respect to the initial prototype, the second platform prototype the following updates were made available:

- Upgrade of **Smart Meter Data Management** component.
- Upgrade of **Customer Portal.**
- Advanced (gamified) version **of the Customer Portal** which includes the **Gamification Engine** (new release).
- The initial version of the **Games Platform** (new release).
- **DROP! online game** (new release).
- **DROP! the question game** (new release).
- The initial version of the **Water Utility Admin portal** for the water utilities (new release).
- The initial version of the **Social Network Crawler and Data Analyzer** (released on development site).
- **Models of User Behaviour** (released on development site).
- **Agent Based Modelling** (released on development site).
- An upgrade of the **ESB integration services.**
- **Upgrade of the database** model.

2. Installation guide

The software sources referred in the following sections are available as public projects on the SmartH2O project account on Bitbucket– available at https://bitbucket.org.

Bitbucket – is a free Git based source code management and collaboration solution in the cloud.

The SmartH2O project credentials for software source management on Bitbucket are:

`User:` `smarth2o-guest`

`Password:` *`smarth2oguest`*

2.1 SES Demo Case

2.1.1 SmartH2O Database

The **SmartH2O Database** is the central repository of the information that is either common to all the SmartH2O components or supports the coordination and exchange of messages among them. Not all the data of SmartH2O will reside in the SmartH2O database; for example, commercial data about the water consumers maintained by the water utility will be stored in the proprietary systems of the company.

The database server for the Smarth2O platform database is an instance of MySql 5.5+.

The Data Definition Language (DDL) script for creating the database structure is made available as Appendix A, section 1 of the present document.

The dump file of the database corresponding to the second prototype of the platform, is available on Bitbucket at:

`https://smarth2o-guest@bitbucket.org/smarth2o/ses-smarth2odb-dump.git`

2.1.2 Smart Meter Data Management Component

The **Smart Meter Data Manager** (SMDMC) deals with the acquisition of data streams from smart meters and with their consolidation within the SmartH2O database. It implements the data privacy and security policy of the utility company and ensures that only admissible (e.g., aggregated, anonymised) data is stored in the platform database.

It is implemented using Big Data parallel processing technologies the main advantage of which is obtaining scalability when processing increasing amounts of data by just adding and registering new hardware without making any software changes.

This component implements the ETL (Extract, Transform, Load) process with no assumption of the utility of the data, so it can be reused in other Big Data processing projects.

Requirements:

- OS: Unix
- Java 7+: `https://home.java.net`
- Apache Maven 3.3+: `https://maven.apache.org`
- MySQL 5.1: `https://www.mysql.com`

For a better view of the component functionality, in the following we depict a representation of the processing flow performed by the SMDMC .

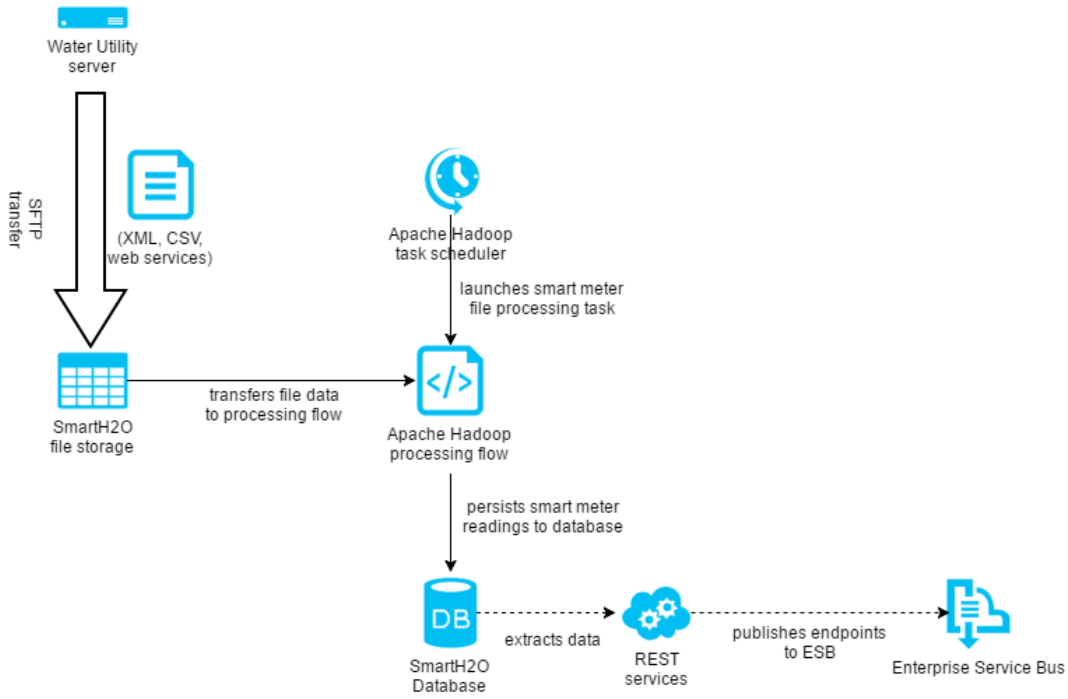

Figure 1.1 Smart Meter Data Management - data acquisition flow

The SMDM component interface specification has been presented in D6.2 Platform architecture and design, section 3.7, page 43

The component services have been described In D2.3 Functional specification, section 4.2, page 21

The business, technical and security issues regarding the SMDM component integration into the water utility service infrastructure have been addresses in D2.2 Final requirements, section 17.4, page 141

SMDMC uses a Big Data processing approach using an Apache Hadoop[1] cluster over HDFS (Hadoop Distributed File System) DataNodes[2], running MapReduce 2.0 (MRv2)[3] for aggregation, PIG[4] (SQL like) scripts for performing logical operations, scheduled by Oozie[5] jobs and loading the data in the database with SQOOP[6].

For installing the SMDMC component, first a Hadoop cluster must be created, using Apache Ambari.

To install Apache Ambari using wget (https://www.gnu.org/software/wget).

```
wget http://public-repo-1.hortonworks.com/ambari/centos6/1.x/GA/ambari.repo
cp ambari.repo /etc/yum.repos.d
yum install ambari-server
ambari-server setup
```

[1] Apache Hadoop https://hadoop.apache.org/

[2] HDFS DataNode https://wiki.apache.org/hadoop/DataNode

[3] YARN http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html

[4] PIG https://pig.apache.org/

[5] OOZIE http://oozie.apache.org/

[6] SQOOP http://sqoop.apache.org/

After the setup is completed, start Ambari service:

```
ambari-server start
```

The Ambari setup is straightforward and can be customized for the infrastructure that is deployed on by using the web interface. To start the web interface, open an internet browser (Mozilla Firefox or Chrome) and login to the admin screen: http://localhost:8095/#/login.

In Figure 2 it is shown how to add new hosts to the Hadoop cluster.

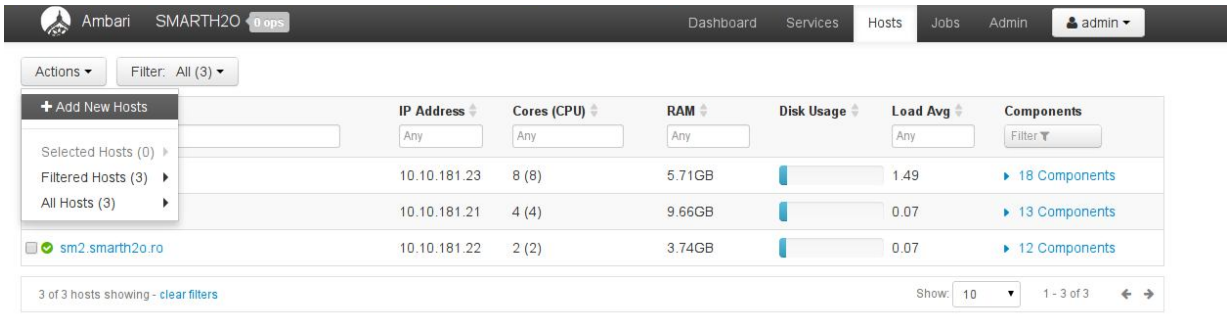

Figure 2. Adding new hosts to Hadoop cluster.

In Figure 3 it is shown how to install a new Ambari component.

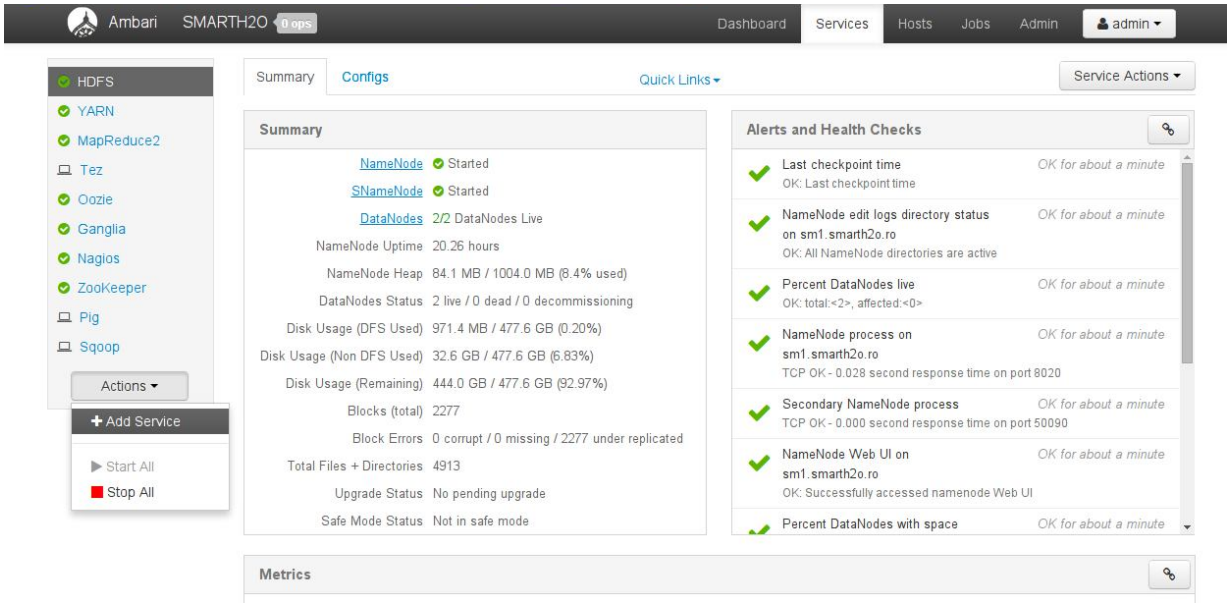

Figure 3. Installing a new Ambari component.

The source for the manager and transform component are available at:

https://smarth2o-guest@bitbucket.org/smarth2o/sh2osmdmctransform.git

https://smarth2o-guest@bitbucket.org/smarth2o/sh2osmdmcmanager.git

The second prototype divided the processing workflow in two stages: a basic daily workflow and a weekly workflow to ensure meter processing redundancy and recovery of possible data lost. Both workflows contain developments (e.g. adding various data level aggregations) and optimizations in order to ensure optimum data access for the data wrapping backend services.

The updates in the SMDM workflows corresponding to the second prototype of the platform are available on Bitbucket at the following addresses.

Daily workflow:
https://bitbucket.org/smarth2o/sh2osmdmcmanager/src/3a0383853b274ac367cafbac46760ae39841f430?at=master

Weekly workflow:
https://bitbucket.org/smarth2o/sh2osmdmcmanager/src/1e0d69f5efa5dbab6cb291e6b6d9523ea7134063/?at=weekly

To build the components this command must be executed in the folder where the file pom.xml is located:

```
mvn package
```

As a result a target folder is created and inside that a jar file containing the application.

On the current platform that is based on Linux CentOS 6.5 an init file must be created to start the SMDMC Manager component. A sample file is located in the Bitbucket repository at https://bitbucket.org/smarth2o/sh2osmdmcmanager/src/8f52860fc13df9b78f8bed3f4d217f5f27cea46a/centos6.5-smdmc-init?at=master

The **SMDMC** subcomponent relies on the fact that a FTP server is previously configured and the water utility has access to store the XML files with the meter data on the server. Also an Oozie server must be installed and configured to work with the underlying Apache Hadoop infrastructure using the Apache Ambari platform. As a dependency for the jobs scheduled by Oozie, Apache Pig and Apache Sqoop must be also installed using the Ambari platform. Other dependencies include Jdom 2.0.5 and MySQL java connector for MySQL 5.1, while jdom must be built using maven as does the other SMDMC components, the MySQL connector can be downloaded in jar form. The configuration of the FTP path and access to the Oozie server, through ssh, must be done in the src/main/resources folder prior to building the package.

The path for storing data and the application Oozie workflow and Pig scripts on the Hadoop HDFS must be created with the necessary rights for access. Credentials for the Oozie server must be configured in the SMDMC source before building the package with Maven.

The workflow.xml and the Pig scripts are located in the **SMDMC Transform** repository. The workflow must be uploaded into HDFS to the path configured in the SMDMC Manager. The jdom and MySQL connector dependencies must be uploaded to the lib folder where the workflow is stored in the HDFS.

The **SMDMC Transform subcomponent** must be built using Maven pom.xml script from:

https://bitbucket.org/smarth2o/sh2osmdmctransform/src/3d52d50bab5496d849fd8e67847290f2fb2c7631?at=master

The update of the SMDMC Transform component corresponding to the second prototype is available at:

https://bitbucket.org/smarth2o/sh2osmdmctransform/src/26cf5b49f241e2a76b598cf665844bcded0d1fce?at=master

Using the:

```
mvn package
```

command and then uploaded in the Hadoop HDFS in the lib folder where the Pig scripts are located.

2.1.3 Enterprise Service Bus based on JBoss Fuse

SmartH2O platform runs on JBoss Fuse Enterprise Service Bus. It allows:

- using Apache Camel with OSGI integration to dynamically route messages to new or updated OSGi bundles. OSGI is a standard specification describing a modular system and a service platform for the Java programming language that implements a complete and dynamic component model.
- Combining use of the Camel Recipient List, which allows at runtime to specify the Camel Endpoint to route to, and use of the Camel VM Component. It provides a SEDA (*staged event-driven architecture*) queue that can be accessed from different OSGi bundles running in the same Java virtual machine.

Requirements

- JBoss Fuse 6.1.0 or later (https://access.redhat.com/jbossnetwork, installation guide: https://access.redhat.com/documentation/en-US/Fuse_ESB_Enterprise/7.1/html/Installation_Guide/files/InstallingText.html)
- Maven 2.2.1 or 3.0 (http://maven.apache.org)
- Java SE 6 or Java SE 7

Building and Running

In the current development state, the SmartH2O ESB includes specific web-services bundles integrated for the usage of the Customer Portal, the Gamification Engine and the Games portal. The bundles are permanently updated during the development process . The bundles areavailable for download at https://bitbucket.org/smarth2o-setmob1/smarth20-esb/src/a4948cf699deb76d88cf8c1b990a4cf9e4c9779e?at=master

To build the project use Apache Maven in the base directory of this project

```
mvn clean install
```

Project components

1. Base service (main ESB project)
2. Client (ESB client)
3. Newservice service (new integration service): each new service that needs to be integrated with ESB should have a similar component in project, and it can be deployed at the runtime, with no impact on running processes

Bundles installation and testing

1. Install the Base Service

Within the Base Camel route there are 3 routes:

- an HTTP listener that routes to other endpoints based on the contents of the request
- a “simple” route that responds with a “Simple Response: {body of message}”
- a “other” route that responds with a “Other Response: {body of message}”

Start JBoss Fuse by running the included starting script:

```
<JBoss Fuse Home>/bin/fuse
```

In the JBoss Fuse console, launch the following commands:

```
features:addurl          mvn:ro.setmobile.sh2o.dynamic/features/0.0.1-SNAPSHOT/xml/features
features:install dynamic-routing-base
```

2. Testing the Base Service

Change to the client sub-project, and run:

```
mvn -P simple
```

You should see log entries in both JBoss Fuse console, and the Command Prompt that messages are flowing to the Simple Route.

3. Deploying the Newservice Service

In the JBoss Fuse console, launch the following command:

```
features:install dynamic-routing-newservice
```

4. Testing the Newservice Service

Change to the client sub-project, and run the command:

```
mvn -P newservice
```

The log entries should be available in both JBoss Fuse console, and the Command Prompt showing the messages that are flowing on the Service Route.

The original Base service doesn't need to be reloaded or restarted in order to keep forwarding messages to the newly loaded routes. The original tests can be re-run using:

`mvn -Psimple` and `mvn -Pother` commands, while observing that the previous registered services still work correctly, without being affected by the newly installed services.

Newservice gets the messages from the Camel VM component, and puts them onto an ActiveMQ Queue. This shows how to route to new endpoints and integration routes at runtime -- it does not have to be ActiveMQ, it could easily be WS, REST or other JMS.

In the following, the development project specific configurations are presented.

In Figure 4 we report the Howtio schemas of deployed services – corresponding to first platform prototype.

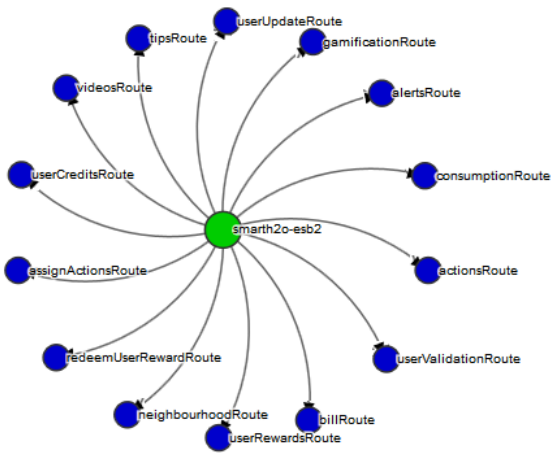

Figure 4. Schema of deployed Fuse services – first platform prototype.

In Figure 5 we report the Howtio schemas of deployed services – as the services have been extended to meet the requirements of the second platform prototype.

