PLATFORM IMPLEMENTATION AND INTEGRATION – FINAL PROTOTYPE

SmartH2O

Project FP7-ICT-619172

Deliverable D6.5 WP6

Deliverable
Version 1.0 – 31 March 2017
Document. ref.:
D6.5.SETMOB.WP6.V1.0

Programme Name:	ICT
Project Number:	619172
Project Title:	SmartH2O
Partners:	Coordinator: SUPSI Contractors: POLMI, UoM, SETMOB, EIPCM, TWUL, SES, MOONSUB
Document Number:	smarth2o.D6.5.SETMOB.WP6.V2.0
Work-Package:	WP6
Deliverable Type:	Document
Contractual Date of Delivery:	31 March 2017
Actual Date of Delivery:	31 March 2017
Title of Document:	Platform Implementation and Integration
Author(s):	Luigi Caldararu, Piero Fraternali, Chiara Pasini, Sorin Dumitru, Eduard Gabrian, Catalin Cristea, Sergio Luis Herrera Gonzalez, Giuseppe Pasceri, Fausto Dasseno, Isabel Micheel, Joan Carles Guardiola, Natalia Bakkalian, Marco Bertocchi, Andrea Emilio Rizzoli.
Approval of this report	Submitted to the EC
Summary of this report:	D6.5 Platform Implementation and Integration- Final 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
0.1	03/10/2016	First draft	Luigi Caldararu
0.2	20/10/2016	SETMOB contribution: Code repository URLs alligned	Luigi Caldararu
0.3	15/11/2016	SETMOB contribution: SMDM update	Luigi Caldararu
0.4	17/11/2016	POLIMI contribution: Swiss case study	Chiara Pasini
0.5	07/12/2016	POLIMI contribution: Drop!	Sergio Luis Gonzalez
0.51	17/02/2017	POLIMI contribution: Drop! backend, customer portal	Chiara Pasini
0.6	22/02/2017	SETMOB: Spanish case study	Luigi Caldararu
0.61	03/03/2017	SETMOB: state of development	Luigi Caldararu
0.62	16/03/2017	SETMOB: cloud deployment documented	Luigi Caldararu
0.7	20/03/2017	EIPCM: initial check against requirements	Isabel Micheel
0.8	27/03/2017	Added component Social Network Crawler	Chiara Pasini
0.81	29/03/2017	IFML and Javadoc inserted	Luigi Caldararu
0.82	30/03/2017	SETMOB: Deliverable finalized	Luigi Caldararu
0.9	30/03/2017	Quality check	Piero Fraternali
1.0	31/03/2017	Final revision	Andrea Emilio Rizzoli

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 **D6.5: Platform Implementation and Integration – Final prototype**. According to the Description of Work (DoW), **D6.5** is a software deliverable (the nature of the deliverable is **P = Prototype**) containing the final prototype of the SmartH2O platform.

The SmartH2O platform and its software components have been gradually developed, updated and deployed on the demo sites and on the cloud, permanently during the project lifetime. Here we detail the features included in the Final prototype of the SmartH2O platform in regard to the First and the Second prototypes.

Beyond the features included in the First platform prototype documented in the accompanying document **D6.3 Platform Implementation and Integration-initial prototype** and in the Second prototypes documented in the accompanying document **D6.4 Platform Implementation and Integration-2ndPrototype**, the Final prototype includes the final releases for the demo cases deployed in Swiss case study (Locarno, Switzerland) and in Spanish case study (Valencia, Spain) as well as the SmartH2O platform deployment for cloud.

Overall, the SmartH2O platform has been released in its final state.

The following is the state of the SmartH2O platform components:

- Customer Portal - Basic version: final version released (installed in Swiss case study, Spanish case study, cloud);
- Customer Portal - Advanced (gamified) version which includes the Gamification Engine: final version released (installed in Swiss case study, Spanish case study, cloud);
- Water Utility Admin portal for water utilities: final version released (installed in Swiss case study, Spanish case study, cloud);
- Games Platform: final version released (installed in Swiss case study, Spanish case study, cloud);
- Enterprise Service Bus which includes integration services: final version released (installed in Swiss case study, Spanish case study, cloud);
- Authentication Gateway: final version released (installed in Swiss case study, Spanish case study, cloud);
- Enterprise Service Bus: final version released (installed in Swiss case study, Spanish case study, cloud);
- Social Network Crawler and Data Analyzer: final version released (installed on the development site);
- Models of User Behaviour and Agent Based Modelling: final version released (installed on the development site).

The Final prototype of the SmartH2O platform accomplishes the objectives stated in the DoW:

- The water utilities are able to use the portal to evaluate the impact of policy changes and examine historical data on the performance of the consumers.
- The end users use the portal as an effective complement to the SmartH2O app.

This accompanying document is structured as an installation guide of SmartH2O platform – final prototype, describing specific demo-cases and the cloud distribution. Also, it includes the source code and the documentation for using the platform.

The final prototype of the SmartH2O platform is available in the form of a demo installation on the internet:

The cloud distribution site is available in internet at http://cloud.smarth2o.ro

Testing credentials for Customer portal

1. username: ses.smarth2o
2. password: *smarth2o*

The administration profile of the SmartH2O platform is available in internet at http://cloud.smarth2o.ro/community/admin

Testing credentials for Water Utility admin

3. username: admin
4. password: *admin123admin*

1. SmartH2O platform – Final prototype

1.1 Finalizing the development process

The implementation of the SmartH2O platform conformed to the evolutionary prototyping, by undergoing several iterations following incremental refinements of the functional specification.

In the First prototype we provided the infrastructure to collect and organize the water consumption data of the end users (Smart Meter Data Management component) and the first implementations of the first user behavior models (Customer Portal) connected through a service interface – the Enterprise Service Bus (ESB).

In the Second prototype we provided beyond the features included in the First prototype:

- the implementation of the simulation platform to predict user behaviour under various types of stimuli,
- initial versions of the: Games Platform, Water Utility Admin portal, Social Network Crawler and Data Analyzer, Models of User Behaviour and Agent Based Modelling
- upgrades of the Customer Portal (Basic and Advanced) and ESB integration service

The Final prototype offers to water utilities a full portal to evaluate the impact of policy changes and examine historical data on the performance of the consumers while the Customer portal fully allows to end users an effective complement to the SmartH2O app.

During the development of the platform prototypes, the software has been deployed in a development and testing environment – using a server provided and managed by SETMOB. Water consumption data using smart metering data have been provided by SES, for the Swiss case study, through a FTP connection using an automated nightly job.

The Final SmartH2O platform prototype has been deployed in two production software environments hosted by SES in Locarno, for the Swiss case study respectively by EMIVASA in Valencia, for the Spanish case study.

Also, the Final prototype has been deployed on a cloud infrastructure in order to accommodate testing and production accounts for the next water utilities that will assess and implement the SmartH2O platform.

Prior to every launch or component update into production environment, the software have been tested and 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 was 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 final prototype of the SmartH2O platform. The components' background colors have the following meaning:

- Green – completed (all the platform components have been completed)
- Yellow – in progress
- Red – development not started

The yellow and red states have been kept in the figure in order to ensure legacy to the previous versions of the development state diagram.

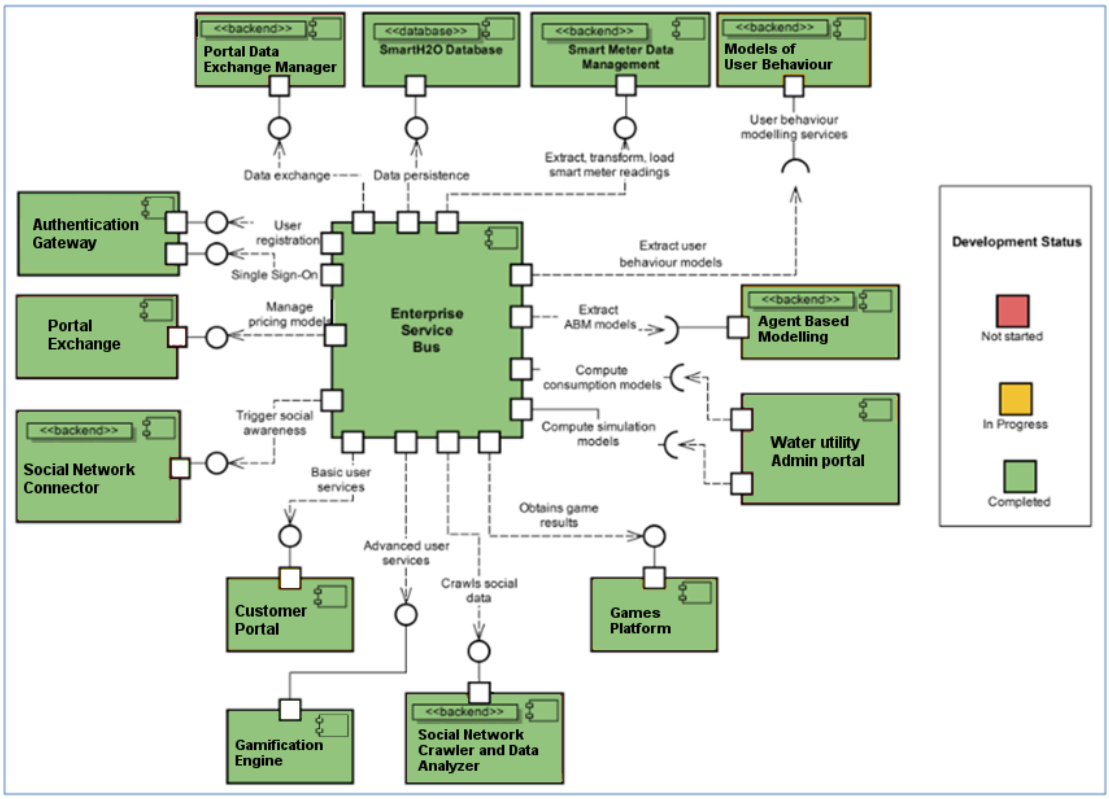

Figure 1: Overview of the main components of the SmartH2O architecture within the final prototype.

With respect to the Initial and the Second prototype, the following components are available in the Final platform prototype:

- **Smart Meter Data Management** component: final version released
- **Customer Portal**: final version released
- Advanced (gamified) version **of the Customer Portal** which includes the **Gamification Engine**: final version released
- **Games Platform**: final version released
- **DROP! online game**: final version released
- **Water Utility Admin portal** for the water utilities: final version released
- **Social Network Crawler and Data Analyzer**: final version released
- **Models of User Behaviour**: final version released
- **Agent Based Modelling**: final version released
  - **Enterprise Service Bus** the centralizing component acting as:
    - Single Point of Access
    - Transaction Manager
    - Security Manager
    - Integration Manager
    - Exchange Service Manager

    final version released

SmartH2O Database: final version released

1.2 SmartH2O platform deployment in the cloud

The process of SmartH2O platform deployment in the cloud had the objective to select a service that automates the creation and management of the platform instances. Different instances of the SmartH2O platform are needed for different water utilities intending to implement the platform with quick availability and without requiring resources from the potential client. Also, while the computation resource needs of a current platform implementation can increase in a specific period of time or just over night, we have to be able to scale them when necessary.

We considered the following objectives for selecting the best suited cloud infrastructure for SmartH2O platform distribution, set-up, tunning and day-to-day administration:

- an application programming interfaces for automatically creating, scaling and destroying an application instance;
- built-in tools for continuously testing and monitoring the reliability and performance of the application instances and the possibility to take adequate countermeasures when needed in order to guarantee the expected level of service;
- ability to control key aspects such as component and user access to resources, automatic resource balancing and replication, security, and accounting;
- a proper costs vs. benefits ratio for allowing qualitative service delivery at the best price;
- worldwide availability;
- compliancy to EU regulations for data storage.

Figure 2 represents Gartner's Magic Quadrant Cloud services as Ability to execution vs. Vision.

Figure 2: Gartner's Magic Quadrant Cloud services as of May 2015.

Following the above objectives and the industry best practices we studied the technical and

economical offers of the following cloud services providers: Google Cloud, Microsoft Azure, Digital Ocean, Rackspace and Amazon Web Services deciding that Amazon Web Services is the best suited cloud provider for the cloud distribution of the SmartH2O platform.

2. Software development kit and installation guide

The SmartH20 development kit contains the software resources deployed as 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.

Unless otherwise specified, the following credentials for accessing SmartH2O project component source code on Bitbucket are used:

User: *smarth2o-guest*

Password: *smarth2oguest*

Based on the resources in the Bitbucket repository, developers can install and extend the platform with their customizations.

2.1 Swiss case study

2.1.1 SmartH2O Database

The **SmartH2O Database** is the central repository of the information common to all the SmartH2O components. It supports the coordination and exchange of messages among the platform components. 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.

SmartH2O platform central and component database runs on MySql 5.5+.

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

A dump file containing testing data corresponding to the final prototype of the platform, is available on Bitbucket at:

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

2.1.2 Smart Meter Data Management Component

The **Smart Meter Data Manager (SMDM)** 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 (aggregated, anonymised) data is stored in the platform database.

It is implemented using Big Data parallel processing. The main benefit of Big Data frameworks is getting 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/

SMDM uses a Big Data processing approach using an Apache Hadoop[1] cluster over HDFS[2]

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

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

(Hadoop Distributed File System) DataNodes, running MapReduce 2.0 (MRv2) for aggregation, PIG[3] (SQL like) scripts for performing logical operations, scheduled by Oozie[4] jobs and loading the data in the database with SQOOP[5] .

For installing the SMDM 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
```

After the setup is completed, Ambari service has to be started with the command:

ambari-server start

Figure 3 is shown how to add new hosts to the Hadoop cluster.

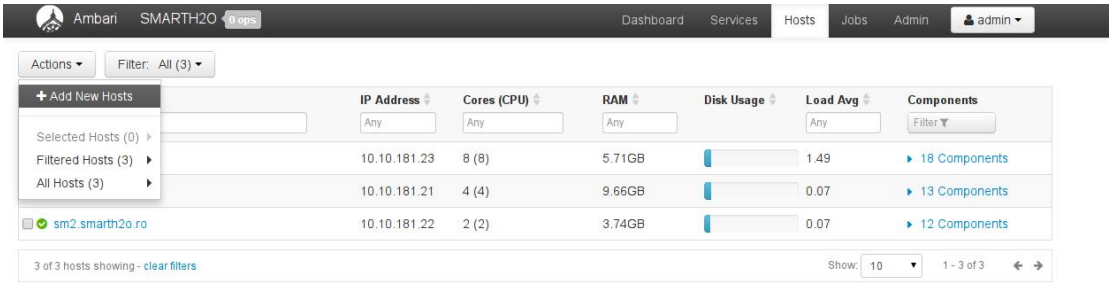

Figure 3: Adding new hosts to Hadoop cluster.

In Figure 4 it is shown how to install a new Ambari component.

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

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

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

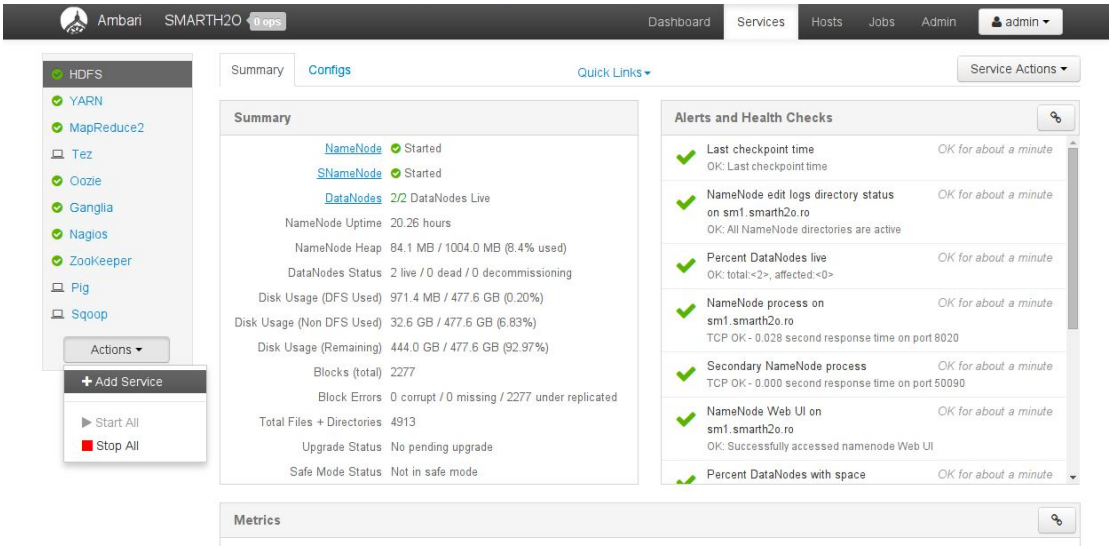

Figure 4: Installing a new Ambari component.

The code source for SMDM manager and transform components are available at:

https://bitbucket.org/smarth2o/sh2osmdmctransform
https://bitbucket.org/smarth2o/sh2osmdmcmanager/

The final 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 **SMDM** workflows corresponding to the final 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 the following command must be executed in the folder where the file pom.xml is located:

```
mvn package
```

A folder named “target” is created containing the application jar file.

An initialisation file must be created to start the SMDM Manager component. A sample file is located in the Bitbucket repository within the source code of the project at https://bitbucket.org/smarth2o/sh2osmdmcmanager/

A FTP server must be configured for receiving consumption data from the water utility. Also an Oozie server must be installed and configured to work with the underlying Apache Hadoop infrastructure using the Apache Ambari platform. The prcessing 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.

Jdom can be downloaded from http://www.jdom.org/downloads then it must be built using

Maven.

MySQL java connector can be downloaded from https://dev.mysql.com/downloads/connector/j/5.1.html.

The configuration of the FTP consumption files path and the access to the Oozie server using SSH protocol is performed in the config.properties file from src/main/resources path before building the package.

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

The workflow.xml and the Pig scripts are located in the **SMDM** 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 SMDM Transform subcomponent source code is available at https://bitbucket.org/smarth2o/sh2osmdmctransform

It must be built using Maven using the command:

```
mvn package
```

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 to:

- Dynamically routing messages to new or updated OSGi bundles, using Apache Camel with OSGI integration. 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.
- 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

SmartH2O ESB includes specific web-services bundles integrated for the usage of the Customer Portal, the Gamification Engine and the Games portal. The bundles were permanently updated during the development process. The final version of the service bundles is available for download at:

https://bitbucket.org/smarth2o/ses-esb-customerportal-smarth2o.git

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 a third party may

need to integrate with the ESB should have a similar component in project, that 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
```

A log entries in both JBoss Fuse console and the Command Prompt will be displayed sgowing messages triggered by 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.

Figure 5 presents the Howtio schemas of deployed services – corresponding to final platform prototype.

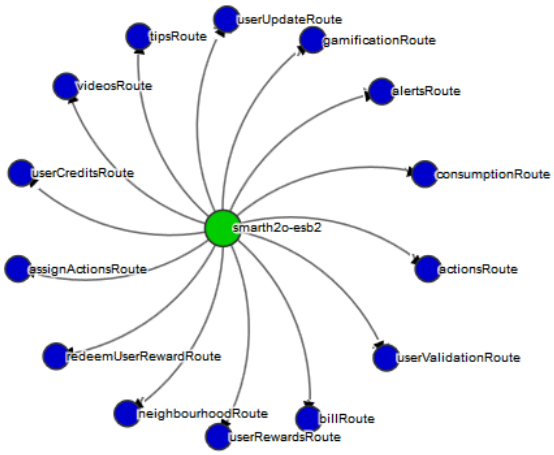

Figure 5: Schema of deployed Fuse services - final platform prototype.

Figure 6 presents the Howtio schemas of deployed services – as the services have been extended to meet the requirements of the final platform prototype.

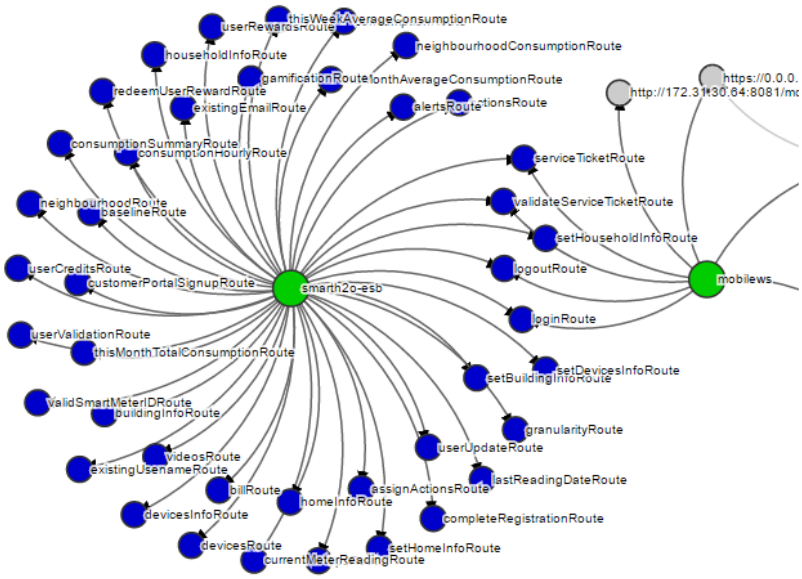

Figure 6: Schema of deployed Fuse services - final platform prototype.

Figure 7 presents the Camel route list.

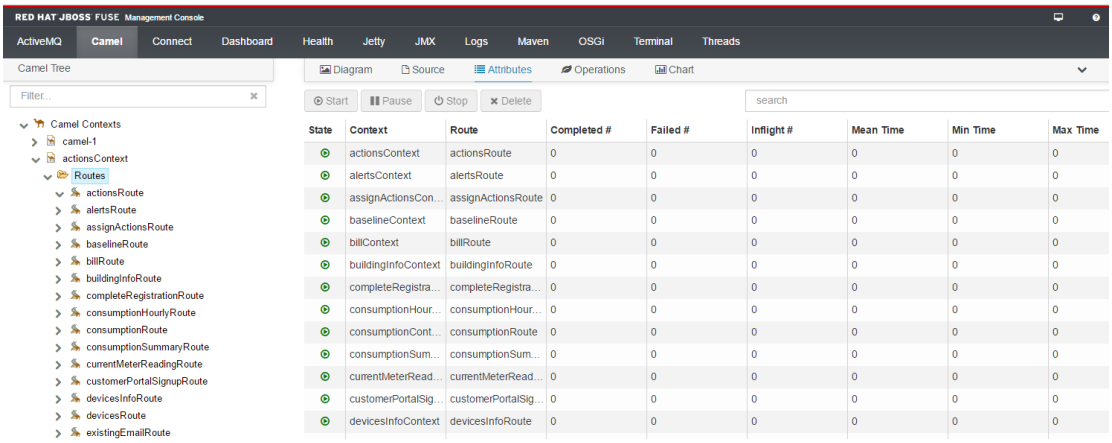

Figure 7: List of Camel routes.

The Apache Camel service endpoints are listed in Figure 8.

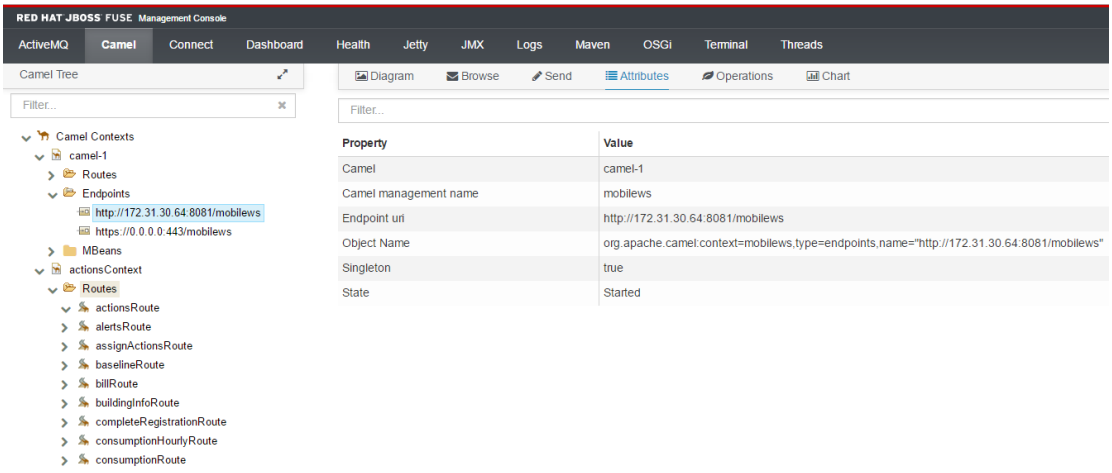

Figure 8: List of Endpoints.

The Enterprise Service Bus instance configured for the SmartH2O cloud distribution server is available online at http://cloud.smarth2o.ro:8084

`Username: smarth2o`

`Password: dsfsmarth2o`

2.1.4 Customer Portal and Gamification Engine

With respect with the previous version of the customer portal and gamification engine described on the D6.4, a set of new features was added in the current version. This are the most relevant ones:

- **Social Sharing**: a social network connector was implemented to enable user to gain points by sharing their water saving achievements and tips on Facebook and twitter.
- **Maps**: an interactive neighborhood map that enable users to compare their achievement one-on-one with their neighbors, to encourage competition within the platform beside the existing leaderboards.
- **Notifications**: this feature enables constant communication and engagement with the user by email notifications related to their activity on the portal. There are 2

periodic notifications that are send to users:

- **Weekly Summary**: email that contains all the information related with the user achievements and points gain during the week, it also includes a water saving tip for the week, the rank on the leaderboards, the weekly winner and the Top 3 leaders.
- **Come Back**: is an email send to those users that haven't login during the last week, inviting them to participate again in the platform.

This version of the Web Project also includes the Water Utility Admin Portal.

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/Release3 and it contains:

- Authentication, BootstrapStyleRarolab, Gamification, GamificationBackEndStyle, GamificationCustomRarolab, GamificationFrontEndStyle, NotificationMessageSample: the webratio source projecs.
- Deployment/lib.zip: external libraries.
- build/community.war: the deployed web application.

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/Release3

2. Application installation on Tomcat

- Unzip the **Deployment/community.war** 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**:
  - 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).
  - Note3: reward_shipment and reward_score_decrease are set on true for the Swiss case study.

```
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
```

- 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

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`

- Admin portal

  `username: admin / password: admin`

2.2 Spanish case study

In the Spanish case study, the SmartH2O platform integration has been performed at web service level by connecting it to EMIVASA Virtual Office – the online customer portal. The virtual office is offered to EMIVASA customers as an online service that provides basic water consumption information and e-billing services. User authentication is uniquely performed by Virtual Office, while a ticketing system allows Single Sign On user access to SmartH2O platform.

Figure 9 displays a screen capture of the EMIVASA Virtual Office featuring integrated SmartH2O service.

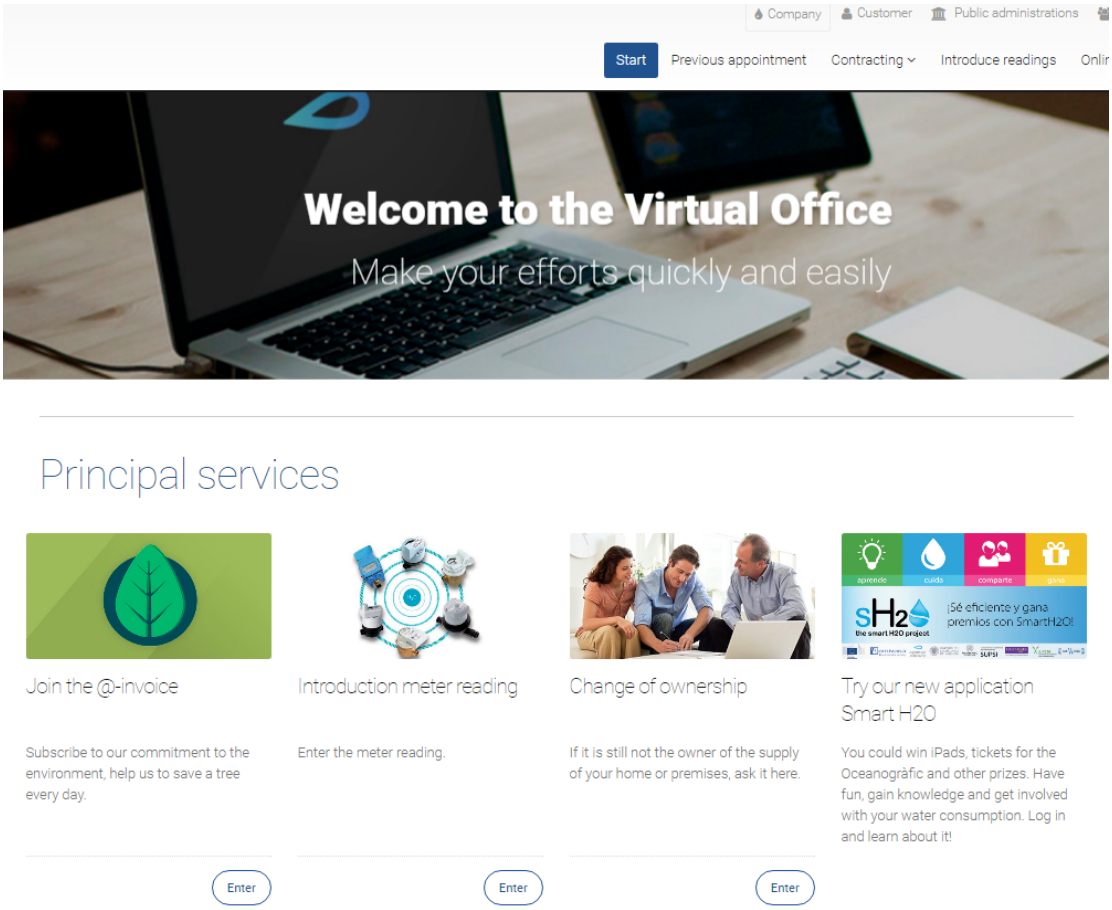

Figure 9: SmartH2O integration to EMIVASA Virtual Office.

2.2.1 SmartH2O Database

In the Spanish case study, not all the data of SmartH2O platform reside in the SmartH2O database. Gamification data are stored in the Gamification 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.

SmartH2O platform central and component database runs on MySql 5.5+.

