D6.5 PLATFORM FINAL PROTOTYPE

Continuous development of the platform and its modules

Project title	**Collaborative Recommendations and Adaptive Control for Personalised Energy Saving**
Project acronym	**enCOMPASS**
Project call	**EE-07-2016-2017 Behavioural change toward energy efficiency through ICT**
Work Package	**WP6**
Lead Partner	**SMOB**
Contributing Partner(s)	**PMI, CERTH, GRA, PDX, SUPSI, EICPM**
Security classification	**Public**
Contractual delivery date	**31/10/2019**
Actual delivery date	**29/10/2019**
Version	**1.1**
Reviewers	**PMI, CERTH, SUPSI**

History of changes

Version	Date	Comments	Main Authors
0.1	02/09/2019	Defining Table of Content	L.Caldararu (SMOB)
0.2	09/09/2019	Defining DDP	L.Caldararu (SMOB)
0.3	20/09/2019	SMOB contribution: Code repository URLs aligned	L.Caldararu (SMOB)
0.4	07/10/2019	GRA contribution: Recommendation Engine update	K. Tarjányi (GRA)
0.5	09/10/2019	SUPSI contribution: Disaggregation Engine update	M. Derboni (SUPSI)
0.6	09/10/2019	SMOB contribution: platform database update	L.Caldararu (SMOB)
0.7	11/10/2019	CERTH contribution: Inference Engine update	S.Krinidis (CERTH)
0.8	14/10/2019	CERTH contribution: Energy Efficiency Console Platform for Building Managers update	S.Krinidis (CERTH)
0.9	15/10/2019	POLIMI contribution: Awareness Application, Gamification engine, Digital Game Extension	S. Herrera (POLIMI)
1.0	25/10/2019	Final quality check	P. Fraternali (POLIMI)
1.1	06/04/2020	Revised version to address Reviews Request	L.Caldararu (SMOB) S. Herrera(POLIMI)

Disclaimer

This document contains confidential information in the form of the enCOMPASS project findings, work and products and its use is strictly regulated by the enCOMPASS Consortium Agreement and by Contract no. 723059.

Neither the enCOMPASS 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 contents of this document are the sole responsibility of the enCOMPASS consortium and can in no way be taken to reflect the views of the European Union.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 723059.

Table of contents

EXECUTIVE SUMMARY

This is the **accompanying document** of the Deliverable **D6.5: Platform Final Prototype**, which, according to the Description of Action (DoA), is:

Software deliverable of the Final Prototype of the enCOMPASS platform, which will provide all the functions specified in WP2. Source code and documentation for the use of the 3rd party APIs and Software Development Kit to expand the platform.

Beyond the features included in the First Prototype of the enCOMPASS platform (the infrastructure to collect and organize the energy consumption and sensor data from end-consumers and public buildings, with the first integration of the user interfaces) and in the Second Prototype (advanced gamified interfaces integrated with the energy efficiency game and connected with the information systems of the energy utilities), the Final Prototype includes component releases and upgrades that have been rolled out on the Pilot sites of the case studies SES (Locarno, Switzerland), SHF (Hassfurt, Germany) and WVT (Athens, Greece):

- Final version of the Digital Extension game;
- Final version of the Energy Efficiency game;
- Final version of the Disaggregation Engine component;
- Final version of the Inference Engine component;
- Final version of the Recommendation Engine component;
- Final version of the Integration Services;
- Final version of Awareness Application for web and mobile access (Google and iOS versions);
- Final version of Energy Efficiency Console for Buildings;
- Final version of the Board Game (offline component).

The software making the Final Prototype of the enCOMPASS platform has been deployed on the Pilot sites and is available on the Internet for all the case studies.

While the deliverable itself is made by the source code of the software components making the Final Prototype of the platform and the documentation for using it, the current document is an accompanying text providing instructions for the installation and running of the Final Prototype of the enCOMPASS platform.

For a better understanding of this document and of the software deliverable, the following deliverables can be consulted:

- *D6.1 Delivery management plan and testing specification*: contains a description of the software development process; code management tools; documentation to set up the development and testing environments.
- *D6.2 Platform architecture and design*: describes the architectural design of the enCOMPASS platform: the information and data models, platform components, services, and applications, communication protocols. Integration model.
- *D6.3 Platform Initial Prototype*: software deliverable including the source code of the Initial Prototype of the enCOMPASS platform providing the infrastructure to collect energy consumption data, the first integration of the user interfaces designed in WP2, for early testing in the Pilots as well as the installation documentation provided in the accompanying document.
- *D6.4 Platform Second Prototype*: software deliverable including the source code of the Second Prototype of the enCOMPASS platform providing advanced gamified interfaces and integrated with the energy efficiency game and connected with the information systems of the energy utility.

1 enCOMPASS PLATFORM – FINAL PROTOTYPE

1.1 STATE OF THE DEVELOPMENT PROCESS

In the context already set by the Initial Platform Prototype and Second Platform Prototype, during the development of the Platform Final Prototype the software developments were continuously deployed and tested on the development and testing environment, using a server provided and managed by SMOB. The development and testing environments have been continuously provisioned with energy consumption and sensor data coming from SES and PDX through a secured FTP connection using an automated nightly job.

The Final enCOMPASS Platform Prototype has been deployed on three production software environments hosted by SES in Locarno, Switzerland, SHF in Hassfurt, Germany and WVT in Athens, Greece. Prior to the launch into production, all Final Prototype deployments have passed standard incremental tests by undergoing: alpha tests using pools of selected users known to the developers, and beta tests with real-world users that accepted to provide feedback for the enCOMPASS project.

At the same time, during the project phases the development environment has been continuously maintained for deploying and testing new platform features, security configurations and demonstrations of the enCompass platform for various prospects or industry events. In addition of using the development server, software deployments have been performed using the partners' own infrastructure, as for example in the cases of CERTH's Energy Efficiency Console for Building Managers and SUPSI's Disaggregation Engine.

In the UML representation of the Component Diagram from Figure 1 all the components are represented in green as they have been completed.

The main components of the integrated enCOMPASS platform developed and installed for the first platform prototype and for the second platform prototype were:

- enCOMPASS platform database;
- Smart Meter and Sensor Data Management (data ingestion component);
- Inference Engine;
- Gamification Engine;
- Efficiency Exploration Console for Building;
- Notification Engine;
- Service Integration and Semaphorization (orchestration component);
- Awareness Applications;
- Efficiency Exploration Console for Building Managers;

With respect to the Second Platform Prototype, the Final Platform Prototype provided updates to the following components:

- Update of the **Smart Meter and Sensor Data Management** component;
- **Inference Engine** component**;**
- **Disaggregation Engine** component;
- **Digital Extension game**;
- **FUNERGY - Energy Efficiency game**;
- Update of the **Recommendation Engine** component**;**
- Update of the **Integration Services**;
- **Awareness Application for web and mobile access** (Google and iOS versions);
- **Energy Efficiency Console for Building**;
- Update of the **Database Model;**

- **Board Game** (offline component).

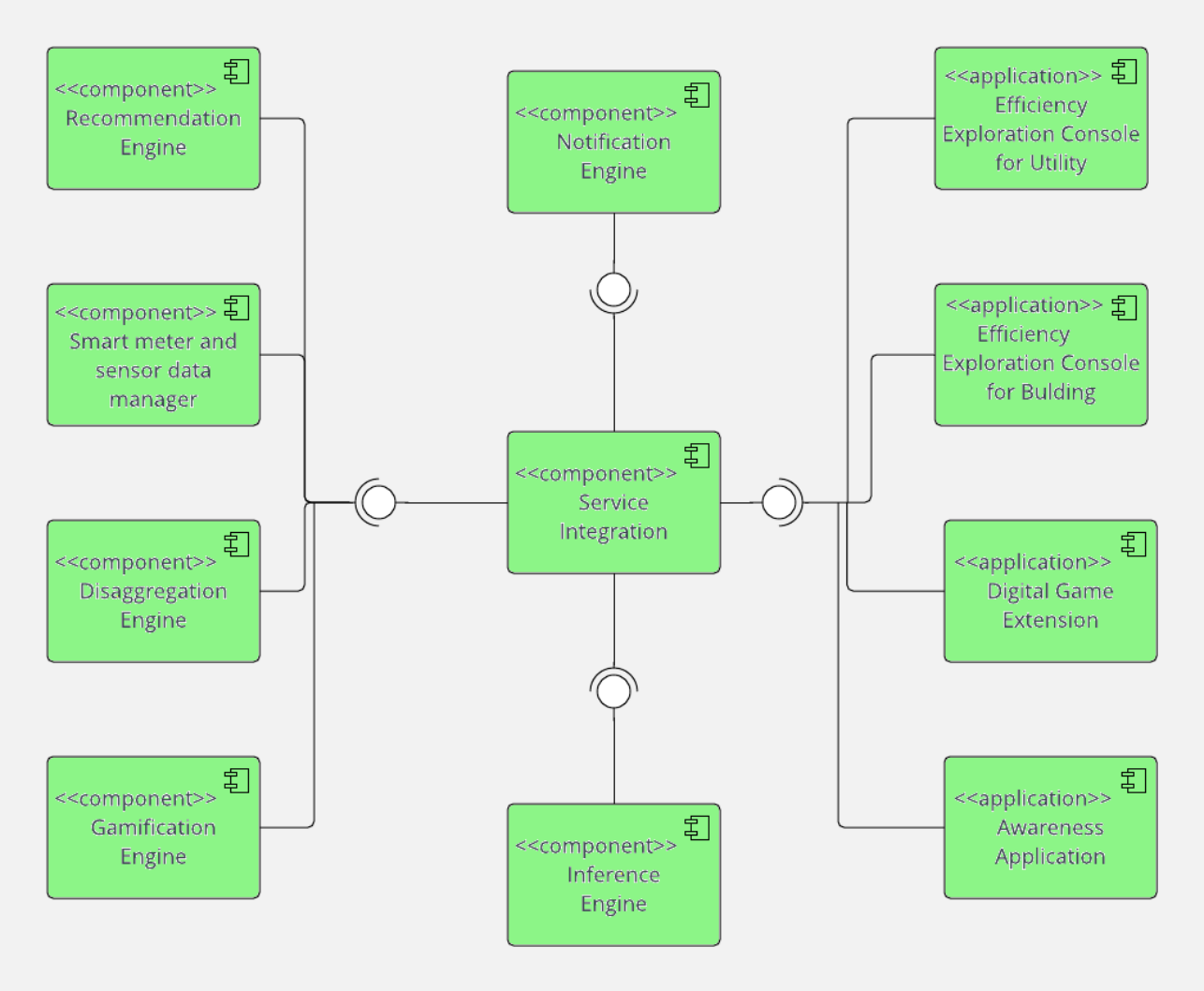

Figure 1 – Overview of the main components of the enCOMPASS architecture and their state of development for the Final Prototype.

The final prototype of the enCOMPASS platform is available on the internet in the form of a demo installation, such installation is based on the Greek Pilot which includes all the features of the platform. The Awareness Application is the central access point for the users to access enCOMPASS platform, it is available as follows:

The enCOMPASS demo is available at https://myencompass.watt-volt.gr/

Testing credentials:

1. username: wtv.encompass
2. password: *password*

2 INSTALLATION GUIDE AND SOFTWARE DEVELOPMENT KITS

The software sources referred in the following sections are available as public projects on the enCOMPASS project account on the project development Git server. They can be downloaded and installed from the following addresses:

ssh://18.184.32.200:22/var/git/encompass-database.git

ssh://18.184.32.200:22/var/git/encompass-service-integration.git

ssh://18.184.32.200:22/var/git/AA_android_app.git

ssh://18.184.32.200:22/var/git/AA_ios_app.git

ssh://18.184.32.200:22/var/git/encompass-inference-engine.git

ssh://18.184.32.200:22/var/git/gamification_engine.git

ssh://18.184.32.200:22/var/git/GamificationEngine_executable.git

ssh://18.184.32.200:22/var/git/gamification_engine_db.git

ssh://18.184.32.200:22/var/git/funergy_db.git

ssh://18.184.32.200:22/var/git/funergy_portal_executable.git

ssh://18.184.32.200:22/var/git/funergy_engine.git

ssh://18.184.32.200:22/var/git/funergy_app.git

2.1 enCOMPASS DATABASE

The **enCOMPASS Database** is the central repository of the information that is either common to all the enCOMPASS components or supports the coordination and exchange of messages among them. Not all the data of enCOMPASS will reside in the enCOMPASS database; for example, commercial data about the energy consumers maintained by the Energy Utilities are stored in the proprietary systems of the respective companies.

In the platform database of the Final Prototype have been included new objects and updates of the objects used in the First and in the Second platform prototype, as the data model was continuously updated according to the platform development plan and feedback from the testing sites.

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

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

The updated script file for creating and initializing the enCOMPASS platform database with a testing data set is available on the Git project account at:

ssh://18.184.32.200:22/var/git/encompass-database.git

2.2 SMART METER AND SENSOR DATA MANAGEMENT OR DATA INGESTION COMPONENT

The **Smart Meter and Sensor Data Manager** (SMSDM) component deals with the acquisition of data streams from smart meters and with their consolidation within the enCOMPASS 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.

For accomplishing its task of provisioning data to enCOMPASS platform, SMSDM component process smart meter and sensor readings, uploaded as CSV files by the Energy Utility through SFTP and ingests these data into the platform database. The data ingestion may occur periodically e.g. hourly, daily, weekly, monthly or at any moment the CSV files containing readings are available.
The SMSDM component:

- Retrieves user consumption data from smart meters for past time intervals;
- Retrieves data from sensors (presence, luminosity, temperature, humidity, etc.) for past time intervals.