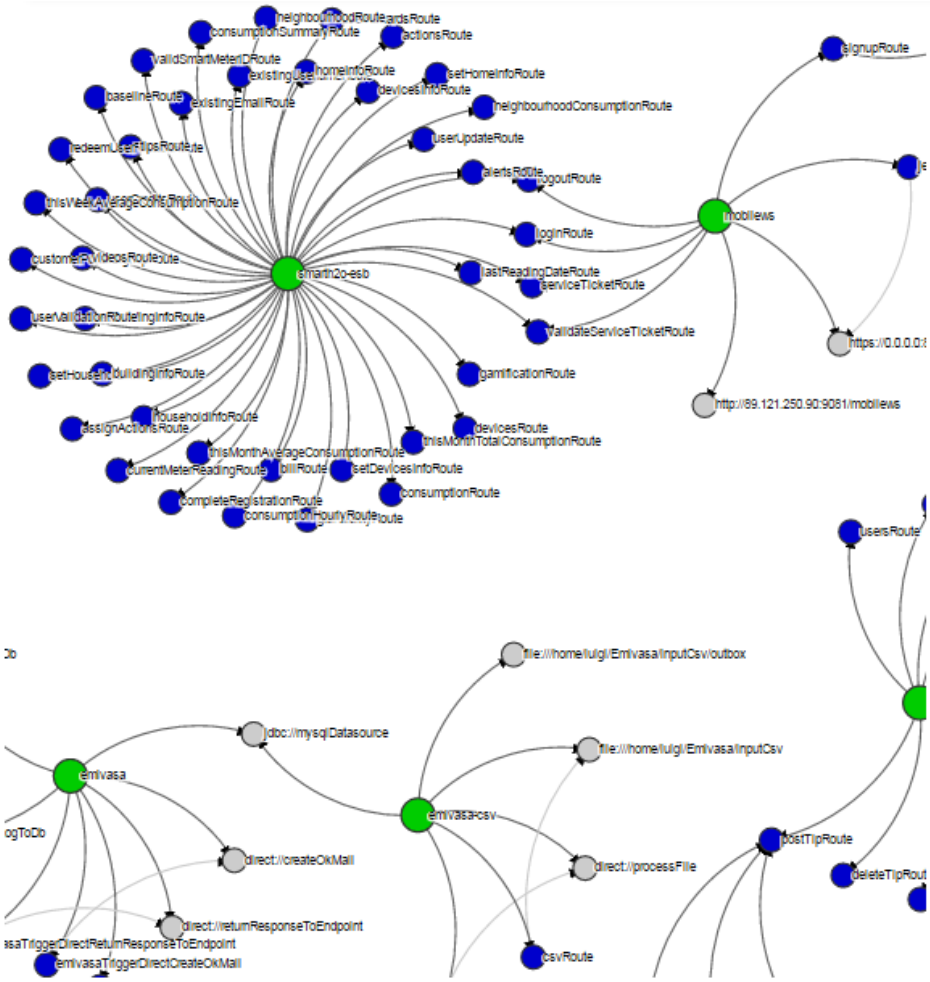

Figure 5. Schema of deployed Fuse services – second platform prototype.

In Figure 6 the Camel routes are shown.

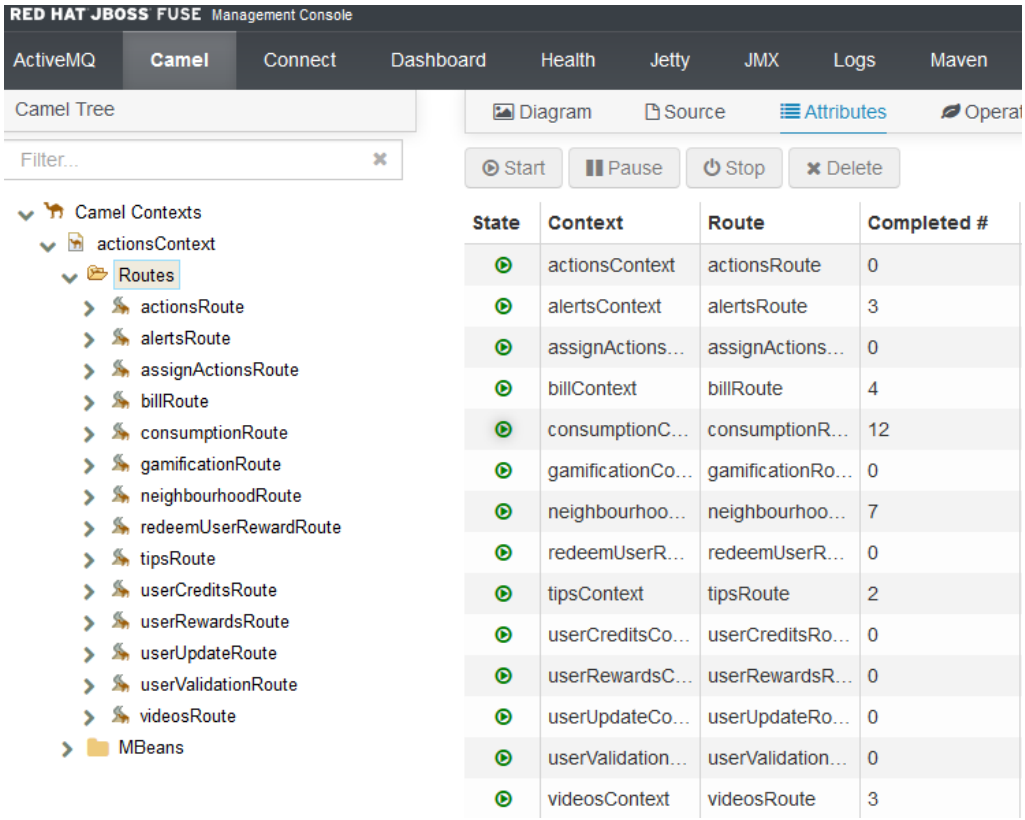

Figure 6. List of Camel routes.

The Camel service endpoints are listed in Figure 7.

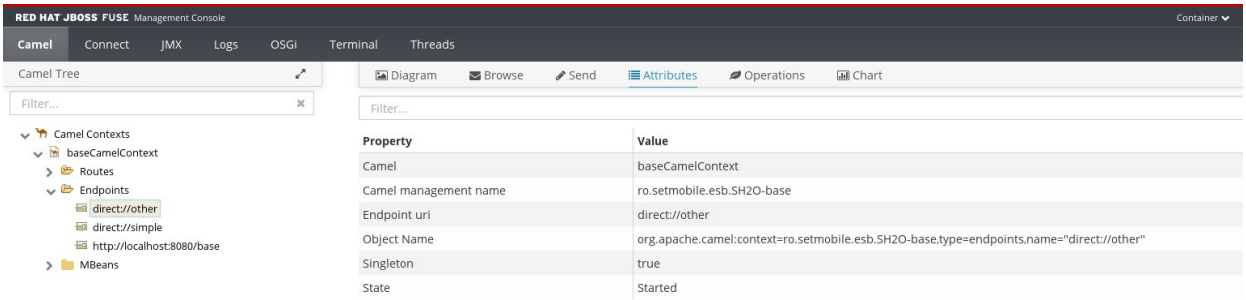

Figure 7. List of endpoints.

The Enterprise Service Bus instance configured for the SmartH2O development and testing server is available online at http://esb.smarth2o.ro:9080

`Username:` `smarth2o`

`Password:` *`dsfsmarth2o`*

2.1.4 Customer Portal and Gamification Engine

Requirements:

- MySQL DBMS 5.6 (link)
- Java 1.8.0 or later (link)
- Apache Tomcat 6.x or later (link).

Installation package

The installation package is available here:

https://bitbucket.org/smarth2o/frontend-customerportal-smarth2o/commits/tag/Release2.1

and it contains:

- Authentication, BootstrapStyleRarolab, Gamification, GamificationBackEndStyle, GamificationCustomRarolab, GamificationFrontEndStyle, NotificationMessageSample: the webratio source projecs.
- Deployment/lib.zip: external libraries.
- Deployment/community.7z: the deployed webapplication.

Steps for installation:

1. DB Installation (usr: root, pwd: *password*):

Create a new database " community_new_newdata " and import the sql file community_new_newdata.sql from https://bitbucket.org/smarth2o/ses-gedb-dump/commits/tag/Release2

2. Application installation on Tomcat

- Unzip the **Deployment/community.7z** file and copy the **community** folder into the webapps folder of your Tomcat installation (henceforth tomcat_webapps)
- Change configuration path of the invoked web services **tomcat_webapps*/community/WEB-INF/classes/Webratio.properties***:

```
var services_host_url = "http://89.121.250.90:8083/";
var local_host_url = "http://localhost:8080/"; tomcat address

services_host_url=http://89.121.250.90:8083/SmartH2O  address of the invoked services
my_host_url=http://localhost tomcat address
my_host_port=8080 tomcat port for basic
my_host_internal_url=http://localhost tomcat address for internal invocations, tipically "localhost"
my_host_name=community name of the application
my_host_port_ssl=8443
my_host_internal_port=8080

reward_shipment=true whether rewards are shipped or collected in place
reward_score_decrease=truse whether a reward claim causes a score descrease

sendNotificationEmail=true whether the email notification is enabled
emailAdmin=youradmin@gmail.com email administrator used when sendNotificationEmail is false
emailContactUs=smarth2o-help@idsia.ch email used for communication with the user

SMTP configuration
smtp_url=smtp.gmail.com
smtp_port= 465
smtp_username=smarth2o.project@gmail.com
smtp_password=yourpass
smtp_default=prova@gmail.com
```

- Note1: the **services_host_url** must be configured with the url of the web server that exposes the required external services (see 2.1.3)
- Note2: the **SMTP server** is used to send notification emails to customers and admin (i.e. reward shipment notification, badge achieved notification etc). To enable the email notification set to true the parameter **sendNotificationEmail**. If it's disabled the GE will send the notification to the email address specified in **emailAdmin** (if any).
  Note2: reward_shipment and reward_score_decrease are set on true for the SES scenario.

- Change the database configuration (url, username and password), updating the "dbx.hibernate.cfg.xml" file in the **tomcat_webapps/*community*/WEB-INF/classes** of the applications:

```
<property name="connection.url">
      jdbc:mysql://localhost:<your_port>/community_new_newdata
</property>
<property name="connection.username">
      <your_user>
</property>
<property name="connection.password">
      <your_password>
</property>
```

- From **Deployment/lb.zip** add the two jar files under the **lib** folder of your Tomcat installation
- Grant R/W permission to Tomcat to the entire folder **community.** For UNIX:

```
sudo chown -R tomcat7 community/
sudo chgrp tomcat7 community/
```

- Start Tomcat

Testing

The applications can be accessed using the following url:

- http://localhost:<your_port>/community (customer frontend). The portal can be accessed using the following credentials:
- Basic profile
  - username: ses.smarth2o
  - password: *smarth2o*
- Advanced profile
  - username: ses2.smarth2o
  - password: *smarth2o*
- https://localhost:< your_port >/community/admin (admin frontend). The portal can be accessed using the following credentials:
  - username: admin
  - password: *admin*

Source code

The Webratio projects can be imported into thw Webratio IDE using the projects:

- Authentication,
- BootstrapStyleRarolab,
- Gamification,
- GamificationBackEndStyle,
- GamificationCustomRarolab,
- GamificationFrontEndStyle,
- NotificationMessageSample:

Testing

The applications can be invoked locally at the following url:

- `http://localhost:8080/community` (frontend)
- `http://localhost:8080/community/admin` (backend)

The two versions of the portals can be accessed using the following credentials:

- Basic Version:
  username: ses.smarth2o / password: *password*
- Advanced Version (gamified)
  username: ses2.smarth2o / password: *password*

2.2 EMIVASA Demo Case

In EMIVASA demo case, SmartH2O platform integration has been performed at web service level by connecting it to EMIVASA Virtual Office – the online customer portal. It is offered to EMIVASA customers as an online service, enriching the water utility efforts to open a new kind of relationship to its customers. User authentication is uniquely performed by Virtual Office, while a ticketing system allows Single Sign On user acces to SmartH2O platform.

In Figure 8 it is displayed a screen capture of EMIVASA Virtual Office featuring integrated SmartH2O service.

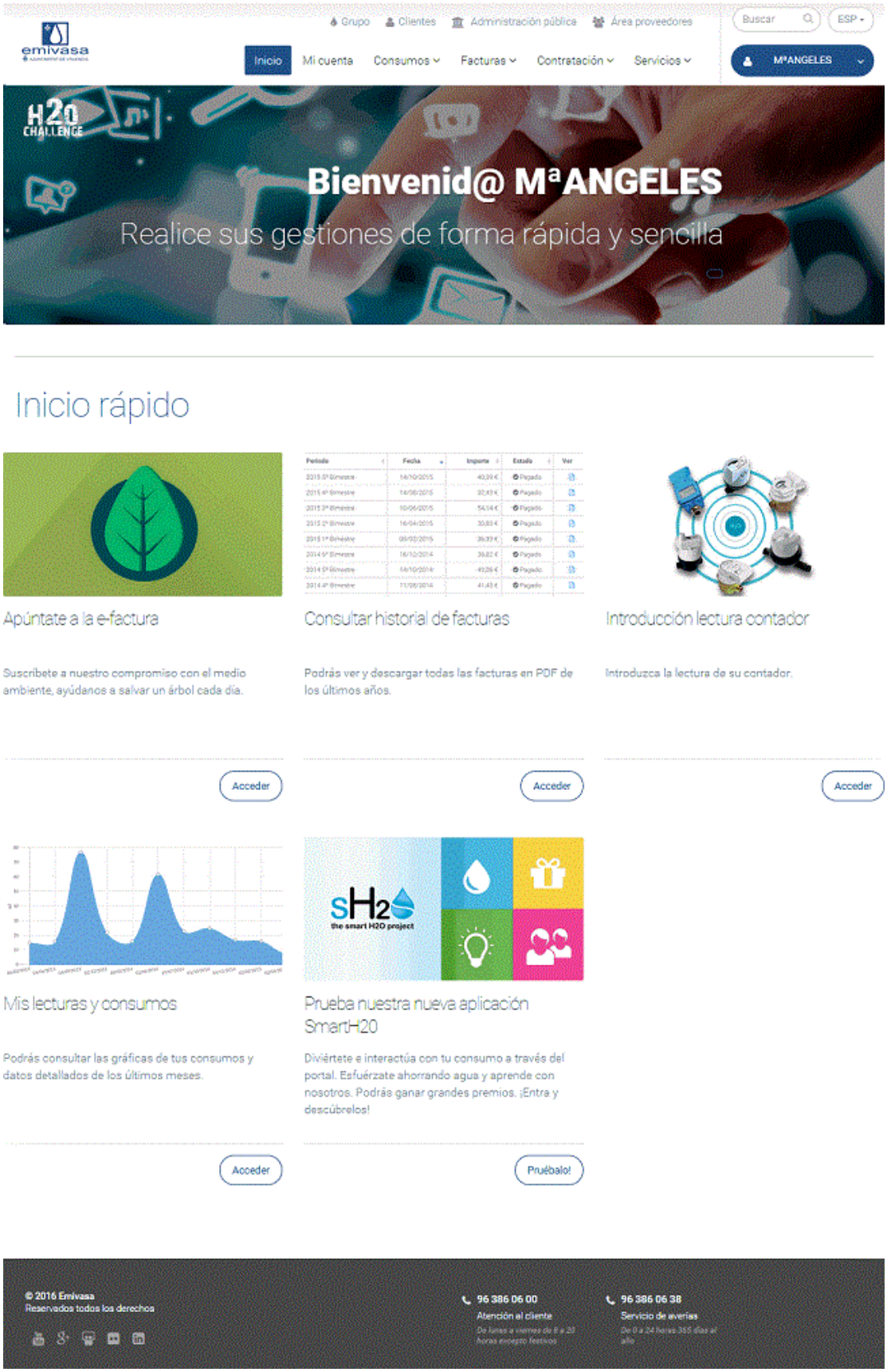

Figure 8. SmartH2O integration to EMIVASA Virtual Office.

2.2.1 SmartH2O Database

The **SmartH2O Database** is the central repository of the information that is either common to all the SmartH2O components or supports the coordination and exchange of messages among them. In EMIVASA demo case, not all the data of SmartH2O platform reside in the SmartH2O database. Gamification data are stored in the Gmification engine database, while Commercial data about the water consumers maintained by the water utility are stored in the proprietary systems of the water utility as stipulated by the service contract.

The database server for the Smarth2O platform database is a MySql 5.5+.

The DDL script for creating the database structure is available as Appendix A, section 2 of the present document.

The dump file of the database containing alpha users testing data corresponding to the second prototype of the platform, is available on Bitbucket at:

https://smarth2o-guest@bitbucket.org/smarth2o/emivasa-smarth2odb-dump.git

2.2.2 Emivasa Smart Meter Data Provisioning component

In EMIVASA demo case, SmartH2O platform takes advantage of existing database of users and consumptions provided by the water utility. Therefore the EMIVASA – SmartH2O integration architecture had to be adapted in the sense that SmartH2O platform has been connected to EMIVASA Virtual Office at web service level instead of data level.

The Smart Meter Data Management (SMDM) component - whose role is to process periodic (daily, weekly, monthly) smart meter readings received by FTP as XML files – has been replaced by a dedicated component **EMIVASA Smart Meter Data Provisioning** (ESMDP) component, the functionality of which is to:

1. Retrieve user consumption data from past (parametrized) intervals.
2. Process daily user consumption data (CSV file).

Let's examine these two steps in more detail.

Retrieving user consumption data from past (parametrized) intervals

The business logic is implemented following the steps:

1. ESDMP exposes **TriggerUserSubscription** GET web service to be called by the Gamification Engine when a Virtual Office user subscribes to SmartH2O.
   A sample WADL is:

```
<application>
    <grammars />
    <resources base="http://esb.smarth2o.ro:9081/trigger">
    <resource path="/TriggerUserSubscription">
            <resource path="/triggerUserSubscription">
            <method name="GET">
                    <request>
                    <param name="user_id" style="query" type="xs:string" />
                    <param name="data_start" style="query" type="xs:string" />
                    <param name="data_stop" style="query" type="xs:string" />
                    </request>
                    <response>
                    <representation mediaType="application/json">
                            <param name="result" style="plain" type="xs:string" />
                    </representation>
                    </response>
            </method>
            </resource>
    </resource>
    </resources>
</application>
```

Example call for consuming the web service (as from Virtual Office):

http://www.server.com/trigger/TriggerUserSubscription/triggerUserSubscription?user_id=ABC123&data_start=2015-08-20&data_stop=2015-10-20

2. The Gamification Engine manages the user_id of the Virtual Office users who subscribe to SmartH2O and calls **TriggerUserSubscription** web service only for the new users (as well as the display of the Privacy Policy). Already registered users access SmartH2O platform, without calling this web service.

3. Emivasa publishes **GetConsumptionDataFromVirtualOffice** GET web service which is called by ESMDP. This service provides a JSON with the user consumption data for the past interval specified in the ESDMP call (2 month or 1 year).

```
JSON structure:
{
  user_id: global_id,
  email: email,
  zipcode: zipcode,
  meter_readings:[
    {
   smartmeter_id:smartmeter_id,
   value:value,
   timestamp:timestamp
    },
    {
   smartmeter_id:smartmeter_id,
   value:value,
   timestamp:timestamp
    }
  ]
}
```

GetConsumptionDataFromVirtualOffice GET web service that is published by Emivasa is of the the following URL type:
http://www.server.com/esmdp/EsmdpServices/DataProvisioningExample/dataProvisioningExample?user_id=ABC123&data_start=2015-08-20&data_stop=2015-10-21

4. ESDMP calls **GetConsumptionDataFromVirtualOffice** to process the returned JSON and it consistently saves the data in the SmartH2O central database.

Processing daily user consumption data (CSV file)

Following an automated process, Emivasa prepares a CSV file with daily user consumption data. The file is uploaded by a nightly job to a transfer location (E.g. C:\esdmp\daily**SmartH2O_Emivasa_Daily_151019-000000-151020-000000.csv** having the meaning "daily data from day 19-10-2015 hour 00:00:00 to day 20-10-2015 hour 00:00:00).

Also, Emivasa prepares a MD5 file containing the signature of the CSV file loaded to the transfer location. The processing workflow uses the MD5 file in order to ensure the integrity of the CSV file during the transfer. A non consistent CSV is not processed and it is be reported via e-mail alert to the ESDMP administrator in order to be sent again.

The uploading process is accomplished by FTP/SFTP. The recommended interval of time to program the running of the job is nightly (E.g. starting from 4:00 o'clock) in order for the users to have fresh data available in the following morning.