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

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

2.2.2 Emivasa Smart Meter Data Provisioning component

In the Spanish case study, the 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, whose functionality 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 months or 1 year).

```
JSON structure:
{
  user_id: global_id,
  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 logs 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

With respect with the previous version of the customer portal and gamification engine for the Spanish case study, as described on the D6.4, a set of new features was added in the current version. This are the most relevant ones:

- **Social Sharing**: A social network connector was implemented to enable user to gain points by sharing their water saving achievements and tips on Facebook and twitter.
- **Maps**: An interactive neighborhood map that enable users to compare their achievement one-on-one with their neighbors, to encourage competition within the platform beside the existing leaderboards.
- **Notifications**: This feature enables constant communication and engagement with the user by sending mobile and email notifications related to their activity on the portal. There are 2 types of notification.
  - **Periodic notifications**: Notification send weekly to the users, such as the Summary mail that contains all the information related with the user achievement and point of the week, the rank on the leaderboards, the weekly winner and the Top 3 leaders; Come Back mail is send to those user that haven't login during the last week, inviting them to participate again in the platform; and Tip of the week Notification, is a mobile notification with a water saving hint to keep users engage on water preservation.
  - **Direct Notificatio**n: Notifications send to users based on their activities such as badge achieved, new reward available, goal achieved, etc.

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/Release3

and it contains:

- Authentication, BootstrapStyleRarolab, Gamification, GamificationBackEndStyle,

GamificationCustomRarolab, GamificationFrontEndStyle, NotificationMessageSample: the webratio source projecs
- Deployment/lib.zip: external libraries
- Deployment/community.war: 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/Release3

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).
  - Note3: 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

  username: ses.smart2o

  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 the Webratio IDE using the projects:

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

2.3 Games Platform

With respect with the previous version of Drop! The question application, as described on the

D6.4, changes on the login process and the point assignment were made to allow multi-platform integration, allowing user from the both pilots (Swiss case study and Spanish case study), to register to the app and receive point from the game to their platform accounts.

DropTheQuestion 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/Release3

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.

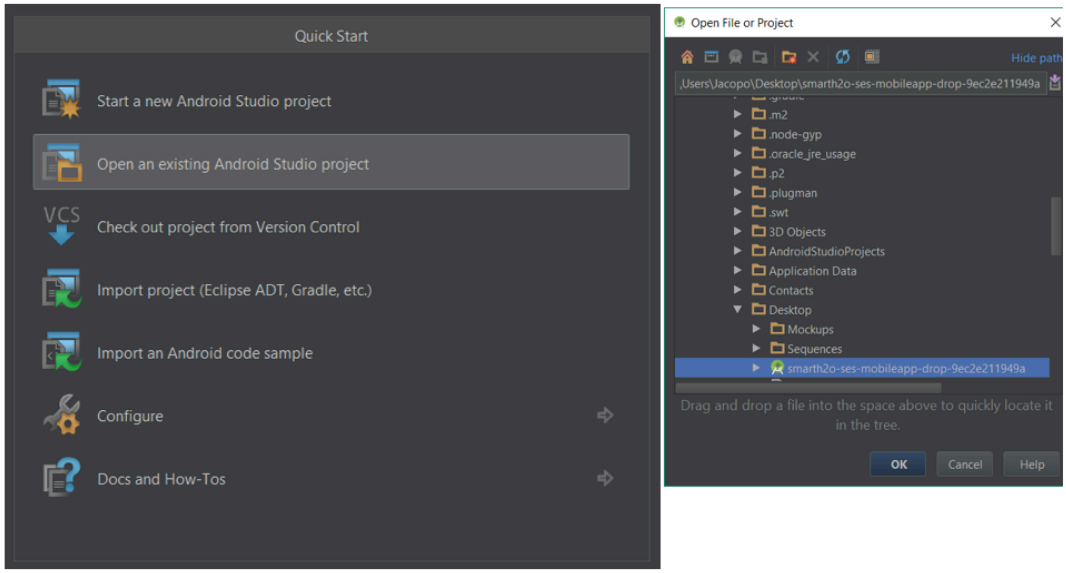

Figure 10: building the mobile application.

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 (Figure 11). 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.

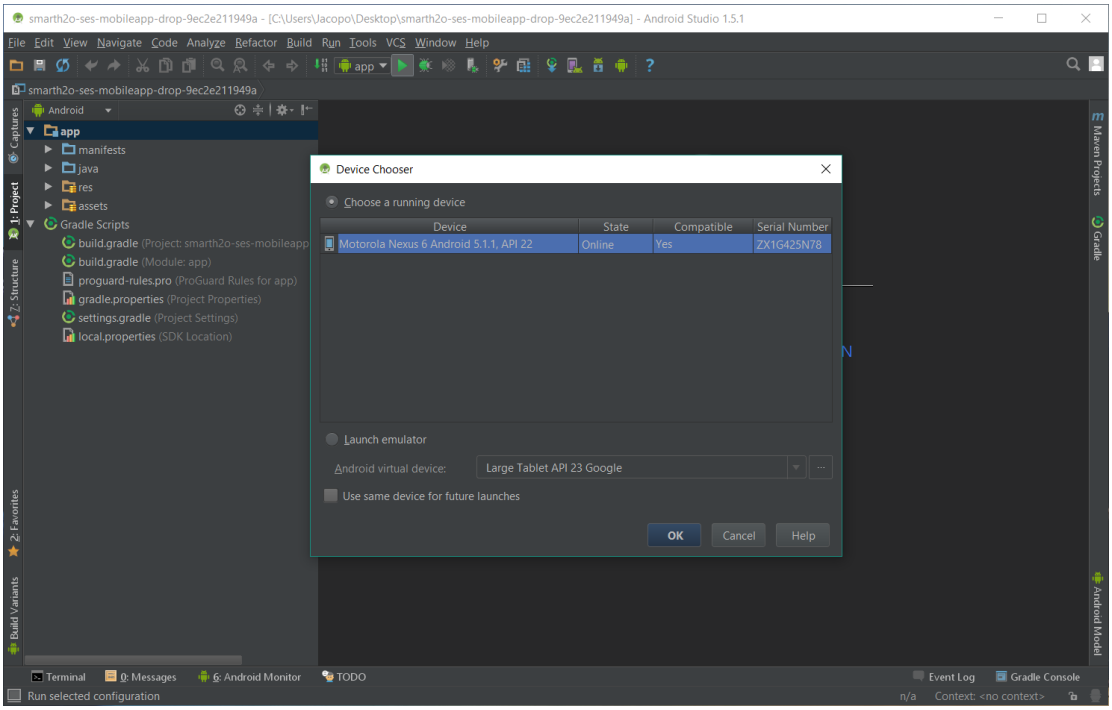

Figure 11: Running the mobile application.

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 enables the user to play the game in conjunction with the Drop! board game by scanning the QR code on specific playing cards.

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/Release3

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/Release3

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
services_host_url_emi=https://www.aguasdevalencia.es:8443 address of the invoked services for Emivasa

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

endpoint_assign_action = ${services_host_url}/community/…/assign….do WebService path
endpoint_assign_action_emi = ${services_host_url}/community/…/assign….do WebService path
```

- 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 of the Drop! Boardgame with the "decode a card" option.

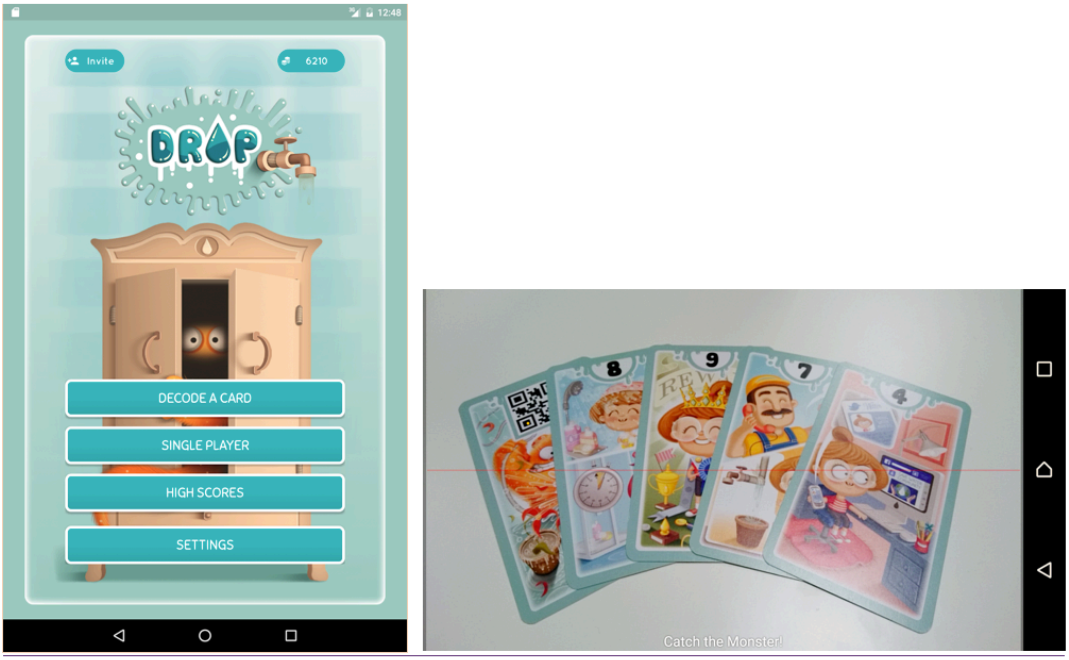

Figure 12: Testing the mobile application Drop!TheQuestion with the Drop! board game.

2.4 Social Network Connector

Social sharing is a new feature added to the platform for the final prototype, it enables user of the platform to get points by inviting other people to join the platform, or by sharing their water saving achievements, status and tips through facebook, twitter and mail.

In order to enable this feature, several components need to be set up, this component include an application internally develop to display twitter cards and facebook stories; a facebook and a twitter application to manage the user actions and award points accordingly; and an imgur application to create and store the images that are share through the social channel. In the next sections the configuration of each component will be explain in detail.

2.4.1 Facebook Application Configuration

Access the facebook developer web site (Facebook Developer) and register with a facebook account, once registered proceed to the “My Apps” menu, and select the “Create a new App” option.

Select a display name for the application and choose a category, click on the “Create App ID” to proceed. You will be redirect to the Application Dashboard where the application ID is display along with the application secret password (Figure 13), this information will be required to connect with the SmartH2O platform.

On Application settings option the following information should be provided:

- Application Domain: Domain of the server where the SmartH2O platform is installed
- Privacy Policy URL: A web page describing all the privacy policies about data protection and data use of the platform.

- Terms of Service URL: A web page describing the obligation and responsibilities of the platform to the users.
- App Icon: The application logo or icon image, in a 1024x1024 resolution.

After providing the information select the “Add platform” option and select “Web Site”, in the site URL field insert the main platform URL, in example: Figure 13 , where <server> indicate the name of the server where the platform is deployed. The Figure 14 presents a complete example of the information that should be provided in this section.

Finally, go to the “App Review” menu and toggle the “Make application public?” option to “Yes” to make it accessible for every one on Facebook.

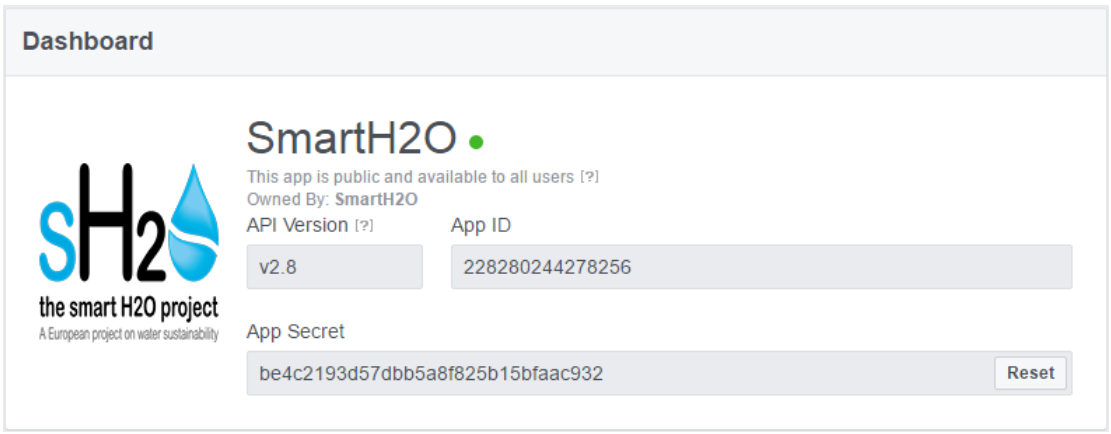

Figure 13: Facebook application ID and Secret word.

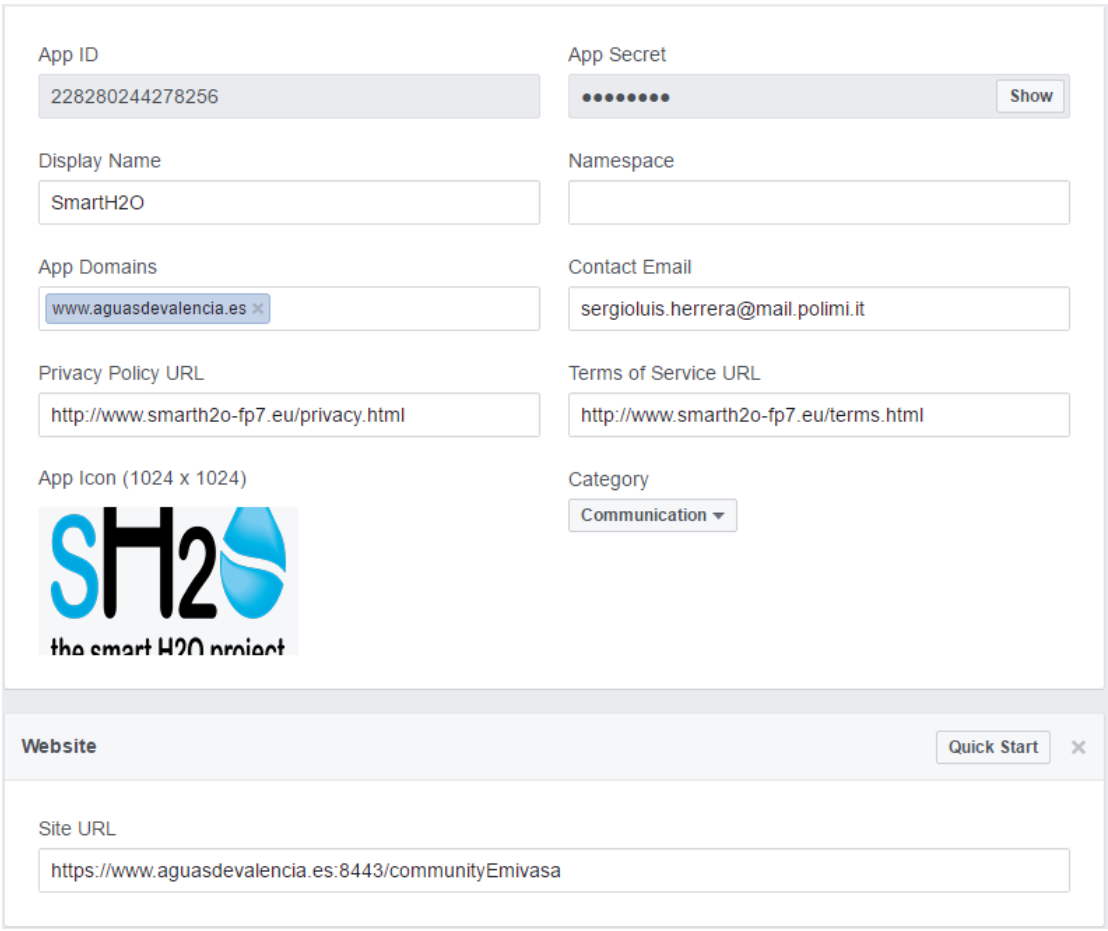

Figure 14:Facebook application configuration.

2.4.2 Twitter Application Configuration

Access the twitter application management web site (Twitter App Management), and register with an existing twitter account. Once on the management page select the “Create New App” option.

On the create application page the following information will be required (Figure 15):

- Application name: The display name of the application, i.e. SmartH2O App.
- Description: A brief description of the application to be display to users while asking permissions for the app.
- Web site: A website publicly accessible, for the users find out more information about the application objectives, this information is also display while asking for user authorization.
- Callback URL: The URL where twitter will respond to the platform request. The values should be the name of the server were smartH2O is installed plus the suffix /callBackUrlFromTwitter, in example: https://<server>/community/callBackUrlFromTwitter
- Developer Agreement: A checkbox asking for acceptance of the development agreement.

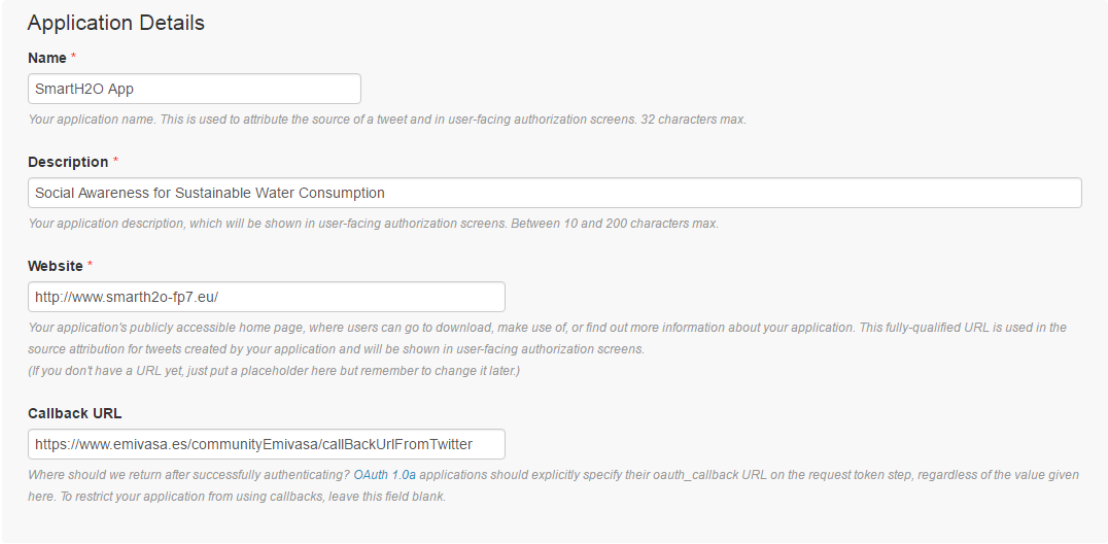

Figure 15: Twitter Application registration.

