the smartH2O project
A European project on water sustainability

PLATFORM IMPLEMENTATION AND INTEGRATION – INITIAL PROTOTYPE

SmartH2O

Project FP7-ICT-619172

Deliverable D6.3 WP6

Deliverable
Version 1.0 – 30 Mar 2015

Document. ref.:
D6.3.SETMOB.WP6.V1.0

Programme Name: .....ICT
Project Number:.....619172
Project Title: SmartH2O
Partners: .....Coordinator: SUPSI
Contractors: POLIMI, SETMOB, TWUL, SES, MOONSUB

Document Number: smarth2o. D6.3.SETMOB.WP6.V1.0
Work-Package:.....WP6
Deliverable Type: .....Document
Contractual Date of Delivery: .....31 March 2015
Actual Date of Delivery: .....31 March 2015
Title of Document: .....Platform Implementation and Integration
Author(s): .....Luigi Caldararu, Piero Fraternali, Chiara Pasini, Giorgia Baroffio, Marco Tagliasacchi, Andrea Emilio Rizzoli.

Approval of this report .....Submitted for approval by EC

Summary of this report:.....D6.3 Platform Implementation and Integration-Initial prototype

History: .....n/a

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

Disclaimer

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.

Table of Contents

Executive Summary

This is the accompanying document of the Deliverable **D6.3: Platform Implementation and Integration – Initial prototype**, which, according to the Description of Work, is a software deliverable containing the initial prototype of the SmartH2O platform. It provides the infrastructure to collect and organize the water consumption data of the consumers. Also, it contains the first implementation of the first user behaviour models produced in Task3.3

The deliverable includes the source code and the documentation for use and installation of the platform.

1. SmartH2O platform – Initial prototype

1.1 State of the development process

The figure 1 is the UML component representation of the SmartH2O platform.

The main components of the integrated platform are:

- **SmartH2O platform database**
- **Smart Meter Data Management** component - having the role of meter reading data provider for the platform
- **Enterprise Service Bus** which is the centralizing component acting as
  - Single Point of Access
  - Transaction Manager
  - Security Manager.

In the representation of the current state of development of the software components of the SmartH2O platform, the states of the components have the following meaning:

Green – completed

Yellow – in progress

Red – not started

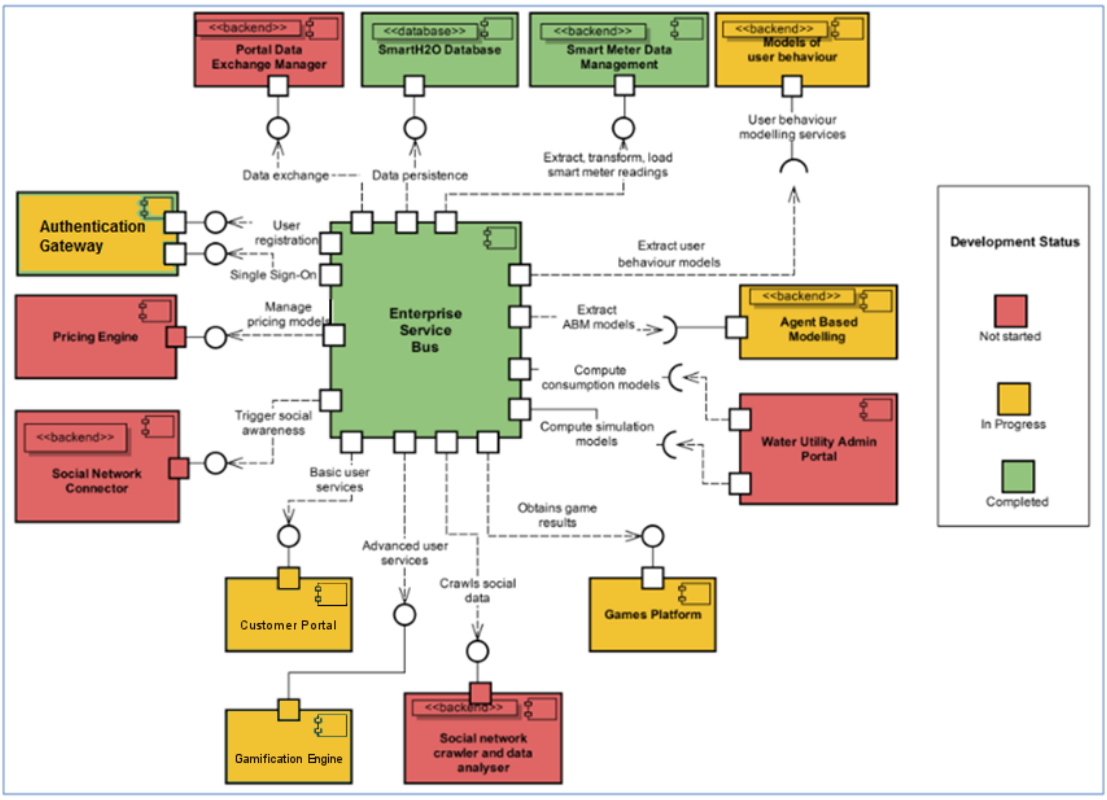


Figure 1: overview of the main components of the SmartH2O architecture and their current state of development.

2. Installation guide

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

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

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

User: smarth2o-guest

Password: smarth2oguest

2.1 SmartH2O Database

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

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

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

The dump file for creating and initializing the SmartH2O platform database with a testing data set is available on the Bitbucket project account at:

https://bitbucket.org/smarth2o-setmob1/smarth2o_p1_dump/src/c6da51c32818d22184b3faccd5b36ea4c6933d33?at=master

2.2 Smart Meter Data Management Component

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

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

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

Requirements:

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

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

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

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

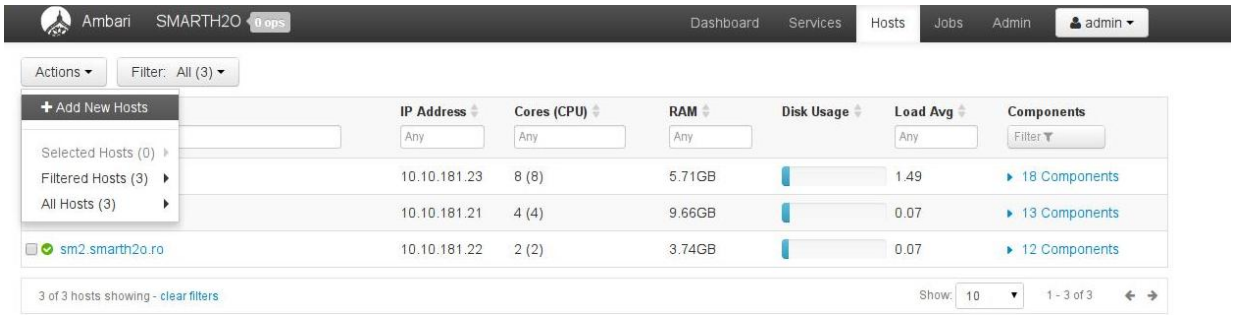
```
wget http://public-repo-1.hortonworks.com/ambari/centos6/1.x/GA/ambari.repo
cp ambari.repo /etc/yum.repos.d
yum install ambari-server
ambari-server setup
```

After the setup is completed, start Ambari service:

```
ambari-server start
```

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

To add new hosts to the Hadoop cluster:



To install a new Ambari component:

---

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

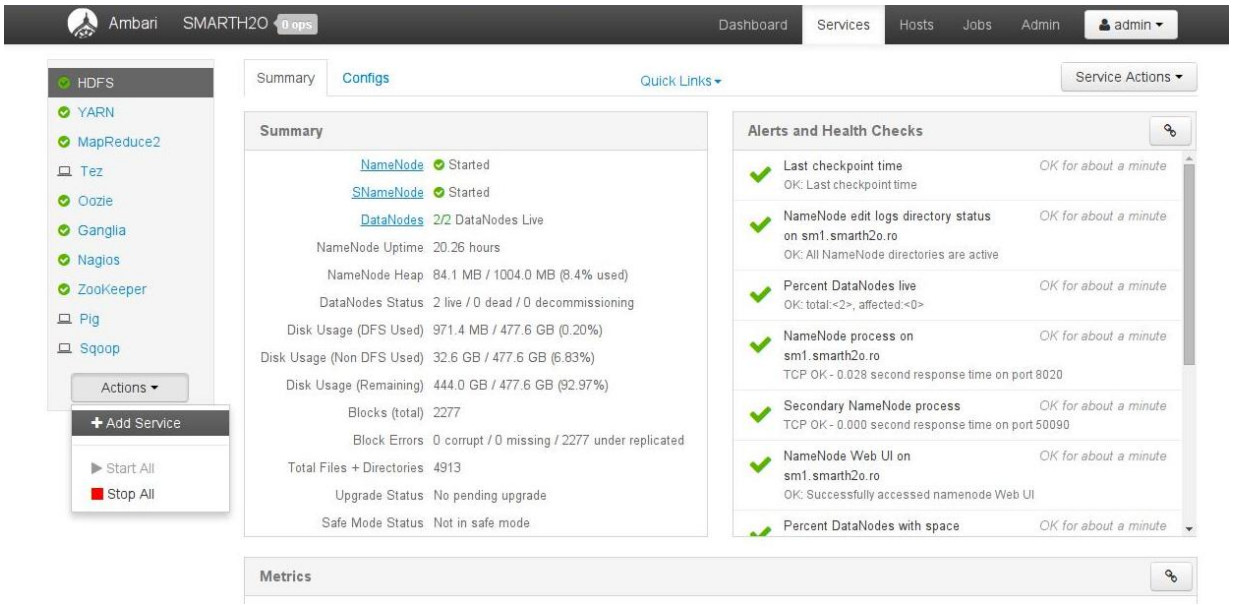
[2] HDFS DataNode https://wiki.apache.org/hadoop/DataNode

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

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

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

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



The source for the manager and transform component are available at:
https://bitbucket.org/smarth2o/sh2osmdmctransform
https://bitbucket.org/smarth2o/sh2osmdmcmanager

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

```
mvn package
```

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

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

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

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

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

The **SMDMC Transform subcomponent** must be built using Maven pom.xml script from https://bitbucket.org/smarth2o/sh2osmdmctransform/src/3d52d50bab5496d849fd8e67847290f2fb2c7631?at=master

Using:

`mvn package` command

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

2.2 Enterprise Service Bus based on JBoss Fuse

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

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

Requirements

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

Building and Running

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

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

mvn clean install

Project components

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

Bundles installation and testing

1. Install 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 Base Service

Change to the client sub-project, and run
mvn -P simple

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

3. Deploying Newservice Service

In the JBoss Fuse console, launch the following command:
features:install dynamic-routing-newservice

4. Testing 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.

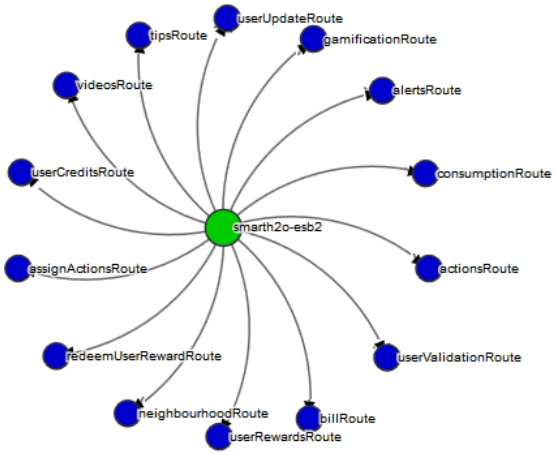
The original Base service doesn’t need to be reloaded or restarted in order to keep forwarding messages to the newly loaded routes. The original tests can be re-run using:

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

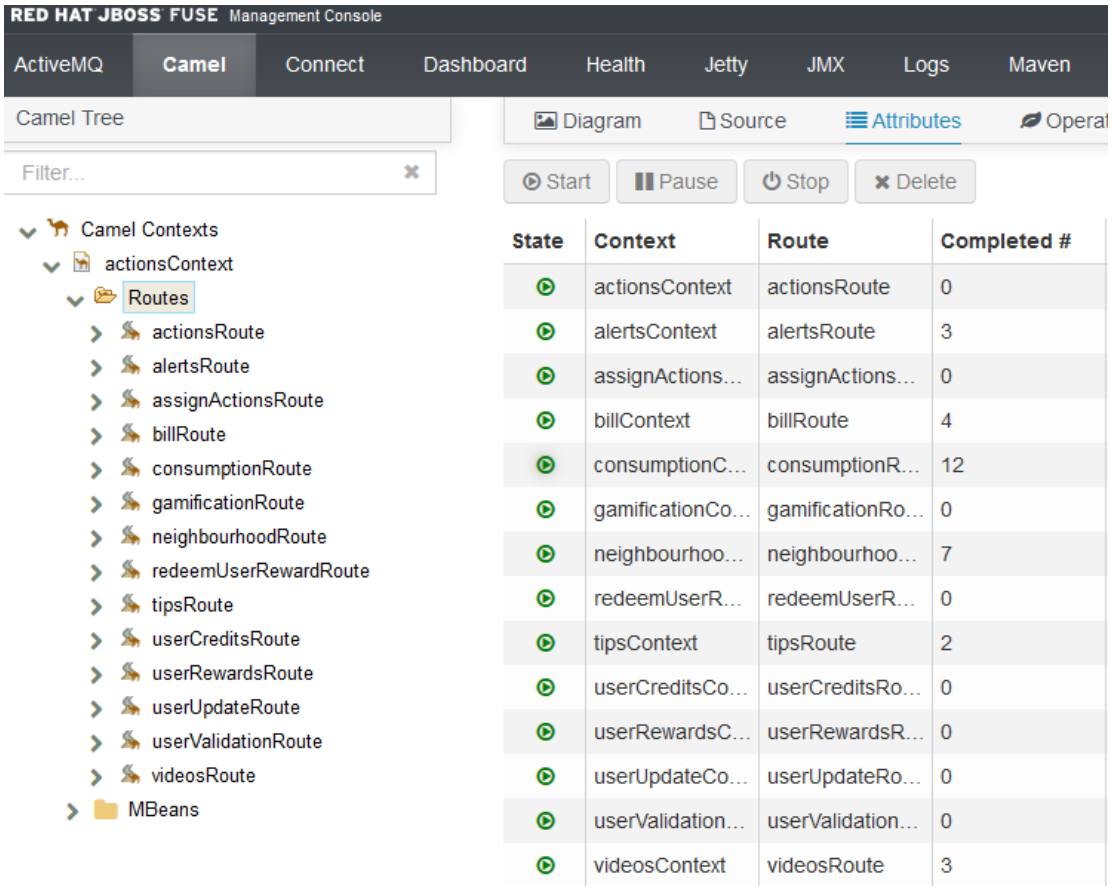
Newservice gets the messages from the Camel VM component, and puts them onto an ActiveMQ Queue. This shows how to route to new endpoints and integration routes at runtime -- it does not have to be ActiveMQ, it could easily be WS, REST or other JMS.

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

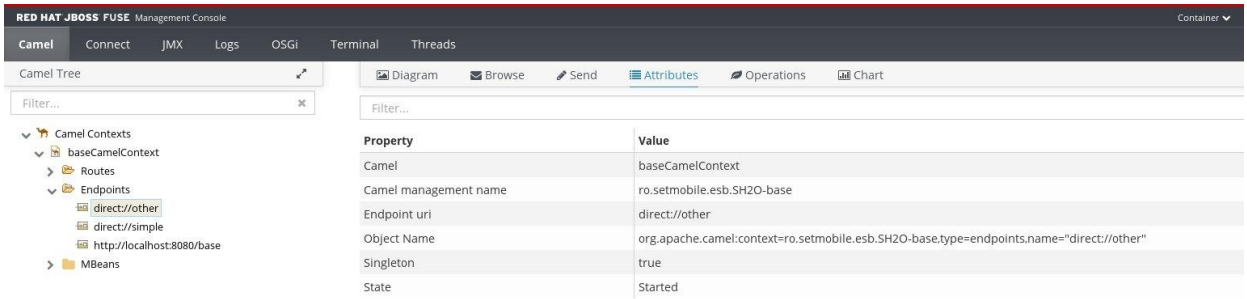
Howtio schemas of deployed services:



Camel routes:



Enpoints:



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

Username: smarth2o

Password: dsfsmarth2o

2.3 Customer Portal and Gamification Engine

Requirements:

- Apache Tomcat 6.x or later: http://tomcat.apache.org/
- MySQL DBMS 5.6

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-setmob1/smarth2o_p1_dump/src/0fd0a92de4e4fca9148cf18e1f97db551d1e2b4e/community_new_newdata.sql?at=smarth2o-setmob1/community_new_newdatasql-created-online--1427826477536
- Create a new database "gamified_app_new" and import the sql file gamified_app_new.sql from https://bitbucket.org/smarth2o-setmob1/smarth2o_p1_dump/raw/0fd0a92de4e4fca9148cf18e1f97db551d1e2b4e/gamified_app_new.sql