The CSV structure is the following:

```
user_id,smart_meter_id,timestamp,value,unit_of_measurement, zipcode
```

CSV sample:

"U123VAL1", "ES_VAL_12345678", "2015-10-20 01:00:00", 123.45, "m3", 91354
"U123VAL1", "ES_VAL_12345678", "2015-10-20 02:00:00", 124.04, "m3", 91354
"U123VAL1", "ES_VAL_12345678", "2015-10-20 03:00:00", 124.16, "m3", 91354
"U123VAL1", "ES_VAL_12345678", "2015-10-20 04:00:00", 125.00, "m3", 91354
"U123VAL1", "ES_VAL_12345678", "2015-10-20 05:00:00", 125.25, "m3", 91354
"U123VAL1", "ES_VAL_12345678", "2015-10-20 06:00:00", 126.05, "m3", 91354

ESDMP applies a business logic based on the user_id in order to decide if to overwrite an old smart_meter_id when changed with a newer smart_meter_id.

Regarding the unit of measurement, ESDMP applies the necessary transformation ratio to liters which is the default unit of measurement for calculations and display within SmartH2O. When the unit of measurement for consumption readings is always the same, then the unit of measurement is removed from the CSV file for decreasing the CSV file transfer size.

After a successful workflow processing, ESDMP automatically loggs the outcome in the SmartH2O database and sends by e-mail an OK message to the ESDMP administrator.

2.2.3 Enterprise Service Bus based on JBoss Fuse

The Enterprise Service Bus is the endpoint publishing:

- **TriggerUserSubscription** - a web service for retrieving user consumption data from Emivasa Virtual Office for past parametrized intervals and save them to the SmartH2O platform database.
- **EmivasaCSV** - Apache Camel route for processing daily consumption data as CSV files.

In order to configure the port exposing the list of the available web services, you have to edit `$ESB-path\jboss-fuse\etc\jetty.xml` and change the value for property `jetty.port`.

E.g.: `http://localhost:9080/cxf`

Also, ESB exposes its GUI on this port. Eg. http://localhost:9080 (admin / admin).

```
<!-- default port will be overwritten by pax-web configuration -->
              <Set name="port">
                     <Property name="jetty.port" default="9080"/>
```

The user and password can be changed in `$ESB-path\jboss-fuse\etc\user.properties` file

```
# The password of the first user in the file will also be used
# as a registry (zookeeper) password
# unless a password is explicitly specified.


admin=admin,admin
```

In order to configure the port exposing the bundle of web services you have to edit `$ESB-path\jboss-fuse\etc\system.properties` and change the value for property `org.osgi.service.http.port`.

```
#
# Default port for the OSGI HTTP Service
#
org.osgi.service.http.port=9081
```

ESB exposes the services on this port. E.g.: `http://localhost:9081/cxf`

To start the ESB, in a Command Prompt go to `$ESB-path\jboss-fuse\bin` folder and launch `start.bat`.
To stop the ESB, in a Command Prompt go to `$ESB-path\jboss-fuse\bin` folder and launch `stop.bat`.
To get the status of the ESB, in a Command Prompt go to `$ESB-path\jboss-fuse\bin` folder and launch `status.bat`.

Installing jboss-fuse as service:

```
$ESB-path\jboss-fuse\bin\fuse
JBossFuse:karaf@root> features:install wrapper
JBossFuse:karaf@root> wrapper:install -n JBossFuseService -d JBossFuseService -D ESB service
JBossFuse:karaf@root> exit
```

Run as administrator:

```
$ESB-path\jboss-fuse\bin\JBossFuseService-service.bat install
```

From `Services.msc` set JBossFuseService-service as Automatic

```
Start JBossFuseService-service
```

ESB backend bundles

The ESB bundles represent unique access points for the backend services.

- `emivasa-1.0.1-SNAPSHOT.jar` is the bundle embedding **TriggerUserSubscription** web service
  The source code is available on Bitbucket at https://smarth2o-guest@bitbucket.org/smarth2o/emivasa-rest-triggerusersubscription.git
- `emivasa-csv-0.0.1-SNAPSHOT.jar` is the bundle embedding **EmivasaCSV** - Apache Camel route for processing daily consumption data as CSV files
  The source code is available on Bitbucket at https://smarth2o-guest@bitbucket.org/smarth2o/emivasa-esb-csv-dailyconsumptionroute.git

The bundles are installed by copying the .jar files in the $ESB-path\jboss-fuse\deploy folder while the ESB server is running. After a successful deployment, the services are extracted and listed at http://localhost:9081/cxf

2.2.4 Customer Portal and Gamification Engine

The following figure presents an overview of the user interaction with the Customer Portal and Gamification Engine.

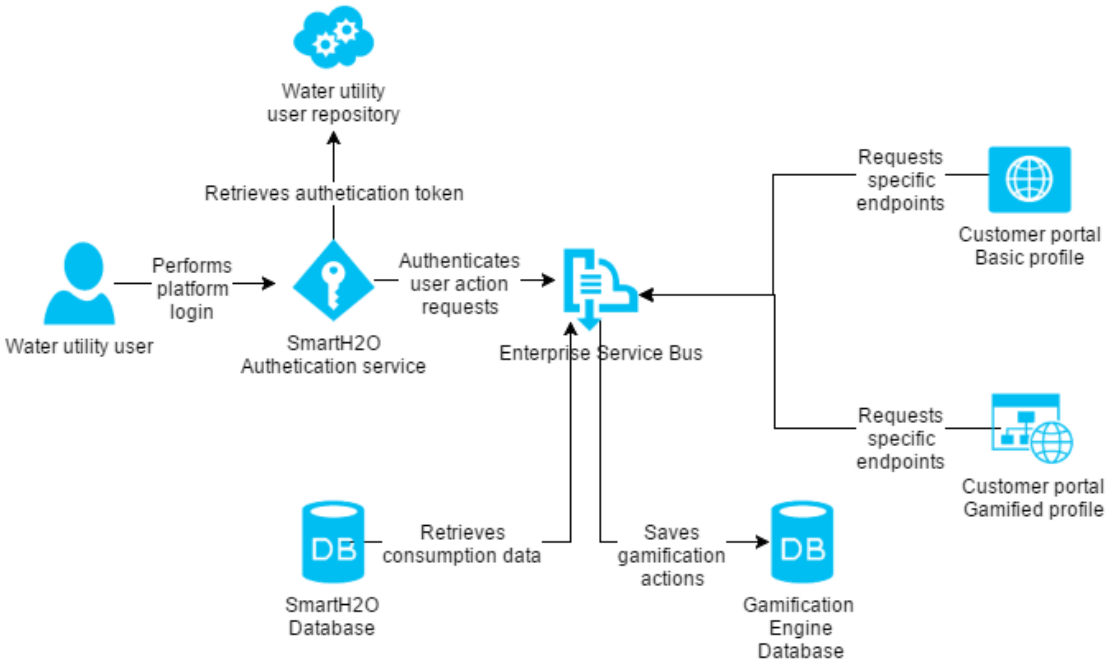

Figure 8.1 Overview of the user interaction with the Customer Portal and Gamification Engine

The Customer Portal (Basic profile) component interface specification has been defined in D6.2 Platform architecture and design, section 3.3, page 20 (under the name Water Utility Customer Portal).

The Customer Portal (Basic profile) services have been described In D2.3 Functional specification, section 6, page 34.

The Gamification Engine (Customer Portal - Gamified profile) component interface specification has been defined in D6.2 Platform architecture and design, section 3.4, page 24.

The Gamification Engine (Customer Portal - Gamified profile)services have been described In D2.3 Functional specification, section 7, page 45

Requirements:

- MySQL DBMS 5.6 (link)
- Java 1.8.0 or later (link)
- Apache Tomcat 6.x or later (link).

Installation package

The installation package is available here:

https://bitbucket.org/smarth2o/frontend-customerportal-smarth2o/commits/tag/Release2.1

and it contains:

- Authentication, BootstrapStyleRarolab, Gamification, GamificationBackEndStyle, GamificationCustomRarolab, GamificationFrontEndStyle, NotificationMessageSample: the webratio source projecs
- Deployment/lib.zip: external libraries
- Deployment/community.7z: the deployed webapplication.

Steps for installation:

1. DB Installation (usr: root, pwd: *password*):

Create a new database " community_new_newdata " and import the sql file community_new_newdata.sql from:
https://bitbucket.org/smarth2o/emivasa-gedb-dump/commits/tag/Release2

2. Application installation on Tomcat

- Unzip the **Deployment/community.7z** file and copy the **community** folder into the webapps folder of your Tomcat installation (henceforth tomcat_webapps)
- Change configuration path of the invoked web services **tomcat_webapps/*community*/WEB-INF/classes/*Webratio.properties***:

```
var services_host_url = "http://89.121.250.90:8083/";
var local_host_url = "http://localhost:8080/"; tomcat address

services_host_url=http://89.121.250.90:8083/SmartH2O  address of the invoked services
my_host_url=http://localhost tomcat address
my_host_port=8080 tomcat port for basic
my_host_internal_url=http://localhost tomcat address for internal invocations, tipically
"localhost"
my_host_name=community name of the application
my_host_port_ssl=8443
my_host_internal_port=8080

reward_shipment=false whether rewards are shipped or collected in place
reward_score_decrease=false whether a reward claim causes a score decrease

sendNotificationEmail=true whether the email notification is enabled
emailAdmin=youradmin@gmail.com email administrator used when sendNotificationEmail is false
emailContactUs=smarth2o-help@idsia.ch email used for communication with the user

SMTP configuration
smtp_url=smtp.gmail.com
smtp_port= 465
smtp_username=smarth2o.project@gmail.com
smtp_password=yourpass
smtp_default=prova@gmail.com
```

- Note1: the **services_host_url** must be configured with the url of the web server that exposes the required external services (see 2.2.3)
- Note2: the **SMTP server** is used to send notification emails to customers and admin (i.e. reward shipment notification, badge achieved notification etc). To enable the email notification set to true the parameter **sendNotificationEmail**. If it's disabled the GE will send the notification to the email address specified in **emailAdmin** (if any).
  Note2: reward_shipment and reward_score_decrease are set on false for the EMIVASA scenario.
- Change the database configuration (url, username and password), updating the "dbx.hibernate.cfg.xml" file in the **tomcat_webapps/*community*/WEB-INF/classes** of the applications:

```
<property name="connection.url">
      jdbc:mysql://localhost:<your_port>/community_new_newdata
</property>
<property name="connection.username">
      <your_user>
</property>
<property name="connection.password">
      <your_password>
</property>
```

- From **Deployment/lb.zip** add the two jar files under the **lib** folder of your Tomcat installation

- Grant R/W permission to Tomcat to the entire folder **community.** For UNIX:

```
sudo chown –R tomcat7 community/
sudo chgrp tomcat7 community/
```

- Start Tomcat

Testing

The applications can be accessed using the following url:

- http://localhost:<your_port>/community (customer frontend). The portal can be accessed using the following credentials:
- Basic profile
    1. username: ses.smart2o
    2. password: *smarth2o*

- Advanced profile
    3. username: ses2.smarth2o
    4. password: *smarth2o*

- https://localhost:< your_port >/community/admin (admin frontend). The portal can be accessed using the following credentials:
    - username: admin
    - password: *admin*

Source code

The Webratio projects can be imported into thw Webratio IDE using the projects:

- Authentication,
- BootstrapStyleRarolab,
- Gamification,
- GamificationBackEndStyle,
- GamificationCustomRarolab,
- GamificationFrontEndStyle,
- NotificationMessageSample:

Testing

The applications can be invoked locally at the following url:

- `http://localhost:8080/community` (frontend)
- `http://localhost:8080/community/admin` (backend)

The two versions of the portals can be accessed using the following credentials:

- Basic Version:
  username: ses.smarth2o / password: *smarth2o*
- Advanced Version (gamified)
  username: Ses2.smarth2o / password: *smarth2o*

2.4 Games Platform

As an awareness and educatiaonal tool for water consumers Drop! The Question is the first production deployed game which composes the SmartH2O Games Platform, The following flow presents an overview of the user interaction with Drop! The Question game.

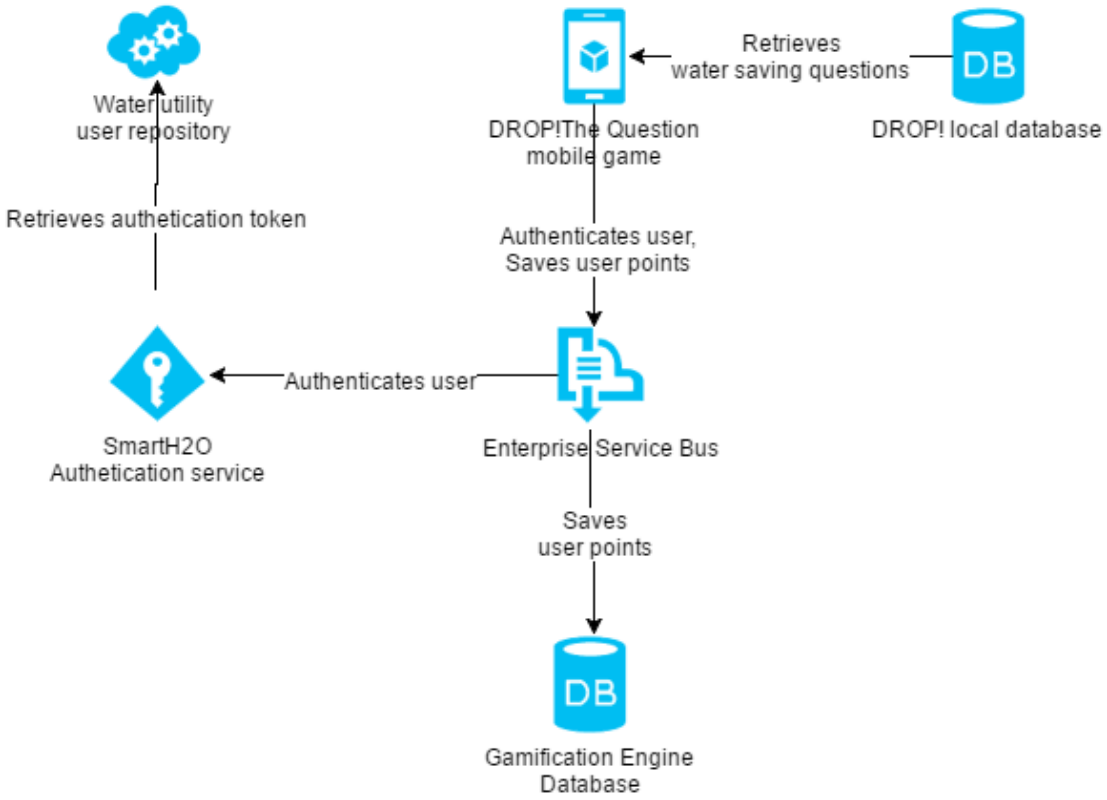

Figure 8.2 Overview of the user interaction with Drop! The Question mobile game

The Games Platform component interface specification has been defined in D6.2 Platform architecture and design, section 3.5, page 36

The Games Platform services have been described In D2.3 Functional specification, section 11, page 112.

Drop!TheQuestion has been developed as a native Android application using the Android SDK bundled with Android Studio, the official Android IDE, available for free at:

http://developer.android.com/sdk/index.html.

The complete sources of DropTheQuestion are available for download at:

https://bitbucket.org/smarth2o/ses-mobileapp-drop/commits/tag/release2

The source code of the game can be found under the "/app/src/main" folder. The game has been developed in compliance with the latest development guidelines published by Google, to achieve good performance on the widest range of targetable devices. More specifically, the application is designed to run on both smartphones and tablets and its minimum supported platform version is API level 15 (Android 4.0.3).

Deploy and testing procedure

The application has been developed as a native Android application to achieve fast performance and the maximum degree of reliability. The application is therefore coded in Java.

Requirements

- Android Studio (any version, http://developer.android.com/sdk/index.html, installation guide: http://developer.android.com/sdk/installing/index.html?pkg=studio)
- A working Android Virtual Device created via AVD Manager, included in Android Studio (guide: http://developer.android.com/tools/devices/managing-avds.html) or a physical Android device having USB debug enabled. In both cases, the device must support API level 15 at least.
- Backend: see later

Building and Running Mobile App

Extract the content of the downloaded repository zip in a directory, launch Android Studio and choose "Open an existing Android Studio project" from the "Quick Start" menu. Navigate to the path of the directory in which the project was extracted and select it, as shown in the pictures below.

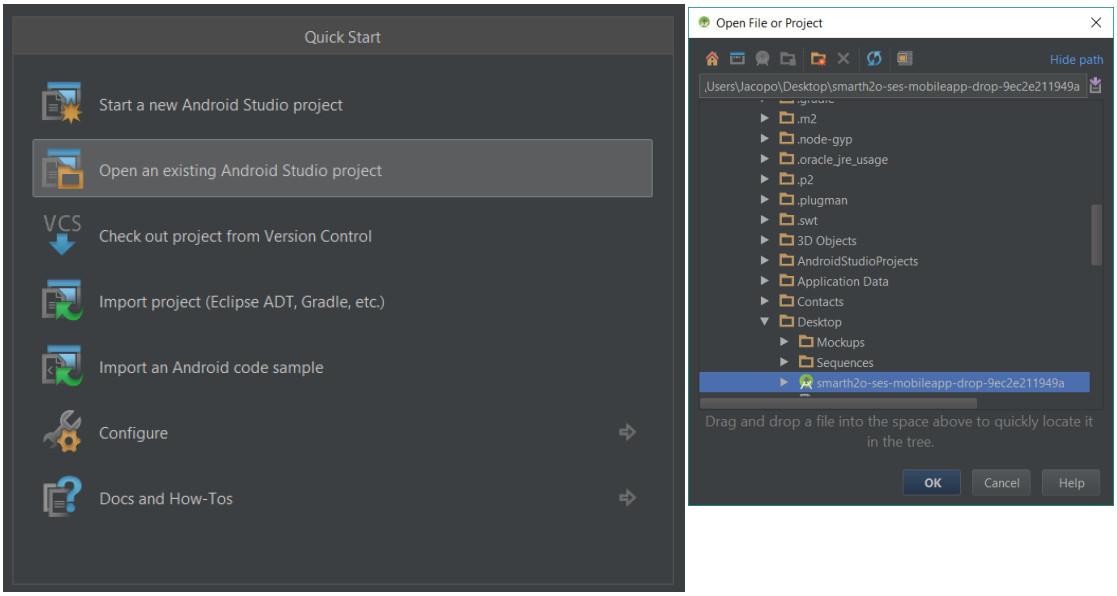

Figure 9. Building the mobile application.

Android Studio will automatically recreate the project structure and try to generate a Gradle build. During the compilation process, if any of the platform dependency fails, Android Studio will launch the embedded SDK Manager to download the needed packages. As this phase ends and the build is generated, the testing phase can start.

The project can be executed by clicking on the "Run" icon, choosing the "Run 'app'" from the contextual menu or by triggering the "Shift + f10" shortcut. A "Device Chooser" window will pop up, asking for a target testing device. As already mentioned, it is possible to choose a physical connected device (given that "USB debug" is enabled under the "Developer Options" of the device) or a virtual one (check "Launch emulator" and select the virtual device). Further information on running Android applications is provided at the following page: http://developer.android.com/training/basics/firstapp/running-app.html.