It is implemented using Apache Camel routes and 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 data integration projects.

The version of the SMSDM component delivered in the Platform Final Prototype include updates of the processing routes as well as a new route for processing smart plug consumption (detailed consumption data).

2.2.1 Requirements

- OS: Cent OS 7.0
- Java 8+: https://home.java.net/
- MySQL 5.1+: https://www.mysql.com/
- Apache Maven 3.5+: https://maven.apache.org/download.cgi
- SpringBoot: https://projects.spring.io/spring-boot/

2.2.2 Installation of the SMSDM component

For installing and testing the SMSDMC component, the following steps have to be performed:

1. **Route configuration**

The SMSDM (Data Ingestion) component is a SpringBoot project containing the Apache Camel libraries – an open-source Java-based framework focusing on data and service integration following a rule-based routing and mediation engine. The first requirement is the configuration of the routes the SMSDM component uses to process the consumption and sensor data files according to the local settings. The route configuration is performed by populating the table **route_config** from the central database of the enCOMPASS platform, as in the sample from Table 1: Populating **route_config** table

*Table 1: Populating **route_config** table*

route	source_path	error_path	md5_path	md...	outbox_path
importConsumption	/srv/ses/meterdata?readLock=rename&include=enCOMPASS_SES_consumption.*.csv...	/srv/ses/error/	/srv/ses/md5/	26	/srv/ses/out/
importCO2Data	/srv/ses?include=enCOMPASS_SES_co2.*.csv&sortBy=file:name	/srv/ses/error/	/srv/ses/md5/	26	/srv/ses/out/
importHumidityData	/srv/ses?include=enCOMPASS_SES_humidity.*.csv&sortBy=file:name	/srv/ses/error/	/srv/ses/md5/	26	/srv/ses/out/
importLuminanceData	/srv/ses?include=enCOMPASS_SES_illuminance.*.csv&sortBy=file:name	/srv/ses/error/	/srv/ses/md5/	26	/srv/ses/out/
importTemperatureData	/srv/ses?include=enCOMPASS_SES_temperature.*.csv&sortBy=file:name	/srv/ses/error/	/srv/ses/md5/	26	/srv/ses/out/
importMotionData	/srv/ses?include=enCOMPASS_SES_motion.*.csv&sortBy=file:name	/srv/ses/error/	/srv/ses/md5/	26	/srv/ses/out/

2. **Project deployment**
   a) Make a **git clone** of the master service project from **ssh://18.184.32.200:22/var/git/encompass-service-integration.git**
   b) Open a **Command Prompt** and go to the **encompass-data-ingestion** subfolder of the project
   c) Configure the **application.properties** file by ensuring the CAMEL PROPERTIES and ROUTE PROPERTIES are set according to local configuration, as in the following configuration sample:

```
#----------CAMEL PROPERTIES----------
camel.routes.import.consumption=importConsumption  ;energy consumption data ingesting route
camel.routes.import.co2=importCO2Data          ;CO2 sensor data ingesting route
camel.routes.import.humidity=importHumidityData   ;humidity sensor data ingesting route
camel.routes.import.luminance=importLuminanceData  ;illuminance sensor data ingesting route
camel.routes.import.temperature=importTemperatureData ;temperature sensor data ingesting route
camel.routes.import.motion=importMotionData      ;occupancy sensor data ingesting route
camel.routes.import.consumption.warning.null.percentage=10 ;percentage of reading lines having null values when to
trigger an alert for too many nulls
#----------/CAMEL PROPERTIES----------

#----------MySQL DATABASE PROPERTIES----------
spring.datasource.driver-class-name=com.mysql.jdbc.Driver ;MySQL DB driver
spring.datasource.url=jdbc:mysql://18.184.32.200:3306/encompass_model?useUnicode=yes&characterEncoding=UTF
-8&useSSL=false ;MySQL connection URL
spring.datasource.username=puser      ;MySQL username
spring.datasource.password=sj5$mfD&k37z1c;MySQL password
spring.datasource.test-while-idle=true  ;perform ping to avoid disconnection
spring.datasource.test-on-borrow=true   ;perform
spring.datasource.tomcat.max-active=10  ;number of tomcat simultaneous active connections
spring.datasource.max-active=10       ;number of simultaneous active connections
spring.datasource.tomcat.max-wait=10000  ;tomcat connection timeout
spring.datasource.max-wait=10000      ;connection timeout
spring.datasource.validation-query=SELECT 1 ;validation query
#----------/MySQL DATABASE PROPERTIES----------


#----------MAIL PROPERTIES----------
spring.mail.host=smtp.gmail.com              ;email server address
spring.mail.port=465                    ;email server port
spring.mail.protocol=smtps                ;email server protocol
spring.mail.username=ses.encompass@gmail.com      ;username for email authentication
spring.mail.password=**********           ;password for email authentication
spring.mail.properties.mail.transport.protocol=smtps ;email server protocol for spring
spring.mail.properties.mail.smtps.auth=true       ;email auth parameter for spring
spring.mail.properties.mail.smtps.starttls.enable=true ;email tls parameter for spring
spring.mail.properties.mail.smtps.timeout=2000    ;email timeout parameter for spring
application.email=encompass.no-reply          ;email address for alert originator
application.email.to=admin@encompass-project.eu    ;email address of the enCOMPASS platform administrator
#----------/MAIL PROPERTIES----------
```

d) Launch the command **mvn clean install** to build the project distribution
e) Deploy the project on the server
f) Launch the server
g) In the transfer path specified in the **route_config.source_path** upload CSV files containing consumption and sensor data. The SMSDM component automatically starts processing the file according to the ingestion rules.
h) Check the outcome of the SMSDM component in the following tables from the central enCOMPASS database: **semaphore_log, meter_consumption**, **indoor_conditions_humidity**, **indoor_conditions_temperature**, **indoor_conditions_luminanace**, **indoor_conditions_occupancy**.
i) Check the processed files in the paths specified in the **route_config.outbox_path** column, if the data ingestion has been successfully performed or respectively in the **route_config.error_path** column if the data ingestion failed.
j) Check the email account specified by **application.email.to** configuration variable to consult the success or fail alert message of the data ingesting processing. This alert message is meant for the platform administrator that is in charge of maintaining the data ingestion.

2.2.3 Access to the source code of the SMSDM component

The SMSDM component source code can be downloaded from the following repository:

ssh://18.184.32.200:22/var/git/encompass-service-integration.git

using the standard Git credentials, then importing the **encompass-data-ingestion** project.

2.3 DISAGGREGATION ENGINE

The **Disaggregation Engine** (DE) component processes aggregated consumption data and returns the estimated end-uses of the single devices in a household. The services provided by this component are the producer and the consumer.

The business logic and the code of DE for the final prototype is the same as the one used for the second release, since no changes were necessary for the third release. The following is a summary of the business logic and the main methods used, whose code is available in Appendix A, provides further documentation for the Disaggregation Engine Component.

2.3.1 Producer

The producer service is invoked by the API /de/runDisaggregation, which is responsible to ingest the data from the database, disaggregate the consumption data of the 2nd day before, and store the results in the database.

The data ingested by the producer from the database are the following:

- User profile data: the presence/absence of the electrical appliances (fridge, washing machine, tumble dryer, dishwasher, air conditioning, electric car, electric oven, heat pump). These data are retrieved from the encompass_model.user_profile table;
- Consumption measurements: the 24-hour consumption data of the 2nd day before
- Building information: used to check if the data are those of is a household. These data are retrieved from the encompass_model.building table.

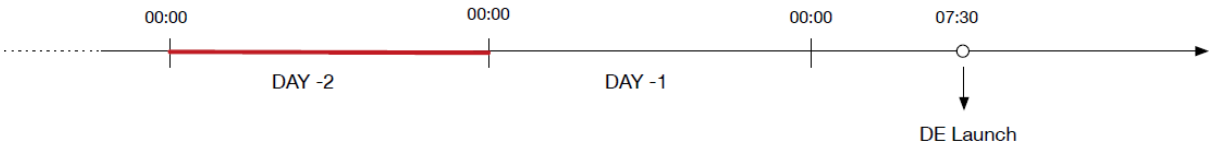

Figure 2 – In red, the data processed every day by the disaggregation engine

Every day these data are sent to the disaggregation process, provided by the SUPSI backend, through this HTTPS POST: https://encompass.idsia.ch/webscript/cgi-bin/disaggregation_engine. The results of the external service are stored in the encompass_model.disaggregation_data table.

This service is not fully integrated into the orchestration model of the encompass platform, because the solver of the optimization model requires more computational power. Hosting the service in a dedicated server avoids locking the orchestration services for a long time and improves the overall performances of the encompass platform. Moreover, since the DE service should not be invoked by the three instances of the enCOMPASS platform (one for each pilot) at the same time due to the hardware constraints, it was decided to schedule the

service sequentially with a cron job. The installation and the monitoring of the external service are provided by SUPSI on their own server.

2.3.2 Consumer

The consumer service provides the estimated end-uses of the single devices in a household. It provides for each appliance the mean of the disaggregated data in the last 30 days, starting the 3rd day before and the total consumption in that time window.

This service is invoked by the API /de/getDisaggregatedDataMean

2.3.3 Requirements

OS: Cent OS 7.0

Java 8+: https://home.java.net/

MySQL 5.1+: https://www.mysql.com/

Apache Maven 3.5+: https://maven.apache.org/download.cgi

SpringBoot: https://projects.spring.io/spring-boot/

2.3.4 Installation

For installing the DE component, the following steps must be performed:

Make a **git clone** of the master service project from **ssh://18.184.32.200:22/var/git/encompass-service-integration.git**

Open a **Command Prompt** and go to the **encompass-component-services** subfolder of the project.

Configure the **application.properties** file according to local configuration, as in the following configuration sample:

```
#--------------DE PROPERTIES---------------------------------
supsi.de.username=idsia    ; disaggregation producer service username
supsi.de.password=**********     ; disaggregation producer service password
supsi.de.url=https://encompass.idsia.ch/webscript/cgi-bin/disaggregation_engine     ; disaggregation producer link
#--------------/DE PROPERTIES----------------------
```

Launch the command **mvn clean install** to build the project distribution.

Deploy the project distribution to the server.

Launch the server.

2.3.5 Testing

After successfully deploying the Service Integration and Semaphoring component, the Swagger console is available at the URL http://<IP>: 8081/swagger-ui.html . For the development and testing server, the Swagger console is available at http://18.184.32.200:8081/swagger-ui.html

2.3.6 Source code

The DE component source code can be downloaded from the following repository:

ssh://18.184.32.200:22/var/git/encompass-service-integration.git

using the standard Git credentials, then importing the **encompass-component-services** project.

2.4 INFERENCE ENGINE

The inference engine (IE) is composed of the occupancy and activity inference tool, of the user comfort levels inference tools, and of the scheduler of comfort feedback notifications.

2.4.1 Occupancy and Activity Inference Tool

The Occupancy and Activity Inference Tool is responsible for estimating the occupancy (presence or absence) and the activity performed in a building (e.g. cooking, washing, sleeping, etc.).
The tool takes as input the raw data from the sensors (i.e. the energy consumption and the indoor environmental conditions) and the disaggregated information (i.e. detailed information regarding the individual device energy consumption). These data are analyzed, while a probabilistic model (HMM) is utilized. This model is able to infer if space is occupied or not, as well as the kind of activity performed at a given time interval.

2.4.2 User Comfort Levels Inference Tool

The Comfort Levels Inference Engine is responsible for estimating the subjective comfort level of the occupants in a building/ space. Both thermal and visual comfort levels. The engine requires as input indoor environmental conditions, i.e. temperature, humidity and luminance, for the estimation of the users' comfort levels. However, users can provide their personalized feedback about their actual thermal and visual comfort feelings. This feedback is utilized by the engine, in order to retrain the models and make precise calculations for their clothing and metabolic rate values at different time intervals, revising the initial assumptions (ASHRAE model) and converging gradually to the personalized values.

2.4.3 The scheduler of comfort feedback notifications

The Scheduler of comfort feedback notifications is a tool that regulates when notifications will be sent to users, engaging them to provide information about their comfort levels feeling. The main scope of this tool is to gather users' personalized information, without irritating them.

2.4.4 Requirements

1. MySQL DBMS 5.6 or newer (link)
2. Java 1.8.0 or later (link)
3. Apache Maven 3.5.3 or newer
4. Spring Boot

2.4.5 Installation and execution

1. Download the project from ssh://18.184.32.200:22/var/git/encompass-inference-engine.git and store it in a folder of your choice.
2. In the files that regard the pure components (in Scheduler.java, for example) change the URL of the database and its access credentials (username and password) at your own will.
3. Open a terminal on your computer, and change to the folder where you have stored the project.
4. Build the project by writing `./mvnw clean package`
5. Run the Project by writing `./mvnw spring-boot:run`, which starts Tomcat on port 8083 and the Spring Boot Application.

2.4.6 Testing

This project contains a POST web service, which can be called with any sample JSON as input (even no input at all is accepted). In order to run the project, the Tomcat server should start on port 8083, and call the following URL http://{myIP}:8083/ie/component/external/start, where "{myIP}" is the IP address of the machine where the project runs. When this service is called, then the IE components will be executed.

The IE component source code can be downloaded from the following repository:

ssh://18.184.32.200:22/var/git/encompass-inference-engine.git

using the standard Git credentials.

The business logic of the Inference Engine included in the Final prototype was mainly delivered already for the Second release, while during the Final release minor updates and parameterizations have been performed.