2. Application installation on Tomcat

- In the \Tomcat\webapps folder download and add the following:
  - From https://bitbucket.org/smarth2o-setmob1/gamificationfrontendstyle/src/853a948a75b6912e71189d461c989637d25806af?at=master download and add the frontend and backend of the gamified portal.
  - From https://bitbucket.org/smarth2o-setmob1/gamifiedapplicationtest/src/08bda21d62d5341995d14258ee75b299eb33602e?at=master download and add GamifiedApplicationTest: simulator for the community app.
  - From https://bitbucket.org/smarth2o-setmob1/smarth20restservices/src/74d32d56ce900b25051936a175d1e3343fa068de?at=master download and add SmartH20RestServices: rest services

called by the community app.

- From https://bitbucket.org/smarth2o-setmob1/json-simple/src/01bf1f30adf777e915994f0f79436b548cefcb13/json-simple-1.1.1.jar?at=master# add the "json-simple-1.1.1.jar" file under the "Tomcat/lib" folder

This configuration allows to execute the applications without any change.

It is possible to change the database configuration, updating the "dbx.hibernate.cfg.xml" file in the /WEB-INF/classes of the applications.

Testing

The applications can be invoked locally at the following url:

- http://localhost:8080/community (frontend)
- http://localhost:8080/community/admincommunity (backend)
- http://localhost:8080/GamifiedApplicationTest
- http://localhost:8080/SmartH20RestServices

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

- Basic Version:
  username: ChiaraPasini / password: *password*
- Advanced Version (gamified)
  username: LucaGalli / password: *password*

2.4 Games Platform

Drop! The Game has been developed using the Unity 3D Game Engine, available for free at http://unity3d.com/. The sources of the Drop! Game are available for download from: https://bitbucket.org/moonsubmarine/drop **.** The source code for the Unity game can be found under the "src\Drop-Unity\Drop" folder. The game has been developed with Android and iOS in mind.

Deploy and testing procedure

Unity has been chosen has the development platform since it is a multiplatform engine that allows multiplatform deployment by sharing the same codebase for all the platform, easing the development and integration among different members of the development team.

Requirements

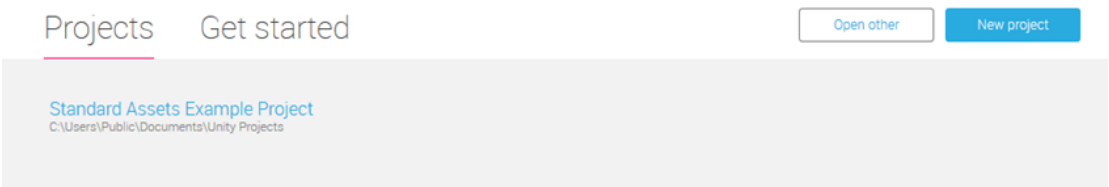
- Unity3D 5.0 or later (http://unity3d.com/get-unity/download, installation guide: http://docs.unity3d.com/Manual/index.html)
- Easy Code Scanner Plugin (commercial plugin, cannot be released freely) http://u3d.as/content/c4mprod/easy-code-scanner/3sM