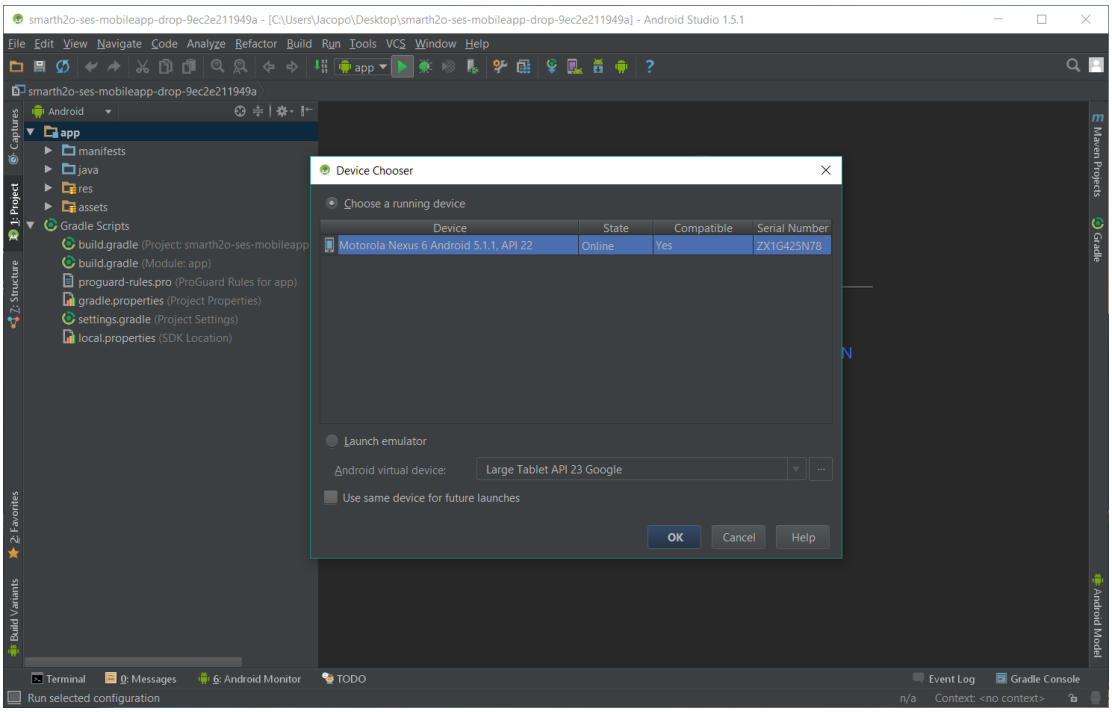

Figure 10. Running the mobile application.

After having chosen the target device, the application will be deployed to the device, installed and finally launched. Notice that on devices running Android API level 23+ (Android Marshmallow), the application will ask for the permission to access the camera of the device at runtime, when the "Decode a card" mode is selected. This mode is conceived to play the game in conjunction with the Drop! board game.

Building and Running (Backend)

Requirements:

- MySQL DBMS 5.6 (link)
- Java 1.8.0 or later (link)
- Apache Tomcat 6.x or later (link).

Installation package

The installation package is available here:

https://bitbucket.org/smarth2o/ses-rest-backend-drop/commits/tag/Release2 and contains:

- DropBackend and DropBackendStyle: the webratio source projects
- Deploy/Drop.war: the deployed webapplication.

Steps for installation:

1. DB Installation (usr: root, pwd: *password*):

Create a new database "drop" and import the sql file SES-DROP!DB-Dump.sqlfrom:
https://bitbucket.org/smarth2o/ses-dropdb-dump/commits/tag/Release2

2. Application installation on Tomcat

- Unzip the **Deploy/Drop.war** file and copy the **Drop** folder into the webapps folder of your Tomcat installation (henceforth tomcat_webapps)
- Change configuration path of the invoked web services **tomcat_webapps/*Drop/WEB-INF/classes/Webratio.properties***:

```
services_host_url=http://89.121.250.90:8083/SmartH2O  address of the invoked services
my_host_url=http://localhost tomcat address
my_host_port=8080 tomcat port for basic
my_host_internal_url=http://localhost  tomcat  address  for  internal  invocations,  tipically
"localhost"
my_host_name=drop name of the application
```

- Note1: the **services_host_url** must be configured with the url of the web server that exposes the required external services.
  Note2: the **SMTP server** is used to send notification emails to customers and admin (i.e. reward shipment notification, badge achieved notification etc). To enable the email notification set to true the parameter **sendNotificationEmail**. If it's disabled the GE will send the notification to the email address specified in **emailAdmin** (if any).
- Change the database configuration (url, username and password), updating the "dbx.hibernate.cfg.xml" file in the **tomcat_webapps/*community*/WEB-INF/classes** of the applications:

```
<property name="connection.url">
      jdbc:mysql://localhost:<your_port>/community_new_newdata
</property>
<property name="connection.username">
      <your_user>
</property>
<property name="connection.password">
      <your_password>
</property>
```

- Grant R/W permission to Tomcat to the entire folder **community.** For UNIX:

```
sudo chown -R tomcat7 community/
sudo chgrp tomcat7 community/
```

- Start Tomcat

Testing

Once deployed on the target device, the application can be run by clicking on the Drop icon that has been created on the dock panel of the device. It is possible to test the application in conjunction with the physical cards belonging to the Drop! Boardgame.

Figure 11. Testing the mobile application DROP!

2.4 Social network crawler and data analyser

This is a new component added to the SmartH2O platform second prototype.

Requirements

These components use the MongoDB database. Thus, download and install the MongoDB version that suits your operating system.

2.4.1 Crawler

Download the Eclipse project "TwitterCrawler" from the bitbucket repository here:

https://bitbucket.org/smarth2o/common-socialnetworkanalytics-smarth2o

Look for the "relevantFiles" folder, and specify your search criteria in the text files:

- relevantHashtags.txt: hashtags (e.g., #WaterSustainability, #WaterConsumption)
- relevantKeywords.txt: common words (e.g., water, environment)
- relevantUsers.txt: Twitter users to track

Then, use Eclipse to extract a JAR file, selecting TwitterCrawlingPipeline as the main class.

Via the terminal, start the crawler

- either by using the command "java –jar <<JARNAME>>" (in this case you are forced to maintain the session open, otherwise the crawler will be dropped and will stop retrieving tweets)
- or by using the command "nohup java –jar <<JARNAME>> &" (in case you want to close the terminal and let the crawler run alone; in this case, logs will be stored in the file nohup.out)

Tweets will be captured and stored in their raw format in the rawTweets collection.

2.4.2 Analyzer

NOTE: This project integrates with the crawler component, so you need to configure and start it first.

NOTE 2: The analysed (and positively classified) tweets will be stored in the Tweet collection in the MongoDB database.

Download the Eclipse project "TwitterStreamAnalyzer" from the bitbucket repository here:

https://smarth2o-guest@bitbucket.org/smarth2o/common-socialnetworkanalytics-smarth2o.git

Installing OpenCV

The project has an important dependency that is not included in the provided Eclipse project, since there does not exist a portable version: the OpenCV library. Thus, to run the analyser, you will have to go through the following steps:

1. Download and install/compile the OpenCV (preferably 2.4.9) for your operating system
2. Insert the compiled version in the "lib/opencv-lib" folder in the Eclipse project. Include OpenCV in the build path of the Eclipse project.
    a. If you cannot install/compile the suggested version (i.e., 2.4.9), then you will have to download the OpenCV JAR file related to the version you installed/compiled, and substitute the current version in the "lib/" folder with the one you downloaded

Extracting and executing the JAR file

Export the project with all the dependencies

Via the terminal, start the analyzer

- either by using the command "java –jar <<JARNAME>>" (in this case you are forced to maintain the session open, otherwise the analyzer will be dropped and will stop analyzing tweets)
- or by using the command "nohup java –jar <<JARNAME>> &" (in case you want to close the terminal and let the analyzer run alone; in this case, logs will be stored in the file nohup.out)

Executing the analyser without JAR file

Please notice that, in case the JAR extraction will fail, you can still extract the "src/" folder of the Eclipse project together with all the dependencies, and compile it via terminal, as follows:

```
javac -cp .:lib:lib/mongo-java-driver-2.12.3.jar:lib/jsonic-
1.3.0.jar:lib/langdetect.jar:lib/opencv-249.jar:lib/weka.jar:lib/json-
20090211.jar:lib/snowball-20051010.jar:lib/commons-lang3-
3.3.2.jar:lib/libsvm.jar:lib/wlsvm.jar:src
src/it/polimi/tweetcrawlingpipeline/pipeline/*.java
src/it/polimi/tweetcrawlingpipeline/classifiers/analyzers/*.java
src/it/polimi/tweetcrawlingpipeline/crawler/crawlers/*.java
src/it/polimi/tweetcrawlingpipeline/utils/*.java
src/it/polimi/tweetcrawlingpipeline/mongodb/*.java
src/it/polimi/tweetcrawlingpipeline/classifier/utils/*.java
src/it/polimi/tweetcrawlingpipeline/classifier/classificationresults/*.java
src/it/polimi/tweetcrawlingpipeline/classifier/objects/samples/*.java
src/it/polimi/tweetcrawlingpipeline/classifier/objects/dictionaries/*.java
src/it/polimi/tweetcrawlingpipeline/classifier/parameters/*.java
src/it/polimi/tweetcrawlingpipeline/classifier/objects/annotatedinstances/*.java
src/it/polimi/tweetcrawlingpipeline/classifier/objects/annotateddatasets/*.java
src/it/polimi/tweetcrawlingpipeline/classifier/visualobjects/*.java
src/it/polimi/tweetcrawlingpipeline/classifier/classifiers/*.java
src/it/polimi/tweetcrawlingpipeline/classifier/objects/*.java
src/it/polimi/tweetcrawlingpipeline/classifier/objects/trainingsets/*.java
src/it/polimi/tweetcrawlingpipeline/crawler/analyzers/*.java
src/it/polimi/tweetcrawlingpipeline/classifier/classifiers/factories/*.java
src/it/polimi/tweetcrawlingpipeline/classifier/objects/instances/*.java
src/it/polimi/tweetcrawlingpipeline/dataset/*.java
src/it/polimi/tweetcrawlingpipeline/mongodb/objects/*.java
src/it/polimi/tweetcrawlingpipeline/filters/*.java
```

Then, to launch the analyser, please use the following command:

```
nohup java -Djava.library.path="path-to-your-compiled-opencv" -cp
lib/*:src
it.polimi.tweetcrawlingpipeline.pipeline.TweetCrawlingPipeline &
```

2.5 Agent based modelling platform

The agent based model to test the main elements of the incentive model in Valencia, presented in the Deliverable 6.4, is based on the SmartH2O simulator discussed in the Deliverable D3.4, and whose first prototype has been extensively presented and discussed in the Deliverable D3.3. Remember that the model is not linked to the platform.

Here we report just the essential information concerning the software used for its development and the procedure to run a simulation.

The SH2O_D33 model is implemented within the commercial simulation software AnyLogic (www.anylogic.com).

The AnyLogic file SH2O_D33.alp of the model is located in the folder SH2O_D33. The folder also contains:

- meteo_table.xlsx: an excel file containing the chosen meteorological data that are going to be used during the simulation,
- *results_meteo_classes*.xlsx: an excel file containing the results on the available data described in the previous subsection, that is: (a) the properties of the obtained 5 meteorological classes ("meteo classes" page), (b) the obtained probability distributions for each of the 30 obtained consumption classes ("consumption classes" page), (c) the obtained single user behavioural model, that is, for each household, the probability distribution associated to each meteorological class ("user classification" page). In addition to that, a page "*results validation*" contains results discussed in section 3.6 about validation.
- Tegna-Google *Maps-Cartina*.png: a png map of the Tegna area,
- Tegna-Google *Maps-Terreno*.png: a satellite png map of the Tegna area
- a folder x3d: it contains the households' image files used as presentation shapes during the simulation;
- a folder containing the AnyLogic's database files used in the implementation of the model.

Start AnyLogic 7.2. If the Welcome page displays, close it.

Open the SH2O_D33 model by selecting: *File* > *Open* and then selecting SH2O_D33.alp located in the SH2O_D33 folder. The model opens.

The ABM model is implemented within the commercial simulation software AnyLogic 7, University Researcher Edition (www.anylogic.com).

AnyLogic 7 is Java and Eclipse based application and has been tested on the following platforms:

- Microsoft Windows 10, x86-32 and x64
- Microsoft Windows 8, x86-32 and x64
- Microsoft Windows 7 SP1, x86-32 and x64
- Microsoft Windows Vista SP2, x86-32
- Apple Mac OS X 10.7.3 (Lion) or later, Universal

SuSE Linux, x86-32 (with installed GTK+, libwebkitgtk-1.0-0, libudev, libssl 0.9.8 and newer)

Ubuntu Linux 10.04 or above, x86-32 (with installed GTK+, libwebkitgtk-1.0-0, libudev, libssl 0.9.8 and newer)

Java 2 Standard Edition 8.0 or later is needed to run AnyLogic model development environment. JRE is included in AnyLogic installation package for Windows, but needs to be installed independently on other platforms. For what concerns hardware recommendations, AnyLogic 7 installation requires 500MB of free disk space. It is recommended to have 2GB of

memory and modern processor for optimal performance.

Once the appropriate license has been bought and you have received the serial number, AnyLogic 7 University Researcher Edition can be downloaded directly from: http://www.anylogic.com/downloads

Once the correct version has been downloaded, some activation steps have to be followed.

Firstly, AnyLogic has to be installed. In order to do so, one has just to click on the downloaded installation application, and follow its simple instructions. Once the installation is finished, run AnyLogic. An activation wizard will show up automatically. Select the option “Request a permanent key. The key will be sent to you by e-mail”. Click the “Next” button. The fact that such a request from the computer where you are going to use AnyLogic is needed, is because AnyLogic key is computer specific.

Next, in the new wizard page, fill in the identifying information and specify the serial number in the “Order ID” field. Check if you have chosen the correct License type (“University research”). Click the “Next” button again.

Shortly after sending the request you’ll receive a confirmation of its receipt. The actual key will be sent to you in a separate email within one or two days.

Having received the activation key, run AnyLogic. The activation wizard will show up automatically (If you accidentally close it, select “Help | Activate product”). Select the option "Enter the Permanent Key that you received by email" and click the "Next" button. Copy the received activation key, paste it in the wizard field and click the "Next" button.

Activation is completed. The software is now fully functional and we can start running the model.

To conclude, we briefly present how to run the model SH2O_RewardValencia_statisticsNonLinearExpand. The corresponding AnyLogic file SH2O_RewardValencia_statisticsNonLinearExpand.alp is located in the folder called SH2O_RewardValencia. The folder also contains two png images (back.png and statistics.png) used in the model, and some jar files automatically generated by the model itself.

Start AnyLogic 7. If the Welcome page displays, close it.

Open the SH2O_RewardValencia_statisticsNonLinearExpand model by selecting: *File* > *Open* and then selecting SH2O_RewardValencia_statisticsNonLinearExpand.alp located in the SH2O_RewardValencia folder. The model opens, see Figure 12.

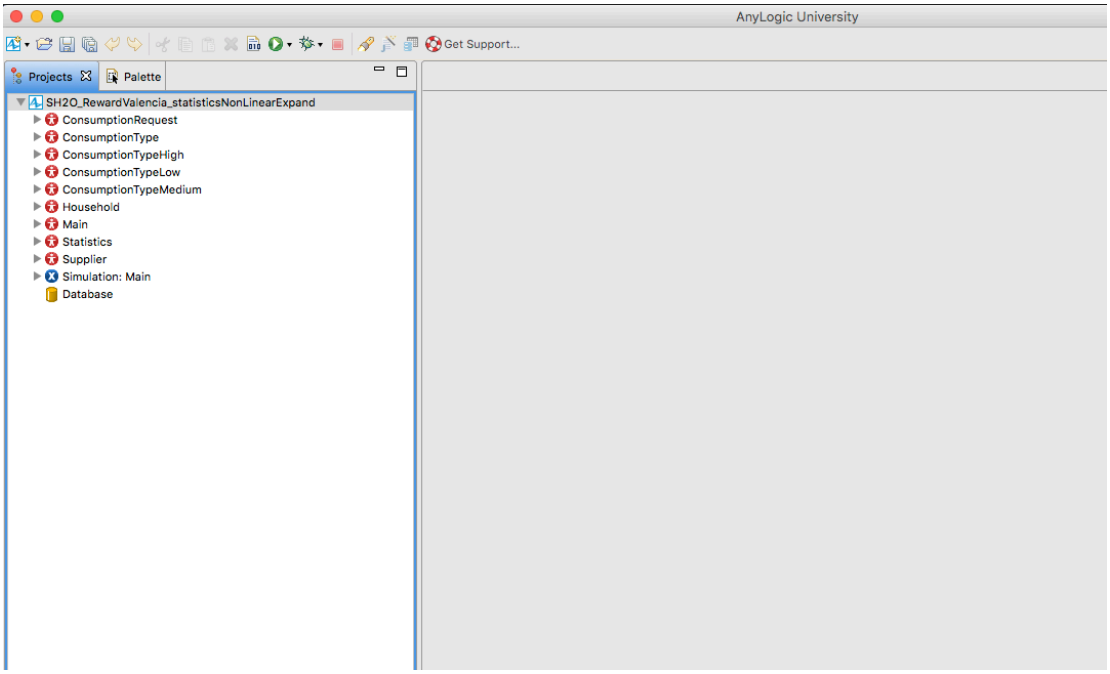

Figure 12: Opening window of the ABM model.

To start the simulation, proceeds as follows: locate the Run button and choose SH2O_RewardValencia_statisticsNonLinearExpand / Simulation from the list, as shown in Figure 13. After you start the model, the presentation window displays the launches experiment *Simulation*. Click the "Run" button to run the model

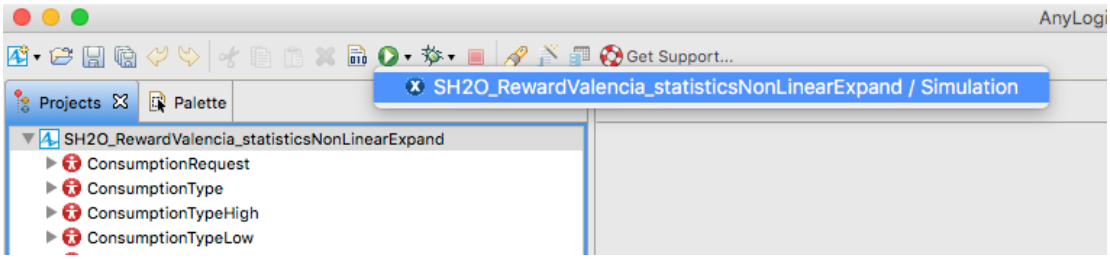

Figure 13. Selection of the Run button in the opening window.