2.5 GAMIFICATION ENGINE

The **Gamification Engine** is a component that listens to the actions of the users and transforms them into a variety of rewards, for improving activity and participation. For the Final Prototype of the enCompass platform no changes were made to this component business logic. Nevertheless, for this release, a number of rules have been added by using the component configurator.

The definition of points, achievements and rewards allow energy utility companies to challenge the customers in their pilot with energy-saving goals to be achieved each month, to achieve individual energy consumption reduction. The **Gamification Engine** is also used as a mean for raising energy consumption awareness by promoting sustainable behaviour.

The Gamification Engine has been developed using WebRatio, a platform for the development of web and mobile applications based on the OMG standard IFML.

2.5.1 Requirements

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

2.5.2 Installation

1. **DB Installation.** Create a new database "community" and import the sql file community.sql from ssh://18.184.32.200:22/var/git/gamification_engine_db.git
2. **Application installation on Tomcat**.
   - Unzip the ssh://18.184.32.200:22/var/git/gamificationEngine_executable.git 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://18.184.32.200:8083/";
var local_host_url = "http://localhost:8080/"; tomcat address
```

```
services_host_url=http://18.184.32.200:8083/si  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, typically "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
```

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

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

- Grant "read" and "write" permission to Tomcat to the entire folder **community.** For UNIX:

```
sudo chown -R tomcat7 community/

sudo chgrp tomcat7 community/
```

- Start Tomcat

2.5.3 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:
  Advanced profile:

  username: Chiara

  password: password

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

  username: admin

  password: password

2.5.4 Source code

The WebRatio projects can be downloaded from: ssh://18.184.32.200:22/var/git/gamification_engine.git and can be imported into the WebRatio IDE, the following project are required and need to be added:

- Authentication
- BootstrapStyleRarolab
- Gamification

- GamificationBackEndStyle
- GamificationCustomRarolab
- GamificationFrontEndStyle
- NotificationMessageSample

The business logic of the Gamification Engine included in the Final prototype was mainly delivered already for the First and the Second release, while during the Final release minor updates have been performed.

2.5.5 Configuration

The Gamification Engine has been designed to be configurable, the rules and the actions are stored in the gamification database which allows changing the rules dynamically without modifying the code or restarting the component.

The configuration can be changed through the Gamification Administration Portal that can be accessed using the following URL:

```
https://localhost:< your_port >/community/admin
```

Default credentials:

username: admin

password: password

The Administration portal allows the configuration of the following aspects:

- Thematic Areas
  - Create/Remove Thematic Areas
  - Modify existing Areas (Name, Description, Logo, etc.)
- Gamification actions
  - Create/Remove actions
  - Modify existing Actions (Awarded points, repeatability, etc.)
- Badges
  - Create / Remove Badges
  - Modify existing Badges (Name, description, image, required score, etc.)
  - Add badge Levels
- Rewards
  - Create/ Remove Rewards
  - Modify existing rewards (Name, description, image, required points, availability, etc.)
- Thematic Area – Action relationship
  - Add/Remove/Update the set of actions that contribute to the progress of the different thematic areas and badges.
- Goals
  - Change the default target goal
- Users
  - Manage user with administration profile.
- Localization
  - Add translated labels for all the described elements.

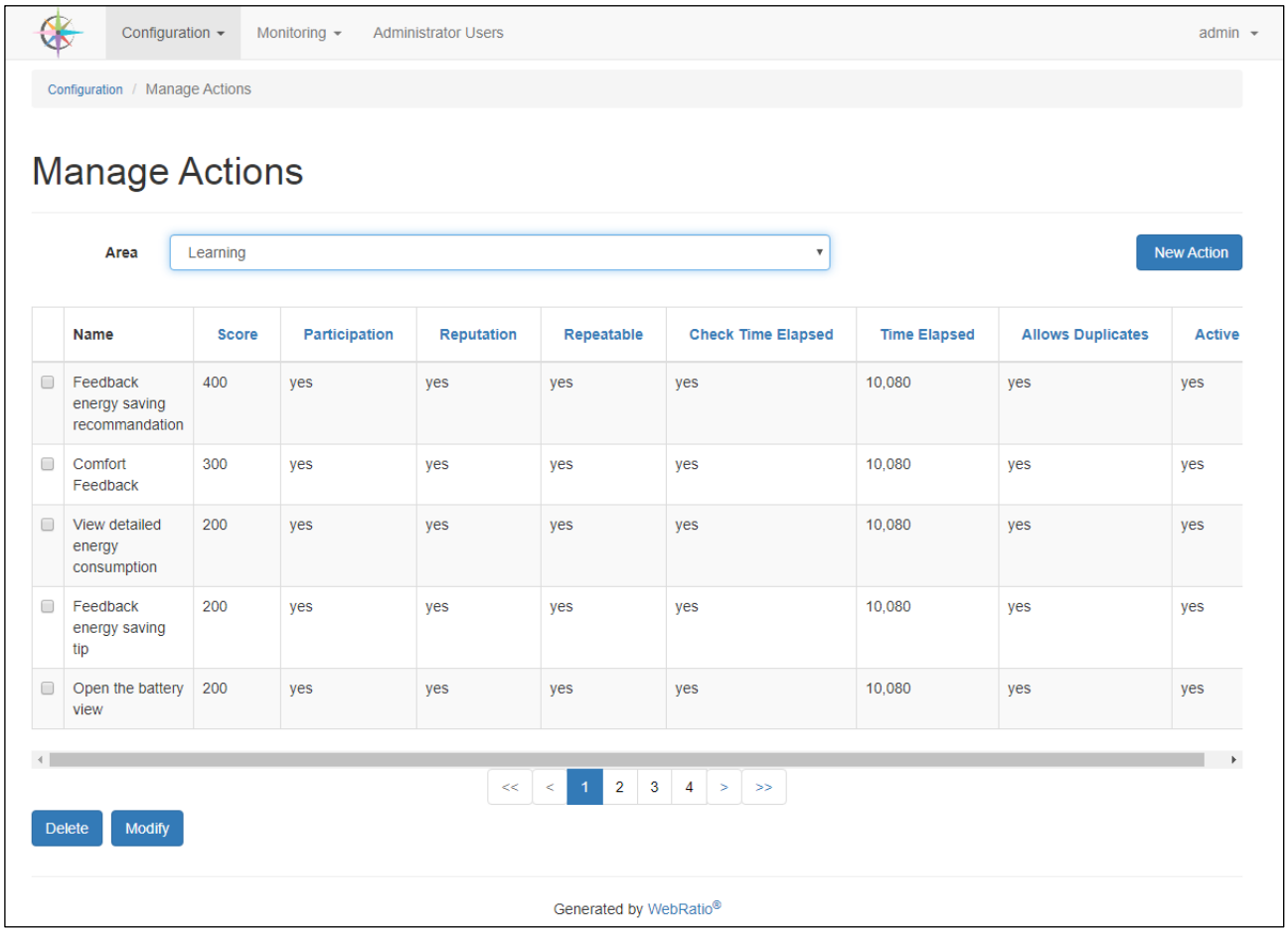

Figure 3 - Gamification Administration Portal – Manage Gamification actions for Thematic Area.

The configured rules currently stored in the Gamification DB script correspond to the enCOMPASS needs and behavioural change strategy, described on Deliverables D5.3 and D5.5. Third-party developers can use the administration feature to adapt the rules and incentives to their own strategy and requirements.

2.6 Energy Efficiency Console Platform for Utility

This component performs the monitoring of consumption behaviour and household characteristics. It is included in the Gamification Engine project as a user access role. The installation of this component is already performed when installing the Gamification Engine component.

2.7 Recommendation Engine

As it was originally planned, this component produces action recommendations based on disaggregated consumption and sensor data. Proposed recommendations are sent to the Central database to be further sent to End User by Notification Engine.

The Recommendation Engine ingests the following data available in the Platform:

- consumption and sensor data from the pilots:
  - By the end of the project (by setting up various sensors), type and amount of available data increased compared to R1. There are different sensors in Germany and Switzerland (luminance, motion, temperature, humidity) and in Greece (temperature, humidity);
- user profile data:
  - information about residents: number of adults, kids and pets
  - features of the house: number of rooms, type of dwelling (e.g. single-family house, apartment) and heating source (electricity, oil, gas, wood)

    - information about devices in the household: presence or absence of several device types
    - responses to the first and intermediate questionnaires;
- data from the Disaggregation Engine
- data regarding the thermal and visual comfort of the users
- data from Activity Inference Engine
- user actions on the Platform including user responses to Recommendation Engine suggestions, the so-called, Feedbacks.

The more data is available, the more accurate the RE is and the more specific are the texts of the recommendations it seems more personalized and useful for the users. During the project both the quantity and quality of the processed data was increased by the pilot partners, however RE would have performed even better in bigger – e.g. countrywide - experiment. In order to increase user experience and the usefulness of the recommendations aimed to better off the logic and the texts of the recommendations during the project. Therefore, in R3 the partners issued a bigger (and more personalized) set of texts for recommendations.

In relation to more details regarding the logic of RE, please check *D4.4. Final User Behavior and Recommender*. Final logic of the RE was elaborated for R3 and described in this document.

In terms of the technical implementation of RE almost nothing has changed since the previous release. For the better understanding of the platform, we keep here the structural description of the RE with minor modifications in the following text (comparing to *D6.4. Platform Second Prototype*)

Recommendation Engine backend **is a SaaS like service**; thus, it is not an integral part of the enCOMPASS platform. Its installation and monitoring are provided by Gravity on their own server.

The Recommendation backend in high level is an API-less service which takes input in the form of an uploaded directory of database exports from the enCOMPASS platforms, calculates recommendations for the next day on it, and then sends next day's recommendations to the enCOMPASS platform in the form of an API call to the postRecommendationList.

Internally the Recommendation backend restricts reruns of these tasks to once a day in order to prevent recalculation and resending the data (data import is done in every case).

As a result of the orchestration, the cronjob that is running the Recommendation Engine is running every minute, which will prevent large delay between data export from the enCOMPASS platform, and the data import in the Recommendation Engine.

Also, to make orchestration work in every case, the Recommendation Engine backend is implemented in a way that even when the calculation did not run (because of the calculation once per day restriction), the call to the API which starts the next task in the orchestration will be called.

2.7.1 Installation and configuration

For every enCOMPASS platform installation (development, and pilot instances), a new Recommendation Engine needs to be installed and deployed. These instances are deployed in Gravity cloud and are configured, operated, and maintained by the Gravity's operation team.

Configuration of the Recommendation Engine backend is also internal, any changes to it (like changing server address, cron timers, etc.) must be done by Gravity operation team.

2.7.2 Testing

Because there is no real trigger mechanism (as there is no separate API), the Recommendation Engine backend is running as a cronjob on the server and will check the existence of a [date], go directory into the SFTP directory. If it exists, it will import the data, calculate, and send the recommendation.

The state of the import can be checked externally by connecting to the SFTP server (credentials has been provided in a separate encrypted archive) and checking the state of the data import (not the whole recommendation calculation process). The process has four states:

- [date].go: the Recommendation Engine not yet started to process this directory
- [date].processing: the Recommendation Engine is importing the data
- [date].failed: the Recommendation Engine is failed to import the data
- [date].finished: the Recommendation Engine is finished with the data import

2.7.3 Source code

The RE SaaS's Service Integration source code is available from the following repository:

ssh://18.184.32.200:22/var/git/encompass-service-integration.git

using the standard Git credentials, then importing the **encompass-component-services** project.

2.8 NOTIFICATION ENGINE

This component is responsible for sending PUSH notifications to End User Client application, such as the Awareness Application. It reads messages to be sent from the Central Database and queues them for sending. Any Platform component that needs to send a message needs to store the message in a message table specifying the destination application (endpoint), message content and the schedule.

Minor software optimizations have been included in the Platform Final Prototype, while no business logic updates have been performed.

2.8.1 Requirements

- OS: Cent OS 7.0
- Java 8+: https://home.java.net/
- MySQL 5.1+: https://www.mysql.com/
- Apache Maven 3.5+: https://maven.apache.org/download.cgi
- SpringBoot: https://projects.spring.io/spring-boot/
- ActiveMQ 5.15+: http://activemq.apache.org/download.html

2.8.2 Installation

For installing the NE component, the following steps have to be performed:

1. **Project deployment**

   a) Make a **git clone** of the master service project from **ssh://18.184.32.200:22/var/git/encompass-service-integration.git**
   b) Open a **Command Prompt** and go to the **encompass-component-services** subfolder of the project
   c) Configure the **application.properties** file according to local configuration, as in the following configuration sample:

```
#---------------ACTIVEMQ---------------------------------------
spring.activemq.in-memory=false              ;parameter
spring.activemq.pool.enabled=false           ;parameter
```

```
activemq.broker.url=tcp://localhost:61616        ;message broker connection  protocol
activemq.queue=queue://                          ;internal storage structure
activemq.queue.low=notification.lowpriority.queue   ;low priority queue id
activemq.queue.high=notification.highpriority.queue
activemq.queue.onthefly=notification.onthefly.queue
#--------------/ACTIVEMQ-------------------------------------

#--------------MySQL DATABASE PROPERTIES----------------------
spring.datasource.driver-class-name=com.mysql.jdbc.Driver ;MySQL DB driver
spring.datasource.url=jdbc:mysql://18.184.32.200:3306/encompass_model?useUnicode=yes&characterEncoding=UTF
-8&useSSL=false ;MySQL connection URL
spring.datasource.username=puser       ;MySQL username
spring.datasource.password=sj5$mfD&k37z1c;MySQL password
spring.datasource.test-while-idle=true   ;perform ping to avoid disconnection
spring.datasource.test-on-borrow=true    ;perform
spring.datasource.tomcat.max-active=10   ;number of tomcat simultaneous active connections
spring.datasource.max-active=10        ;number of simultaneous active connections
spring.datasource.tomcat.max-wait=10000  ;tomcat connection timeout
spring.datasource.max-wait=10000       ;connection timeout
spring.datasource.validation-query=SELECT 1 ;validation query
#--------------/MySQL DATABASE PROPERTIES----------------------