Building and Running

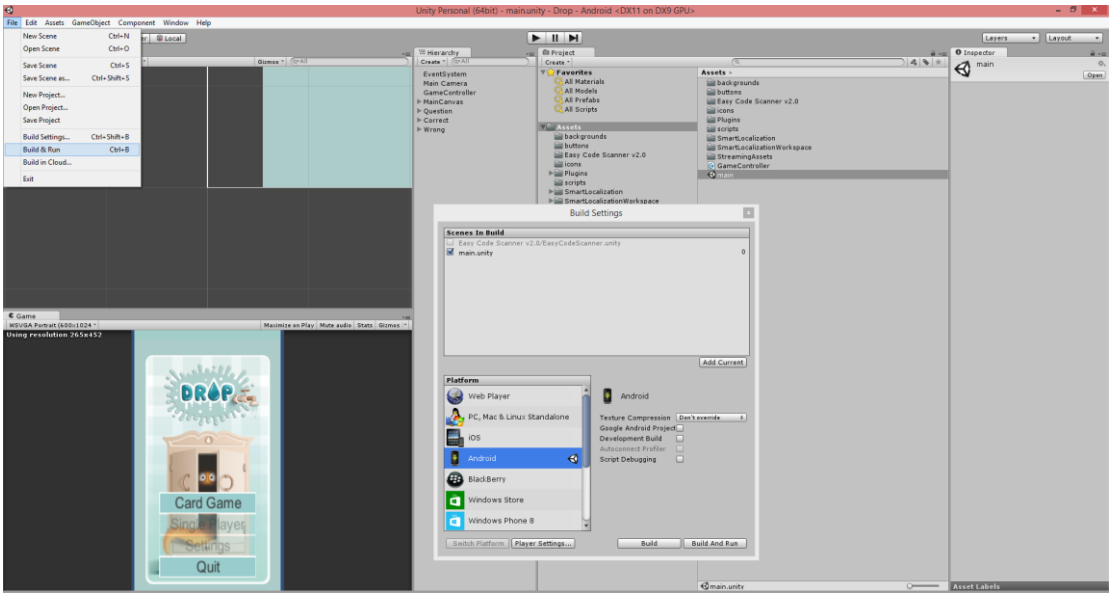
Extract the content of the "Drop" source code in a folder.

As the QR code plugin cannot be released in this package, once it has been downloaded from the developer page of the plugin listed in the requirements section, the content of the “Plugin” folder of the downloaded component must be copied into the “Asset\Plugin” folder of the Drop source code folder.

It is now possible to open the Drop! Project in Unity Game Engine. Open the project by clicking on “Open Other” and choose the folder in which the source code for Drop! has been extracted.



Once the project has been opened, it is possible to compile it and eventually deploy it for the platform of interest. It has to be noted that, since the game is making use of handheld cameras and specific calls for iOS and Android operative systems, it is not possible to run the game on desktop platforms.



Based on the specific platform that should be targeted, it is possible to follow the detailed guides available on the Game Engine's Website:

- Android Development
- iOS Development http://docs.unity3d.com/Manual/iphone-GettingStarted.html

It has to be noted that the iOS deployment of the application requires an Apple Developer Account to be fulfilled.

Testing

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

3. Appendix A: SmartH2O platform database DDL script

```
# MySQL-Front 5.1  (Build 4.13)

/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE */;
/*!40101 SET
SQL_MODE='STRICT_TRANS_TABLES,NO_ENGINE_SUBSTITUTION' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES */;
/*!40103 SET SQL_NOTES='ON' */;


# Host: 89.121.250.90:3308    Database: smarth2o
# ------------------------------------------------------
# Server version 5.6.14-log

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

#
# Source for table alert
#

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

#
# Source for table bill
#

CREATE TABLE `bill` (
  `oid` int(11) NOT NULL,
  `account_number` varchar(255) DEFAULT NULL,
  `bill_date` date DEFAULT NULL,
  `company` varchar(255) DEFAULT NULL,
  `volume_charge` decimal(19,2) DEFAULT NULL,
  `service_charge` decimal(19,2) DEFAULT NULL,
  `currency` varchar(255) DEFAULT NULL,
  `volume_eur_charge` decimal(19,2) DEFAULT NULL,
  `service_eur_charge` decimal(19,2) DEFAULT NULL,
  `exchange_rate` decimal(19,2) DEFAULT NULL,
  `exchange_date` date DEFAULT NULL,
```

```
  `household_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_bill_household` (`household_oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table billing_price
#

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

#
# Source for table billing_price_bill
#

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

#
# Source for table building
#

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

#
# Source for table consumer_segment
#

CREATE TABLE `consumer_segment` (
```

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

#
# Source for table consumer_segment_neutral_user
#

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

#
# Source for table device_class
#

CREATE TABLE `device_class` (
  `oid` int(11) NOT NULL,
  `name` varchar(255) DEFAULT NULL,
  `number` int(11) DEFAULT NULL,
  `household_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_device_class_household` (`household_oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table device_consumption
#

CREATE TABLE `device_consumption` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `start_date` datetime DEFAULT NULL,
  `end_date` datetime DEFAULT NULL,
  `device_consumption` decimal(19,3) DEFAULT NULL,
  `device_class_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_device_consumption_device_c` (`device_class_oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table district
#