3. Appendix A: SmartH2O platform database DDL script – second prototype

3.1 SES Demo Case script

```
CREATE DATABASE `smarth2o` /*!40100 DEFAULT CHARACTER SET utf8 */;
USE `smarth2o`;

#
# Source for table alert
#

CREATE TABLE `alert` (
  `oid` int(11) NOT NULL,
  `type` varchar(255) DEFAULT NULL,
  `level` int(11) DEFAULT NULL,
  `date` datetime DEFAULT NULL,
  `neutral_user_oid` int(11) DEFAULT NULL,
  `mail_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_alert_neutral_user` (`neutral_user_oid`),
  KEY `fk_alert_mail` (`mail_oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table baseline
#

CREATE TABLE `baseline` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `smart_meter_id` varchar(255) DEFAULT NULL,
  `smart_meter_oid` int(11) DEFAULT NULL,
  `total_consumption` decimal(19,3) DEFAULT NULL,
  `year` year(4) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_baseline_smart_meter` (`smart_meter_oid`)
) ENGINE=InnoDB AUTO_INCREMENT=689 DEFAULT CHARSET=utf8;

#
# Source for table baseline_corrado
#

CREATE TABLE `baseline_corrado` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `smart_meter_oid` int(11) NOT NULL,
```

```
  `user` varchar(255) DEFAULT NULL,
  `household_type` varchar(100) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `meter_oid_idx` (`smart_meter_oid`)
) ENGINE=InnoDB AUTO_INCREMENT=242 DEFAULT CHARSET=utf8;

#
# Source for table bill
#

CREATE TABLE `bill` (
  `oid` int(11) NOT NULL,
  `account_number` varchar(255) DEFAULT NULL,
  `bill_date` date DEFAULT NULL,
  `company` varchar(255) DEFAULT NULL,
  `volume_charge` decimal(19,2) DEFAULT NULL,
  `service_charge` decimal(19,2) DEFAULT NULL,
  `currency` varchar(255) DEFAULT NULL,
  `volume_eur_charge` decimal(19,2) DEFAULT NULL,
  `service_eur_charge` decimal(19,2) DEFAULT NULL,
  `exchange_rate` decimal(19,2) DEFAULT NULL,
  `exchange_date` date DEFAULT NULL,
  `household_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_bill_household` (`household_oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table billing_price
#

CREATE TABLE `billing_price` (
  `oid` int(11) NOT NULL,
  `month` varchar(255) DEFAULT NULL,
  `year` int(11) DEFAULT NULL,
  `company` varchar(255) DEFAULT NULL,
  `monthly_service_charge` decimal(19,2) DEFAULT NULL,
  `monthly_volume_charge` decimal(19,2) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table billing_price_bill
#

CREATE TABLE `billing_price_bill` (
  `billing_price_oid` int(11) NOT NULL,
```

```
  `bill_oid` int(11) NOT NULL,
  PRIMARY KEY (`billing_price_oid`,`bill_oid`),
  KEY `fk_billing_price_bill_billing` (`billing_price_oid`),
  KEY `fk_billing_price_bill_bill` (`bill_oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table building
#

CREATE TABLE `building` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `age` int(11) DEFAULT NULL,
  `building_size` decimal(19,2) DEFAULT NULL,
  `address` varchar(255) DEFAULT NULL,
  `district_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_building_district` (`district_oid`)
) ENGINE=InnoDB AUTO_INCREMENT=534 DEFAULT CHARSET=utf8;

#
# Source for table complex_device_instance
#

CREATE TABLE `complex_device_instance` (
  `oid` int(11) NOT NULL,
  `efficiency` varchar(255) DEFAULT NULL,
  `ecomode` bit(1) DEFAULT NULL,
  `timer` bit(1) DEFAULT NULL,
  `device_type_oid` int(11) DEFAULT NULL,
  `household_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_complex_device_instance_dev` (`device_type_oid`),
  KEY `fk_complex_device_instance_hou` (`household_oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table consumer_segment
#

CREATE TABLE `consumer_segment` (
  `oid` int(11) NOT NULL,
  `name` varchar(255) DEFAULT NULL,
  `description` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
#
# Source for table consumer_segment_neutral_user
#

CREATE TABLE `consumer_segment_neutral_user` (
  `consumer_segment_oid` int(11) NOT NULL,
  `neutral_user_oid` int(11) NOT NULL,
  PRIMARY KEY (`consumer_segment_oid`,`neutral_user_oid`),
  KEY `fk_consumer_segment_neutral_us` (`consumer_segment_oid`),
  KEY `fk_consumer_segment_neutral_2` (`neutral_user_oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table device_consumption
#

CREATE TABLE `device_consumption` (
  `oid` int(11) NOT NULL DEFAULT '0',
  `start_date` datetime DEFAULT NULL,
  `end_date` datetime DEFAULT NULL,
  `device_consumption` decimal(19,2) DEFAULT NULL,
  `complex_device_instance_oid` int(11) DEFAULT NULL,
  `simple_device_instance_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY                 `fk_device_consumption_complex_device_instance`
(`complex_device_instance_oid`),
  KEY                  `fk_device_consumption_simple_device_instance`
(`simple_device_instance_oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table device_type
#

CREATE TABLE `device_type` (
  `oid` int(11) NOT NULL,
  `name` varchar(255) DEFAULT NULL,
  `icon` text,
  `type` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table district
#

CREATE TABLE `district` (
```

```
  `oid` int(11) NOT NULL,
  `zipcode` varchar(255) DEFAULT NULL,
  `country` varchar(255) DEFAULT NULL,
  `city` varchar(255) DEFAULT NULL,
  `name` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table feature
#

CREATE TABLE `feature` (
  `oid` int(11) NOT NULL,
  `type` varchar(255) DEFAULT NULL,
  `level` int(11) DEFAULT NULL,
  `consumer_segment_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_feature_consumer_segment` (`consumer_segment_oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table granularity
#

CREATE TABLE `granularity` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `granularity` varchar(255) DEFAULT NULL,
  `start_date` date DEFAULT NULL,
  `end_date` date DEFAULT NULL,
  `user_oid` int(11) NOT NULL DEFAULT '0',
  PRIMARY KEY (`oid`),
  KEY `fk_granularity_user` (`user_oid`)
)    ENGINE=InnoDB    AUTO_INCREMENT=6201    DEFAULT    CHARSET=utf8
ROW_FORMAT=COMPACT;

#
# Source for table group
#

CREATE TABLE `group` (
  `oid` int(11) NOT NULL,
  `groupname` varchar(255) DEFAULT NULL,
  `module_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_group_module` (`module_oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
#
# Source for table group_module
#

CREATE TABLE `group_module` (
  `group_oid` int(11) NOT NULL,
  `module_oid` int(11) NOT NULL,
  PRIMARY KEY (`group_oid`,`module_oid`),
  KEY `fk_group_module_group` (`group_oid`),
  KEY `fk_group_module_module` (`module_oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table household
#

CREATE TABLE `household` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `household_size` decimal(19,2) DEFAULT NULL,
  `ownership` bit(1) DEFAULT NULL,
  `number_pets` int(11) DEFAULT NULL,
  `household_garden_area` decimal(19,2) DEFAULT NULL,
  `household_pool_volume` decimal(19,2) DEFAULT NULL,
  `second` bit(1) DEFAULT NULL,
  `public` bit(1) DEFAULT NULL,
  `visible` bit(1) DEFAULT NULL,
  `household_pool` bit(1) DEFAULT NULL,
  `household_garden` bit(1) DEFAULT NULL,
  `family_id` varchar(255) DEFAULT NULL,
  `smart_meter_oid` int(11) DEFAULT NULL,
  `building_oid` int(11) DEFAULT NULL,
  `children9` int(11) DEFAULT NULL,
  `children5_9` int(11) DEFAULT NULL,
  `children0_4` int(11) DEFAULT NULL,
  `residency_type` varchar(255) DEFAULT NULL,
  `number_bathrooms` varchar(255) DEFAULT NULL,
  `number_adults` int(11) DEFAULT NULL,
  `environmental_attitude` varchar(255) DEFAULT NULL,
  `irrigation_system` bit(1) DEFAULT NULL,
  `house_plants` bit(1) DEFAULT NULL,
  `balcony_plants` bit(1) DEFAULT NULL,
  `balcony_irrigation` bit(1) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_household_smart_meter` (`smart_meter_oid`),
  KEY `fk_household_building` (`building_oid`)
) ENGINE=InnoDB AUTO_INCREMENT=686 DEFAULT CHARSET=utf8;
```

```
#
# Source for table household_consumption
#

CREATE TABLE `household_consumption` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `consumption` decimal(19,3) DEFAULT NULL,
  `start_date` datetime DEFAULT NULL,
  `end_date` datetime DEFAULT NULL,
  `household_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_household_consumption_house` (`household_oid`)
) ENGINE=InnoDB AUTO_INCREMENT=96983 DEFAULT CHARSET=utf8;

#
# Source for table mail
#

CREATE TABLE `mail` (
  `oid` int(11) NOT NULL,
  `description` varchar(255) DEFAULT NULL,
  `subject` varchar(255) DEFAULT NULL,
  `body` longtext,
  `language` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table media_asset
#

CREATE TABLE `media_asset` (
  `oid` int(11) NOT NULL,
  `title` varchar(255) DEFAULT NULL,
  `description` varchar(255) DEFAULT NULL,
  `duration` decimal(19,2) DEFAULT NULL,
  `author` varchar(255) DEFAULT NULL,
  `media` varchar(255) DEFAULT NULL,
  `img_preview` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table meter_reading
#
```

```
CREATE TABLE `meter_reading` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `reading_date_time` datetime DEFAULT NULL,
  `company` varchar(255) DEFAULT NULL,
  `total_consumption` decimal(19,3) DEFAULT NULL,
  `smart_meter_oid` int(11) DEFAULT NULL,
  `total_consumption_adjusted` decimal(19,3) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  UNIQUE KEY `unique_reading`
(`reading_date_time`,`smart_meter_oid`),
  KEY `fk_meter_reading_smart_meter` (`smart_meter_oid`)
) ENGINE=InnoDB AUTO_INCREMENT=3818788 DEFAULT CHARSET=utf8;

#
# Source for table meter_reading_smdm
#

CREATE TABLE `meter_reading_smdm` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `reading_date_time` datetime DEFAULT NULL,
  `total_consumption` decimal(19,3) DEFAULT NULL,
  `smart_meter_id` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  UNIQUE KEY `unique_reading` (`reading_date_time`,`smart_meter_id`)
) ENGINE=InnoDB AUTO_INCREMENT=1497689 DEFAULT CHARSET=utf8;

#
# Source for table meter_reading_temp
#

CREATE TABLE `meter_reading_temp` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `reading_date_time` datetime DEFAULT NULL,
  `total_consumption` decimal(19,3) DEFAULT NULL,
  `smart_meter_oid` int(11) DEFAULT NULL,
  `deviation` decimal(10,3) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB AUTO_INCREMENT=462 DEFAULT CHARSET=utf8;

#
# Source for table meter_reading_weekly
#

CREATE TABLE `meter_reading_weekly` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `reading_date_time` datetime DEFAULT NULL,
  `company` varchar(255) DEFAULT NULL,
```

```
  `total_consumption` decimal(19,3) DEFAULT NULL,
  `smart_meter_oid` int(11) DEFAULT NULL,
  `total_consumption_adjusted` decimal(19,3) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  UNIQUE                    KEY                    `unique_reading`
(`reading_date_time`,`smart_meter_oid`),
  KEY `fk_meter_reading_weekly_smart_meter` (`smart_meter_oid`)
) ENGINE=InnoDB AUTO_INCREMENT=534270 DEFAULT CHARSET=utf8;

#
# Source for table meter_reading_weekly_smdm
#

CREATE TABLE `meter_reading_weekly_smdm` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `reading_date_time` datetime DEFAULT NULL,
  `company` varchar(255) DEFAULT NULL,
  `total_consumption` decimal(19,3) DEFAULT NULL,
  `smart_meter_oid` int(11) DEFAULT NULL,
  `total_consumption_adjusted` decimal(19,3) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  UNIQUE                    KEY                    `unique_reading`
(`reading_date_time`,`smart_meter_oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table module
#

CREATE TABLE `module` (
  `oid` int(11) NOT NULL,
  `moduleid` varchar(255) DEFAULT NULL,
  `moduledomainname` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table neutral_user
#

CREATE TABLE `neutral_user` (
  `user_oid` int(11) NOT NULL AUTO_INCREMENT,
  `registration_date` date DEFAULT NULL,
  `family_role` varchar(255) DEFAULT NULL,
  `house_holder` bit(1) DEFAULT NULL,
  `educational_level` varchar(255) DEFAULT NULL,
  `income_rate` varchar(255) DEFAULT NULL,
```

```
  `currency` varchar(255) DEFAULT NULL,
  `public` bit(1) DEFAULT NULL,
  `language` varchar(255) DEFAULT NULL,
  `temperature_unit` varchar(255) DEFAULT NULL,
  `length_unit` varchar(255) DEFAULT NULL,
  `household_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`user_oid`),
  KEY `fk_neutral_user_household` (`household_oid`),
  KEY `fk_neutral_user_user` (`user_oid`)
) ENGINE=InnoDB AUTO_INCREMENT=928 DEFAULT CHARSET=utf8;

#
# Source for table neutral_user_media_asset
#

CREATE TABLE `neutral_user_media_asset` (
  `neutral_user_oid` int(11) NOT NULL,
  `media_asset_oid` int(11) NOT NULL,
  PRIMARY KEY (`neutral_user_oid`,`media_asset_oid`),
  KEY `fk_neutral_user_mediaasset_neu` (`neutral_user_oid`),
  KEY `fk_neutral_user_mediaasset_med` (`media_asset_oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table neutral_user_tip
#

CREATE TABLE `neutral_user_tip` (
  `neutral_user_oid` int(11) NOT NULL,
  `tip_oid` int(11) NOT NULL,
  PRIMARY KEY (`neutral_user_oid`,`tip_oid`),
  KEY `fk_neutral_user_tip_neutral_us` (`neutral_user_oid`),
  KEY `fk_neutral_user_tip_tip` (`tip_oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table newdata
#

CREATE TABLE `newdata` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `family_ses_id` varchar(255) DEFAULT NULL,
  `description` varchar(255) DEFAULT NULL,
  `smart_meter_ses_id` varchar(255) DEFAULT NULL,
  `building_ses_id` int(11) DEFAULT NULL,
  `building_real_id` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`)
```

```
) ENGINE=InnoDB AUTO_INCREMENT=695 DEFAULT CHARSET=latin1;

#
# Source for table simple_device_instance
#

CREATE TABLE `simple_device_instance` (
  `oid` int(11) NOT NULL,
  `number` int(11) DEFAULT NULL,
  `device_type_oid` int(11) DEFAULT NULL,
  `household_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_simple_device_instance_devi` (`device_type_oid`),
  KEY `fk_simple_device_instance_hous` (`household_oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table smart_meter
#

CREATE TABLE `smart_meter` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `smart_meter_id` varchar(255) DEFAULT NULL,
  `building_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_smart_meter_building` (`building_oid`)
) ENGINE=InnoDB AUTO_INCREMENT=410 DEFAULT CHARSET=utf8;

#
# Source for table smart_meter_new
#

CREATE TABLE `smart_meter_new` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `smart_meter_id` varchar(255) DEFAULT NULL,
  `building_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_smart_meter_building` (`building_oid`)
) ENGINE=InnoDB AUTO_INCREMENT=317 DEFAULT CHARSET=utf8;

#
# Source for table sso_token
#

CREATE TABLE `sso_token` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `token` varchar(255) DEFAULT NULL,
```

```
  `username` varchar(255) DEFAULT NULL,
  `date` timestamp NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
  `ttl` varchar(255) DEFAULT NULL,
  `state` varchar(1) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8;

#
# Source for table tip
#

CREATE TABLE `tip` (
  `oid` int(11) NOT NULL,
  `name` varchar(255) DEFAULT NULL,
  `header` varchar(255) DEFAULT NULL,
  `body` longtext,
  `tipdate` date DEFAULT NULL,
  `language` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table unit_of_measurement
#

CREATE TABLE `unit_of_measurement` (
  `oid` int(11) NOT NULL,
  `physical_quantity` varchar(255) DEFAULT NULL,
  `primary_unit` varchar(255) DEFAULT NULL,
  `secondary_unit` varchar(255) DEFAULT NULL,
  `conversion_coefficient` decimal(19,2) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table user
#

CREATE TABLE `user` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `username` varchar(255) DEFAULT NULL,
  `password` varchar(255) DEFAULT NULL,
  `email` varchar(255) DEFAULT NULL,
  `first_name` varchar(255) DEFAULT NULL,
  `last_name` varchar(255) DEFAULT NULL,
  `birth_date` varchar(255) DEFAULT NULL,
```

```
  `internal` bit(1) DEFAULT NULL,
  `group_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_user_group` (`group_oid`)
) ENGINE=InnoDB AUTO_INCREMENT=931 DEFAULT CHARSET=utf8;

#
# Source for table user_group
#

CREATE TABLE `user_group` (
  `user_oid` int(11) NOT NULL,
  `group_oid` int(11) NOT NULL,
  PRIMARY KEY (`user_oid`,`group_oid`),
  KEY `fk_user_group_user` (`user_oid`),
  KEY `fk_user_group_group` (`group_oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table weather_condition
#

CREATE TABLE `weather_condition` (
  `oid` int(11) NOT NULL,
  `start_date` date DEFAULT NULL,
  `end_date` date DEFAULT NULL,
  `rain_fall` decimal(19,2) DEFAULT NULL,
  `average_temperature` decimal(19,2) DEFAULT NULL,
  `district_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_weather_condition_district` (`district_oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for procedure adjustMeterReadings
#

CREATE DEFINER=`superadmin`@`%` PROCEDURE `adjustMeterReadings`()
BEGIN

 DECLARE c_finished INTEGER DEFAULT 0;
 DECLARE c_finished_mr INTEGER DEFAULT 0;
 DECLARE c_finished_amr INTEGER DEFAULT 0;
 DECLARE v_smart_meter_oid INTEGER DEFAULT 0;

 DECLARE mr_smart_meter_oid INTEGER DEFAULT 0;
 DECLARE mr_reading_date_time DATETIME;
```

```
 DECLARE mr_total_consumption DECIMAL(19,3);

 DECLARE amr_smart_meter_oid INTEGER DEFAULT 0;
 DECLARE amr_start_date_time DATETIME;
 DECLARE amr_end_date_time DATETIME;
 DECLARE amr_deviation DECIMAL(19,3);

 DECLARE old_smart_meter_oid INTEGER;
 DECLARE old_reading_date_time DATETIME;
 DECLARE old_total_consumption DECIMAL(19,3);


 -- filtering out of range consumption and saving into
meter_reading_temp table

 -- declare cursor for smart meter oid
 DECLARE smart_meter_oid_cursor CURSOR FOR
         SELECT oid FROM smart_meter;
 DECLARE CONTINUE HANDLER FOR SQLSTATE '02000' SET c_finished = 1;

 TRUNCATE meter_reading_temp;
 SET old_total_consumption :=0;

 OPEN smart_meter_oid_cursor;

 get_smart_meter_id_loop: LOOP
 FETCH NEXT FROM smart_meter_oid_cursor INTO v_smart_meter_oid;