#---------------FIREBASE PROPERTIES----------------------------
firebase.api.key=AAAA1Nqhg9E:APA91bHSIFc7WoQKNkgCF1AhvTd_nncEO3M3GIQEtXRm-
nVP7YRqMcPwjTfrcLvk72mOkvF5DTw4JYuAmTKbXuvq_ozx7OtAizovlUOH-Oyeo8yoRjPoVNyjzTdCspWQeTNzEkre-
Ts3   ;Firebase project API Key
firebase.api.project-id=914201084881  ;Firebase project ID
firebase.endpoint.operation.group=https://android.googleapis.com/gcm/notification   ;endpoint for creating Firebase
group
firebase.endpoint.operation.message=https://fcm.googleapis.com/fcm/send ;endpoint for sending notification
firebase.prefix.group=encompass-     ;Firebase group prefix
#--------------/FIREBASE PROPERTIES----------------------------
```

d) Launch the command **mvn clean install** to build the project distribution
e) Deploy the project distribution to the server
f) Launch the server

2.8.3 Testing

After successfully deploying the NE component, the ActiveMQ console is available for testing at the URL http://<IP>:8161/ . For the development and testing server, the ActiveMQ console is available at http://18.184.32.200:8161/admin/

2.8.4 Source code

The NE component source code can be downloaded from the following repository:

ssh://18.184.32.200:22/var/git/encompass-service-integration.git

using the standard Git credentials, then importing the **encompass-component-services** project.

The business logic of the Notification Engine included in the Final prototype was mainly delivered already since the first release, as this component was critical in the functioning of the enCOMPASS platform.

2.9 Service Integration and Component Orchestration (Semaphorization)

Service Integration component manages all interaction between Platform components and between Platform component and Central Database. All interactions are achieved only through Web Services enlisted in this component.

With respect to the First Platform Prototype and the Second Platform Prototype, the Final Prototype of the Platform contains updated versions of existing integration services.
As specified in the accompanying document of the Second Platform Prototype, also, Component Orchestration (Semaphorization) is a service that performs a chained loop of synchronized calls to the Smart Meter and Sensor Data Ingestion component, Inference Engine, Recommendation Engine and Notification Engine, on daily bases. Although it is part of the daily processing flow, the Disaggregation Engine is not part of the component orchestration. Given the fact that Disaggregation Engine process very large amount of data (energy consumption, temperature, humidity, illuminance, building occupancy) at each call, it is independently called by a cron job while its outcome is joined to the outcome of the other components. Therefore, the Disaggregation Engine call is not part of the Component Orchestration (Semaphorization) service.

2.9.1 Requirements

- OS: Cent OS 7.0
- Java 8+: https://home.java.net/
- MySQL 5.1+: https://www.mysql.com/
- Apache Maven 3.5+: https://maven.apache.org/download.cgi
- SpringBoot: https://projects.spring.io/spring-boot/

2.9.2 Installation

For installing the Service Integration and Semaphoring component, the following steps have to be performed:

1. **Project deployment**

   a) Make a **git clone** of the master service project from **ssh://18.184.32.200:22/var/git/encompass-service-integration.git**
   b) Open a **Command Prompt** and go to the **encompass-component-services** subfolder of the project
   c) Configure the **application.properties** file according to local configuration, as in the following configuration sample:

```
#---------------SEMAPHORE---------------------------------------
semaphore.component.internal.url=http://18.184.32.200:8081/%s/component/start ;endpoint for starting the next component within the component orchestration
semaphore.component.ie.url=http://18.184.32.200:8083/ie/component/external/start ;endpoint for starting the Inference Engine component within the component orchestration
semaphore.post_component_subprocess_result_url=http://18.184.32.200:8081/si/semaphore/subprocess/result ;endpoint for saving a subprocess result within the component orchestration
semaphore.post_component_processing_finalization_url=http://18.184.32.200:8081/si/semaphore/result ;endpoint for saving a process finalization and return the control to semaphore within the component orchestration

#---------------/SEMAPHORE---------------------------------------

#--------------MySQL DATABASE PROPERTIES-----------------------
spring.datasource.driver-class-name=com.mysql.jdbc.Driver ;MySQL DB driver
spring.datasource.url=jdbc:mysql://18.184.32.200:3306/encompass_model?useUnicode=yes&characterEncoding=UTF-8&useSSL=false ;MySQL connection URL
spring.datasource.username=puser ;MySQL username
spring.datasource.password=sj5$mfD&k37z1c;MySQL password
spring.datasource.test-while-idle=true ;perform ping to avoid disconnection
spring.datasource.test-on-borrow=true ;perform
spring.datasource.tomcat.max-active=10 ;number of tomcat simultaneous active connections
spring.datasource.max-active=10 ;number of simultaneous active connections
spring.datasource.tomcat.max-wait=10000 ;tomcat connection timeout
spring.datasource.max-wait=10000 ;connection timeout
spring.datasource.validation-query=SELECT 1 ;validation query
#--------------/MySQL DATABASE PROPERTIES----------------------

#---------------SWAGGER PROPERTIES-----------------------------
```

```
swagger.info.title=ENCOMPASS COMPONENTS API
swagger.info.description=ENCOMPASS COMPONENTS API
swagger.info.version=v0.1
swagger.info.terms=Terms
swagger.info.contact.name=enCompass
swagger.info.contact.url=http://www.encompass-project.eu
swagger.info.contact.mail=contact@encompass-project.eu
swagger.info.license=enCompass License
swagger.info.license.url=http://www.encompass-project.eu
#--------------/SWAGGER PROPERTIES--------------------------
```

d) Launch the command **mvn clean install** to build the project distribution
e) Deploy the project distribution to the server
f) Launch the server

2.9.3 Testing

After successfully deploying the Service Integration and Semaphoring component, the Swagger console is available at the URL http://<IP>: 8081/swagger-ui.html . For the development and testing server, the Swagger console is available at http://18.184.32.200:8081/swagger-ui.html

2.9.4 Source code

The Service Integration and Semaphoring component source code can be downloaded from the following repository:

ssh://18.184.32.200:22/var/git/encompass-service-integration.git

using the standard Git credentials, then importing the **encompass-component-services** project.

2.10 AWARENESS APPLICATIONS

The enCOMPASS platform enables its users to track their energy consumption and obtain useful information that helps them to improve their energy consumption efficiency while keeping it comfort level and having fun with gamified activities.
Such interaction with the user happens through a web-based application and a mobile client (available for android and iOS). These components provide to end-user the following functions:

- Consumption data tracking
- Energy end-use visualization
- Energy reduction goal setting and tracking
- Tips and personalized recommendations assigning
- Semi-automatic execution of personalized recommendations
- Achievement tracking and leaderboard visualization
- Energy-saving impact metaphor visualization
- Visual and thermal comfort visualization and feedback
- Event notification delivery

The Awareness Application interacts directly with the Gamification Engine to provide the Gamification functionalities, and with the Service Integration component to retrieve the consumption, comfort, and awareness information (tips and recommendations).

The Platform Final Prototype includes a new feature the semi-automatic execution of recommendations. For the semi-automatic execution of recommendations, the application communicates directly with the utility smart-control devices through WIFI and the smart-home gateway. This communication is configured within the app

manually by the user so that the app can then later automatically communicate with the devices selected by the user. Thereby every such communication to execute a recommendation is explicitly confirmed by the user.

The mobile clients' code can be found in the following repositories:

- Android: ssh://18.184.32.200:22/var/git/encompass-androidApp.git
- iOS: ssh://18.184.32.200:22/var/git/encompass-ios.git

2.10.1 Requirements

Android:

- Android 3.0 or later
- Java JDK 1.7 or later
- GIT 2.0 or later

iOS:

- macOS 10.13.2 or later
- XCode 9.1
- GIT 2.0 or later

2.10.2 Installation of the Android project

In order to get the application code into Android Studio, create a new project, from the Menu File, select New – Project from Version Control – Git.

On the pop-up window, provide the above URL as the Git Repository URL, and select a path to save the project as in Figure 3. Click clone and provide the required user and password when prompt.

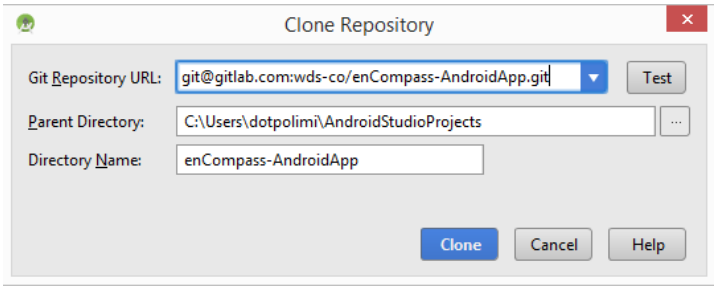

Figure 4 – Cloning the Awareness Application repository

Once the cloning process is finished, the build process will execute automatically, in case it does not it can be started manually by going to the menu Build –> rebuild.

2.10.3 Testing the Android app

Now the application can be tested in 2 ways:

1. Running the app on a connected device, as seen in Figure 4, or on a device Emulator: Go the menu Run –> run, a window will display a list of available devices (physical or virtual), select one and click ok. Android Studio will compile, pack and install the application on the selected device, the application will execute once the installation process is finished.

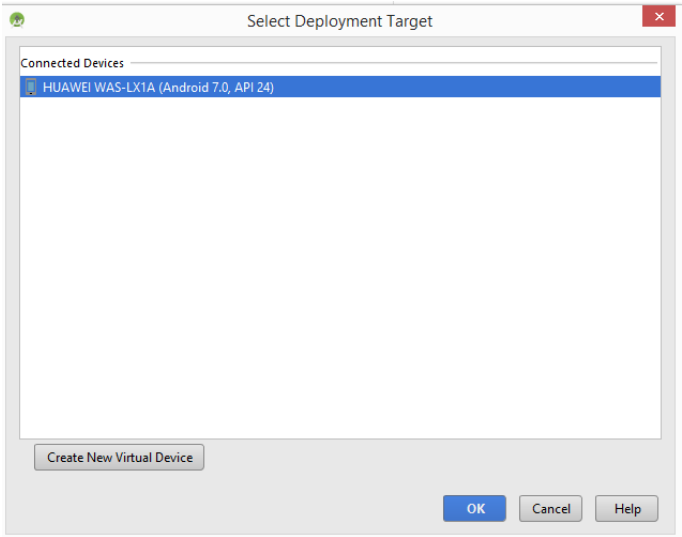

Figure 5 – Running the Awareness Application on a connected device

2. Generate an APK and install manually on a device: Android studio can generate and Android Package for the application, to do so go to the menu Build –> Build APK(s). The apk file will be created automatically on the /app/build/outputs/apk folder. Copy the file to the testing device, and click on it, a pop-up menu will ask to trust the application and allow installation from unknown sources, click on accept on the device. The application will be installed and can be executed.

Notice that the application was built to support android API level 25 (Android Nougat 7.1) and minimum supported API level is 21 (Android Lollipop 5.0) which provides a wide range of supported devices, this information can be found on the build file (/app/build.gradle) but changing it would result in compilation errors.

Additionally, the application uses Firebase Cloud Messaging Services, which are automatically configured by android studio base on the content of the file google-services.json, located on the /app folder, Firebase services will be detail in the following sections.

2.10.4 Android Client Configuration

In order to configure the Android client to work with different backend servers the following files need to be changed:

String.xml:

The file /app/src/main/res/values/String.xml contains the list of available providers, in order to add and remove. In the property “provider_array”:

```
<string-array name="provider_array">
    <item>Service provider</item>
    <item>SES</item>
    <item>SHF</item>
    <item>WTV</item>
</string-array>
```

Config.properties:
This file contains the configuration of the servers corresponding to the previous list, therefore they should be aligned. The file is located in: /app/src/main/assets/config.properties.

The properties represent the host server of the web application, and they are divided as protocol, host and port. Each property has a prefix corresponding with the provider, as follows:

```
ses.protocol=http://
ses.host=18.184.32.200
ses.port=:8080
```

2.10.5 App Store Deployment

Deployment to Google Play Store for beta testing and distribution requires to register as Developer at the play store by paying a licensing fee. After registration, the development console provides with certificates that should be used to sign the APK file to be distributed by Google. Full details of this process can be found at google developers and android developers.

2.10.6 Installation of the iOS Project

To get the code of the iOS project, it is necessary to clone the git repository on your local environment. Create a folder, and go to that location on a terminal, execute the following command:

```
git clone ssh: ://18.184.32.200:22/var/git/encompass-ios.git
```

Once the process is finished open Xcode 9, go to the menu File –> Open, navigate to the folder where the repository was copied, open the file enCOMPASS.xcworkspace and click open. Note that the project uses a series of Pods and libraries, while clicking the file .xcodeproj will open the app code, it will not load these dependencies.

2.10.7 Testing the iOS app

The application can be tested in 2 ways:

1. Installing the app in a device emulator: In Xcode go to the menu Product –> Destination iOS Simulators, and select the device to emulate, Xcode will download the device emulation files, once finished click on the menu Product –> Run.
2. Installing the app in a physical device: Connect a device via USB to the computer, go to the menu Product –> Destination –> Device, the device will be listed in the menu, select it and go to menu Product –> Run.

Consider that the app was built to work with iOS 11 and later, this can be seen in the main project configuration view, changing the target deployment will generate compilation errors and warnings.

2.10.8 iOS Configuration

In order to configure the iOS client to work with other backend servers the data.plist file needs to be changed. The file is located on the path /EnCOMPASS/encompass/src/data.plist.

The property providersArray, contains the list of the service providers that are shown in the login view. In order to add or remove providers, the property has to be changed.