Proceed to create the twitter application, the details of the app will be display, go to the settings tab to complete the application information, as in the Facebook application, configuration URLs for the privacy policy and terms of service are required, as well as an application icon, all this information in optional.

On the “Keys and Access Tokens” tab the consumer key (API Key) and Consumer Secret (API Secret) (Figure 16), this information is required to connect the SmartH2O platform with the twitter application. The process will be explained in detail in the following sections.

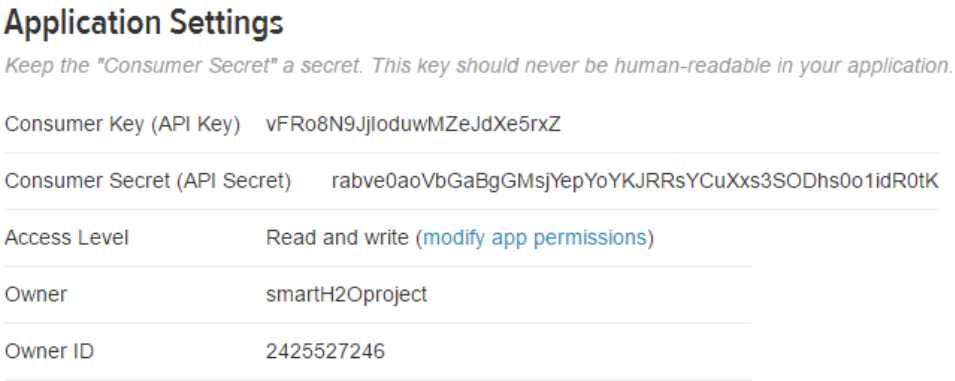

Figure 16: Twitter consumer key and consumer secret word.

2.4.3 Imgur Application Configuration

Access the imgur API web site (Imgur API) with an existing imgur account or create a new account, and “select the register an application” option.

The register application form will require the following information (Figure 17):

- Application Name: Display name for the application on imgur, i.e. SmartH2O.
- Authorization Type: 3 choices of authentication available, select the third option “Anonymous usage without user authorization”, since in this case the application will manage the post and not the users.
- Authorization Callback URL: The callback URL for the authorization, insert the value: “https://imgur.com“, since there will not be authorization process.

- Application Website: A website publicly accessible, for the users find out more information about the application, this field is optional.
- Email: Primary contact for the application.
- Description: A brief description of the application to be display to users while watching at the images through the imgur website.

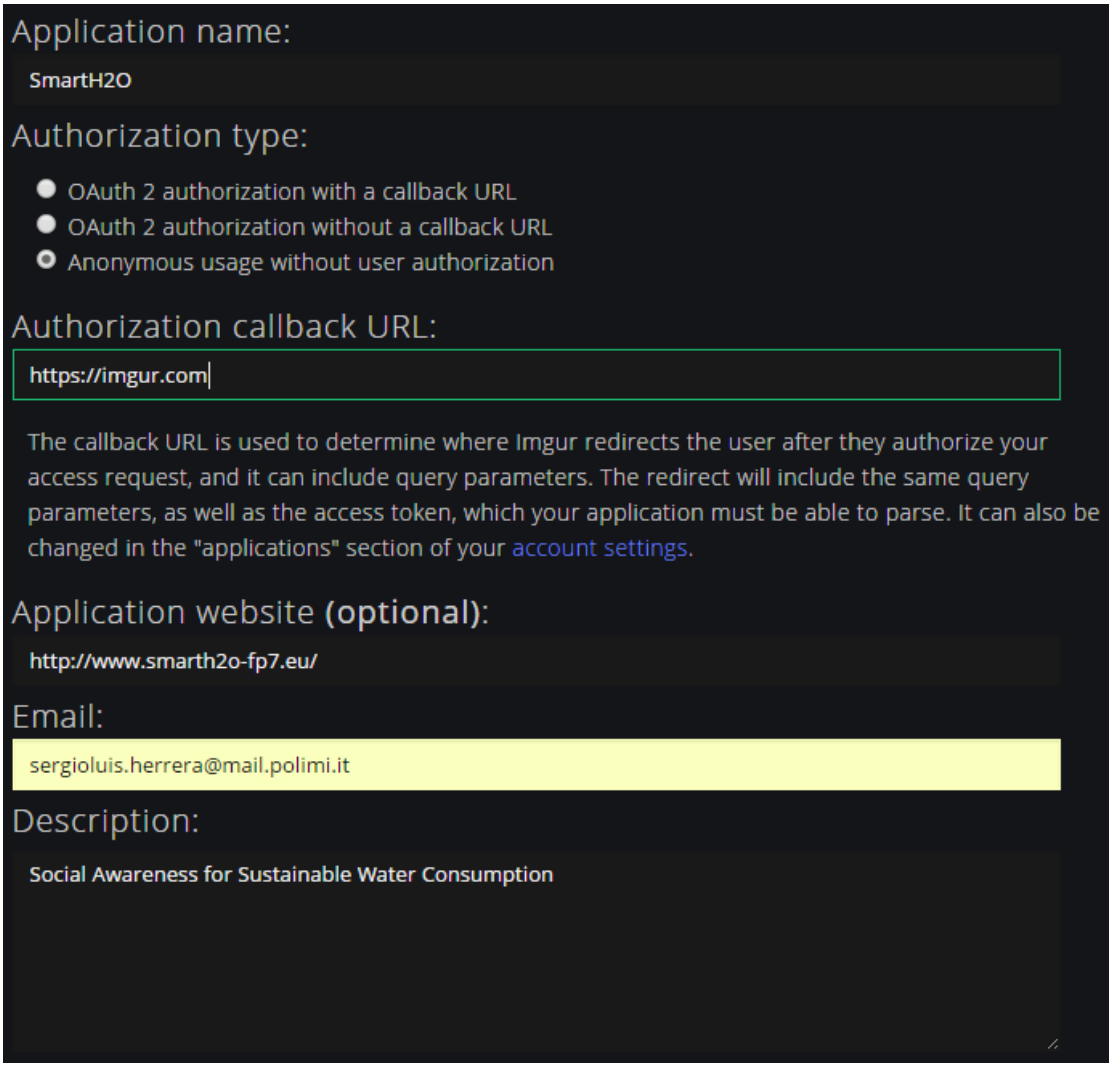

Figure 17: Imgur application registration.

After proceeding with the registration, the client ID and the client secret word will be display (Figure 18), as in previous section this information will be required to connect the smartH2O platform with the external resources.

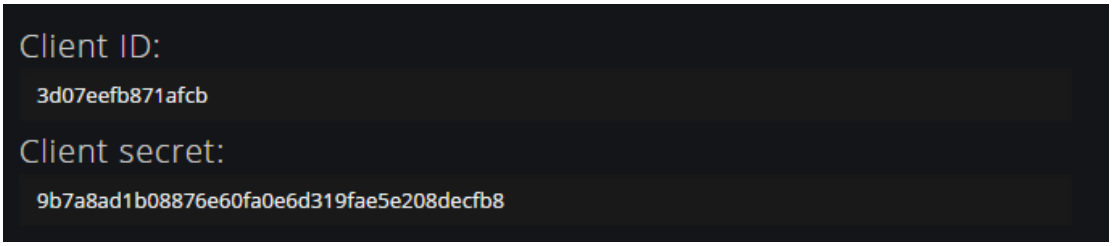

Figure 18: Imgur client ID and client secret word.

2.4.4 Twitter Cards Application

It is a web application that handles the display of the images generated by the SmartH2O application sharing features, including in them meta/tags that enable the visualization of the of twitter cards or facebook stories.

The source code is available on the bitbucket repository:
https://bitbucket.org/smarth2o/common-socialnetworkconnector/commits/tag/Release3

Application Requirements:

- Java 7+: https://home.java.net
- Apache Maven 3.3+: https://maven.apache.org
- Apache Tomcat 6.x or later: http://tomcat.apache.org/
- Eclipse Java EE IDE for Web Developer 4.5 or later: https://www.eclipse.org/

Building and Set up

Open eclipse, on the file menu select Import, on the pop up window select Maven, and the existing maven project. Click Browse and select the folder that is the root of the Maven project downloaded from the repository, make sure the folder is where the pom.xml file is.

Click OK and Under “Projects” click the checkbox next to the pom.xml file and Click Finish.

Go to the location of the pom.xml file and execute:

```
mvn install
```

A war file will be created, extract the file to the tomcat installation folder under “webapps”, at this point you can rename the application folder to any name you prefer, following the naming conventions. Browser to /WEB-INF/classes, on this location there will be several configuration files with name: “messages_XX”, each one corresponding to one language, the suffix of each file indicates the language it corresponds to.

Open the files and set the content as following, for every corresponding language:

country=Provide the name of the country for the deployment, i.e. Italia
twitter_site=The twitter account of the project, i.e. @smarth2oproject
twitter_url=The application web site where users will be redirect.
twitter_title=A title to be display on the page, i.e. Find out if SmartH2O is available in your area and start saving water in a fun way!
twitter_description=A message about the web site for the viewer.

The other properties in the file should not be modified.

Testing

Start tomcat, and on a web browser go to:

http://localhost:8080/common-socialnetworkconnector/

*If you rename the application folder you should provide the same name on the url.

If the application was successfully deployed, the page will display the title and description configure in previous steps, as shown in Figure 19.

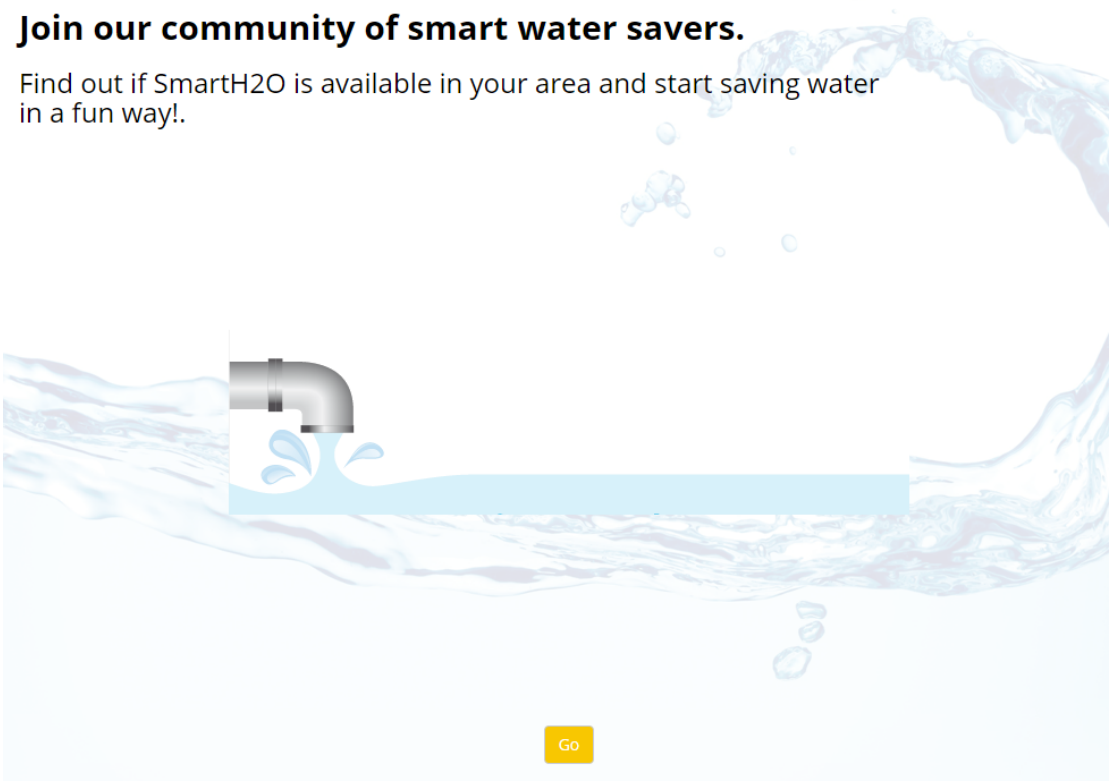

Figure 19: Default Page of the Twitter Cards Application.

2.4.5 SmartH2O Social Connector Configuration

In order to allow SmartH2O platform to communicate with the previously configure applications, some properties need to be added and updated accordingly on the webratio.properties, the following section shows an example of those properties, a comment in green specifies those that need to be updated with information from the previous sections.

```
#Social
#Facebook
social_fb_app_id = …xxx… #Facebook App ID from section 2.5.1
social_tw_app_id = 1
social_host_image = 2
social_images_path = c:/images_wr/
social_images_ext = .png

#Twitter
twitter_consumer_key=…xxx… #Twitter Consumer ID from section 2.5.2
twitter_consumer_secret=…xxx… #Twitter Consumer Secret
twitter_callback_url=http://89.121.250.90:8083/community/callBackUrlFromTwitter #Twitter Callback URL as provided in #section 5.2.2

#Imgur
social_imgur_ID = …xxx… #Imgur Client ID from section 2.5.3
social_imgur_URL = https://api.imgur.com/3/image
```

```
social_img_static_banner = /GamificationCustom/images/social-banner.png


#Twitter Cards
url_app_twitter_cards = http://89.121.250.90:8083/common-socialnetworkconnector #URL where the application for twitter cards (section 2.5.4) is deployed
lang_app_twitter_cards = en #Default language for the Twitter cards application.
```

Once the parameters are updated, start tomcat.

Testing

In order to test the social sharing feature, access the portal with an advance profile user:

- http://localhost:<your_port>/community (customer frontend). The portal can be accessed using the following credentials:
- Advanced profile:

  username: ses2.smarth2o

  password: smarth2o
- Select the social invite option on the top right menu.
- On the pop up window select “Share on Facebook”.
- Provide your facebook credential, if not already login to facebook.
- Click the “Share” button.
- Verify that the post appears on your facebook time lime.

Similar steps can be follow to test the social sharing on twitter.

2.5 Social network crawler and data analyser

The new version of this component allows to scrape twitter to obtain tweets that match a specified query. The crawler is generic and can be adapded to any scenario. It has been used to crawl a list of seeds and the results are specified in “D4.5 - Final social network analysis trust & people search techniques”. The list of seed accounts used for the analysis presented in the D4.5 can be found in the Appendix A of that deliverable.

Requirements:

- NodeJS 4.x or later (link)
- MongoDB (link)

Installation package

The installation package is available here:

https://bitbucket.org/smarth2o-setmob1/common-socialnetworkanalytics-v2-smarth2o/commits/tag/Release3

and it contains the NodeJs project ready to be executed.

Usage

To execute the script, open the command line and run:

```
node cli.js -q "chipasini include:nativeretweets since:2015-09-01 until:2016-01-01”
```

Options

The **query** can be specified inline via the -q argument, or using the -Q flag and specifing the

file containing the query, for more info seeFigure 20.

Flag	Long flag	Description	Required	Default
-Q	--query-file	Twitter query file	**Yes**	*N/D*
		OR		
-q	--query	Twitter query	**Yes**	*N/D*

Figure 20: Query parameter configuration for the Social Network Crawler component.

The mongo options can be specified in a json file or via command line arguments.

If either -M or -d flags are specified then the data will be saved in the corresponding configuration (more info in Figure 21).

Flag	Long flag	Description	Required	Default
-M	--mongo	Mongo configuration file	**Yes**	*N/D*
		OR		
-m	--host	Mongo host	**No**	`localhost`
-p	--port	Mongo port	**No**	`27017`
-d	--database	Mongo database name	**Yes**	*N/D*
-c	--collection	Mongo collection name	**No**	`tweets`

Figure 21: MongoDB parameters configuration for the Social Network Crawler component.