 IF c_finished = 1 THEN
     CLOSE smart_meter_oid_cursor;
     LEAVE get_smart_meter_id_loop;
 END IF;


     BEGIN
         -- declare cursor for meter_reading
         DECLARE meter_reading_cursor CURSOR FOR
         SELECT          smart_meter_oid,          reading_date_time,
total_consumption   FROM   meter_reading   WHERE   smart_meter_oid   =
v_smart_meter_oid ORDER BY reading_date_time;
         DECLARE   CONTINUE   HANDLER   FOR     SQLSTATE   '02000'   SET
c_finished_mr = 1;

         OPEN meter_reading_cursor;

         get_meter_reading_loop: LOOP
             FETCH     NEXT     FROM     meter_reading_cursor     INTO
mr_smart_meter_oid, mr_reading_date_time, mr_total_consumption;
```

```
             IF c_finished_mr = 1 THEN
             SET c_finished_mr = 0;
                CLOSE meter_reading_cursor;
                LEAVE get_meter_reading_loop;
            END IF;


            IF old_total_consumption != 0  THEN
               IF (mr_total_consumption / old_total_consumption) >=
9 THEN
                  INSERT   INTO   meter_reading_temp   VALUES(null,
mr_reading_date_time,    mr_total_consumption,    mr_smart_meter_oid,
mr_total_consumption / old_total_consumption);
               END IF;
            END IF;

            IF old_total_consumption != 0  THEN
               IF (old_total_consumption / mr_total_consumption) >=
9 THEN
                  INSERT   INTO   meter_reading_temp   VALUES(null,
old_reading_date_time,  old_total_consumption,  old_smart_meter_oid,
old_total_consumption / mr_total_consumption);
               END IF;
            END IF;

            SET old_smart_meter_oid = mr_smart_meter_oid;
            SET old_reading_date_time = mr_reading_date_time;
            SET old_total_consumption = mr_total_consumption;

         END LOOP get_meter_reading_loop;

         SET old_total_consumption =0;
   END;

 END LOOP get_smart_meter_id_loop;

 --       adjusting       meter_reading.total_consumption       into
total_consumption_adjusted

 UPDATE meter_reading
 SET total_consumption_adjusted = total_consumption;

 BEGIN
     -- declare cursor for adjusting smart meter
     DECLARE adjusting_meter_reading_cursor CURSOR FOR
             SELECT      smart_meter_oid,      MIN(reading_date_time)
start_date_time,        MAX(reading_date_time)        end_date_time,
MIN(deviation)
```

```
            FROM meter_reading_temp
            GROUP BY smart_meter_oid;
    DECLARE CONTINUE HANDLER FOR SQLSTATE '02000' SET
c_finished_amr = 1;

        OPEN adjusting_meter_reading_cursor;

        get_adjusting_meter_reading_loop: LOOP
            FETCH NEXT FROM adjusting_meter_reading_cursor INTO
amr_smart_meter_oid, amr_start_date_time, amr_end_date_time,
amr_deviation;

             IF c_finished_amr = 1 THEN
             SET c_finished_amr = 0;
                CLOSE adjusting_meter_reading_cursor;
                LEAVE get_adjusting_meter_reading_loop;
            END IF;

        UPDATE meter_reading
        SET total_consumption_adjusted =
        (CASE WHEN amr_deviation > 800 THEN total_consumption/1000
        ELSE CASE WHEN amr_deviation > 90 THEN
total_consumption/100
        ELSE CASE WHEN amr_deviation > 9 THEN total_consumption/10
ELSE total_consumption END END END)
        WHERE smart_meter_oid = amr_smart_meter_oid and
reading_date_time >= amr_start_date_time AND reading_date_time <=
amr_end_date_time;

        END LOOP get_adjusting_meter_reading_loop;
 END;

END;


#
# Source for procedure adjustMeterReadingsWeekly
#

CREATE DEFINER=`superadmin`@`%` PROCEDURE
`adjustMeterReadingsWeekly`()
BEGIN

 DECLARE c_finished INTEGER DEFAULT 0;
 DECLARE c_finished_mr INTEGER DEFAULT 0;
 DECLARE c_finished_amr INTEGER DEFAULT 0;
 DECLARE v_smart_meter_oid INTEGER DEFAULT 0;
```

```
 DECLARE mr_smart_meter_oid INTEGER DEFAULT 0;
 DECLARE mr_reading_date_time DATETIME;
 DECLARE mr_total_consumption DECIMAL(19,3);

 DECLARE amr_smart_meter_oid INTEGER DEFAULT 0;
 DECLARE amr_start_date_time DATETIME;
 DECLARE amr_end_date_time DATETIME;
 DECLARE amr_deviation DECIMAL(19,3);

 DECLARE old_smart_meter_oid INTEGER;
 DECLARE old_reading_date_time DATETIME;
 DECLARE old_total_consumption DECIMAL(19,3);


 -- filtering out of range consumption and saving into
meter_reading_temp table

 -- declare cursor for smart meter oid
 DECLARE smart_meter_oid_cursor CURSOR FOR
         SELECT oid FROM smart_meter;
 DECLARE CONTINUE HANDLER FOR SQLSTATE '02000' SET c_finished = 1;

 TRUNCATE meter_reading_temp;
 SET old_total_consumption :=0;

 OPEN smart_meter_oid_cursor;

 get_smart_meter_id_loop: LOOP
 FETCH NEXT FROM smart_meter_oid_cursor INTO v_smart_meter_oid;

 IF c_finished = 1 THEN
     CLOSE smart_meter_oid_cursor;
     LEAVE get_smart_meter_id_loop;
 END IF;


     BEGIN
         -- declare cursor for meter_reading_weekly
         DECLARE meter_reading_weekly_cursor CURSOR FOR
         SELECT          smart_meter_oid,          reading_date_time,
total_consumption  FROM  meter_reading_weekly  WHERE  smart_meter_oid  =
v_smart_meter_oid ORDER BY reading_date_time;
         DECLARE   CONTINUE   HANDLER   FOR     SQLSTATE   '02000'   SET
c_finished_mr = 1;

         OPEN meter_reading_weekly_cursor;
```

```
        get_meter_reading_weekly_loop: LOOP
            FETCH   NEXT   FROM   meter_reading_weekly_cursor   INTO
mr_smart_meter_oid, mr_reading_date_time, mr_total_consumption;

             IF c_finished_mr = 1 THEN
             SET c_finished_mr = 0;
                CLOSE meter_reading_weekly_cursor;
                LEAVE get_meter_reading_weekly_loop;
            END IF;



            IF old_total_consumption != 0  THEN
               IF (mr_total_consumption / old_total_consumption) >=
9 THEN
                  INSERT   INTO   meter_reading_temp   VALUES(null,
mr_reading_date_time,    mr_total_consumption,    mr_smart_meter_oid,
mr_total_consumption / old_total_consumption);
               END IF;
            END IF;

            IF old_total_consumption != 0  THEN
               IF (old_total_consumption / mr_total_consumption) >=
9 THEN
                  INSERT   INTO   meter_reading_temp   VALUES(null,
old_reading_date_time,  old_total_consumption,  old_smart_meter_oid,
old_total_consumption / mr_total_consumption);
               END IF;
            END IF;

            SET old_smart_meter_oid = mr_smart_meter_oid;
            SET old_reading_date_time = mr_reading_date_time;
            SET old_total_consumption = mr_total_consumption;

        END LOOP get_meter_reading_weekly_loop;

        SET old_total_consumption =0;
   END;

 END LOOP get_smart_meter_id_loop;

 --     adjusting     meter_reading_weekly.total_consumption     into
total_consumption_adjusted

 UPDATE meter_reading_weekly
 SET total_consumption_adjusted = total_consumption;

 BEGIN
     -- declare cursor for adjusting smart meter
```

```
    DECLARE adjusting_meter_reading_weekly_cursor CURSOR FOR
            SELECT      smart_meter_oid,      MIN(reading_date_time)
start_date_time,        MAX(reading_date_time)        end_date_time,
MIN(deviation)
            FROM meter_reading_temp
            GROUP BY smart_meter_oid;
    DECLARE   CONTINUE   HANDLER   FOR   SQLSTATE   '02000'   SET
c_finished_amr = 1;

        OPEN adjusting_meter_reading_weekly_cursor;

        get_adjusting_meter_reading_weekly_loop: LOOP
            FETCH  NEXT  FROM  adjusting_meter_reading_weekly_cursor
INTO  amr_smart_meter_oid,  amr_start_date_time,  amr_end_date_time,
amr_deviation;

             IF c_finished_amr = 1 THEN
             SET c_finished_amr = 0;
                CLOSE adjusting_meter_reading_weekly_cursor;
                LEAVE get_adjusting_meter_reading_weekly_loop;
            END IF;

        UPDATE meter_reading_weekly
        SET total_consumption_adjusted =
        (CASE WHEN amr_deviation > 800 THEN total_consumption/1000
        ELSE    CASE    WHEN    amr_deviation    >    90    THEN
total_consumption/100
        ELSE CASE WHEN amr_deviation > 9 THEN total_consumption/10
ELSE total_consumption END END END)
        WHERE    smart_meter_oid    =    amr_smart_meter_oid    and
reading_date_time  >=  amr_start_date_time  AND  reading_date_time  <=
amr_end_date_time;

        END LOOP get_adjusting_meter_reading_weekly_loop;
 END;

END;



#
# Source for procedure getGranularity
#

CREATE DEFINER=`superadmin`@`%` PROCEDURE `getGranularity`(`user_id`
int(11))
BEGIN

 DECLARE c_finished INTEGER DEFAULT 0;
 DECLARE granularity VARCHAR(255) DEFAULT '';
```

```
 DECLARE v_user_oid INTEGER DEFAULT 0;
 DECLARE v_freq INTEGER DEFAULT 0;
 DECLARE v_day_start DATE;
 DECLARE v_day_end DATE;
 DECLARE v_recno INTEGER DEFAULT 0;
 DECLARE old_v_user_oid INTEGER DEFAULT 0;
 DECLARE old_v_freq INTEGER DEFAULT 0;
 DECLARE old_v_day_start DATE;
 DECLARE old_v_day_end DATE;
 DECLARE old_v_recno INTEGER DEFAULT 0;

 -- declare cursor for granularity
 DECLARE c_granularity_cursor CURSOR FOR


 select oid, min(dif) as frequency, ziua1 as day_start, ziua2 as
day_end, recno
 from
 (select T2.oid, DATEDIFF(T2.ziua2, T1.ziua1) dif, T1.ziua1,
T2.ziua2, T1.recno1 recno from
 (select u.oid, date(m.reading_date_time) ziua1,
count(date(m.reading_date_time)) recno1
            from meter_reading m
            left outer join smart_meter sm on sm.oid =
m.smart_meter_oid
            left outer join household h on sm.oid =
h.smart_meter_oid
            left outer join building b on b.oid = h.building_oid
            left outer join neutral_user nu on nu.household_oid =
h.oid
        left outer join user u on u.oid = nu.user_oid
            where u.oid = user_id
 group by date(m.reading_date_time)
 order by date(m.reading_date_time) desc) T1 JOIN
 (select u.oid, date(m.reading_date_time) ziua2,
count(date(m.reading_date_time)) recno2
            from meter_reading m
            left outer join smart_meter sm on sm.oid =
m.smart_meter_oid
            left outer join household h on sm.oid =
h.smart_meter_oid
            left outer join building b on b.oid = h.building_oid
            left outer join neutral_user nu on nu.household_oid =
h.oid
        left outer join user u on u.oid = nu.user_oid
            where u.oid = user_id
 group by date(m.reading_date_time)
 order by date(m.reading_date_time) desc) T2 ON T1.ziua1 <
```

```
T2.ziua2)e
 group by e.ziua2
 order by day_end;

 DECLARE CONTINUE HANDLER FOR SQLSTATE '02000' SET c_finished = 1;

 DELETE FROM granularity WHERE user_oid = user_id;

 OPEN c_granularity_cursor;

 granularity_loop: LOOP
 FETCH NEXT FROM c_granularity_cursor INTO v_user_oid, v_freq,
v_day_start, v_day_end, v_recno;

 IF c_finished = 1 THEN
     CLOSE c_granularity_cursor;
     LEAVE granularity_loop;
 END IF;

 IF old_v_freq != v_freq THEN

   IF v_freq = 1 THEN
      IF v_recno > 1 THEN
         SET granularity = 'hourly';
      ELSE
         SET granularity = 'daily';
      END IF;
   ELSE
          IF v_freq >= 7 THEN
            SET granularity = 'weekly';
          ELSE
                 IF v_freq >= 30 THEN
                   SET granularity = 'monthly';
                 END IF;
          END IF;
   END IF;

   INSERT INTO granularity(granularity, start_date, end_date,
user_oid) VALUES(granularity, v_day_start, v_day_end, v_user_oid);
 END IF;

 SET old_v_user_oid = v_user_oid;
 SET old_v_freq = v_freq;
 SET old_v_day_start = v_day_start;
 SET old_v_day_end = v_day_end;
 SET old_v_recno = v_recno;
```

```
 END LOOP granularity_loop;

END;


#
# Source for function RowNum
#

CREATE DEFINER=`superadmin`@`%` FUNCTION `RowNum`() RETURNS int(11)
BEGIN
     SET @current_row = 0;
     SET @current_row := @current_row + 1;

  RETURN @current_row;
END;



#
#  Foreign keys for table alert
#

ALTER TABLE `alert`
ADD CONSTRAINT `fk_alert_mail` FOREIGN KEY (`mail_oid`) REFERENCES
`mail` (`oid`),
ADD     CONSTRAINT     `fk_alert_neutral_user`     FOREIGN     KEY
(`neutral_user_oid`) REFERENCES `neutral_user` (`user_oid`);


#
#  Foreign keys for table baseline
#

ALTER TABLE `baseline`
ADD     CONSTRAINT     `fk_baseline_smart_meter`     FOREIGN     KEY
(`smart_meter_oid`) REFERENCES `smart_meter` (`oid`);


#
#  Foreign keys for table baseline_corrado
#

ALTER TABLE `baseline_corrado`
ADD   CONSTRAINT   `meter_oid`   FOREIGN   KEY   (`smart_meter_oid`)
REFERENCES   `smart_meter`   (`oid`)   ON   DELETE   CASCADE   ON   UPDATE
CASCADE;


#
#  Foreign keys for table bill
#
```

```
ALTER TABLE `bill`
ADD CONSTRAINT `fk_bill_household` FOREIGN KEY (`household_oid`)
REFERENCES `household` (`oid`);

#
#  Foreign keys for table billing_price_bill
#

ALTER TABLE `billing_price_bill`
ADD CONSTRAINT `fk_billing_price_bill_bill` FOREIGN KEY (`bill_oid`)
REFERENCES `bill` (`oid`),
ADD CONSTRAINT `fk_billing_price_bill_billing` FOREIGN KEY
(`billing_price_oid`) REFERENCES `billing_price` (`oid`);

#
#  Foreign keys for table building
#

ALTER TABLE `building`
ADD CONSTRAINT `fk_building_district` FOREIGN KEY (`district_oid`)
REFERENCES `district` (`oid`);

#
#  Foreign keys for table complex_device_instance
#

ALTER TABLE `complex_device_instance`
ADD CONSTRAINT `fk_complex_device_instance_dev` FOREIGN KEY
(`device_type_oid`) REFERENCES `device_type` (`oid`),
ADD CONSTRAINT `fk_complex_device_instance_hou` FOREIGN KEY
(`household_oid`) REFERENCES `household` (`oid`);

#
#  Foreign keys for table consumer_segment_neutral_user
#

ALTER TABLE `consumer_segment_neutral_user`
ADD CONSTRAINT `fk_consumer_segment_neutral_2` FOREIGN KEY
(`neutral_user_oid`) REFERENCES `neutral_user` (`user_oid`),
ADD CONSTRAINT `fk_consumer_segment_neutral_us` FOREIGN KEY
(`consumer_segment_oid`) REFERENCES `consumer_segment` (`oid`);

#
#  Foreign keys for table device_consumption
#

ALTER TABLE `device_consumption`
```

```
ADD    CONSTRAINT    `fk_device_consumption_complex_device_instance`
FOREIGN      KEY      (`complex_device_instance_oid`)      REFERENCES
`complex_device_instance` (`oid`),
ADD    CONSTRAINT    `fk_device_consumption_simple_device_instance`
FOREIGN      KEY      (`simple_device_instance_oid`)      REFERENCES
`simple_device_instance` (`oid`);

#
#  Foreign keys for table feature
#

ALTER TABLE `feature`
ADD    CONSTRAINT    `fk_feature_consumer_segment`    FOREIGN    KEY
(`consumer_segment_oid`) REFERENCES `consumer_segment` (`oid`);

#
#  Foreign keys for table granularity
#

ALTER TABLE `granularity`
ADD   CONSTRAINT   `fk_granularity_user`   FOREIGN   KEY   (`user_oid`)
REFERENCES `user` (`oid`);

#
#  Foreign keys for table group
#

ALTER TABLE `group`
ADD   CONSTRAINT   `fk_group_module`   FOREIGN   KEY   (`module_oid`)
REFERENCES `module` (`oid`);

#
#  Foreign keys for table group_module
#

ALTER TABLE `group_module`
ADD  CONSTRAINT  `fk_group_module_group`  FOREIGN  KEY  (`group_oid`)
REFERENCES `group` (`oid`),
ADD CONSTRAINT `fk_group_module_module` FOREIGN KEY (`module_oid`)
REFERENCES `module` (`oid`);

#
#  Foreign keys for table household
#

ALTER TABLE `household`
ADD CONSTRAINT `fk_household_building` FOREIGN KEY (`building_oid`)
REFERENCES `building` (`oid`),
ADD    CONSTRAINT    `fk_household_smart_meter`    FOREIGN    KEY
```

```
(`smart_meter_oid`) REFERENCES `smart_meter` (`oid`);

#
#  Foreign keys for table household_consumption
#

ALTER TABLE `household_consumption`
ADD   CONSTRAINT   `fk_household_consumption_house`   FOREIGN   KEY
(`household_oid`) REFERENCES `household` (`oid`);

#
#  Foreign keys for table meter_reading
#

ALTER TABLE `meter_reading`
ADD   CONSTRAINT   `fk_meter_reading_smart_meter`   FOREIGN   KEY
(`smart_meter_oid`) REFERENCES `smart_meter` (`oid`);

#
#  Foreign keys for table meter_reading_weekly
#

ALTER TABLE `meter_reading_weekly`
ADD  CONSTRAINT  `fk_meter_reading_weekly_smart_meter`  FOREIGN  KEY
(`smart_meter_oid`) REFERENCES `smart_meter` (`oid`);

#
#  Foreign keys for table neutral_user
#

ALTER TABLE `neutral_user`
ADD  CONSTRAINT  `fk_neutral_user_user`  FOREIGN  KEY  (`user_oid`)
REFERENCES `user` (`oid`),
ADD    CONSTRAINT    `fk_neutral_user_household`    FOREIGN    KEY
(`household_oid`) REFERENCES `household` (`oid`);

#
#  Foreign keys for table neutral_user_media_asset
#

ALTER TABLE `neutral_user_media_asset`
ADD   CONSTRAINT   `fk_neutral_user_mediaasset_med`   FOREIGN   KEY
(`media_asset_oid`) REFERENCES `media_asset` (`oid`),
ADD   CONSTRAINT   `fk_neutral_user_mediaasset_neu`   FOREIGN   KEY
(`neutral_user_oid`) REFERENCES `neutral_user` (`user_oid`);

#
#  Foreign keys for table neutral_user_tip
```

```
#

ALTER TABLE `neutral_user_tip`
ADD CONSTRAINT `fk_neutral_user_tip_neutral_us` FOREIGN KEY
(`neutral_user_oid`) REFERENCES `neutral_user` (`user_oid`),
ADD CONSTRAINT `fk_neutral_user_tip_tip` FOREIGN KEY (`tip_oid`)
REFERENCES `tip` (`oid`);

#
#  Foreign keys for table simple_device_instance
#

ALTER TABLE `simple_device_instance`
ADD CONSTRAINT `fk_simple_device_instance_devi` FOREIGN KEY
(`device_type_oid`) REFERENCES `device_type` (`oid`),
ADD CONSTRAINT `fk_simple_device_instance_hous` FOREIGN KEY
(`household_oid`) REFERENCES `household` (`oid`);

#
#  Foreign keys for table smart_meter
#

ALTER TABLE `smart_meter`
ADD CONSTRAINT `fk_smart_meter_building` FOREIGN KEY
(`building_oid`) REFERENCES `building` (`oid`);

#
#  Foreign keys for table user
#

ALTER TABLE `user`
ADD CONSTRAINT `fk_user_group` FOREIGN KEY (`group_oid`) REFERENCES
`group` (`oid`);

#
#  Foreign keys for table user_group
#

ALTER TABLE `user_group`
ADD CONSTRAINT `fk_user_group_group` FOREIGN KEY (`group_oid`)
REFERENCES `group` (`oid`),
ADD CONSTRAINT `fk_user_group_user` FOREIGN KEY (`user_oid`)
REFERENCES `user` (`oid`);

#
#  Foreign keys for table weather_condition
#
```

```
ALTER TABLE `weather_condition`
ADD    CONSTRAINT    `fk_weather_condition_district`    FOREIGN    KEY
(`district_oid`) REFERENCES `district` (`oid`);

/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;
/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
```