```
<key>prodiverList</key>
<array>
      <string>SES</string>
      <string>SHF</string>
      <string>WVT</string>
</array>
```

Then separate properties for the servers, corresponding to each provider, can be found in the same file, with the following syntax:

```
<key>ses.protocol</key>
<string>http://</string>
<key>ses.host</key>
<string>18.184.32.200</string>
<key>ses.port</key>
```

```
<string>:8080</string>
```

The properties have been separated into 3 parts to allow secure or regular http protocol, and different ports that might be used by different application servers. Both the properties and the provider array above should be in line.

2.10.9 Development and App Store Deployment

In order to download the tools and be able to test the application on a device, registration on the Apple developer site is required: https://developer.apple.com/.

Deployment to apple TestFlight for beta testing and apple store for distribution requires registration to the apple development program at the developer website by paying a licensing fee. After registration the console provides with certificates that should be used to sign the application file to be distributed by Apple. Full details of this process can be found at the program website and enrollment procedure site.

2.10.10 Firebase Cloud Messaging

As mention before, the gamification engine uses google Firebase services to deliver messages that are present by the clients as notifications. In order to create and set up a Firebase project, go to the Firebase Console with a google user, create a new project, and provide a name.

Go to the project settings and create 2 applications one for Android and one for iOS:

- Android: The android package name will be requested, make sure to provide the app id of the android application: “com.eu.encompass”
- iOS: The App bundle id will be requested, provide the id on the iOS project: “it.polimi.encompass”

If the above properties are not aligned with the corresponding application ids the messages will not be delivered to the clients.

2.10.11 Configuration

Once the project is configured, firebase will generate a json configuration file for each app, it should be included with the code of each app to authenticate the client with the backend servers.

For android go the project settings in the firebase console, and select the android application, and download the google-service.json file. Copy the file into the android project on the /app folder.

For iOS repeat the process, go to the firebase console and select the iOS application, download the googleServices-info.plist file, finally add the file to the iOS project. Select enCompass project on the file navigation of Xcode, on the menu File –> Add Files to “enCOMPASS” and select the google services file.

In order to fully configure the iOS app for Firebase, the APN (Apple Push Notification) certificate need to be created on the apple developer console on the section Certificates, Identifiers & Profiles, 2 certificates need to be created 1 for development and testing and 1 for distribution.

Once created the certificates need to be uploaded to firebase console, on the project overview –> settings –> Cloud Messaging –> iOS application –> APNs Certificate.

Fully detail process of how to create the certificates and upload them to firebase can be found in the following links: Configuring APNs with FCM

2.11 Energy Efficiency Console Platform for Buildings Managers

The **Energy Efficiency Console Platform for Buildings Managers** is a platform where the user can see information (e.g. energy consumption, indoor environmental, etc.) from the building he/she owns or manages. He/she is able to see the current status as well as historical information. Furthermore, the platform supports multiple dashboards, which are fully parameterized, where the user can visualize different data depending on his/her interests.

The version delivered in the Platform Final Prototype is an evolution of the initial version. Monitoring of the energy consumption is supported in both aggregated and disaggregated form, allowing the building managers to understand the energy behaviour of the building. The building managers are given many options (graphs, plots, bars, etc.) regarding the customization of viewing the data, as new widgets can be added according to his/her preferences, raw data can be visualized and data from the same or different time periods may be compared.

The platform has been based on Java for the backend, while the frontend has been based on Angular-1 with parts developed at PhP.

The platform is secured and is hosted at http://encompass.iti.gr

The credentials for demo purposes are:

Login: encompass-demo

Password: encomp@$$-d3mo

The business logic of the Energy Efficiency Console platform of Building Managers for the final prototype was mainly delivered during the second release, while during the final release minor updates and parameterizations have been performed.

2.12 Digital Game Extension

This is a component released for the Second Prototype and included in the platform Final Prototype also. It is a companion app for the Funergy physical board-game, that is part of the enCompass platform. It consists of 2 parts: a trivia-like mobile game and a question management portal.

The Funergy trivia game presents the players with energy-related questions and 2 possible answers from which they have to select the correct one, the app gives feedback to the players about whether the selected answer was correct or incorrect and offers an explanation about the question; Players with a high rate of correct answers will level-up and will get harder and more interesting questions as their level gets higher, on the other hand, players with a low rate of correct answers will remain in the same level.

The Funergy question management portal is a web application that enables the platform administrator to create, edit and translate the questions and the corresponding answers that appear the Funergy app. The management portal allows the admin user to create a question by inserting the text of the question, a correct answer, an incorrect answer, the difficulty level of the question and an explanation about the topic of the question; once a question is created, translations of the question can be added for different languages, and they finally become available for the funergy app.

2.12.1 Funergy Question Management Portal

The question management portal has been developed using WebrRtio, a platform for the development of web and mobile applications based on the OMG standard IFML.

It allows the administrator user to:

- Create energy-related questions to be used by the funergy app.
- Add translations to the questions in different languages
- Keep track and statistic of the missing translations and the distribution of the questions over the difficulty levels.

2.12.2 Requirements

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

2.12.3 Installation

1. **DB Installation.** Create a new database "funergy" and import the sql file funergy.sql from ssh:// 18.184.32.200:22/var/git/funergy_db.git
2. **Application installation on Tomcat**.
   - Unzip the ssh:// 18.184.32.200:22/var/git/funergy_portal_executable.git file and copy the funergy folder into the webapps folder of your Tomcat installation (henceforth tomcat_webapps);
   - Change the database configuration (url, username and password), updating the "db1.hibernate.cfg.xml" file in the tomcat_webapps/funergy/WEB-INF/classes of the applications:

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

3. Grant "read" and "write" permission to Tomcat to the entire folder **funergy.** For UNIX:

```
sudo chown -R tomcat7 funergy/

sudo chgrp tomcat7 funergy/
```

4. Start Tomcat

2.12.4 Testing

The applications can be accessed using the following URL:

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

  username: admin

  password: admin

2.12.5 Source code

The WebRatio projects can be downloaded from: ssh://18.184.32.200:22/var/git/funergy_engine.git and can be imported into the WebRatio IDE, the following project are required:

- DropBackend
- DropBackendStyle

2.12.6 Funergy App

The funergy game has been developed using IFMLEdit, an open-source platform for the development of web and mobile applications based on the OMG standard IFML. This platform generates a Cordova project that can be used to deploy the application on Android and iOS.

Funergy App features are:

- Trivia-like mode: This feature presents players with questions and 2 options for them to select the correct one, once an option has been selected it provides feedback of whether the selected answer was correct and an explanation about the question.
- Card decode: This option was created to be used during the gameplay of the funergy boardgame, the player has the chance to scan a physical card (with a QR code) that will show a random question, answering the question correctly or incorrectly will have effects over the player situation on the board game.
- Settings: Players can select the language they want to use for the user interface and the questions of the game.

2.12.7 Requirements

Cordova:

- Node 8
- Npm 6.4
- Cordova 7.1
- GIT 2.0 or later

Android:

- Android Studio 3.0 or later
- Java JDK 1.7 or later
- GIT 2.0 or later

iOS:

- macOS 10.13.2 or later
- XCode 9.1
- GIT 2.0 or later

2.12.8 Build of the android project:

The Cordova project can be downloaded from: ssh://18.184.32.200:22/var/git/funergy_app.git, once the project has been cloned a change in configuration is needed to point the app services to the server deployed in the previous section.

Navigate to the clone project folder, and go to the path /src/js/actions, with a text editor open the following files:

- src/js/actions/action-random-question-1.js
- src/js/actions/action-random-question-2.js

- src/js/actions/action-read-card.js

Find and replace the current server URL "*funergy.ifmledit.org*" with the URL of the recently installed Funergy Question Manager URL. After this change, the code can be compiled, the android version can be created following these steps:

- Open a Command-line window and go to the project folder
- Execute the following commands sequentially:

```
Cordova platform add android

Cordova platform build Android
```

- The android project will be created on <project_folder>/platforms/android.
- Open Android Studio, on the File menu, select "Open Project", navigate to the path of the previous step and click open.
- The build process will execute automatically, in case it does not it can be started manually by going to the menu Build –> rebuild.

2.12.9 Testing of the android project:

The application can be tested in 2 ways:

1. Running the app on a connected device, as seen in section 2.10.3, or on a device Emulator: Go the menu Run –> run, a window will display a list of available devices (physical or virtual), select one and click ok. Android Studio will compile, pack and install the application on the selected device, the application will execute once the installation process is finished.
2. Generate an APK and install it manually on a device: Android studio can generate an Android Package for the application, to do so go to the menu Build –> Build APK(s). The apk file will be created automatically on the /app/build/outputs/apk folder. Copy the file to the testing device, and click on it, a pop-up menu will ask to trust the application and allow installation from unknown sources, click on accept on the device. The application will be installed and can be executed.

Notice that the application was built to support android API level 25 (Android Nougat 7.1) and minimum supported API level is 21 (Android Lollipop 5.0) which provides a wide range of supported devices, this information can be found on the build file (/app/build.gradle) but changing it would result in compilation errors.

2.12.10 Android App Store deployment:

Deployment to Google Play Store for beta testing and distribution requires to register as Developer at the play store by paying a licensing fee. After registration, the development console provides with certificates that should be used to sign the APK file to be distributed by Google. Full details of this process can be found at google developers and android developers.

2.12.11 Build of the iOS project:

The Cordova project can be downloaded from: ssh://18.184.32.200:22/var/git/funergy_app.git, once the project has been cloned a change in configuration is needed to point the app services to the server deployed in the previous section.

Navigate to the clone project folder, and go to the path /src/js/actions, with a text editor open the following files:

- src/js/actions/action-random-question-1.js
- src/js/actions/action-random-question-2.js
- src/js/actions/action-read-card.js

Find and replace the current server URL “funergy.ifmledit.org” with the URL of the recently installed Funergy Question Manager URL. After this change, the code can be compiled, the iOS version can be created following these steps:

- Open a Command-line window and go to the project folder
- Execute the following commands sequentially:

```
Cordova platform add iOS

Cordova platform build iOS
```

- The iOS project will be created on <project_folder>/platforms/ios.
- Open Xcode 9, go to the menu File –> Open, navigate to the folder of the previous step, select the file funergy.xcodeproj.

2.12.12 Testing of the iOS project:

The application can be tested in 2 ways:

1. Installing the app in a device emulator: In Xcode go to the menu Product –> Destination iOS Simulators, and select the device to emulate, Xcode will download the device emulation files, once finished click on the menu Product –> Run.
2. Installing the app in a physical device: Connect a device via USB to the computer, go to the menu Product –> Destination –> Device, the device will be listed in the menu, select it and go to menu Product –> Run.

Consider that the app was built to work with iOS 11 and later, this can be seen in the main project configuration view, changing the target deployment will generate compilation errors and warnings.

2.12.13 Apple Store deployment:

In order to download the tools and be able to test the application on a device, registration on the Apple developer site is required: https://developer.apple.com/.

Deployment to apple TestFlight for beta testing and apple store for distribution requires registration to the apple development program at the developer website by paying a licensing fee. After registration the console provides with certificates that should be used to sign the application file to be distributed by Apple. Full details of this process can be found at the program website and enrollment procedure site.

3 APPENDIX A

3.1 enCOMPASS DATABASE CREATION SCRIPT (UPDATED VERSION FOR THE PLATFORM FINAL PROTOTYPE)

```
-- --------------------------------------------------------
-- Host:                    18.184.32.200
-- Server version:          5.5.58 - MySQL Community Server (GPL)
-- Server OS:               Linux
-- HeidiSQL Version:        9.5.0.5196
-- --------------------------------------------------------
/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET NAMES utf8 */;
/*!50503 SET NAMES utf8mb4 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
-- Dumping database structure for encompass_model
CREATE DATABASE IF NOT EXISTS `encompass_model` /*!40100 DEFAULT CHARACTER SET utf16 */;
USE `encompass_model`;
-- Dumping structure for table encompass_model.activity_clothing_values
CREATE TABLE IF NOT EXISTS `activity_clothing_values` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `user_oid` int(11) NOT NULL,
  `interval` int(11) DEFAULT NULL,
  `clothing` decimal(19,3) DEFAULT NULL,
  `activity` int(11) DEFAULT NULL,
  `season` varchar(45) DEFAULT NULL,
  `datetime_updated` datetime DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_clot_act_user_idx` (`user_oid`),
  CONSTRAINT `fk_clot_act_user_idx` FOREIGN KEY (`user_oid`) REFERENCES `user` (`oid`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=81 DEFAULT CHARSET=utf16;
-- Dumping structure for table encompass_model.activity_inference
CREATE TABLE IF NOT EXISTS `activity_inference` (
  `oid` int(11) NOT NULL,
  `dwelling_room_oid` int(11) DEFAULT NULL,
```

```
  `activity` varchar(45) DEFAULT NULL,

  `timestamp` datetime DEFAULT NULL,

  `timestamp_received` datetime DEFAULT NULL,

  PRIMARY KEY (`oid`),

  KEY `fk_activity_dwelling_room_idx` (`dwelling_room_oid`),

  CONSTRAINT `fk_activity_dwelling_room` FOREIGN KEY (`dwelling_room_oid`) REFERENCES `dwelling_room` (`oid`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Dumping structure for table encompass_model.baseline