The **Twitter** options can be specified in a json file or via command line arguments.

If either -T or all the other twitter flags (-k -s -t -y) are specified then the data will be enriched with the full tweet info (more info in Figure 22).

Flag	Long flag	Description	Required	Default
-T	--twitter	Twitter config file	**Yes**	N/D
		OR		
-k	--key	Twitter key	**Yes**	N/D
-s	--secret	Twitter secret	**Yes**	N/D
-t	--token	Twitter access token	**Yes**	N/D
-y	--token-secret	Twitter access token secret	**Yes**	N/D

Figure 22: Twitter parameters configuration for the Social Network Crawler component.

2.6 Agent based modelling platform

The agent based model to test the main elements of the incentive model in Valencia, presented in the Deliverable 6.5, 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 23.

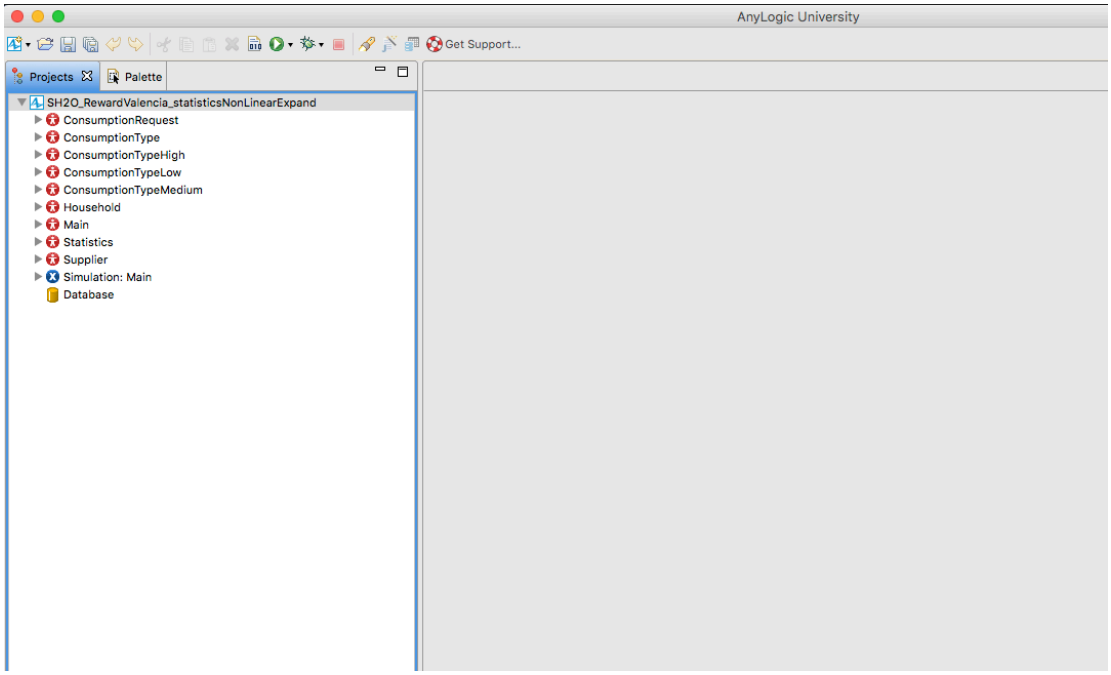

Figure 23: Opening window of the ABM model.

To start the simulation, proceeds as follows: locate the Run button and choose SH2O_RewardValencia_statisticsNonLinearExpand / Simulation from the list, as shown in Figure 24. After you start the model, the presentation window displays the launches experiment Simulation. Click the “Run” button to run the model,

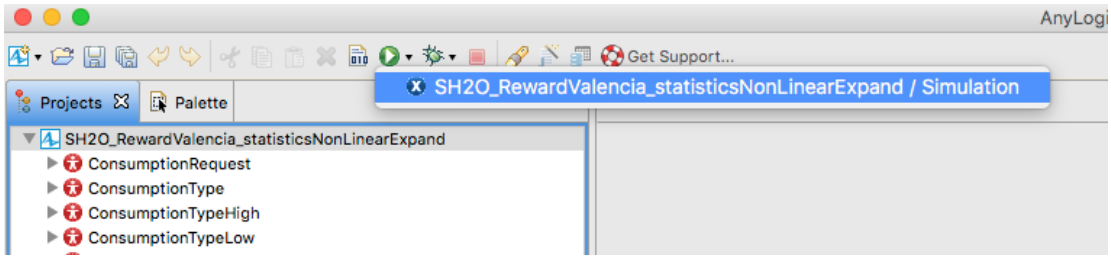

Figure 24: Selection of the Run button in the opening window.

3. Appendix A: SmartH2O platform javadoc and database DDL script – final prototype

3.1 Javadoc

The documentation generated for the Gamification Engine is available at the following (link).
The documentation generated for the integration services is available at the following (link).

3.2 SmartH2O central database script

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

-- Dumping structure for procedure smarth2o.adjustMeterReadings
DELIMITER //
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//
DELIMITER ;

-- Dumping structure for procedure smarth2o.adjustMeterReadingsWeekly
DELIMITER //
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//
DELIMITER ;

-- Dumping structure for table smarth2o.alert
CREATE TABLE IF NOT EXISTS `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`),
  CONSTRAINT `fk_alert_mail` FOREIGN KEY (`mail_oid`) REFERENCES `mail` (`oid`),
  CONSTRAINT `fk_alert_neutral_user` FOREIGN KEY (`neutral_user_oid`) REFERENCES
`neutral_user` (`user_oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for table smarth2o.baseline
CREATE TABLE IF NOT EXISTS `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`),
  CONSTRAINT `fk_baseline_smart_meter` FOREIGN KEY (`smart_meter_oid`)
REFERENCES `smart_meter` (`oid`)
) ENGINE=InnoDB AUTO_INCREMENT=1197 DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for table smarth2o.bill
CREATE TABLE IF NOT EXISTS `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`),
  CONSTRAINT `fk_bill_household` FOREIGN KEY (`household_oid`) REFERENCES `household` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for table smarth2o.billing_price
CREATE TABLE IF NOT EXISTS `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;

-- Data exporting was unselected.
-- Dumping structure for table smarth2o.billing_price_bill
CREATE TABLE IF NOT EXISTS `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`),
  CONSTRAINT `fk_billing_price_bill_bill` FOREIGN KEY (`bill_oid`) REFERENCES `bill` (`oid`),
  CONSTRAINT `fk_billing_price_bill_billing` FOREIGN KEY (`billing_price_oid`) REFERENCES `billing_price` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for table smarth2o.building
CREATE TABLE IF NOT EXISTS `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`),
  CONSTRAINT `fk_building_district` FOREIGN KEY (`district_oid`) REFERENCES `district` (`oid`)
) ENGINE=InnoDB AUTO_INCREMENT=494 DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for table smarth2o.complex_device_instance
CREATE TABLE IF NOT EXISTS `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`),
  CONSTRAINT `fk_complex_device_instance_dev` FOREIGN KEY (`device_type_oid`) REFERENCES `device_type` (`oid`),
  CONSTRAINT `fk_complex_device_instance_hou` FOREIGN KEY (`household_oid`) REFERENCES `household` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for table smarth2o.consumer_segment
CREATE TABLE IF NOT EXISTS `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;

-- Data exporting was unselected.
-- Dumping structure for table smarth2o.consumer_segment_neutral_user
CREATE TABLE IF NOT EXISTS `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`),
  CONSTRAINT `fk_consumer_segment_neutral_2` FOREIGN KEY (`neutral_user_oid`) REFERENCES `neutral_user` (`user_oid`),
  CONSTRAINT `fk_consumer_segment_neutral_us` FOREIGN KEY
```

```
(`consumer_segment_oid`) REFERENCES `consumer_segment` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for table smarth2o.device_consumption
CREATE TABLE IF NOT EXISTS `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`),
  CONSTRAINT `fk_device_consumption_complex_device_instance` FOREIGN KEY
(`complex_device_instance_oid`) REFERENCES `complex_device_instance` (`oid`),
  CONSTRAINT `fk_device_consumption_simple_device_instance` FOREIGN KEY
(`simple_device_instance_oid`) REFERENCES `simple_device_instance` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for table smarth2o.device_type
CREATE TABLE IF NOT EXISTS `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;

-- Data exporting was unselected.
-- Dumping structure for table smarth2o.district
CREATE TABLE IF NOT EXISTS `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;

-- Data exporting was unselected.
-- Dumping structure for table smarth2o.feature
CREATE TABLE IF NOT EXISTS `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`),
  CONSTRAINT `fk_feature_consumer_segment` FOREIGN KEY (`consumer_segment_oid`)
REFERENCES `consumer_segment` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for procedure smarth2o.getGranularity
DELIMITER //
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//
DELIMITER ;

-- Dumping structure for table smarth2o.granularity
CREATE TABLE IF NOT EXISTS `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`),
  CONSTRAINT `fk_granularity_user` FOREIGN KEY (`user_oid`) REFERENCES `user`
(`oid`)
) ENGINE=InnoDB AUTO_INCREMENT=23821 DEFAULT CHARSET=utf8
ROW_FORMAT=COMPACT;

-- Data exporting was unselected.
-- Dumping structure for table smarth2o.group
CREATE TABLE IF NOT EXISTS `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`),
  CONSTRAINT `fk_group_module` FOREIGN KEY (`module_oid`) REFERENCES `module`
(`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for table smarth2o.group_module
```

```
CREATE TABLE IF NOT EXISTS `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`),
  CONSTRAINT `fk_group_module_group` FOREIGN KEY (`group_oid`) REFERENCES `group` (`oid`),
  CONSTRAINT `fk_group_module_module` FOREIGN KEY (`module_oid`) REFERENCES `module` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for table smarth2o.household
CREATE TABLE IF NOT EXISTS `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`),
  CONSTRAINT `fk_household_building` FOREIGN KEY (`building_oid`) REFERENCES `building` (`oid`),
  CONSTRAINT `fk_household_smart_meter` FOREIGN KEY (`smart_meter_oid`)
```

```
REFERENCES `smart_meter` (`oid`)
) ENGINE=InnoDB AUTO_INCREMENT=646 DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for table smarth2o.household_consumption
CREATE TABLE IF NOT EXISTS `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=198742 DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for table smarth2o.mail
CREATE TABLE IF NOT EXISTS `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;

-- Data exporting was unselected.
-- Dumping structure for table smarth2o.media_asset
CREATE TABLE IF NOT EXISTS `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;

-- Data exporting was unselected.
-- Dumping structure for table smarth2o.meter_reading
CREATE TABLE IF NOT EXISTS `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`),
  CONSTRAINT `fk_meter_reading_smart_meter` FOREIGN KEY (`smart_meter_oid`)
REFERENCES `smart_meter` (`oid`)
) ENGINE=InnoDB AUTO_INCREMENT=15282194 DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for table smarth2o.meter_reading_log
CREATE TABLE IF NOT EXISTS `meter_reading_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=417 DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
-- Dumping structure for table smarth2o.meter_reading_smdm
CREATE TABLE IF NOT EXISTS `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=500677 DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for table smarth2o.meter_reading_temp
CREATE TABLE IF NOT EXISTS `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(19,3) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB AUTO_INCREMENT=1312 DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for table smarth2o.meter_reading_weekly
CREATE TABLE IF NOT EXISTS `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;

-- Data exporting was unselected.
-- Dumping structure for table smarth2o.meter_reading_weekly_smdm
CREATE TABLE IF NOT EXISTS `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;

-- Data exporting was unselected.
-- Dumping structure for table smarth2o.module
CREATE TABLE IF NOT EXISTS `module` (
  `oid` int(11) NOT NULL,
  `moduleid` varchar(255) DEFAULT NULL,
  `moduledomainname` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for table smarth2o.neutral_user
CREATE TABLE IF NOT EXISTS `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`),
  CONSTRAINT `fk_neutral_user_household` FOREIGN KEY (`household_oid`) REFERENCES `household` (`oid`),
  CONSTRAINT `fk_neutral_user_user` FOREIGN KEY (`user_oid`) REFERENCES `user` (`oid`)
) ENGINE=InnoDB AUTO_INCREMENT=130 DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for table smarth2o.neutral_user_media_asset
CREATE TABLE IF NOT EXISTS `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;

-- Data exporting was unselected.
-- Dumping structure for table smarth2o.neutral_user_tip
CREATE TABLE IF NOT EXISTS `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;

-- Data exporting was unselected.
-- Dumping structure for table smarth2o.newdata
CREATE TABLE IF NOT EXISTS `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;

-- Data exporting was unselected.
```

```
-- Dumping structure for function smarth2o.RowNum
DELIMITER //
CREATE DEFINER=`superadmin`@`%` FUNCTION `RowNum`() RETURNS int(11)
BEGIN
    SET @current_row = 0;
    SET @current_row := @current_row + 1;

  RETURN @current_row;
END//
DELIMITER ;

-- Dumping structure for table smarth2o.simple_device_instance
CREATE TABLE IF NOT EXISTS `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;

-- Data exporting was unselected.
-- Dumping structure for table smarth2o.smart_meter
CREATE TABLE IF NOT EXISTS `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`),
  CONSTRAINT `fk_smart_meter_building` FOREIGN KEY (`building_oid`) REFERENCES
`building` (`oid`)
) ENGINE=InnoDB AUTO_INCREMENT=379 DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for table smarth2o.smart_meter_new
CREATE TABLE IF NOT EXISTS `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;

-- Data exporting was unselected.
```

```
-- Dumping structure for table smarth2o.sso_token
CREATE TABLE IF NOT EXISTS `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;

-- Data exporting was unselected.
-- Dumping structure for table smarth2o.tip
CREATE TABLE IF NOT EXISTS `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;

-- Data exporting was unselected.
-- Dumping structure for table smarth2o.unit_of_measurement
CREATE TABLE IF NOT EXISTS `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;

-- Data exporting was unselected.
-- Dumping structure for table smarth2o.user
CREATE TABLE IF NOT EXISTS `user` (
  `oid` int(11) 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`),
  CONSTRAINT `fk_user_group` FOREIGN KEY (`group_oid`) REFERENCES `group` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for table smarth2o.user_group
CREATE TABLE IF NOT EXISTS `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;