3.2 EMIVASA Demo Case script

```
CREATE DATABASE `smarth2o_emi` /*!40100 DEFAULT CHARACTER SET utf8
*/;
USE `smarth2o_emi`;
#
# Source for table alert
#

CREATE TABLE `alert` (
  `oid` int(11) NOT NULL,
  `type` varchar(255) DEFAULT NULL,
  `level` int(11) DEFAULT NULL,
  `date` datetime DEFAULT NULL,
  `neutral_user_oid` varchar(255) DEFAULT NULL,
  `mail_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_alert_neutral_user` (`neutral_user_oid`),
  KEY `fk_alert_mail` (`mail_oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table baseline
#

CREATE TABLE `baseline` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `smart_meter_id` varchar(255) DEFAULT NULL,
  `smart_meter_oid` int(11) DEFAULT NULL,
  `total_consumption` decimal(19,3) DEFAULT NULL,
  `year` year(4) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB AUTO_INCREMENT=29 DEFAULT CHARSET=utf8;

#
# Source for table bill
#

CREATE TABLE `bill` (
  `oid` int(11) NOT NULL,
```

```
  `account_number` varchar(255) DEFAULT NULL,
  `bill_date` date DEFAULT NULL,
  `company` varchar(255) DEFAULT NULL,
  `volume_charge` decimal(19,2) DEFAULT NULL,
  `service_charge` decimal(19,2) DEFAULT NULL,
  `currency` varchar(255) DEFAULT NULL,
  `volume_eur_charge` decimal(19,2) DEFAULT NULL,
  `service_eur_charge` decimal(19,2) DEFAULT NULL,
  `exchange_rate` decimal(19,2) DEFAULT NULL,
  `exchange_date` date DEFAULT NULL,
  `household_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_bill_household` (`household_oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table billing_price
#

CREATE TABLE `billing_price` (
  `oid` int(11) NOT NULL,
  `month` varchar(255) DEFAULT NULL,
  `year` int(11) DEFAULT NULL,
  `company` varchar(255) DEFAULT NULL,
  `monthly_service_charge` decimal(19,2) DEFAULT NULL,
  `monthly_volume_charge` decimal(19,2) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table billing_price_bill
#

CREATE TABLE `billing_price_bill` (
  `billing_price_oid` int(11) NOT NULL,
  `bill_oid` int(11) NOT NULL,
  PRIMARY KEY (`billing_price_oid`,`bill_oid`),
  KEY `fk_billing_price_bill_billing` (`billing_price_oid`),
  KEY `fk_billing_price_bill_bill` (`bill_oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table building
#

CREATE TABLE `building` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
```

```
  `age` int(11) DEFAULT NULL,
  `building_size` decimal(19,2) DEFAULT NULL,
  `address` varchar(255) DEFAULT NULL,
  `district_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_building_district` (`district_oid`)
) ENGINE=InnoDB AUTO_INCREMENT=1616 DEFAULT CHARSET=utf8;

#
# Source for table complex_device_instance
#

CREATE TABLE `complex_device_instance` (
  `oid` int(11) NOT NULL,
  `efficiency` varchar(255) DEFAULT NULL,
  `ecomode` bit(1) DEFAULT NULL,
  `timer` bit(1) DEFAULT NULL,
  `device_type_oid` int(11) DEFAULT NULL,
  `household_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_complex_device_instance_dev` (`device_type_oid`),
  KEY `fk_complex_device_instance_hou` (`household_oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table consumer_segment
#

CREATE TABLE `consumer_segment` (
  `oid` int(11) NOT NULL,
  `name` varchar(255) DEFAULT NULL,
  `description` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table consumer_segment_neutral_user
#

CREATE TABLE `consumer_segment_neutral_user` (
  `consumer_segment_oid` int(11) NOT NULL,
  `neutral_user_oid` varchar(255) NOT NULL,
  PRIMARY KEY (`consumer_segment_oid`,`neutral_user_oid`),
  KEY `fk_consumer_segment_neutral_us` (`consumer_segment_oid`),
  KEY `fk_consumer_segment_neutral_2` (`neutral_user_oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
#
# Source for table device_consumption
#

CREATE TABLE `device_consumption` (
  `oid` int(11) NOT NULL DEFAULT '0',
  `start_date` datetime DEFAULT NULL,
  `end_date` datetime DEFAULT NULL,
  `device_consumption` decimal(19,2) DEFAULT NULL,
  `complex_device_instance_oid` int(11) DEFAULT NULL,
  `simple_device_instance_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY                 `fk_device_consumption_complex_device_instance`
(`complex_device_instance_oid`),
  KEY                  `fk_device_consumption_simple_device_instance`
(`simple_device_instance_oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table device_type
#

CREATE TABLE `device_type` (
  `oid` int(11) NOT NULL,
  `name` varchar(255) DEFAULT NULL,
  `icon` text,
  `type` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table district
#

CREATE TABLE `district` (
  `oid` int(11) NOT NULL,
  `zipcode` varchar(255) DEFAULT NULL,
  `country` varchar(255) DEFAULT NULL,
  `city` varchar(255) DEFAULT NULL,
  `name` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table emivasa_log
#
```

```
CREATE TABLE `emivasa_log` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `timestamp` timestamp NULL DEFAULT NULL,
  `user_id` varchar(255) DEFAULT NULL,
  `readings_inserted` int(11) DEFAULT NULL,
  `message_payload` varchar(1000) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=2405 DEFAULT CHARSET=latin1;

#
# Source for table emivasa_readings_completed
#

CREATE TABLE `emivasa_readings_completed` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `timestamp` timestamp NULL DEFAULT NULL,
  `user_id` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=2405 DEFAULT CHARSET=latin1;

#
# Source for table feature
#

CREATE TABLE `feature` (
  `oid` int(11) NOT NULL,
  `type` varchar(255) DEFAULT NULL,
  `level` int(11) DEFAULT NULL,
  `consumer_segment_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_feature_consumer_segment` (`consumer_segment_oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table granularity
#

CREATE TABLE `granularity` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `granularity` varchar(255) DEFAULT NULL,
  `start_date` date DEFAULT NULL,
  `end_date` date DEFAULT NULL,
  `user_oid` varchar(255) NOT NULL DEFAULT '',
  PRIMARY KEY (`oid`),
  KEY `fk_granularity_user` (`user_oid`)
)    ENGINE=InnoDB    AUTO_INCREMENT=236    DEFAULT    CHARSET=utf8
ROW_FORMAT=COMPACT;
```

```
#
# Source for table group
#

CREATE TABLE `group` (
  `oid` int(11) NOT NULL,
  `groupname` varchar(255) DEFAULT NULL,
  `module_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_group_module` (`module_oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table group_module
#

CREATE TABLE `group_module` (
  `group_oid` int(11) NOT NULL,
  `module_oid` int(11) NOT NULL,
  PRIMARY KEY (`group_oid`,`module_oid`),
  KEY `fk_group_module_group` (`group_oid`),
  KEY `fk_group_module_module` (`module_oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table household
#

CREATE TABLE `household` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `household_size` decimal(19,2) DEFAULT NULL,
  `ownership` bit(1) DEFAULT NULL,
  `number_pets` int(11) DEFAULT NULL,
  `household_garden_area` decimal(19,2) DEFAULT NULL,
  `household_pool_volume` decimal(19,2) DEFAULT NULL,
  `second` bit(1) DEFAULT NULL,
  `public` bit(1) DEFAULT NULL,
  `visible` bit(1) DEFAULT NULL,
  `household_pool` bit(1) DEFAULT NULL,
  `household_garden` bit(1) DEFAULT NULL,
  `family_id` varchar(255) DEFAULT NULL,
  `smart_meter_oid` int(11) DEFAULT NULL,
  `building_oid` int(11) DEFAULT NULL,
  `children9` int(11) DEFAULT NULL,
  `children5_9` int(11) DEFAULT NULL,
  `children0_4` int(11) DEFAULT NULL,
```

```
  `residency_type` varchar(255) DEFAULT NULL,
  `number_bathrooms` varchar(255) DEFAULT NULL,
  `number_adults` int(11) DEFAULT NULL,
  `environmental_attitude` varchar(255) DEFAULT NULL,
  `irrigation_system` bit(1) DEFAULT NULL,
  `house_plants` bit(1) DEFAULT NULL,
  `balcony_plants` bit(1) DEFAULT NULL,
  `balcony_irrigation` bit(1) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_household_smart_meter` (`smart_meter_oid`),
  KEY `fk_household_building` (`building_oid`)
) ENGINE=InnoDB AUTO_INCREMENT=1616 DEFAULT CHARSET=utf8;

#
# Source for table household_consumption
#

CREATE TABLE `household_consumption` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `consumption` decimal(19,3) DEFAULT NULL,
  `start_date` datetime DEFAULT NULL,
  `end_date` datetime DEFAULT NULL,
  `household_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_household_consumption_house` (`household_oid`)
) ENGINE=InnoDB AUTO_INCREMENT=58078 DEFAULT CHARSET=utf8;

#
# Source for table mail
#

CREATE TABLE `mail` (
  `oid` int(11) NOT NULL,
  `description` varchar(255) DEFAULT NULL,
  `subject` varchar(255) DEFAULT NULL,
  `body` longtext,
  `language` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table media_asset
#

CREATE TABLE `media_asset` (
  `oid` int(11) NOT NULL,
  `title` varchar(255) DEFAULT NULL,
```

```
  `description` varchar(255) DEFAULT NULL,
  `duration` decimal(19,2) DEFAULT NULL,
  `author` varchar(255) DEFAULT NULL,
  `media` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table meter_reading
#

CREATE TABLE `meter_reading` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `reading_date_time` datetime DEFAULT NULL,
  `company` varchar(255) DEFAULT NULL,
  `total_consumption` decimal(19,3) DEFAULT NULL,
  `smart_meter_oid` int(11) DEFAULT NULL,
  `total_consumption_adjusted` decimal(19,3) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  UNIQUE                    KEY                    `unique_reading`
(`reading_date_time`,`smart_meter_oid`),
  KEY `fk_meter_reading_smart_meter` (`smart_meter_oid`)
) ENGINE=InnoDB AUTO_INCREMENT=509495 DEFAULT CHARSET=utf8;

#
# Source for table meter_reading_temp
#

CREATE TABLE `meter_reading_temp` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `reading_date_time` datetime DEFAULT NULL,
  `total_consumption` decimal(19,3) DEFAULT NULL,
  `smart_meter_oid` int(11) DEFAULT NULL,
  `deviation` decimal(10,3) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table module
#

CREATE TABLE `module` (
  `oid` int(11) NOT NULL,
  `moduleid` varchar(255) DEFAULT NULL,
  `moduledomainname` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
#
# Source for table neutral_user
#

CREATE TABLE `neutral_user` (
  `user_oid` varchar(255) NOT NULL,
  `registration_date` date DEFAULT NULL,
  `family_role` varchar(255) DEFAULT NULL,
  `house_holder` bit(1) DEFAULT NULL,
  `educational_level` varchar(255) DEFAULT NULL,
  `income_rate` varchar(255) DEFAULT NULL,
  `currency` varchar(255) DEFAULT NULL,
  `public` bit(1) DEFAULT NULL,
  `language` varchar(255) DEFAULT NULL,
  `temperature_unit` varchar(255) DEFAULT NULL,
  `length_unit` varchar(255) DEFAULT NULL,
  `household_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`user_oid`),
  KEY `fk_neutral_user_household` (`household_oid`),
  KEY `fk_neutral_user_user` (`user_oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table neutral_user_media_asset
#

CREATE TABLE `neutral_user_media_asset` (
  `neutral_user_oid` varchar(255) NOT NULL,
  `media_asset_oid` int(11) NOT NULL,
  PRIMARY KEY (`neutral_user_oid`,`media_asset_oid`),
  KEY `fk_neutral_user_mediaasset_neu` (`neutral_user_oid`),
  KEY `fk_neutral_user_mediaasset_med` (`media_asset_oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table neutral_user_tip
#

CREATE TABLE `neutral_user_tip` (
  `neutral_user_oid` varchar(255) NOT NULL,
  `tip_oid` int(11) NOT NULL,
  PRIMARY KEY (`neutral_user_oid`,`tip_oid`),
  KEY `fk_neutral_user_tip_neutral_us` (`neutral_user_oid`),
  KEY `fk_neutral_user_tip_tip` (`tip_oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
#
# Source for table simple_device_instance
#

CREATE TABLE `simple_device_instance` (
  `oid` int(11) NOT NULL,
  `number` int(11) DEFAULT NULL,
  `device_type_oid` int(11) DEFAULT NULL,
  `household_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_simple_device_instance_devi` (`device_type_oid`),
  KEY `fk_simple_device_instance_hous` (`household_oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table smart_meter
#

CREATE TABLE `smart_meter` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `smart_meter_id` varchar(255) DEFAULT NULL,
  `building_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_smart_meter_building` (`building_oid`)
) ENGINE=InnoDB AUTO_INCREMENT=1604 DEFAULT CHARSET=utf8;

#
# Source for table sso_token
#

CREATE TABLE `sso_token` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `token` varchar(255) DEFAULT NULL,
  `username` varchar(255) DEFAULT NULL,
  `date` timestamp NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
  `ttl` varchar(255) DEFAULT NULL,
  `state` varchar(1) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table tip
#

CREATE TABLE `tip` (
  `oid` int(11) NOT NULL,
```

```
  `name` varchar(255) DEFAULT NULL,
  `header` varchar(255) DEFAULT NULL,
  `body` longtext,
  `tipdate` date DEFAULT NULL,
  `language` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table unit_of_measurement
#

CREATE TABLE `unit_of_measurement` (
  `oid` int(11) NOT NULL,
  `physical_quantity` varchar(255) DEFAULT NULL,
  `primary_unit` varchar(255) DEFAULT NULL,
  `secondary_unit` varchar(255) DEFAULT NULL,
  `conversion_coefficient` decimal(19,2) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table user
#

CREATE TABLE `user` (
  `oid` varchar(255) NOT NULL,
  `username` varchar(255) DEFAULT NULL,
  `password` varchar(255) DEFAULT NULL,
  `email` varchar(255) DEFAULT NULL,
  `first_name` varchar(255) DEFAULT NULL,
  `last_name` varchar(255) DEFAULT NULL,
  `birth_date` varchar(255) DEFAULT NULL,
  `internal` bit(1) DEFAULT NULL,
  `group_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_user_group` (`group_oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table user_csv
#

CREATE TABLE `user_csv` (
  `user_oid` varchar(255) NOT NULL DEFAULT '',
  PRIMARY KEY (`user_oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
#
# Source for table user_group
#

CREATE TABLE `user_group` (
  `user_oid` varchar(255) NOT NULL,
  `group_oid` int(11) NOT NULL,
  PRIMARY KEY (`user_oid`,`group_oid`),
  KEY `fk_user_group_user` (`user_oid`),
  KEY `fk_user_group_group` (`group_oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table weather_condition
#

CREATE TABLE `weather_condition` (
  `oid` int(11) NOT NULL,
  `start_date` date DEFAULT NULL,
  `end_date` date DEFAULT NULL,
  `rain_fall` decimal(19,2) DEFAULT NULL,
  `average_temperature` decimal(19,2) DEFAULT NULL,
  `district_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_weather_condition_district` (`district_oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for procedure adjustMeterReadings
#

CREATE DEFINER=`superadmin`@`%` PROCEDURE `adjustMeterReadings`()
BEGIN

 DECLARE c_finished INTEGER DEFAULT 0;
 DECLARE c_finished_mr INTEGER DEFAULT 0;
 DECLARE c_finished_amr INTEGER DEFAULT 0;
 DECLARE v_smart_meter_oid INTEGER DEFAULT 0;

 DECLARE mr_smart_meter_oid INTEGER DEFAULT 0;
 DECLARE mr_reading_date_time DATETIME;
 DECLARE mr_total_consumption DECIMAL(19,3);

 DECLARE amr_smart_meter_oid INTEGER DEFAULT 0;
 DECLARE amr_start_date_time DATETIME;
 DECLARE amr_end_date_time DATETIME;
```

```
 DECLARE amr_deviation DECIMAL(19,3);

 DECLARE old_smart_meter_oid INTEGER;
 DECLARE old_reading_date_time DATETIME;
 DECLARE old_total_consumption DECIMAL(19,3);


 -- filtering out of range consumption and saving into
meter_reading_temp table

 -- declare cursor for smart meter oid
 DECLARE smart_meter_oid_cursor CURSOR FOR
         SELECT oid FROM smart_meter;
 DECLARE CONTINUE HANDLER FOR SQLSTATE '02000' SET c_finished = 1;

 TRUNCATE meter_reading_temp;
 SET old_total_consumption :=0;

 OPEN smart_meter_oid_cursor;

 get_smart_meter_id_loop: LOOP
 FETCH NEXT FROM smart_meter_oid_cursor INTO v_smart_meter_oid;

 IF c_finished = 1 THEN
     CLOSE smart_meter_oid_cursor;
     LEAVE get_smart_meter_id_loop;
 END IF;


     BEGIN
         -- declare cursor for meter_reading
         DECLARE meter_reading_cursor CURSOR FOR
         SELECT          smart_meter_oid,          reading_date_time,
total_consumption   FROM   meter_reading   WHERE   smart_meter_oid   =
v_smart_meter_oid ORDER BY reading_date_time;
         DECLARE   CONTINUE   HANDLER   FOR     SQLSTATE   '02000'   SET
c_finished_mr = 1;

         OPEN meter_reading_cursor;

         get_meter_reading_loop: LOOP
             FETCH     NEXT     FROM     meter_reading_cursor     INTO
mr_smart_meter_oid, mr_reading_date_time, mr_total_consumption;

              IF c_finished_mr = 1 THEN