CREATE TABLE IF NOT EXISTS `baseline` (

  `oid` int(11) NOT NULL AUTO_INCREMENT,

  `smart_meter_oid` int(11) NOT NULL,

  `total_consumption` decimal(19,3) DEFAULT NULL,

  `month` date DEFAULT NULL,

  PRIMARY KEY (`oid`),

  KEY `fk_baseline_smart_meter1_idx` (`smart_meter_oid`),

  CONSTRAINT `fk_baseline_smart_meter1` FOREIGN KEY (`smart_meter_oid`) REFERENCES `smart_meter` (`oid`) ON DELETE NO ACTION ON UPDATE NO ACTION

) ENGINE=InnoDB AUTO_INCREMENT=1855 DEFAULT CHARSET=utf8;

-- Dumping structure for table encompass_model.building

CREATE TABLE IF NOT EXISTS `building` (

  `oid` int(11) NOT NULL AUTO_INCREMENT,

  `district_oid` int(11) DEFAULT NULL,

  `name` varchar(255) DEFAULT NULL,

  `construction_year` year(4) DEFAULT NULL,

  `building_size` decimal(19,2) DEFAULT NULL,

  `address` varchar(255) DEFAULT NULL,

  `renovation_year` year(4) DEFAULT NULL,

  `building_type_oid` int(11) DEFAULT NULL,

  PRIMARY KEY (`oid`),

  KEY `fk_building_district` (`district_oid`),

  KEY `fk_building_type_idx` (`building_type_oid`),

  CONSTRAINT `fk_building_district` FOREIGN KEY (`district_oid`) REFERENCES `district` (`oid`),

  CONSTRAINT `fk_building_type` FOREIGN KEY (`building_type_oid`) REFERENCES `building_type` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION

) ENGINE=InnoDB AUTO_INCREMENT=189 DEFAULT CHARSET=utf8;

-- Dumping structure for table encompass_model.building_type
```

```
CREATE TABLE IF NOT EXISTS `building_type` (
 `id` int(11) NOT NULL,
 `language` enum('en','it','de','gr') NOT NULL,
 `text` varchar(255) DEFAULT NULL,
 PRIMARY KEY (`id`,`language`)
) ENGINE=InnoDB DEFAULT CHARSET=utf16;
-- Dumping structure for procedure encompass_model.calculate_motivation
DELIMITER //
CREATE DEFINER=`root`@`%` PROCEDURE `calculate_motivation`(IN id INT, money INT, env INT,  fun INT)
BEGIN
	declare motivation int;
	IF  money >= fun AND  money > env AND money >= 3
		THEN SET motivation=1;
	ELSE
		IF  env >= fun AND  env >= money AND env >= 3
			THEN SET motivation=2;
		ELSE
			IF  fun > env AND  fun > money  AND fun >= 3
				THEN SET motivation=3;
			ELSE
				SET motivation=2;
			END IF;
		END IF;
	END IF;
    UPDATE `user` SET `motivation`= motivation , `saving_goal_money` = money, `saving_goal_fun` = fun, `saving_goal_environmental` = env WHERE `oid`=id;
END//
DELIMITER ;
-- Dumping structure for table encompass_model.complex_device_instance
CREATE TABLE IF NOT EXISTS `complex_device_instance` (
 `oid` int(11) NOT NULL AUTO_INCREMENT,
 `dwelling_room_oid` int(11) DEFAULT NULL,
 `device_type_oid` int(11) DEFAULT NULL,
 `efficiency` varchar(255) DEFAULT NULL,
 `ecomode` bit(1) DEFAULT NULL,
 `timer` bit(1) DEFAULT NULL,
```

```
  `nominal_power` decimal(19,3) DEFAULT NULL,
  `remote_control` bit(1) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_complex_device_instance_dev` (`device_type_oid`),
  KEY `fk_complex_device_instance_hou_idx` (`dwelling_room_oid`),
  CONSTRAINT `fk_complex_device_instance_dev` FOREIGN KEY (`device_type_oid`) REFERENCES `device_type` (`oid`),
  CONSTRAINT `fk_complex_device_instance_hou` FOREIGN KEY (`dwelling_room_oid`) REFERENCES `dwelling_room` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
-- Dumping structure for table encompass_model.cost_estimation
CREATE TABLE IF NOT EXISTS `cost_estimation` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `dwelling_oid` int(11) DEFAULT NULL,
  `date_start` date DEFAULT NULL,
  `date_end` date DEFAULT NULL,
  `unit_eur_charge` decimal(19,3) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_bill_household` (`dwelling_oid`),
  CONSTRAINT `fk_bill_household` FOREIGN KEY (`dwelling_oid`) REFERENCES `dwelling` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
-- Dumping structure for table encompass_model.device_consumption
CREATE TABLE IF NOT EXISTS `device_consumption` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `complex_device_instance_oid` int(11) DEFAULT NULL,
  `simple_device_instance_oid` int(11) DEFAULT NULL,
  `datetime` datetime DEFAULT NULL,
  `datetime_received` datetime DEFAULT NULL,
  `consumption` decimal(19,3) DEFAULT NULL,
  `is_disaggregated` tinyint(1) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_simple_cons_idx` (`simple_device_instance_oid`),
  KEY `fk_complex_cons_idx` (`complex_device_instance_oid`),
  CONSTRAINT `fk_complex_cons` FOREIGN KEY (`complex_device_instance_oid`) REFERENCES `complex_device_instance` (`oid`)
ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `fk_simple_cons` FOREIGN KEY (`simple_device_instance_oid`) REFERENCES `simple_device_instance` (`oid`) ON
DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
-- Dumping structure for table encompass_model.device_reading
```

```
CREATE TABLE IF NOT EXISTS `device_reading` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `complex_device_instance_oid` int(11) DEFAULT NULL,
  `simple_device_instance_oid` int(11) DEFAULT NULL,
  `datetime` datetime DEFAULT NULL,
  `datetime_received` datetime DEFAULT NULL,
  `reading` decimal(19,3) DEFAULT NULL,
  `is_disaggregated` tinyint(1) 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;

-- Dumping structure for table encompass_model.device_type
CREATE TABLE IF NOT EXISTS `device_type` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(255) DEFAULT NULL,
  `icon` text,
  `type` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Dumping structure for table encompass_model.disaggregation_data
CREATE TABLE IF NOT EXISTS `disaggregation_data` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `user_oid` int(11) DEFAULT NULL,
  `date` date DEFAULT NULL,
  `datetime_received` datetime DEFAULT NULL,
  `datetime_proc` datetime DEFAULT NULL,
  `fridge` decimal(19,4) DEFAULT NULL,
  `washing_machine` decimal(19,4) DEFAULT NULL,
  `tumble_dryer` decimal(19,4) DEFAULT NULL,
  `dishwasher` decimal(19,4) DEFAULT NULL,
  `AC` decimal(19,4) DEFAULT NULL,
  `electric_car` decimal(19,4) DEFAULT NULL,
```

```
  `electric_oven` decimal(19,4) DEFAULT NULL,
  `heat_pump` decimal(19,4) DEFAULT NULL,
  `other` decimal(19,4) DEFAULT NULL,
  `total_consumption` decimal(19,4) DEFAULT NULL,
  `v_mod` smallint(6) NOT NULL,
  `v_dat` smallint(6) NOT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_disaggr_user_idx` (`user_oid`),
  CONSTRAINT `fk_disaggr_user_idx` FOREIGN KEY (`user_oid`) REFERENCES `user` (`oid`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=13510 DEFAULT CHARSET=utf16;

-- Dumping structure for table encompass_model.district
CREATE TABLE IF NOT EXISTS `district` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(255) DEFAULT NULL,
  `country` varchar(255) DEFAULT NULL,
  `city` varchar(255) DEFAULT NULL,
  `zipcode` varchar(45) DEFAULT NULL,
  `timezone` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB AUTO_INCREMENT=46 DEFAULT CHARSET=utf8;

-- Dumping structure for table encompass_model.dwelling
CREATE TABLE IF NOT EXISTS `dwelling` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `building_oid` int(11) NOT NULL,
  `name` varchar(255) DEFAULT NULL,
  `dwelling_size` decimal(19,2) DEFAULT NULL,
  `ownership` bit(1) DEFAULT NULL,
  `public` bit(1) DEFAULT NULL,
  `number_rooms` int(11) DEFAULT NULL,
  `number_adults` int(11) DEFAULT NULL,
  `number_children` int(11) DEFAULT NULL,
  `number_visitors` int(11) DEFAULT NULL,
  `windows_position` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_dwelling_building1_idx` (`building_oid`),
```

```
  CONSTRAINT `fk_dwelling_building1` FOREIGN KEY (`building_oid`) REFERENCES `building` (`oid`) ON DELETE NO ACTION ON UPDATE NO ACTION

) ENGINE=InnoDB AUTO_INCREMENT=194 DEFAULT CHARSET=utf8;

-- Dumping structure for table encompass_model.dwelling_room

CREATE TABLE IF NOT EXISTS `dwelling_room` (

  `oid` int(11) NOT NULL AUTO_INCREMENT,

  `dwelling_oid` int(11) NOT NULL,

  `name` varchar(255) DEFAULT NULL,

  `area` decimal(19,3) DEFAULT NULL,

  PRIMARY KEY (`oid`),

  KEY `fk_dwelling_rooms_dwelling1_idx` (`dwelling_oid`),

  CONSTRAINT `fk_dwelling_rooms_dwelling1` FOREIGN KEY (`dwelling_oid`) REFERENCES `dwelling` (`oid`) ON DELETE NO ACTION ON UPDATE NO ACTION

) ENGINE=InnoDB AUTO_INCREMENT=201 DEFAULT CHARSET=utf8;

-- Dumping structure for table encompass_model.dwelling_type

CREATE TABLE IF NOT EXISTS `dwelling_type` (

  `id` int(11) NOT NULL,

  `language` enum('en','it','de','gr') NOT NULL,

  `text` varchar(255) DEFAULT NULL,

  PRIMARY KEY (`id`,`language`)

) ENGINE=InnoDB DEFAULT CHARSET=utf16;

-- Dumping structure for table encompass_model.encompass_model_ses.message_motivation

CREATE TABLE IF NOT EXISTS `encompass_model_ses.message_motivation` (

  `oid` int(11) NOT NULL,

  `motivation` int(11) DEFAULT NULL,

  `language` varchar(255) DEFAULT NULL,

  `motivation_text` varchar(255) DEFAULT NULL,

  PRIMARY KEY (`oid`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Dumping structure for table encompass_model.generic_message

CREATE TABLE IF NOT EXISTS `generic_message` (

  `oid` int(11) NOT NULL AUTO_INCREMENT,

  `automatic_exec` tinyint(1) DEFAULT NULL,

  `apply_to` varchar(255) DEFAULT NULL,

  `message_type` varchar(255) DEFAULT NULL,

  PRIMARY KEY (`oid`),

  KEY `message_type` (`message_type`)
```

```
) ENGINE=InnoDB AUTO_INCREMENT=214 DEFAULT CHARSET=utf8;
-- Dumping structure for procedure encompass_model.get_savings_sp
DELIMITER //
CREATE DEFINER=`puser`@`%` PROCEDURE `get_savings_sp`(
	IN `user_oid` INT
)
SELECT
meterconsu0_.saving_date  col_0_0,
meterconsu0_.monthly_consumption AS col_1_0,
((baselines4_.total_consumption-meterconsu0_.monthly_consumption)/baselines4_.total_consumption)*100 AS col_2_0
FROM
(SELECT UUID() AS oid,
u.oid AS user_oid,
TRUNCATE(SUM(mc.consumption),2) AS monthly_consumption,
last_day(date(concat_ws('-', YEAR(mc.datetime), MONTH(mc.datetime), 1))) AS saving_date,
sm.oid AS sm_oid
FROM (((meter_consumption mc JOIN smart_meter sm ON((mc.smart_meter_oid = sm.oid)))
JOIN dwelling d ON((sm.dwelling_oid = d.oid)))
JOIN user u ON((d.oid = u.dwelling_oid)))
WHERE (CAST(mc.datetime AS DATE) <= CAST((NOW() + INTERVAL -(1) DAY) AS DATE))
and u.oid=user_oid
GROUP BY u.oid, MONTH(mc.datetime), YEAR(mc.datetime)) meterconsu0_
JOIN baseline baselines4_ ON meterconsu0_.sm_oid = baselines4_.smart_meter_oid
AND MONTH(meterconsu0_.saving_date) = MONTH(baselines4_.month)
WHERE MONTH(meterconsu0_.saving_date) < MONTH(now())
ORDER BY meterconsu0_.saving_date DESC//
DELIMITER ;
-- Dumping structure for view encompass_model.get_tips
-- Creating temporary table to overcome VIEW dependency errors
CREATE TABLE `get_tips` (
	`user_oid` INT(11) NULL,
	`id` INT(11) NOT NULL,
	`name` VARCHAR(100) NULL COLLATE 'utf8_general_ci',
	`header` CHAR(0) NOT NULL COLLATE 'utf8mb4_general_ci',
	`body` VARCHAR(255) NULL COLLATE 'utf8_general_ci',
	`image` VARCHAR(255) NULL COLLATE 'utf8_general_ci',
```

```
	`video` VARCHAR(255) NULL COLLATE 'utf8_general_ci',
	`instance` INT(11) NOT NULL,
	`feedback` VARCHAR(255) NULL COLLATE 'utf8_general_ci',
	`static` TINYINT(4) NULL,
	`timestamp_creation` DATETIME NULL
) ENGINE=MyISAM;