-- Data exporting was unselected.
-- Dumping structure for table smarth2o.weather_condition
CREATE TABLE IF NOT EXISTS `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;
```

3.3 Gamification engine database script

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

CREATE TABLE IF NOT EXISTS `action_instance` (
  `oid` int(11) NOT NULL,
  `executor` varchar(255) DEFAULT '',
  `date` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `score` decimal(19,2) DEFAULT NULL,
  `description` varchar(255) DEFAULT NULL,
  `tag` varchar(255) DEFAULT NULL,
  `link` varchar(255) DEFAULT NULL,
  `rank_oid` int(11) DEFAULT NULL,
  `action_type_oid` int(11) DEFAULT NULL,
```

```
  `object_key` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `idx_action_instance_rank` (`rank_oid`),
  KEY `idx_action_instance_action_typ` (`action_type_oid`),
  CONSTRAINT `fk_action_instance_action_type` FOREIGN KEY (`action_type_oid`) REFERENCES `action_type` (`oid`),
  CONSTRAINT `fk_action_instance_rank` FOREIGN KEY (`rank_oid`) REFERENCES `community_user` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for view community.action_instance_action_area_vi
-- Creating temporary table to overcome VIEW dependency errors
CREATE TABLE `action_instance_action_area_vi` (
	`oid` INT(11) NOT NULL,
	`der_attr` VARCHAR(255) NULL COLLATE 'utf8_general_ci'
) ENGINE=MyISAM;

-- Dumping structure for view community.action_instance_daily_vi
-- Creating temporary table to overcome VIEW dependency errors
CREATE TABLE `action_instance_daily_vi` (
	`action_type_oid` INT(11) NULL,
	`date` DATE NULL,
	`daily_occurrence` BIGINT(21) NOT NULL
) ENGINE=MyISAM;

-- Dumping structure for view community.action_instance_label
-- Creating temporary table to overcome VIEW dependency errors
CREATE TABLE `action_instance_label` (
	`oid` INT(11) NOT NULL,
	`score` DECIMAL(19,2) NULL,
	`date` TIMESTAMP NOT NULL,
	`name` VARCHAR(255) NULL COLLATE 'latin1_swedish_ci',
	`languagecode` VARCHAR(255) NULL COLLATE 'latin1_swedish_ci',
	`rank_oid` INT(11) NULL
) ENGINE=MyISAM;

-- Dumping structure for view community.action_instance_name_view
-- Creating temporary table to overcome VIEW dependency errors
CREATE TABLE `action_instance_name_view` (
	`oid` INT(11) NOT NULL,
	`der_attr` VARCHAR(255) NULL COLLATE 'utf8_general_ci'
) ENGINE=MyISAM;

-- Dumping structure for table community.action_label
```

```
CREATE TABLE IF NOT EXISTS `action_label` (
  `oid` int(11) NOT NULL,
  `language` varchar(255) DEFAULT NULL,
  `name` varchar(255) DEFAULT NULL,
  `description` varchar(255) DEFAULT NULL,
  `done` varchar(255) DEFAULT NULL,
  `action_type_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_action_label_action_type` (`action_type_oid`),
  CONSTRAINT `fk_action_label_action_type` FOREIGN KEY (`action_type_oid`) REFERENCES `action_type` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
-- Dumping structure for view community.action_labels
-- Creating temporary table to overcome VIEW dependency errors
CREATE TABLE `action_labels` (
	`action_type_oid` INT(11) NULL,
	`languagecode` VARCHAR(255) NULL COLLATE 'latin1_swedish_ci',
	`name` VARCHAR(255) NULL COLLATE 'latin1_swedish_ci',
	`descr` VARCHAR(255) NULL COLLATE 'latin1_swedish_ci'
) ENGINE=MyISAM;

-- Dumping structure for view community.action_labels_area
-- Creating temporary table to overcome VIEW dependency errors
CREATE TABLE `action_labels_area` (
	`url` VARCHAR(255) NULL COLLATE 'utf8_general_ci',
	`score` DECIMAL(19,2) NULL,
	`oid` INT(11) NOT NULL,
	`languagecode` VARCHAR(255) NULL COLLATE 'latin1_swedish_ci',
	`name` VARCHAR(255) NULL COLLATE 'latin1_swedish_ci',
	`descr` VARCHAR(255) NULL COLLATE 'latin1_swedish_ci',
	`action_type_oid` INT(11) NULL,
	`area_name` VARCHAR(255) NULL COLLATE 'utf8_general_ci',
	`hd_checked_image` VARCHAR(255) NULL COLLATE 'utf8_general_ci'
) ENGINE=MyISAM;

-- Dumping structure for view community.action_labels_done
-- Creating temporary table to overcome VIEW dependency errors
CREATE TABLE `action_labels_done` (
	`action_type_oid` INT(11) NULL,
	`languagecode` VARCHAR(255) NULL COLLATE 'latin1_swedish_ci',
	`done` VARCHAR(255) NULL COLLATE 'latin1_swedish_ci'
) ENGINE=MyISAM;
```

```
-- Dumping structure for table community.action_type
CREATE TABLE IF NOT EXISTS `action_type` (
  `oid` int(11) NOT NULL,
  `check_time_elapsed` tinyint(1) DEFAULT NULL,
  `milliseconds_time_elapsed` int(11) DEFAULT NULL,
  `name` varchar(255) DEFAULT NULL,
  `repeatable` tinyint(1) DEFAULT NULL,
  `score` decimal(19,2) DEFAULT NULL,
  `reputation` tinyint(1) DEFAULT NULL,
  `participation` tinyint(1) DEFAULT NULL,
  `area` varchar(255) DEFAULT NULL,
  `description` varchar(255) DEFAULT NULL,
  `gamified_application_oid` int(11) DEFAULT NULL,
  `active` bit(1) DEFAULT NULL,
  `thematic_area_oid` int(11) DEFAULT NULL,
  `reward_type_oid` int(11) DEFAULT NULL,
  `url` varchar(255) DEFAULT NULL,
  `allows_duplicates` bit(1) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `idx_action_type_gamified_appli` (`gamified_application_oid`),
  KEY `fk_action_type_thematic_area` (`thematic_area_oid`),
  KEY `fk_action_type_reward_type` (`reward_type_oid`),
  CONSTRAINT `fk_action_type_gamified_applic` FOREIGN KEY (`gamified_application_oid`) REFERENCES `gamified_application` (`oid`),
  CONSTRAINT `fk_action_type_reward_type` FOREIGN KEY (`reward_type_oid`) REFERENCES `reward_type` (`oid`),
  CONSTRAINT `fk_action_type_thematic_area` FOREIGN KEY (`thematic_area_oid`) REFERENCES `thematic_area` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for table community.area_label
CREATE TABLE IF NOT EXISTS `area_label` (
  `oid` int(11) NOT NULL,
  `language` varchar(255) DEFAULT NULL,
  `name` varchar(255) DEFAULT NULL,
  `description` varchar(255) DEFAULT NULL,
  `thematic_area_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_area_label_thematic_area` (`thematic_area_oid`),
  CONSTRAINT `fk_area_label_thematic_area` FOREIGN KEY (`thematic_area_oid`) REFERENCES `thematic_area` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
-- Dumping structure for view community.badgeimportancebyuser
```

```
-- Creating temporary table to overcome VIEW dependency errors
CREATE TABLE `badgeimportancebyuser` (
	`badge_instance` INT(11) NOT NULL,
	`user` INT(11) NOT NULL,
	`nickname_area` VARCHAR(255) NULL COLLATE 'utf8_general_ci',
	`importance` INT(11) NULL
) ENGINE=MyISAM;

-- Dumping structure for view community.badgetype_sortco
-- Creating temporary table to overcome VIEW dependency errors
CREATE TABLE `badgetype_sortco` (
	`oid` INT(11) NOT NULL,
	`der_attr` INT(1) NULL
) ENGINE=MyISAM;

-- Dumping structure for table community.badge_action
CREATE TABLE IF NOT EXISTS `badge_action` (
  `badge_type_oid` int(11) NOT NULL,
  `action_type_oid` int(11) NOT NULL,
  PRIMARY KEY (`badge_type_oid`,`action_type_oid`),
  KEY `idx_badge_action_badge_type` (`badge_type_oid`),
  KEY `idx_badge_action_action_type` (`action_type_oid`),
  CONSTRAINT `fk_badge_action_action_type` FOREIGN KEY (`action_type_oid`) REFERENCES `action_type` (`oid`),
  CONSTRAINT `fk_badge_action_badge_type` FOREIGN KEY (`badge_type_oid`) REFERENCES `badge_type` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for table community.badge_instance
CREATE TABLE IF NOT EXISTS `badge_instance` (
  `oid` int(11) NOT NULL,
  `score` decimal(19,2) DEFAULT NULL,
  `date` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  `rank_oid` int(11) DEFAULT NULL,
  `badge_type_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `idx_badge_instance_rank` (`rank_oid`),
  KEY `idx_badge_instance_badge_type` (`badge_type_oid`),
  CONSTRAINT `fk_badge_instance_badge_type` FOREIGN KEY (`badge_type_oid`) REFERENCES `badge_type` (`oid`),
  CONSTRAINT `fk_badge_instance_rank` FOREIGN KEY (`rank_oid`) REFERENCES `community_user` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
-- Data exporting was unselected.
-- Dumping structure for table community.badge_label
CREATE TABLE IF NOT EXISTS `badge_label` (
  `oid` int(11) NOT NULL,
  `language` varchar(255) DEFAULT NULL,
  `title` varchar(255) DEFAULT NULL,
  `description` varchar(255) DEFAULT NULL,
  `badge_type_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_badge_label_badge_type` (`badge_type_oid`),
  CONSTRAINT `fk_badge_label_badge_type` FOREIGN KEY (`badge_type_oid`) REFERENCES `badge_type` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
-- Dumping structure for table community.badge_type
CREATE TABLE IF NOT EXISTS `badge_type` (
  `oid` int(11) NOT NULL,
  `area` varchar(255) DEFAULT NULL,
  `needed_score` decimal(19,2) DEFAULT NULL,
  `image` varchar(255) DEFAULT NULL,
  `hd_image` varchar(255) DEFAULT NULL,
  `key` varchar(255) DEFAULT NULL,
  `importance` int(11) DEFAULT NULL,
  `description` varchar(255) DEFAULT NULL,
  `checked_image` varchar(255) DEFAULT NULL,
  `title` varchar(255) DEFAULT NULL,
  `hd_checked_image` varchar(255) DEFAULT NULL,
  `sort_number` int(11) DEFAULT NULL,
  `active` bit(1) DEFAULT NULL,
  `image_2` varchar(255) DEFAULT NULL,
  `imageblob` longblob,
  `hd_image_2` varchar(255) DEFAULT NULL,
  `hd_imageblob` longblob,
  `checked_image_2` varchar(255) DEFAULT NULL,
  `checked_imageblob` longblob,
  `hd_checked_image_2` varchar(255) DEFAULT NULL,
  `hd_checked_imageblob` longblob,
  `thematic_area_oid` int(11) DEFAULT NULL,
  `notification_type_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_badge_type_thematic_area` (`thematic_area_oid`),
  KEY `fk_badge_type_notification_typ` (`notification_type_oid`),
  CONSTRAINT `fk_badge_type_notification_typ` FOREIGN KEY (`notification_type_oid`) REFERENCES `notification_type` (`oid`),
```

```
  CONSTRAINT `fk_badge_type_thematic_area` FOREIGN KEY (`thematic_area_oid`) REFERENCES `thematic_area` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for table community.bundle_data
CREATE TABLE IF NOT EXISTS `bundle_data` (
  `oid` int(11) NOT NULL,
  `key` varchar(255) DEFAULT NULL,
  `locale` varchar(255) DEFAULT NULL,
  `message` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for table community.common_data
CREATE TABLE IF NOT EXISTS `common_data` (
  `oid` int(11) NOT NULL,
  `name` varchar(255) DEFAULT NULL,
  `image` varchar(255) DEFAULT NULL,
  `area` varchar(255) DEFAULT NULL,
  `hd_image` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for table community.community_user
CREATE TABLE IF NOT EXISTS `community_user` (
  `oid` int(11) NOT NULL,
  `company_name` varchar(255) DEFAULT NULL,
  `first_name` varchar(255) DEFAULT NULL,
  `last_name` varchar(255) DEFAULT NULL,
  `city` varchar(255) DEFAULT NULL,
  `forum_level` int(11) DEFAULT NULL,
  `small_photo` varchar(255) DEFAULT NULL,
  `twitter` varchar(255) DEFAULT NULL,
  `country` varchar(255) DEFAULT NULL,
  `public_profile` tinyint(1) DEFAULT NULL,
  `geographical_area` varchar(255) DEFAULT NULL,
  `website` varchar(255) DEFAULT NULL,
  `big_photo` varchar(255) DEFAULT NULL,
  `bio` text,
  `linkedin` varchar(255) DEFAULT NULL,
  `certification_level` int(11) DEFAULT NULL,
  `kb_level` int(11) DEFAULT NULL,
```

```
  `store_level` int(11) DEFAULT NULL,
  `participation_monthly` decimal(19,2) DEFAULT NULL,
  `forum_badge` varchar(255) DEFAULT NULL,
  `certification_badge` varchar(255) DEFAULT NULL,
  `kb_badge` varchar(255) DEFAULT NULL,
  `store_badge` varchar(255) DEFAULT NULL,
  `kb_badge_title` varchar(255) DEFAULT NULL,
  `store_badge_title` varchar(255) DEFAULT NULL,
  `forum_badge_title` varchar(255) DEFAULT NULL,
  `certification_badge_title` varchar(255) DEFAULT NULL,
  `birthdate` date DEFAULT NULL,
  `participation` decimal(19,2) DEFAULT NULL,
  `credit` decimal(19,2) DEFAULT NULL,
  `facebook` varchar(255) DEFAULT NULL,
  `google` varchar(255) DEFAULT NULL,
  `iso_code` varchar(255) DEFAULT NULL,
  `small_photo_2` varchar(255) DEFAULT NULL,
  `small_photoblob` longblob,
  `big_photo_2` varchar(255) DEFAULT NULL,
  `big_photoblob` longblob,
  `registration_date` datetime DEFAULT NULL,
  `latitude` decimal(19,6) DEFAULT NULL,
  `longitude` decimal(19,6) DEFAULT NULL,
  `globalid` varchar(255) DEFAULT NULL,
  `initial_awerness_oid` int(11) DEFAULT NULL,
  `token` varchar(255) DEFAULT NULL,
  `optin_notification` bit(1) DEFAULT NULL,
  `optin_message` bit(1) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_community_user_initial_awer` (`initial_awerness_oid`),
  CONSTRAINT `fk_community_user_initial_awer` FOREIGN KEY (`initial_awerness_oid`)
REFERENCES `initial_awerness` (`oid`),
  CONSTRAINT `fk_rank_usertable` FOREIGN KEY (`oid`) REFERENCES `user` (`user_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for view community.community_user_credits_availab
-- Creating temporary table to overcome VIEW dependency errors
CREATE TABLE `community_user_credits_availab` (
        `oid` INT(11) NOT NULL,
        `der_attr` DECIMAL(19,2) NULL
) ENGINE=MyISAM;

-- Dumping structure for view community.community_user_credits_spent_v
-- Creating temporary table to overcome VIEW dependency errors
```

```
CREATE TABLE `community_user_credits_spent_v` (
	`oid` INT(11) NOT NULL,
	`der_attr` DECIMAL(41,2) NULL
) ENGINE=MyISAM;

-- Dumping structure for table community.containers_mail
CREATE TABLE IF NOT EXISTS `containers_mail` (
  `oid` int(11) NOT NULL,
  `language_code` varchar(255) DEFAULT NULL,
  `text` text,
  `alias` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for table community.delivery_status
CREATE TABLE IF NOT EXISTS `delivery_status` (
  `oid` int(11) NOT NULL,
  `name` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
-- Dumping structure for table community.emailmessage
CREATE TABLE IF NOT EXISTS `emailmessage` (
  `oid` int(11) NOT NULL,
  `creation_date` datetime DEFAULT NULL,
  `code` varchar(255) DEFAULT NULL,
  `status` varchar(255) DEFAULT NULL,
  `delivery_date` datetime DEFAULT NULL,
  `language_code` varchar(255) DEFAULT NULL,
  `messagetemplate_oid` int(11) DEFAULT NULL,
  `user_user_id` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_emailmessage_messagetemplat` (`messagetemplate_oid`),
  KEY `fk_emailmessage_user` (`user_user_id`),
  CONSTRAINT `fk_emailmessage_messagetemplat` FOREIGN KEY (`messagetemplate_oid`) REFERENCES `messagetemplate` (`oid`),
  CONSTRAINT `fk_emailmessage_user` FOREIGN KEY (`user_user_id`) REFERENCES `user` (`user_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
-- Dumping structure for table community.gamifiedapplication_thematic_a
CREATE TABLE IF NOT EXISTS `gamifiedapplication_thematic_a` (
```

```
  `gamified_application_oid` int(11) NOT NULL,
  `thematic_area_oid` int(11) NOT NULL,
  PRIMARY KEY (`gamified_application_oid`,`thematic_area_oid`),
  KEY `fk_gamifiedapplication_themati` (`gamified_application_oid`),
  KEY `fk_gamifiedapplication_thema_2` (`thematic_area_oid`),
  CONSTRAINT `fk_gamifiedapplication_themati` FOREIGN KEY (`gamified_application_oid`) REFERENCES `gamified_application` (`oid`),
  CONSTRAINT `fk_gamifiedapplication_thema_2` FOREIGN KEY (`thematic_area_oid`) REFERENCES `thematic_area` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for table community.gamified_application
CREATE TABLE IF NOT EXISTS `gamified_application` (
  `oid` int(11) NOT NULL,
  `name` varchar(255) DEFAULT NULL,
  `api_key` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for table community.geographical_area
CREATE TABLE IF NOT EXISTS `geographical_area` (
  `oid` int(11) NOT NULL,
  `name` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for table community.goal
CREATE TABLE IF NOT EXISTS `goal` (
  `oid` int(11) NOT NULL,
  `value` int(11) DEFAULT NULL,
  `expiration` date DEFAULT NULL,
  `granularity` varchar(255) DEFAULT NULL,
  `active` bit(1) DEFAULT NULL,
  `self` bit(1) DEFAULT NULL,
  `action_type_oid` int(11) DEFAULT NULL,
  `community_user_user_id` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_goal_action_type` (`action_type_oid`),
  KEY `fk_goal_community_user` (`community_user_user_id`),
  CONSTRAINT `fk_goal_action_type` FOREIGN KEY (`action_type_oid`) REFERENCES `action_type` (`oid`),
  CONSTRAINT `fk_goal_community_user` FOREIGN KEY (`community_user_user_id`) REFERENCES `community_user` (`oid`)
```

```
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for table community.grouptable
CREATE TABLE IF NOT EXISTS `grouptable` (
  `oid_2` int(11) NOT NULL,
  `groupname` varchar(255) DEFAULT NULL,
  `siteviewoid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid_2`),
  KEY `idx_grouptable_siteviewtable` (`siteviewoid`),
  CONSTRAINT `fk_grouptable_siteviewtable` FOREIGN KEY (`siteviewoid`) REFERENCES
`siteviewtable` (`oid_2`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for table community.group_moduletable
CREATE TABLE IF NOT EXISTS `group_moduletable` (
  `groupoid` int(11) NOT NULL,
  `moduleoid` int(11) NOT NULL,
  PRIMARY KEY (`groupoid`,`moduleoid`),
  KEY `idx_group_moduletable_grouptab` (`groupoid`),
  KEY `idx_group_moduletable_siteview` (`moduleoid`),
  CONSTRAINT `fk_group_moduletable_grouptabl` FOREIGN KEY (`groupoid`)
REFERENCES `grouptable` (`oid_2`),
  CONSTRAINT `fk_group_moduletable_siteviewt` FOREIGN KEY (`moduleoid`)
REFERENCES `siteviewtable` (`oid_2`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for view community.headquarter_user_partecipation
-- Creating temporary table to overcome VIEW dependency errors
CREATE TABLE `headquarter_user_partecipation` (
        `oid` INT(11) NOT NULL,
        `partecipation` DECIMAL(41,2) NULL
) ENGINE=MyISAM;

-- Dumping structure for view community.headquarter_user_participation_monthly
-- Creating temporary table to overcome VIEW dependency errors
CREATE TABLE `headquarter_user_participation_monthly` (
        `oid` INT(11) NOT NULL,
        `participation_monthly` DECIMAL(41,2) NULL
) ENGINE=MyISAM;

-- Dumping structure for view community.headquarter_user_participation_seven_days
-- Creating temporary table to overcome VIEW dependency errors
CREATE TABLE `headquarter_user_participation_seven_days` (
```

```
	`oid` INT(11) NOT NULL,
	`participation_seven_days` DECIMAL(41,2) NULL
) ENGINE=MyISAM;

-- Dumping structure for table community.initial_awerness
CREATE TABLE IF NOT EXISTS `initial_awerness` (
  `oid` int(11) NOT NULL,
  `awareness` varchar(255) DEFAULT NULL,
  `estimated_consumption` decimal(19,2) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for table community.JOB_BLOB_TRIGGERS
CREATE TABLE IF NOT EXISTS `JOB_BLOB_TRIGGERS` (
  `SCHED_NAME` varchar(120) NOT NULL,
  `TRIGGER_NAME` varchar(200) NOT NULL,
  `TRIGGER_GROUP` varchar(200) NOT NULL,
  `BLOB_DATA` blob,
  PRIMARY KEY (`SCHED_NAME`,`TRIGGER_NAME`,`TRIGGER_GROUP`),
  KEY `SCHED_NAME` (`SCHED_NAME`,`TRIGGER_NAME`,`TRIGGER_GROUP`),
  CONSTRAINT `JOB_BLOB_TRIGGERS_ibfk_1` FOREIGN KEY (`SCHED_NAME`, `TRIGGER_NAME`, `TRIGGER_GROUP`) REFERENCES `JOB_TRIGGERS` (`SCHED_NAME`, `TRIGGER_NAME`, `TRIGGER_GROUP`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
-- Dumping structure for table community.JOB_CALENDARS
CREATE TABLE IF NOT EXISTS `JOB_CALENDARS` (
  `SCHED_NAME` varchar(120) NOT NULL,
  `CALENDAR_NAME` varchar(200) NOT NULL,
  `CALENDAR` blob NOT NULL,
  PRIMARY KEY (`SCHED_NAME`,`CALENDAR_NAME`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
-- Dumping structure for table community.JOB_CRON_TRIGGERS
CREATE TABLE IF NOT EXISTS `JOB_CRON_TRIGGERS` (
  `SCHED_NAME` varchar(120) NOT NULL,
  `TRIGGER_NAME` varchar(200) NOT NULL,
  `TRIGGER_GROUP` varchar(200) NOT NULL,
  `CRON_EXPRESSION` varchar(120) NOT NULL,
  `TIME_ZONE_ID` varchar(80) DEFAULT NULL,
  PRIMARY KEY (`SCHED_NAME`,`TRIGGER_NAME`,`TRIGGER_GROUP`),
  KEY `SCHED_NAME` (`SCHED_NAME`,`TRIGGER_NAME`,`TRIGGER_GROUP`),
```

```
  CONSTRAINT `JOB_CRON_TRIGGERS_ibfk_1` FOREIGN KEY (`SCHED_NAME`, `TRIGGER_NAME`, `TRIGGER_GROUP`) REFERENCES `JOB_TRIGGERS` (`SCHED_NAME`, `TRIGGER_NAME`, `TRIGGER_GROUP`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
-- Dumping structure for table community.JOB_FIRED_TRIGGERS
CREATE TABLE IF NOT EXISTS `JOB_FIRED_TRIGGERS` (
  `SCHED_NAME` varchar(120) NOT NULL,
  `ENTRY_ID` varchar(95) NOT NULL,
  `TRIGGER_NAME` varchar(200) NOT NULL,
  `TRIGGER_GROUP` varchar(200) NOT NULL,
  `INSTANCE_NAME` varchar(200) NOT NULL,
  `FIRED_TIME` bigint(13) NOT NULL,
  `PRIORITY` int(11) NOT NULL,
  `STATE` varchar(16) NOT NULL,
  `JOB_NAME` varchar(200) DEFAULT NULL,
  `JOB_GROUP` varchar(200) DEFAULT NULL,
  `IS_NONCONCURRENT` varchar(1) DEFAULT NULL,
  `REQUESTS_RECOVERY` varchar(1) DEFAULT NULL,
  PRIMARY KEY (`SCHED_NAME`,`ENTRY_ID`),
  KEY `IDX_JOB_FT_TRIG_INST_NAME` (`SCHED_NAME`,`INSTANCE_NAME`),
  KEY `IDX_JOB_FT_INST_JOB_REQ_RCVRY` (`SCHED_NAME`,`INSTANCE_NAME`,`REQUESTS_RECOVERY`),
  KEY `IDX_JOB_FT_J_G` (`SCHED_NAME`,`JOB_NAME`,`JOB_GROUP`),
  KEY `IDX_JOB_FT_JG` (`SCHED_NAME`,`JOB_GROUP`),
  KEY `IDX_JOB_FT_T_G` (`SCHED_NAME`,`TRIGGER_NAME`,`TRIGGER_GROUP`),
  KEY `IDX_JOB_FT_TG` (`SCHED_NAME`,`TRIGGER_GROUP`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
-- Dumping structure for table community.JOB_JOB_DETAILS
CREATE TABLE IF NOT EXISTS `JOB_JOB_DETAILS` (
  `SCHED_NAME` varchar(120) NOT NULL,
  `JOB_NAME` varchar(200) NOT NULL,
  `JOB_GROUP` varchar(200) NOT NULL,
  `DESCRIPTION` varchar(250) DEFAULT NULL,
  `JOB_CLASS_NAME` varchar(250) NOT NULL,
  `IS_DURABLE` varchar(1) NOT NULL,
  `IS_NONCONCURRENT` varchar(1) NOT NULL,
  `IS_UPDATE_DATA` varchar(1) NOT NULL,
  `REQUESTS_RECOVERY` varchar(1) NOT NULL,
  `JOB_DATA` blob,
  PRIMARY KEY (`SCHED_NAME`,`JOB_NAME`,`JOB_GROUP`),
  KEY `IDX_JOB_J_REQ_RECOVERY` (`SCHED_NAME`,`REQUESTS_RECOVERY`),
  KEY `IDX_JOB_J_GRP` (`SCHED_NAME`,`JOB_GROUP`)
```