              SET c_finished_mr = 0;
                 CLOSE meter_reading_cursor;
                 LEAVE get_meter_reading_loop;
```

```
            END IF;


            IF old_total_consumption != 0  THEN
               IF (mr_total_consumption / old_total_consumption) >=
9 THEN
                  INSERT   INTO   meter_reading_temp   VALUES(null,
mr_reading_date_time,    mr_total_consumption,    mr_smart_meter_oid,
mr_total_consumption / old_total_consumption);
               END IF;
            END IF;

            IF old_total_consumption != 0  THEN
               IF (old_total_consumption / mr_total_consumption) >=
9 THEN
                  INSERT   INTO   meter_reading_temp   VALUES(null,
old_reading_date_time,  old_total_consumption,  old_smart_meter_oid,
old_total_consumption / mr_total_consumption);
               END IF;
            END IF;

            SET old_smart_meter_oid = mr_smart_meter_oid;
            SET old_reading_date_time = mr_reading_date_time;
            SET old_total_consumption = mr_total_consumption;

         END LOOP get_meter_reading_loop;

         SET old_total_consumption =0;
   END;

 END LOOP get_smart_meter_id_loop;

 --       adjusting       meter_reading.total_consumption       into
total_consumption_adjusted

 UPDATE meter_reading
 SET total_consumption_adjusted = total_consumption;

 BEGIN
     -- declare cursor for adjusting smart meter
     DECLARE adjusting_meter_reading_cursor CURSOR FOR
            SELECT      smart_meter_oid,      MIN(reading_date_time)
start_date_time,        MAX(reading_date_time)        end_date_time,
MIN(deviation)
            FROM meter_reading_temp
            GROUP BY smart_meter_oid;
     DECLARE   CONTINUE   HANDLER   FOR   SQLSTATE   '02000'   SET
c_finished_amr = 1;
```

```
        OPEN adjusting_meter_reading_cursor;

        get_adjusting_meter_reading_loop: LOOP
            FETCH NEXT FROM adjusting_meter_reading_cursor INTO
amr_smart_meter_oid,     amr_start_date_time,     amr_end_date_time,
amr_deviation;

             IF c_finished_amr = 1 THEN
             SET c_finished_amr = 0;
                CLOSE adjusting_meter_reading_cursor;
                LEAVE get_adjusting_meter_reading_loop;
            END IF;

        UPDATE meter_reading
        SET total_consumption_adjusted =
        (CASE WHEN amr_deviation > 800 THEN total_consumption/1000
        ELSE    CASE    WHEN    amr_deviation    >    90    THEN
total_consumption/100
        ELSE CASE WHEN amr_deviation > 9 THEN total_consumption/10
ELSE total_consumption END END END)
        WHERE    smart_meter_oid    =    amr_smart_meter_oid    and
reading_date_time  >=  amr_start_date_time  AND  reading_date_time  <=
amr_end_date_time;

        END LOOP get_adjusting_meter_reading_loop;
 END;

END;



#
# Source for procedure adjustMeterReadingsWeekly
#

CREATE               DEFINER=`superadmin`@`%`               PROCEDURE
`adjustMeterReadingsWeekly`()
BEGIN

 DECLARE c_finished INTEGER DEFAULT 0;
 DECLARE c_finished_mr INTEGER DEFAULT 0;
 DECLARE c_finished_amr INTEGER DEFAULT 0;
 DECLARE v_smart_meter_oid INTEGER DEFAULT 0;

 DECLARE mr_smart_meter_oid INTEGER DEFAULT 0;
 DECLARE mr_reading_date_time DATETIME;
 DECLARE mr_total_consumption DECIMAL(19,3);

 DECLARE amr_smart_meter_oid INTEGER DEFAULT 0;
```

```
 DECLARE amr_start_date_time DATETIME;
 DECLARE amr_end_date_time DATETIME;
 DECLARE amr_deviation DECIMAL(19,3);

 DECLARE old_smart_meter_oid INTEGER;
 DECLARE old_reading_date_time DATETIME;
 DECLARE old_total_consumption DECIMAL(19,3);


 -- filtering out of range consumption and saving into
meter_reading_temp table

 -- declare cursor for smart meter oid
 DECLARE smart_meter_oid_cursor CURSOR FOR
        SELECT oid FROM smart_meter;
 DECLARE CONTINUE HANDLER FOR SQLSTATE '02000' SET c_finished = 1;

 TRUNCATE meter_reading_temp;
 SET old_total_consumption :=0;

 OPEN smart_meter_oid_cursor;

 get_smart_meter_id_loop: LOOP
 FETCH NEXT FROM smart_meter_oid_cursor INTO v_smart_meter_oid;

 IF c_finished = 1 THEN
     CLOSE smart_meter_oid_cursor;
     LEAVE get_smart_meter_id_loop;
 END IF;


     BEGIN
         -- declare cursor for meter_reading_weekly
         DECLARE meter_reading_weekly_cursor CURSOR FOR
         SELECT          smart_meter_oid,          reading_date_time,
total_consumption  FROM meter_reading_weekly  WHERE  smart_meter_oid =
v_smart_meter_oid ORDER BY reading_date_time;
         DECLARE   CONTINUE   HANDLER   FOR     SQLSTATE   '02000'   SET
c_finished_mr = 1;

         OPEN meter_reading_weekly_cursor;

         get_meter_reading_weekly_loop: LOOP
             FETCH   NEXT   FROM   meter_reading_weekly_cursor   INTO
mr_smart_meter_oid, mr_reading_date_time, mr_total_consumption;

              IF c_finished_mr = 1 THEN
              SET c_finished_mr = 0;
```

```
                CLOSE meter_reading_weekly_cursor;
                LEAVE get_meter_reading_weekly_loop;
            END IF;


            IF old_total_consumption != 0  THEN
               IF (mr_total_consumption / old_total_consumption) >=
9 THEN
                  INSERT   INTO   meter_reading_temp   VALUES(null,
mr_reading_date_time,    mr_total_consumption,    mr_smart_meter_oid,
mr_total_consumption / old_total_consumption);
               END IF;
            END IF;

            IF old_total_consumption != 0  THEN
               IF (old_total_consumption / mr_total_consumption) >=
9 THEN
                  INSERT   INTO   meter_reading_temp   VALUES(null,
old_reading_date_time,  old_total_consumption,  old_smart_meter_oid,
old_total_consumption / mr_total_consumption);
               END IF;
            END IF;

            SET old_smart_meter_oid = mr_smart_meter_oid;
            SET old_reading_date_time = mr_reading_date_time;
            SET old_total_consumption = mr_total_consumption;

         END LOOP get_meter_reading_weekly_loop;

         SET old_total_consumption =0;
   END;

 END LOOP get_smart_meter_id_loop;

 --     adjusting     meter_reading_weekly.total_consumption     into
total_consumption_adjusted

 UPDATE meter_reading_weekly
 SET total_consumption_adjusted = total_consumption;

 BEGIN
     -- declare cursor for adjusting smart meter
     DECLARE adjusting_meter_reading_weekly_cursor CURSOR FOR
             SELECT      smart_meter_oid,      MIN(reading_date_time)
start_date_time,        MAX(reading_date_time)        end_date_time,
MIN(deviation)
             FROM meter_reading_temp
             GROUP BY smart_meter_oid;
     DECLARE   CONTINUE   HANDLER   FOR   SQLSTATE   '02000'   SET
```

```
c_finished_amr = 1;

        OPEN adjusting_meter_reading_weekly_cursor;

        get_adjusting_meter_reading_weekly_loop: LOOP
            FETCH NEXT FROM adjusting_meter_reading_weekly_cursor
INTO amr_smart_meter_oid, amr_start_date_time, amr_end_date_time,
amr_deviation;

             IF c_finished_amr = 1 THEN
             SET c_finished_amr = 0;
                CLOSE adjusting_meter_reading_weekly_cursor;
                LEAVE get_adjusting_meter_reading_weekly_loop;
            END IF;

        UPDATE meter_reading_weekly
        SET total_consumption_adjusted =
        (CASE WHEN amr_deviation > 800 THEN total_consumption/1000
        ELSE CASE WHEN amr_deviation > 90 THEN
total_consumption/100
        ELSE CASE WHEN amr_deviation > 9 THEN total_consumption/10
ELSE total_consumption END END END)
        WHERE smart_meter_oid = amr_smart_meter_oid and
reading_date_time >= amr_start_date_time AND reading_date_time <=
amr_end_date_time;

        END LOOP get_adjusting_meter_reading_weekly_loop;
 END;

END;



#
# Source for procedure getGranularity
#

CREATE DEFINER=`superadmin`@`%` PROCEDURE `getGranularity`(`user_id`
varchar(255))
BEGIN

 DECLARE c_finished INTEGER DEFAULT 0;
 DECLARE granularity VARCHAR(255) DEFAULT '';

 DECLARE v_user_oid VARCHAR(255) DEFAULT '';
 DECLARE v_freq INTEGER DEFAULT 0;
 DECLARE v_day_start DATE;
 DECLARE v_day_end DATE;
 DECLARE v_recno INTEGER DEFAULT 0;
```

```
 DECLARE old_v_user_oid VARCHAR(255) DEFAULT '';
 DECLARE old_v_freq INTEGER DEFAULT 0;
 DECLARE old_v_day_start DATE;
 DECLARE old_v_day_end DATE;
 DECLARE old_v_recno INTEGER DEFAULT 0;

 -- declare cursor for granularity
 DECLARE c_granularity_cursor CURSOR FOR


 select oid, min(dif) as frequency, ziua1 as day_start, ziua2 as
day_end, recno
 from
 (select T2.oid, DATEDIFF(T2.ziua2, T1.ziua1) dif, T1.ziua1,
T2.ziua2, T1.recno1 recno from
 (select u.oid, date(m.reading_date_time) ziua1,
count(date(m.reading_date_time)) recno1
            from meter_reading m
            left outer join smart_meter sm on sm.oid =
m.smart_meter_oid
            left outer join household h on sm.oid =
h.smart_meter_oid
            left outer join building b on b.oid = h.building_oid
            left outer join neutral_user nu on nu.household_oid =
h.oid
        left outer join user u on u.oid = nu.user_oid
            where u.oid = user_id
 group by date(m.reading_date_time)
 order by date(m.reading_date_time) desc) T1 JOIN
 (select u.oid, date(m.reading_date_time) ziua2,
count(date(m.reading_date_time)) recno2
            from meter_reading m
            left outer join smart_meter sm on sm.oid =
m.smart_meter_oid
            left outer join household h on sm.oid =
h.smart_meter_oid
            left outer join building b on b.oid = h.building_oid
            left outer join neutral_user nu on nu.household_oid =
h.oid
        left outer join user u on u.oid = nu.user_oid
            where u.oid = user_id
 group by date(m.reading_date_time)
 order by date(m.reading_date_time) desc) T2 ON T1.ziua1 <
T2.ziua2)e
 group by e.ziua2
 order by day_end;

 DECLARE CONTINUE HANDLER FOR SQLSTATE '02000' SET c_finished = 1;
```

```
 DELETE FROM granularity WHERE user_oid = user_id;

 OPEN c_granularity_cursor;

 granularity_loop: LOOP
 FETCH NEXT FROM c_granularity_cursor INTO v_user_oid, v_freq,
v_day_start, v_day_end, v_recno;

 IF c_finished = 1 THEN
     CLOSE c_granularity_cursor;
     LEAVE granularity_loop;
 END IF;

 IF old_v_freq != v_freq THEN

   IF v_freq = 1 THEN
      IF v_recno > 1 THEN
         SET granularity = 'hourly';
      ELSE
         SET granularity = 'daily';
      END IF;
   ELSE
          IF v_freq >= 7 THEN
            SET granularity = 'weekly';
          ELSE
                 IF v_freq >= 30 THEN
                   SET granularity = 'monthly';
                 END IF;
          END IF;
   END IF;

   INSERT INTO granularity(granularity, start_date, end_date,
user_oid) VALUES(granularity, v_day_start, v_day_end, v_user_oid);
 END IF;

 SET old_v_user_oid = v_user_oid;
 SET old_v_freq = v_freq;
 SET old_v_day_start = v_day_start;
 SET old_v_day_end = v_day_end;
 SET old_v_recno = v_recno;

 END LOOP granularity_loop;

END;


#
```

```
# Source for function RowNum
#

CREATE DEFINER=`superadmin`@`%` FUNCTION `RowNum`() RETURNS int(11)
BEGIN
     SET @current_row = 0;
     SET @current_row := @current_row + 1;

  RETURN @current_row;
END;



#
#  Foreign keys for table alert
#

ALTER TABLE `alert`
ADD CONSTRAINT `fk_alert_mail` FOREIGN KEY (`mail_oid`) REFERENCES
`mail` (`oid`),
ADD     CONSTRAINT     `fk_alert_neutral_user`     FOREIGN     KEY
(`neutral_user_oid`) REFERENCES `neutral_user` (`user_oid`);

#
#  Foreign keys for table bill
#

ALTER TABLE `bill`
ADD CONSTRAINT `fk_bill_household` FOREIGN KEY (`household_oid`)
REFERENCES `household` (`oid`);

#
#  Foreign keys for table billing_price_bill
#

ALTER TABLE `billing_price_bill`
ADD CONSTRAINT `fk_billing_price_bill_bill` FOREIGN KEY (`bill_oid`)
REFERENCES `bill` (`oid`),
ADD   CONSTRAINT   `fk_billing_price_bill_billing`   FOREIGN   KEY
(`billing_price_oid`) REFERENCES `billing_price` (`oid`);

#
#  Foreign keys for table building
#

ALTER TABLE `building`
ADD CONSTRAINT `fk_building_district` FOREIGN KEY (`district_oid`)
REFERENCES `district` (`oid`);
```

```
#
#  Foreign keys for table complex_device_instance
#

ALTER TABLE `complex_device_instance`
ADD   CONSTRAINT   `fk_complex_device_instance_dev`   FOREIGN   KEY
(`device_type_oid`) REFERENCES `device_type` (`oid`),
ADD   CONSTRAINT   `fk_complex_device_instance_hou`   FOREIGN   KEY
(`household_oid`) REFERENCES `household` (`oid`);

#
#  Foreign keys for table consumer_segment_neutral_user
#

ALTER TABLE `consumer_segment_neutral_user`
ADD   CONSTRAINT   `fk_consumer_segment_neutral_2`   FOREIGN   KEY
(`neutral_user_oid`) REFERENCES `neutral_user` (`user_oid`),
ADD   CONSTRAINT   `fk_consumer_segment_neutral_us`   FOREIGN   KEY
(`consumer_segment_oid`) REFERENCES `consumer_segment` (`oid`);

#
#  Foreign keys for table device_consumption
#

ALTER TABLE `device_consumption`
ADD    CONSTRAINT    `fk_device_consumption_complex_device_instance`
FOREIGN     KEY     (`complex_device_instance_oid`)     REFERENCES
`complex_device_instance` (`oid`),
ADD    CONSTRAINT    `fk_device_consumption_simple_device_instance`
FOREIGN     KEY     (`simple_device_instance_oid`)     REFERENCES
`simple_device_instance` (`oid`);

#
#  Foreign keys for table feature
#

ALTER TABLE `feature`
ADD    CONSTRAINT    `fk_feature_consumer_segment`    FOREIGN    KEY
(`consumer_segment_oid`) REFERENCES `consumer_segment` (`oid`);

#
#  Foreign keys for table group
#

ALTER TABLE `group`
ADD   CONSTRAINT   `fk_group_module`   FOREIGN   KEY   (`module_oid`)
REFERENCES `module` (`oid`);

#
```

```
#  Foreign keys for table group_module
#

ALTER TABLE `group_module`
ADD CONSTRAINT `fk_group_module_group` FOREIGN KEY (`group_oid`)
REFERENCES `group` (`oid`),
ADD CONSTRAINT `fk_group_module_module` FOREIGN KEY (`module_oid`)
REFERENCES `module` (`oid`);

#
#  Foreign keys for table household
#

ALTER TABLE `household`
ADD CONSTRAINT `fk_household_building` FOREIGN KEY (`building_oid`)
REFERENCES `building` (`oid`),
ADD CONSTRAINT `fk_household_smart_meter` FOREIGN KEY
(`smart_meter_oid`) REFERENCES `smart_meter` (`oid`);

#
#  Foreign keys for table household_consumption
#

ALTER TABLE `household_consumption`
ADD CONSTRAINT `fk_household_consumption_house` FOREIGN KEY
(`household_oid`) REFERENCES `household` (`oid`);

#
#  Foreign keys for table meter_reading
#

ALTER TABLE `meter_reading`
ADD CONSTRAINT `fk_meter_reading_smart_meter` FOREIGN KEY
(`smart_meter_oid`) REFERENCES `smart_meter` (`oid`);

#
#  Foreign keys for table neutral_user
#

ALTER TABLE `neutral_user`
ADD CONSTRAINT `fk_neutral_user_household` FOREIGN KEY
(`household_oid`) REFERENCES `household` (`oid`),
ADD CONSTRAINT `fk_neutral_user_user` FOREIGN KEY (`user_oid`)
REFERENCES `user` (`oid`);

#
#  Foreign keys for table neutral_user_media_asset
#
```

```
ALTER TABLE `neutral_user_media_asset`
ADD   CONSTRAINT   `fk_neutral_user_mediaasset_med`   FOREIGN   KEY
(`media_asset_oid`) REFERENCES `media_asset` (`oid`),
ADD   CONSTRAINT   `fk_neutral_user_mediaasset_neu`   FOREIGN   KEY
(`neutral_user_oid`) REFERENCES `neutral_user` (`user_oid`);

#
#  Foreign keys for table neutral_user_tip
#

ALTER TABLE `neutral_user_tip`
ADD   CONSTRAINT   `fk_neutral_user_tip_neutral_us`   FOREIGN   KEY
(`neutral_user_oid`) REFERENCES `neutral_user` (`user_oid`),
ADD  CONSTRAINT  `fk_neutral_user_tip_tip`  FOREIGN  KEY  (`tip_oid`)
REFERENCES `tip` (`oid`);

#
#  Foreign keys for table simple_device_instance
#

ALTER TABLE `simple_device_instance`
ADD   CONSTRAINT   `fk_simple_device_instance_devi`   FOREIGN   KEY
(`device_type_oid`) REFERENCES `device_type` (`oid`),
ADD   CONSTRAINT   `fk_simple_device_instance_hous`   FOREIGN   KEY
(`household_oid`) REFERENCES `household` (`oid`);

#
#  Foreign keys for table smart_meter
#

ALTER TABLE `smart_meter`
ADD     CONSTRAINT     `fk_smart_meter_building`     FOREIGN     KEY
(`building_oid`) REFERENCES `building` (`oid`);

#
#  Foreign keys for table user
#

ALTER TABLE `user`
ADD CONSTRAINT `fk_user_group` FOREIGN KEY (`group_oid`) REFERENCES
`group` (`oid`);

#
#  Foreign keys for table user_group
#

ALTER TABLE `user_group`
```

```
ADD CONSTRAINT `fk_user_group_group` FOREIGN KEY (`group_oid`)
REFERENCES `group` (`oid`),
ADD CONSTRAINT `fk_user_group_user` FOREIGN KEY (`user_oid`)
REFERENCES `user` (`oid`);

#
#  Foreign keys for table weather_condition
#

ALTER TABLE `weather_condition`
ADD CONSTRAINT `fk_weather_condition_district` FOREIGN KEY
(`district_oid`) REFERENCES `district` (`oid`);



/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;
/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
```