CREATE TABLE `district` (
  `oid` int(11) NOT NULL,
  `zipcode` varchar(255) DEFAULT NULL,
```

```
  `country` varchar(255) DEFAULT NULL,
  `city` varchar(255) DEFAULT NULL,
  `name` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table feature
#

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

#
# Source for table group
#

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

#
# Source for table group_module
#

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

#
# Source for table household
#

CREATE TABLE `household` (
  `oid` int(11) NOT NULL,
  `utilityid` varchar(255) DEFAULT NULL,
  `household_size` decimal(19,2) DEFAULT NULL,
  `ownership` bit(1) DEFAULT NULL,
  `number_occupants` int(11) 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,
  `building_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_household_building` (`building_oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table household_consumption
#

CREATE TABLE `household_consumption` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `consumption` decimal(19,3) DEFAULT NULL,
  `start_date` datetime DEFAULT NULL,
  `end_date` datetime DEFAULT NULL,
  `household_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_household_consumption_house` (`household_oid`)
) ENGINE=InnoDB AUTO_INCREMENT=190 DEFAULT CHARSET=utf8;

#
# Source for table mail
#

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

#
# Source for table media_asset
#

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

#
```

```
# Source for table meter_reading
#

CREATE TABLE `meter_reading` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `reading_date_time` datetime DEFAULT NULL,
  `meter_id` varchar(255) DEFAULT NULL,
  `company` varchar(255) DEFAULT NULL,
  `total_consumption` decimal(19,3) DEFAULT NULL,
  `building_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_meter_reading_building` (`building_oid`)
) ENGINE=InnoDB AUTO_INCREMENT=166424 DEFAULT CHARSET=utf8;

#
# Source for table module
#

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

#
# Source for table neutral_user
#

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

#
# Source for table neutral_user_media_asset
#

CREATE TABLE `neutral_user_media_asset` (
```

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

#
# Source for table neutral_user_tip
#

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

#
# Source for table tip
#

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

#
# Source for table unit_of_measurement
#

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

#
# Source for table user
#

CREATE TABLE `user` (
  `oid` int(11) NOT NULL,
  `username` varchar(255) DEFAULT NULL,
  `password` varchar(255) DEFAULT NULL,
```

```
  `email` varchar(255) DEFAULT NULL,
  `first_name` varchar(255) DEFAULT NULL,
  `last_name` varchar(255) DEFAULT NULL,
  `birth_date` varchar(255) DEFAULT NULL,
  `internal` bit(1) DEFAULT NULL,
  `group_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_user_group` (`group_oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Source for table user_group
#

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

#
# Source for table weather_condition
#

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

#
#  Foreign keys for table alert
#

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

#
#  Foreign keys for table bill
#

ALTER TABLE `bill`
```

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

#
#  Foreign keys for table billing_price_bill
#

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

#
#  Foreign keys for table building
#

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

#
#  Foreign keys for table consumer_segment_neutral_user
#

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

#
#  Foreign keys for table device_class
#

ALTER TABLE `device_class`
ADD CONSTRAINT `fk_device_class_household` FOREIGN KEY
(`household_oid`) REFERENCES `household` (`oid`);

#
#  Foreign keys for table device_consumption
#

ALTER TABLE `device_consumption`
ADD CONSTRAINT `fk_device_consumption_device_c` FOREIGN KEY
(`device_class_oid`) REFERENCES `device_class` (`oid`);

#
#  Foreign keys for table feature
#

ALTER TABLE `feature`
```

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

#
#  Foreign keys for table group
#

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

#
#  Foreign keys for table group_module
#

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

#
#  Foreign keys for table household
#

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

#
#  Foreign keys for table household_consumption
#

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

#
#  Foreign keys for table meter_reading
#

ALTER TABLE `meter_reading`
ADD CONSTRAINT `fk_meter_reading_building` FOREIGN KEY
(`building_oid`) REFERENCES `building` (`oid`);

#
#  Foreign keys for table neutral_user
#

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

```
ADD CONSTRAINT `fk_neutral_user_user` FOREIGN KEY (`user_oid`)
REFERENCES `user` (`oid`);

#
#  Foreign keys for table neutral_user_media_asset
#

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

#
#  Foreign keys for table neutral_user_tip
#

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

#
#  Foreign keys for table user
#

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

#
#  Foreign keys for table user_group
#

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

#
#  Foreign keys for table weather_condition
#

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


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