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

-- Data exporting was unselected.
-- Dumping structure for table community.JOB_LOCKS
CREATE TABLE IF NOT EXISTS `JOB_LOCKS` (
  `SCHED_NAME` varchar(120) NOT NULL,
  `LOCK_NAME` varchar(40) NOT NULL,
  PRIMARY KEY (`SCHED_NAME`,`LOCK_NAME`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
-- Dumping structure for table community.JOB_PAUSED_TRIGGER_GRPS
CREATE TABLE IF NOT EXISTS `JOB_PAUSED_TRIGGER_GRPS` (
  `SCHED_NAME` varchar(120) NOT NULL,
  `TRIGGER_GROUP` varchar(200) NOT NULL,
  PRIMARY KEY (`SCHED_NAME`,`TRIGGER_GROUP`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
-- Dumping structure for table community.JOB_SCHEDULER_STATE
CREATE TABLE IF NOT EXISTS `JOB_SCHEDULER_STATE` (
  `SCHED_NAME` varchar(120) NOT NULL,
  `INSTANCE_NAME` varchar(200) NOT NULL,
  `LAST_CHECKIN_TIME` bigint(13) NOT NULL,
  `CHECKIN_INTERVAL` bigint(13) NOT NULL,
  PRIMARY KEY (`SCHED_NAME`,`INSTANCE_NAME`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
-- Dumping structure for table community.JOB_SIMPLE_TRIGGERS
CREATE TABLE IF NOT EXISTS `JOB_SIMPLE_TRIGGERS` (
  `SCHED_NAME` varchar(120) NOT NULL,
  `TRIGGER_NAME` varchar(200) NOT NULL,
  `TRIGGER_GROUP` varchar(200) NOT NULL,
  `REPEAT_COUNT` bigint(7) NOT NULL,
  `REPEAT_INTERVAL` bigint(12) NOT NULL,
  `TIMES_TRIGGERED` bigint(10) NOT NULL,
  PRIMARY KEY (`SCHED_NAME`,`TRIGGER_NAME`,`TRIGGER_GROUP`),
  KEY `SCHED_NAME` (`SCHED_NAME`,`TRIGGER_NAME`,`TRIGGER_GROUP`),
  CONSTRAINT `JOB_SIMPLE_TRIGGERS_ibfk_1` FOREIGN KEY (`SCHED_NAME`,
`TRIGGER_NAME`, `TRIGGER_GROUP`) REFERENCES `JOB_TRIGGERS`
(`SCHED_NAME`, `TRIGGER_NAME`, `TRIGGER_GROUP`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
```

```
-- Dumping structure for table community.JOB_SIMPROP_TRIGGERS
CREATE TABLE IF NOT EXISTS `JOB_SIMPROP_TRIGGERS` (
  `SCHED_NAME` varchar(120) NOT NULL,
  `TRIGGER_NAME` varchar(200) NOT NULL,
  `TRIGGER_GROUP` varchar(200) NOT NULL,
  `STR_PROP_1` varchar(512) DEFAULT NULL,
  `STR_PROP_2` varchar(512) DEFAULT NULL,
  `STR_PROP_3` varchar(512) DEFAULT NULL,
  `INT_PROP_1` int(11) DEFAULT NULL,
  `INT_PROP_2` int(11) DEFAULT NULL,
  `LONG_PROP_1` bigint(20) DEFAULT NULL,
  `LONG_PROP_2` bigint(20) DEFAULT NULL,
  `DEC_PROP_1` decimal(13,4) DEFAULT NULL,
  `DEC_PROP_2` decimal(13,4) DEFAULT NULL,
  `BOOL_PROP_1` varchar(1) DEFAULT NULL,
  `BOOL_PROP_2` varchar(1) DEFAULT NULL,
  PRIMARY KEY (`SCHED_NAME`,`TRIGGER_NAME`,`TRIGGER_GROUP`),
  CONSTRAINT `JOB_SIMPROP_TRIGGERS_ibfk_1` FOREIGN KEY (`SCHED_NAME`, `TRIGGER_NAME`, `TRIGGER_GROUP`) REFERENCES `JOB_TRIGGERS` (`SCHED_NAME`, `TRIGGER_NAME`, `TRIGGER_GROUP`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
-- Dumping structure for table community.JOB_TRIGGERS
CREATE TABLE IF NOT EXISTS `JOB_TRIGGERS` (
  `SCHED_NAME` varchar(120) NOT NULL,
  `TRIGGER_NAME` varchar(200) NOT NULL,
  `TRIGGER_GROUP` varchar(200) NOT NULL,
  `JOB_NAME` varchar(200) NOT NULL,
  `JOB_GROUP` varchar(200) NOT NULL,
  `DESCRIPTION` varchar(250) DEFAULT NULL,
  `NEXT_FIRE_TIME` bigint(13) DEFAULT NULL,
  `PREV_FIRE_TIME` bigint(13) DEFAULT NULL,
  `PRIORITY` int(11) DEFAULT NULL,
  `TRIGGER_STATE` varchar(16) NOT NULL,
  `TRIGGER_TYPE` varchar(8) NOT NULL,
  `START_TIME` bigint(13) NOT NULL,
  `END_TIME` bigint(13) DEFAULT NULL,
  `CALENDAR_NAME` varchar(200) DEFAULT NULL,
  `MISFIRE_INSTR` smallint(2) DEFAULT NULL,
  `JOB_DATA` blob,
  PRIMARY KEY (`SCHED_NAME`,`TRIGGER_NAME`,`TRIGGER_GROUP`),
  KEY `SCHED_NAME` (`SCHED_NAME`,`JOB_NAME`,`JOB_GROUP`),
  KEY `IDX_JOB_T_J` (`SCHED_NAME`,`JOB_NAME`,`JOB_GROUP`),
  KEY `IDX_JOB_T_JG` (`SCHED_NAME`,`JOB_GROUP`),
```

```
  KEY `IDX_JOB_T_C` (`SCHED_NAME`,`CALENDAR_NAME`),
  KEY `IDX_JOB_T_G` (`SCHED_NAME`,`TRIGGER_GROUP`),
  KEY `IDX_JOB_T_STATE` (`SCHED_NAME`,`TRIGGER_STATE`),
  KEY `IDX_JOB_T_N_STATE`
(`SCHED_NAME`,`TRIGGER_NAME`,`TRIGGER_GROUP`,`TRIGGER_STATE`),
  KEY `IDX_JOB_T_N_G_STATE`
(`SCHED_NAME`,`TRIGGER_GROUP`,`TRIGGER_STATE`),
  KEY `IDX_JOB_T_NEXT_FIRE_TIME` (`SCHED_NAME`,`NEXT_FIRE_TIME`),
  KEY `IDX_JOB_T_NFT_ST` (`SCHED_NAME`,`TRIGGER_STATE`,`NEXT_FIRE_TIME`),
  KEY `IDX_JOB_T_NFT_MISFIRE`
(`SCHED_NAME`,`MISFIRE_INSTR`,`NEXT_FIRE_TIME`),
  KEY `IDX_JOB_T_NFT_ST_MISFIRE`
(`SCHED_NAME`,`MISFIRE_INSTR`,`NEXT_FIRE_TIME`,`TRIGGER_STATE`),
  KEY `IDX_JOB_T_NFT_ST_MISFIRE_GRP`
(`SCHED_NAME`,`MISFIRE_INSTR`,`NEXT_FIRE_TIME`,`TRIGGER_GROUP`,`TRIGGER_
STATE`),
  CONSTRAINT `JOB_TRIGGERS_ibfk_1` FOREIGN KEY (`SCHED_NAME`, `JOB_NAME`,
`JOB_GROUP`) REFERENCES `JOB_JOB_DETAILS` (`SCHED_NAME`, `JOB_NAME`,
`JOB_GROUP`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
-- Dumping structure for table community.language
CREATE TABLE IF NOT EXISTS `language` (
  `oid` int(11) NOT NULL,
  `code` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for view community.max_date_action_instance
-- Creating temporary table to overcome VIEW dependency errors
CREATE TABLE `max_date_action_instance` (
        `action_type_oid` INT(11) NULL,
        `rank_oid` INT(11) NULL,
        `maxDate` TIMESTAMP NULL
) ENGINE=MyISAM;