-- Dumping structure for procedure encompass_model.get_tips_sp

DELIMITER //

CREATE DEFINER=`puser`@`%` PROCEDURE `get_tips_sp`(IN `user_oid` INT

)

select * from (

select `mi`.`user_oid` AS `user_oid`,`gm`.`oid` AS `id`,`ml`.`title` AS `name`, NULL AS `header`,`ml`.`description` AS `body`,`ml`.`image` AS `image`,`ml`.`video` AS `video`,`mi`.`oid` AS `instance`,`mf2`.`feedback` AS `feedback`,`mi`.`is_static` AS `static`,`mi`.`is_executable` AS `executable`, `mi`.`timestamp_creation` AS `timestamp_creation` from ((((`generic_message` `gm` join `message_instance` `mi` on((`mi`.`generic_message_oid` = `gm`.`oid`))) join `message_localization` `ml` on(((`ml`.`generic_message_oid` = `gm`.`oid`) and (`mi`.`is_static` = 1)))) join `user_preference` `up` on(((`up`.`user_oid` = `mi`.`user_oid`) and (`up`.`language` = `ml`.`language`)))) left join `get_tips_with_feedback` `mf2` on((`mf2`.`message_instance_oid` = `mi`.`oid`))) where (`mi`.`hidden` = 0) and up.user_oid=user_oid

union

select `mi`.`user_oid` AS `user_oid`,`gm`.`oid` AS `id`,`ofml`.`title` AS `name`, NULL AS `header`,`ofml`.`description` AS `body`,NULL AS `image`,NULL AS `video`,`mi`.`oid` AS `instance`,`mf2`.`feedback` AS `feedback`,`mi`.`is_static` AS `static`,`mi`.`is_executable` AS `executable`,`mi`.`timestamp_creation` AS `timestamp_creation` from ((((`generic_message` `gm` join `message_instance` `mi` on((`mi`.`generic_message_oid` = `gm`.`oid`))) join `on_the_fly_message_localization` `ofml` on(((`ofml`.`message_instance_oid` = `mi`.`oid`) and (`mi`.`is_static` = 0)))) join `user_preference` `up` on(((`up`.`user_oid` = `mi`.`user_oid`) and (`up`.`language` = `ofml`.`language`)))) left join `get_tips_with_feedback` `mf2` on((`mf2`.`message_instance_oid` = `mi`.`oid`))) where (`mi`.`hidden` = 0) and up.user_oid=user_oid

) t order by t.timestamp_creation desc//

DELIMITER ;

-- Dumping structure for view encompass_model.get_tips_with_feedback

-- Creating temporary table to overcome VIEW dependency errors

CREATE TABLE `get_tips_with_feedback` (
	`oid` INT(11) NOT NULL,
	`feedback` VARCHAR(255) NULL COLLATE 'utf8_general_ci',
	`message_instance_oid` INT(11) NOT NULL
) ENGINE=MyISAM;