-- Dumping structure for table community.message
CREATE TABLE IF NOT EXISTS `message` (
  `oid` int(11) NOT NULL,
  `body` text,
  `date` datetime DEFAULT NULL,
  `thread_oid` int(11) DEFAULT NULL,
  `user_user_id` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_message_thread` (`thread_oid`),
```

```
  KEY `fk_message_user` (`user_user_id`),
  CONSTRAINT `fk_message_thread` FOREIGN KEY (`thread_oid`) REFERENCES `thread` (`oid`),
  CONSTRAINT `fk_message_user` FOREIGN KEY (`user_user_id`) REFERENCES `user` (`user_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for table community.messagetemplate
CREATE TABLE IF NOT EXISTS `messagetemplate` (
  `oid` int(11) NOT NULL,
  `language` varchar(255) DEFAULT NULL,
  `code` varchar(255) DEFAULT NULL,
  `body` text,
  `object` varchar(255) DEFAULT NULL,
  `volatile` bit(1) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
-- Dumping structure for table community.message_label
CREATE TABLE IF NOT EXISTS `message_label` (
  `oid` int(11) NOT NULL,
  `language` varchar(255) DEFAULT NULL,
  `body` longtext,
  `message_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_message_label_message` (`message_oid`),
  CONSTRAINT `fk_message_label_message` FOREIGN KEY (`message_oid`) REFERENCES `message` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for view community.mostimportant_badge
-- Creating temporary table to overcome VIEW dependency errors
CREATE TABLE `mostimportant_badge` (
	`oid` INT(11) NOT NULL,
	`rankoid` INT(11) NOT NULL,
	`area` VARCHAR(255) NULL COLLATE 'utf8_general_ci',
	`title` VARCHAR(255) NULL COLLATE 'utf8_general_ci',
	`importance` INT(11) NULL,
	`checked_image_2` VARCHAR(255) NULL COLLATE 'utf8_general_ci',
	`checked_imageblob` LONGBLOB NULL,
	`hd_checked_image_2` VARCHAR(255) NULL COLLATE 'utf8_general_ci',
	`hd_checked_imageblob` LONGBLOB NULL,
	`sort_number` INT(11) NULL
```

```
) ENGINE=MyISAM;

-- Dumping structure for table community.notification
CREATE TABLE IF NOT EXISTS `notification` (
  `oid` int(11) NOT NULL,
  `creation_date` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  `code` varchar(255) DEFAULT NULL,
  `status` varchar(255) DEFAULT NULL,
  `delivery_date` timestamp NULL DEFAULT NULL,
  `language_code` varchar(255) DEFAULT NULL,
  `rank_oid` int(11) DEFAULT NULL,
  `reward_type_oid` int(11) DEFAULT NULL,
  `text_mail_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `idx_notification_rank` (`rank_oid`),
  KEY `idx_notification_reward_type` (`reward_type_oid`),
  KEY `idx_notification_text_mail` (`text_mail_oid`),
  CONSTRAINT `fk_notification_rank` FOREIGN KEY (`rank_oid`) REFERENCES `community_user` (`oid`),
  CONSTRAINT `fk_notification_reward_type` FOREIGN KEY (`reward_type_oid`) REFERENCES `reward_type` (`oid`),
  CONSTRAINT `fk_notification_text_mail` FOREIGN KEY (`text_mail_oid`) REFERENCES `text_mail` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for table community.notification_instance
CREATE TABLE IF NOT EXISTS `notification_instance` (
  `oid` int(11) NOT NULL,
  `date` datetime DEFAULT NULL,
  `notification_type_oid` int(11) DEFAULT NULL,
  `community_user_user_id` int(11) DEFAULT NULL,
  `deleted` datetime DEFAULT NULL,
  `read` datetime DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_notification_instance_notif` (`notification_type_oid`),
  KEY `fk_notification_instance_commu` (`community_user_user_id`),
  CONSTRAINT `fk_notification_instance_commu` FOREIGN KEY (`community_user_user_id`) REFERENCES `community_user` (`oid`),
  CONSTRAINT `fk_notification_instance_notif` FOREIGN KEY (`notification_type_oid`) REFERENCES `notification_type` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for table community.notification_label
```

```
CREATE TABLE IF NOT EXISTS `notification_label` (
  `oid` int(11) NOT NULL,
  `lang` varchar(255) DEFAULT NULL,
  `title` longtext,
  `notification_type_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_notification_label_notifica` (`notification_type_oid`),
  CONSTRAINT `fk_notification_label_notifica` FOREIGN KEY (`notification_type_oid`) REFERENCES `notification_type` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for table community.notification_type
CREATE TABLE IF NOT EXISTS `notification_type` (
  `oid` int(11) NOT NULL,
  `type` varchar(255) DEFAULT NULL,
  `title` varchar(255) DEFAULT NULL,
  `icon` varchar(255) DEFAULT NULL,
  `iconblob` longblob,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for table community.reward_instance
CREATE TABLE IF NOT EXISTS `reward_instance` (
  `oid` int(11) NOT NULL,
  `date` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  `score` decimal(19,2) DEFAULT NULL,
  `rank_oid` int(11) DEFAULT NULL,
  `reward_type_oid` int(11) DEFAULT NULL,
  `delivery_status_oid` int(11) DEFAULT NULL,
  `recipient_address` varchar(255) DEFAULT NULL,
  `shipment_date` date DEFAULT NULL,
  `note` longtext,
  PRIMARY KEY (`oid`),
  KEY `idx_reward_instance_rank` (`rank_oid`),
  KEY `idx_reward_instance_reward_typ` (`reward_type_oid`),
  KEY `fk_reward_instance_delivery_st` (`delivery_status_oid`),
  CONSTRAINT `fk_reward_instance_delivery_st` FOREIGN KEY (`delivery_status_oid`) REFERENCES `delivery_status` (`oid`),
  CONSTRAINT `fk_reward_instance_rank` FOREIGN KEY (`rank_oid`) REFERENCES `community_user` (`oid`),
  CONSTRAINT `fk_reward_instance_reward_type` FOREIGN KEY (`reward_type_oid`) REFERENCES `reward_type` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
-- Data exporting was unselected.
-- Dumping structure for table community.reward_label
CREATE TABLE IF NOT EXISTS `reward_label` (
  `oid` int(11) NOT NULL,
  `language` varchar(255) DEFAULT NULL,
  `name` varchar(255) DEFAULT NULL,
  `descr` text,
  `reward_type_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_reward_label_reward_type` (`reward_type_oid`),
  CONSTRAINT `fk_reward_label_reward_type` FOREIGN KEY (`reward_type_oid`) REFERENCES `reward_type` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
-- Dumping structure for view community.reward_labels
-- Creating temporary table to overcome VIEW dependency errors
CREATE TABLE `reward_labels` (
	`reward_type_oid` INT(11) NULL,
	`languagecode` VARCHAR(255) NULL COLLATE 'latin1_swedish_ci',
	`name` VARCHAR(255) NULL COLLATE 'latin1_swedish_ci',
	`descr` VARCHAR(255) NULL COLLATE 'latin1_swedish_ci'
) ENGINE=MyISAM;

-- Dumping structure for table community.reward_type
CREATE TABLE IF NOT EXISTS `reward_type` (
  `oid` int(11) NOT NULL,
  `needed_points` decimal(19,2) DEFAULT NULL,
  `available` tinyint(1) DEFAULT NULL,
  `image` varchar(255) DEFAULT NULL,
  `language_code` varchar(255) DEFAULT NULL,
  `title` varchar(255) DEFAULT NULL,
  `description` text,
  `image_2` varchar(255) DEFAULT NULL,
  `imageblob` longblob,
  `notification_type_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_reward_type_notification_ty` (`notification_type_oid`),
  CONSTRAINT `fk_reward_type_notification_ty` FOREIGN KEY (`notification_type_oid`) REFERENCES `notification_type` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for table community.siteviewtable
```

```
CREATE TABLE IF NOT EXISTS `siteviewtable` (
  `oid_2` int(11) NOT NULL,
  `moduledomainname` varchar(255) DEFAULT NULL,
  `siteviewid` varchar(255) DEFAULT NULL,
  `modulename` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`oid_2`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for table community.text_chunk
CREATE TABLE IF NOT EXISTS `text_chunk` (
  `oid` int(11) NOT NULL,
  `languagecode` varchar(255) DEFAULT NULL,
  `key` varchar(255) DEFAULT NULL,
  `message` text,
  `action_type_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_text_chunk_action_type` (`action_type_oid`),
  CONSTRAINT `fk_text_chunk_action_type` FOREIGN KEY (`action_type_oid`) REFERENCES `action_type` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for table community.text_chunk_gamified
CREATE TABLE IF NOT EXISTS `text_chunk_gamified` (
  `oid` int(11) NOT NULL,
  `languagecode` varchar(255) DEFAULT NULL,
  `message` varchar(255) DEFAULT NULL,
  `key` varchar(255) DEFAULT NULL,
  `action_type_oid` int(11) DEFAULT NULL,
  `reward_type_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_text_chunk_gamified_action` (`action_type_oid`),
  KEY `fk_text_chunk_gamified_reward` (`reward_type_oid`),
  CONSTRAINT `fk_text_chunk_gamified_action` FOREIGN KEY (`action_type_oid`) REFERENCES `action_type` (`oid`),
  CONSTRAINT `fk_text_chunk_gamified_reward` FOREIGN KEY (`reward_type_oid`) REFERENCES `reward_type` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
-- Dumping structure for table community.text_mail
CREATE TABLE IF NOT EXISTS `text_mail` (
  `oid` int(11) NOT NULL,
  `code` varchar(255) DEFAULT NULL,
  `language_code` varchar(255) DEFAULT NULL,
```

```
  `body` text,
  `description` varchar(255) DEFAULT NULL,
  `subject` varchar(255) DEFAULT NULL,
  `containers_oid_header` int(11) DEFAULT NULL,
  `containers_oid_footer` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `idx_text_mail_containers_mail` (`containers_oid_header`),
  KEY `idx_text_mail_containers_mai_2` (`containers_oid_footer`),
  CONSTRAINT `fk_text_mail_containers_mail` FOREIGN KEY (`containers_oid_header`) REFERENCES `containers_mail` (`oid`),
  CONSTRAINT `fk_text_mail_containers_mail_2` FOREIGN KEY (`containers_oid_footer`) REFERENCES `containers_mail` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for table community.thematic_area
CREATE TABLE IF NOT EXISTS `thematic_area` (
  `oid` int(11) NOT NULL,
  `area_name` varchar(255) DEFAULT NULL,
  `checked_image` varchar(255) DEFAULT NULL,
  `hd_image` varchar(255) DEFAULT NULL,
  `hd_checked_image` varchar(255) DEFAULT NULL,
  `checked_imageblob` blob,
  `hd_checked_imageblob` blob,
  `hd_imageblob` blob,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for table community.thread
CREATE TABLE IF NOT EXISTS `thread` (
  `oid` int(11) NOT NULL,
  `subject` varchar(255) DEFAULT NULL,
  `language` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for table community.thread_label
CREATE TABLE IF NOT EXISTS `thread_label` (
  `oid` int(11) NOT NULL,
  `language` varchar(255) DEFAULT NULL,
  `subject` varchar(255) DEFAULT NULL,
  `thread_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
```

```
  KEY `fk_thread_label_thread` (`thread_oid`),
  CONSTRAINT `fk_thread_label_thread` FOREIGN KEY (`thread_oid`) REFERENCES `thread` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for table community.usage_logger
CREATE TABLE IF NOT EXISTS `usage_logger` (
  `oid` int(11) NOT NULL,
  `timestamp` datetime DEFAULT NULL,
  `event` varchar(255) DEFAULT NULL,
  `community_user_user_id` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_usage_logger_community_user` (`community_user_user_id`),
  CONSTRAINT `fk_usage_logger_community_user` FOREIGN KEY (`community_user_user_id`) REFERENCES `community_user` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for table community.user
CREATE TABLE IF NOT EXISTS `user` (
  `user_id` int(11) NOT NULL,
  `email` varchar(255) DEFAULT NULL,
  `password` varchar(255) DEFAULT NULL,
  `internal` tinyint(1) DEFAULT NULL,
  `username` varchar(255) DEFAULT NULL,
  `groupoid` int(11) DEFAULT NULL,
  PRIMARY KEY (`user_id`),
  KEY `idx_usertable_grouptable` (`groupoid`),
  CONSTRAINT `fk_usertable_grouptable` FOREIGN KEY (`groupoid`) REFERENCES `grouptable` (`oid_2`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for table community.user_grouptable
CREATE TABLE IF NOT EXISTS `user_grouptable` (
  `useroid` int(11) NOT NULL,
  `groupoid` int(11) NOT NULL,
  PRIMARY KEY (`useroid`,`groupoid`),
  KEY `idx_user_grouptable_usertable` (`useroid`),
  KEY `idx_user_grouptable_grouptable` (`groupoid`),
  CONSTRAINT `fk_user_grouptable_grouptable` FOREIGN KEY (`groupoid`) REFERENCES `grouptable` (`oid_2`),
  CONSTRAINT `fk_user_grouptable_usertable` FOREIGN KEY (`useroid`) REFERENCES `user` (`user_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
-- Data exporting was unselected.
-- Dumping structure for view community.user_information
-- Creating temporary table to overcome VIEW dependency errors
CREATE TABLE `user_information` (
	`oid` INT(11) NOT NULL,
	`country` VARCHAR(255) NULL COLLATE 'utf8_general_ci',
	`area_geografica` VARCHAR(255) NULL COLLATE 'utf8_general_ci',
	`small_photo` VARCHAR(255) NULL COLLATE 'utf8_general_ci',
	`big_photo` VARCHAR(255) NULL COLLATE 'utf8_general_ci',
	`first_name` VARCHAR(255) NULL COLLATE 'utf8_general_ci',
	`last_name` VARCHAR(255) NULL COLLATE 'utf8_general_ci',
	`twitter` VARCHAR(255) NULL COLLATE 'utf8_general_ci',
	`linkedin` VARCHAR(255) NULL COLLATE 'utf8_general_ci',
	`website` VARCHAR(255) NULL COLLATE 'utf8_general_ci',
	`bio` TEXT NULL COLLATE 'utf8_general_ci',
	`city` VARCHAR(255) NULL COLLATE 'utf8_general_ci',
	`company_name` VARCHAR(255) NULL COLLATE 'utf8_general_ci',
	`email` VARCHAR(255) NULL COLLATE 'utf8_general_ci',
	`internal` TINYINT(1) NULL
) ENGINE=MyISAM;

-- Dumping structure for table community.user_thread_mapping
CREATE TABLE IF NOT EXISTS `user_thread_mapping` (
  `oid` int(11) NOT NULL,
  `read` datetime DEFAULT NULL,
  `deleted` datetime DEFAULT NULL,
  `user_user_id` int(11) DEFAULT NULL,
  `thread_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_user_thread_mapping_user` (`user_user_id`),
  KEY `fk_user_thread_mapping_thread` (`thread_oid`),
  CONSTRAINT `fk_user_thread_mapping_thread` FOREIGN KEY (`thread_oid`) REFERENCES `thread` (`oid`),
  CONSTRAINT `fk_user_thread_mapping_user` FOREIGN KEY (`user_user_id`) REFERENCES `user` (`user_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Data exporting was unselected.
-- Dumping structure for view community.user_thread_mapping_date_view
-- Creating temporary table to overcome VIEW dependency errors
CREATE TABLE `user_thread_mapping_date_view` (
	`oid` INT(11) NOT NULL,
	`der_attr` DATETIME NULL
) ENGINE=MyISAM;
```

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