-- Dumping structure for table encompass_model.group

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

```
-- Dumping structure for table encompass_model.heating_source_type
CREATE TABLE IF NOT EXISTS `heating_source_type` (
  `id` int(11) NOT NULL,
  `language` enum('en','it','de','gr') NOT NULL,
  `text` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`,`language`)
) ENGINE=InnoDB DEFAULT CHARSET=utf16;
-- Dumping structure for table encompass_model.heating_type
CREATE TABLE IF NOT EXISTS `heating_type` (
  `id` int(11) NOT NULL,
  `language` enum('en','it','de','gr') NOT NULL,
  `text` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`,`language`)
) ENGINE=InnoDB DEFAULT CHARSET=utf16;
-- Dumping structure for table encompass_model.indoor_conditions_co2
CREATE TABLE IF NOT EXISTS `indoor_conditions_co2` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `dwelling_room_oid` int(11) DEFAULT NULL,
  `value` double DEFAULT NULL,
  `datetime` datetime DEFAULT NULL,
  `datetime_received` datetime DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_environmental_room_idx` (`dwelling_room_oid`),
  CONSTRAINT `fk_environmental_room000` FOREIGN KEY (`dwelling_room_oid`) REFERENCES `dwelling_room` (`oid`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
-- Dumping structure for table encompass_model.indoor_conditions_humidity
CREATE TABLE IF NOT EXISTS `indoor_conditions_humidity` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `dwelling_room_oid` int(11) DEFAULT NULL,
  `value` double DEFAULT NULL,
  `datetime` datetime DEFAULT NULL,
  `datetime_received` datetime DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_environmental_room_idx` (`dwelling_room_oid`),
  CONSTRAINT `fk_environmental_room00` FOREIGN KEY (`dwelling_room_oid`) REFERENCES `dwelling_room` (`oid`) ON DELETE NO ACTION ON UPDATE NO ACTION
```

```
) ENGINE=InnoDB AUTO_INCREMENT=906654 DEFAULT CHARSET=utf8;

-- Dumping structure for table encompass_model.indoor_conditions_luminance

CREATE TABLE IF NOT EXISTS `indoor_conditions_luminance` (

 `oid` int(11) NOT NULL AUTO_INCREMENT,

 `dwelling_room_oid` int(11) DEFAULT NULL,

 `value` double DEFAULT NULL,

 `datetime` datetime DEFAULT NULL,

 `datetime_received` datetime DEFAULT NULL,

 PRIMARY KEY (`oid`),

 KEY `fk_environmental_room_idx` (`dwelling_room_oid`),

 CONSTRAINT `fk_environmental_room0` FOREIGN KEY (`dwelling_room_oid`) REFERENCES `dwelling_room` (`oid`) ON DELETE NO ACTION ON UPDATE NO ACTION

) ENGINE=InnoDB AUTO_INCREMENT=869603 DEFAULT CHARSET=utf8;

-- Dumping structure for table encompass_model.indoor_conditions_temperature

CREATE TABLE IF NOT EXISTS `indoor_conditions_temperature` (

 `oid` int(11) NOT NULL AUTO_INCREMENT,

 `dwelling_room_oid` int(11) DEFAULT NULL,

 `value` double DEFAULT NULL,

 `datetime` datetime DEFAULT NULL,

 `datetime_received` datetime DEFAULT NULL,

 PRIMARY KEY (`oid`),

 KEY `fk_environmental_room_idx` (`dwelling_room_oid`),

 CONSTRAINT `fk_environmental_room` FOREIGN KEY (`dwelling_room_oid`) REFERENCES `dwelling_room` (`oid`) ON DELETE NO ACTION ON UPDATE NO ACTION

) ENGINE=InnoDB AUTO_INCREMENT=632989 DEFAULT CHARSET=utf8;

-- Dumping structure for table encompass_model.indoor_sensing_occupancy

CREATE TABLE IF NOT EXISTS `indoor_sensing_occupancy` (

 `oid` int(11) NOT NULL AUTO_INCREMENT,

 `dwelling_room_oid` int(11) DEFAULT NULL,

 `value` double DEFAULT NULL,

 `datetime` datetime DEFAULT NULL,

 `datetime_received` datetime DEFAULT NULL,

 PRIMARY KEY (`oid`),

 KEY `fk_environmental_room_idx` (`dwelling_room_oid`),

 CONSTRAINT `fk_environmental_room1` FOREIGN KEY (`dwelling_room_oid`) REFERENCES `dwelling_room` (`oid`) ON DELETE NO ACTION ON UPDATE NO ACTION

) ENGINE=InnoDB AUTO_INCREMENT=563670 DEFAULT CHARSET=utf8;
```

```
-- Dumping structure for table encompass_model.luminance_coefficient
CREATE TABLE IF NOT EXISTS `luminance_coefficient` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `dwelling_room_oid` int(11) NOT NULL,
  `coefficient` float(19,3) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_lumcoef_dwelroom_idx` (`dwelling_room_oid`),
  CONSTRAINT `fk_lumcoef_dwelroom` FOREIGN KEY (`dwelling_room_oid`) REFERENCES `dwelling_room` (`oid`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=utf16;
-- Dumping structure for table encompass_model.main_lighting_type
CREATE TABLE IF NOT EXISTS `main_lighting_type` (
  `id` int(11) NOT NULL,
  `language` enum('en','it','de','gr') NOT NULL,
  `text` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`,`language`)
) ENGINE=InnoDB DEFAULT CHARSET=utf16;


-- Dumping structure for table encompass_model.message_feedback
CREATE TABLE IF NOT EXISTS `message_feedback` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `message_instance_oid` int(11) NOT NULL,
  `timestamp` datetime NOT NULL,
  `feedback` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`message_instance_oid`,`timestamp`),
  KEY `oid` (`oid`),
  CONSTRAINT `fk_message_feedback_message_instance` FOREIGN KEY (`message_instance_oid`) REFERENCES `message_instance` (`oid`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=237 DEFAULT CHARSET=utf8;
-- Dumping structure for table encompass_model.message_instance
CREATE TABLE IF NOT EXISTS `message_instance` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `generic_message_oid` int(11) DEFAULT NULL,
  `user_oid` int(11) DEFAULT NULL,
  `timestamp_creation` datetime DEFAULT NULL,
  `timestamp_read` datetime DEFAULT NULL,
```

```
  `hidden` tinyint(1) DEFAULT NULL,
  `is_static` tinyint(1) DEFAULT NULL,
  `is_processed` tinyint(1) DEFAULT '0',
  `is_executable` tinyint(1) DEFAULT '0',
  PRIMARY KEY (`oid`),
  KEY `fk_tips_users_idx` (`user_oid`),
  KEY `fk_tip_instance_idx` (`generic_message_oid`),
  CONSTRAINT `fk_message_instance_generic_message` FOREIGN KEY (`generic_message_oid`) REFERENCES `generic_message` (`oid`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `fk_message_instance_user` FOREIGN KEY (`user_oid`) REFERENCES `user` (`oid`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=2590 DEFAULT CHARSET=utf8;

-- Dumping structure for table encompass_model.message_localization
CREATE TABLE IF NOT EXISTS `message_localization` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `generic_message_oid` int(11) NOT NULL,
  `language` varchar(255) NOT NULL,
  `title` varchar(100) DEFAULT NULL,
  `description` varchar(255) DEFAULT NULL,
  `image` varchar(255) DEFAULT NULL,
  `video` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`generic_message_oid`,`language`),
  KEY `oid` (`oid`),
  CONSTRAINT `FK_message_localization_generic_message` FOREIGN KEY (`generic_message_oid`) REFERENCES `generic_message` (`oid`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=524 DEFAULT CHARSET=utf8;

-- Dumping structure for table encompass_model.message_motivation
CREATE TABLE IF NOT EXISTS `message_motivation` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `motivation` int(11) DEFAULT NULL,
  `language` varchar(255) DEFAULT NULL,
  `motivation_text` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB AUTO_INCREMENT=25 DEFAULT CHARSET=utf16;

-- Dumping structure for table encompass_model.meter_consumption
CREATE TABLE IF NOT EXISTS `meter_consumption` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
```

```
 `smart_meter_oid` int(11) DEFAULT NULL,
 `datetime` datetime DEFAULT NULL,
 `datetime_received` datetime DEFAULT NULL,
 `consumption` decimal(19,3) DEFAULT NULL,
 PRIMARY KEY (`oid`),
 KEY `fk_meter_consum_meter_idx` (`smart_meter_oid`),
 CONSTRAINT `fk_meter_consum_meter` FOREIGN KEY (`smart_meter_oid`) REFERENCES `smart_meter` (`oid`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB AUTO_INCREMENT=1291939 DEFAULT CHARSET=utf8;
-- Dumping structure for view encompass_model.meter_consumption_daily_avg
-- Creating temporary table to overcome VIEW dependency errors
CREATE TABLE `meter_consumption_daily_avg` (
	`user_oid` INT(11) NOT NULL,
	`daily_consumption` DECIMAL(41,3) NULL,
	`day` INT(2) NULL,
	`month` INT(2) NULL,
	`year` INT(4) NULL
) ENGINE=MyISAM;

-- Dumping structure for procedure encompass_model.meter_consumption_daily_avg_sp
DELIMITER //
CREATE DEFINER=`puser`@`%` PROCEDURE `meter_consumption_daily_avg_sp`(IN `user_oid` INT
)
SELECT t.user_oid, AVG(t.daily_consumption) FROM (
SELECT u.oid AS user_oid, SUM(mc.consumption) AS daily_consumption,
DAYOFMONTH(mc.datetime) AS day, MONTH(mc.datetime) AS month, YEAR(mc.datetime) AS year
FROM (((meter_consumption mc
JOIN smart_meter sm ON((mc.smart_meter_oid = sm.oid)))
JOIN dwelling d ON((sm.dwelling_oid = d.oid)))
JOIN user u ON((d.oid = u.dwelling_oid)))
WHERE (CAST(mc.datetime AS DATE) <= CAST((NOW() + INTERVAL -(1) DAY) AS DATE)) and
u.oid=user_oid
GROUP BY u.oid, DAYOFMONTH(mc.datetime), MONTH(mc.datetime), YEAR(mc.datetime)
ORDER BY u.oid, YEAR(mc.datetime), MONTH(mc.datetime), DAYOFMONTH(mc.datetime)
) t//
DELIMITER ;
```

```
-- Dumping structure for table encompass_model.meter_consumption_error

CREATE TABLE IF NOT EXISTS `meter_consumption_error` (

 `oid` int(11) NOT NULL AUTO_INCREMENT,

 `smart_meter_oid` int(11) DEFAULT NULL,

 `datetime` datetime DEFAULT NULL,

 `datetime_received` datetime DEFAULT NULL,

 `consumption` decimal(19,3) DEFAULT NULL,

 PRIMARY KEY (`oid`),

 KEY `fk_meter_consum_error_idx` (`smart_meter_oid`),

 CONSTRAINT `fk_meter_consum_error_smart_meter` FOREIGN KEY (`smart_meter_oid`) REFERENCES `smart_meter` (`oid`) ON DELETE NO ACTION ON UPDATE NO ACTION

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Dumping structure for view encompass_model.meter_consumption_from_to

-- Creating temporary table to overcome VIEW dependency errors

CREATE TABLE `meter_consumption_from_to` (

	`user_oid` INT(11) NOT NULL,

	`from_` DATETIME NULL,

	`to_` DATETIME NULL

) ENGINE=MyISAM;


-- Dumping structure for procedure encompass_model.meter_consumption_from_to_sp

DELIMITER //

CREATE DEFINER=`puser`@`%` PROCEDURE `meter_consumption_from_to_sp`(

	IN `user_oid` INT

)

select `u`.`oid` AS `user_oid`,min(`mc`.`datetime`) AS `from_`,max(`mc`.`datetime`) AS `to_` from (((`meter_consumption` `mc` join `smart_meter` `sm` on((`mc`.`smart_meter_oid` = `sm`.`oid`))) join `dwelling` `d` on((`sm`.`dwelling_oid` = `d`.`oid`))) join `user` `u` on((`d`.`oid` = `u`.`dwelling_oid`)))

where (cast(`mc`.`datetime` as date) <= cast((now() + interval -(1) day) as date)) and u.oid=user_oid

group by `u`.`oid`//

DELIMITER ;

-- Dumping structure for view encompass_model.meter_consumption_monthly_avg

-- Creating temporary table to overcome VIEW dependency errors

CREATE TABLE `meter_consumption_monthly_avg` (

	`oid` VARCHAR(36) NOT NULL COLLATE 'utf8_general_ci',

	`user_oid` INT(11) NOT NULL,

	`monthly_consumption` DECIMAL(40,2) NULL,
```

```
	`month` INT(2) NULL,
	`year` INT(4) NULL
) ENGINE=MyISAM;
-- Dumping structure for procedure encompass_model.meter_consumption_monthly_avg_sp
DELIMITER //
CREATE DEFINER=`puser`@`%` PROCEDURE `meter_consumption_monthly_avg_sp`(IN `user_oid` INT
)
select t.user_oid, avg(t.monthly_consumption) from (
select uuid() AS `oid`,`u`.`oid` AS `user_oid`,truncate(sum(`mc`.`consumption`),2) AS `monthly_consumption`,month(`mc`.`datetime`) AS `month`,year(`mc`.`datetime`) AS `year` from (((`meter_consumption` `mc` join `smart_meter` `sm` on((`mc`.`smart_meter_oid` = `sm`.`oid`))) join `dwelling` `d` on((`sm`.`dwelling_oid` = `d`.`oid`))) join `user` `u` on((`d`.`oid` = `u`.`dwelling_oid`))) where (cast(`mc`.`datetime` as date) <= cast((now() + interval -(1) day) as date)) and u.oid=user_oid group by `u`.`oid`,month(`mc`.`datetime`),year(`mc`.`datetime`)
) t//
DELIMITER ;
-- Dumping structure for table encompass_model.meter_consumption_normalized
CREATE TABLE IF NOT EXISTS `meter_consumption_normalized` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `meter_consumption_oid` int(11) NOT NULL,
  `datetime_reading` datetime DEFAULT NULL,
  `total_consumption` decimal(19,3) DEFAULT NULL,
  `total_consumption_adjusted` decimal(19,3) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  UNIQUE KEY `unique_reading` (`datetime_reading`),
  KEY `fk_meter_consumption_normalized_meter_consumption1_idx` (`meter_consumption_oid`),
  CONSTRAINT `fk_meter_consumption_normalized_meter_consumption1` FOREIGN KEY (`meter_consumption_oid`) REFERENCES `meter_consumption` (`oid`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=utf16;
-- Dumping structure for view encompass_model.meter_consumption_this_month_total
-- Creating temporary table to overcome VIEW dependency errors
CREATE TABLE `meter_consumption_this_month_total` (
	`user_oid` INT(11) NOT NULL,
	`this_month_total` DECIMAL(41,3) NULL,
	`month` INT(2) NULL,
	`year` INT(4) NULL
) ENGINE=MyISAM;
-- Dumping structure for procedure encompass_model.meter_consumption_this_month_total_sp
DELIMITER //
```

```
CREATE DEFINER=`puser`@`%` PROCEDURE `meter_consumption_this_month_total_sp`(
	IN `user_oid` INT
)
select `u`.`oid` AS `user_oid`,sum(`mc`.`consumption`) AS `this_month_total`,month((now() - interval 1 day)) AS `month`,year(now()) AS `year` from (((`meter_consumption` `mc` join `smart_meter` `sm` on((`mc`.`smart_meter_oid` = `sm`.`oid`))) join `dwelling` `d` on((`sm`.`dwelling_oid` = `d`.`oid`))) join `user` `u` on((`d`.`oid` = `u`.`dwelling_oid`))) where ((month(`mc`.`datetime`) = month((now() - interval 1 day))) and (year(`mc`.`datetime`) = year((now() - interval 1 day))) and (cast(`mc`.`datetime` as date) <= cast((now() + interval -(1) day) as date))) and u.oid=user_oid group by `u`.`oid`//
DELIMITER ;
-- Dumping structure for view encompass_model.meter_consumption_this_week_total
-- Creating temporary table to overcome VIEW dependency errors
CREATE TABLE `meter_consumption_this_week_total` (
	`user_oid` INT(11) NOT NULL,
	`this_week_total` DECIMAL(41,3) NULL,
	`week` INT(2) NULL,
	`year` INT(4) NULL
) ENGINE=MyISAM;
-- Dumping structure for procedure encompass_model.meter_consumption_this_week_total_sp
DELIMITER //
CREATE DEFINER=`puser`@`%` PROCEDURE `meter_consumption_this_week_total_sp`(
	IN `user_oid` INT
)
select t.user_oid, avg(t.weekly_consumption) from (
select `u`.`oid` AS `user_oid`,sum(`mc`.`consumption`) AS `weekly_consumption`,week(`mc`.`datetime`,7) AS `week`,year(`mc`.`datetime`) AS `year` from (((`meter_consumption` `mc` join `smart_meter` `sm` on((`mc`.`smart_meter_oid` = `sm`.`oid`))) join `dwelling` `d` on((`sm`.`dwelling_oid` = `d`.`oid`))) join `user` `u` on((`d`.`oid` = `u`.`dwelling_oid`))) where ( week(DATE_SUB(mc.datetime, INTERVAL 1 DAY), 7) =  week(DATE_SUB(now(), INTERVAL 1 DAY),7) and
(year(mc.datetime) = year(now())) and cast((mc.datetime) as date)) and u.oid=user_oid
group by `u`.`oid`,week(`mc`.`datetime`,7),year(`mc`.`datetime`)
) t//
DELIMITER ;
-- Dumping structure for view encompass_model.meter_consumption_today_total
-- Creating temporary table to overcome VIEW dependency errors
CREATE TABLE `meter_consumption_today_total` (
	`user_oid` INT(11) NOT NULL,
	`today_total` DECIMAL(41,3) NULL,
	`day` INT(2) NULL,
	`month` INT(2) NULL,
	`year` INT(4) NULL
```

```
) ENGINE=MyISAM;

-- Dumping structure for procedure encompass_model.meter_consumption_today_total_sp

DELIMITER //

CREATE DEFINER=`puser`@`%` PROCEDURE `meter_consumption_today_total_sp`(IN `user_oid` INT

)

select t.user_oid, avg(t.today_total) from (

select `u`.`oid` AS `user_oid`,sum(`mc`.`consumption`) AS `today_total`,dayofmonth((now() - interval 1 day)) AS `day`,month((now() - interval 1 day)) AS `month`,year((now() - interval 1 day)) AS `year` from (((`meter_consumption` `mc` join `smart_meter` `sm` on((`mc`.`smart_meter_oid` = `sm`.`oid`))) join `dwelling` `d` on((`sm`.`dwelling_oid` = `d`.`oid`))) join `user` `u` on((`d`.`oid` = `u`.`dwelling_oid`))) where (cast(`mc`.`datetime` as date) = cast((now() + interval -(1) day) as date)) and u.oid=user_oid group by `u`.`oid`

) t//

DELIMITER ;

-- Dumping structure for view encompass_model.meter_consumption_weekly_avg

-- Creating temporary table to overcome VIEW dependency errors

CREATE TABLE `meter_consumption_weekly_avg` (

	`user_oid` INT(11) NOT NULL,

	`weekly_consumption` DECIMAL(41,3) NULL,

	`week` INT(2) NULL,

	`year` INT(4) NULL

) ENGINE=MyISAM;

-- Dumping structure for procedure encompass_model.meter_consumption_weekly_avg_sp

DELIMITER //

CREATE DEFINER=`puser`@`%` PROCEDURE `meter_consumption_weekly_avg_sp`(IN `user_oid` INT

)

select t.user_oid, avg(t.weekly_consumption) from (

select `u`.`oid` AS `user_oid`,sum(`mc`.`consumption`) AS `weekly_consumption`,week(`mc`.`datetime`,0) AS `week`,year(`mc`.`datetime`) AS `year` from (((`meter_consumption` `mc` join `smart_meter` `sm` on((`mc`.`smart_meter_oid` = `sm`.`oid`))) join `dwelling` `d` on((`sm`.`dwelling_oid` = `d`.`oid`))) join `user` `u` on((`d`.`oid` = `u`.`dwelling_oid`)))

where (cast(`mc`.`datetime` as date) <= cast((now() + interval -(1) day) as date)) and u.oid=user_oid

group by `u`.`oid`,week(`mc`.`datetime`,0),year(`mc`.`datetime`)

) t//

DELIMITER ;

-- Dumping structure for table encompass_model.meter_reading

CREATE TABLE IF NOT EXISTS `meter_reading` (

  `oid` int(11) NOT NULL AUTO_INCREMENT,

  `smart_meter_oid` int(11) DEFAULT NULL,

  `datetime` datetime DEFAULT NULL,

  `datetime_received` datetime DEFAULT NULL,
```

```
  `reading` decimal(19,3) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  UNIQUE KEY `unique_reading` (`datetime`,`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 DEFAULT CHARSET=utf8;
-- Dumping structure for table encompass_model.meter_reading_error
CREATE TABLE IF NOT EXISTS `meter_reading_error` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `smart_meter_oid` int(11) DEFAULT NULL,
  `datetime` datetime DEFAULT NULL,
  `datetime_received` datetime DEFAULT NULL,
  `reading` decimal(19,3) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_meter_reading_error_idx` (`smart_meter_oid`),
  CONSTRAINT `fk_meter_reading_error_smart_meter` FOREIGN KEY (`smart_meter_oid`) REFERENCES `smart_meter` (`oid`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=utf8;


-- Dumping structure for table encompass_model.meter_reading_normalized
CREATE TABLE IF NOT EXISTS `meter_reading_normalized` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `meter_reading_oid` int(11) NOT NULL,
  `datetime_reading` datetime DEFAULT NULL,
  `total_consumption` decimal(19,3) DEFAULT NULL,
  `total_consumption_adjusted` decimal(19,3) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  UNIQUE KEY `unique_reading` (`datetime_reading`),
  KEY `fk_meter_reading_normalized_meter_reading1_idx` (`meter_reading_oid`),
  CONSTRAINT `fk_meter_reading_normalized_meter_reading1` FOREIGN KEY (`meter_reading_oid`) REFERENCES `meter_reading` (`oid`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
-- Dumping structure for table encompass_model.notification_delivery
CREATE TABLE IF NOT EXISTS `notification_delivery` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `notification_instance_oid` int(11) NOT NULL,
```

```
  `current_priority` varchar(50) DEFAULT NULL,
  `delivery_timestamp` datetime DEFAULT NULL,
  `delivery_status` varchar(50) DEFAULT NULL,
  `active` tinyint(1) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_user_recommendations_user1_idx` (`notification_instance_oid`),
  CONSTRAINT `fk_notification_delivery_notification_instance` FOREIGN KEY (`notification_instance_oid`) REFERENCES `notification_instance` (`oid`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=1415 DEFAULT CHARSET=utf8;
-- Dumping structure for table encompass_model.notification_feedback
CREATE TABLE IF NOT EXISTS `notification_feedback` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `notification_delivery_oid` int(11) NOT NULL,
  `feedback_timestamp` datetime DEFAULT NULL,
  `feedback_status` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_user_recommendations_user10_idx` (`notification_delivery_oid`),
  CONSTRAINT `fk_user_recommendations_user10` FOREIGN KEY (`notification_delivery_oid`) REFERENCES `notification_delivery` (`oid`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
-- Dumping structure for table encompass_model.notification_instance
CREATE TABLE IF NOT EXISTS `notification_instance` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `user_oid` int(11) DEFAULT NULL,
  `notification_type_oid` int(11) DEFAULT NULL,
  `is_static` tinyint(1) DEFAULT NULL,
  `title` varchar(100) DEFAULT NULL,
  `description` varchar(255) DEFAULT NULL,
  `scheduled_priority` varchar(50) DEFAULT NULL,
  `scheduled_timestamp` datetime DEFAULT NULL,
  `scheduled_status` varchar(50) DEFAULT NULL,
  `object_id` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `act_user_idx` (`user_oid`),
  KEY `fk_notification_type_idx` (`notification_type_oid`),
  CONSTRAINT `act_user0` FOREIGN KEY (`user_oid`) REFERENCES `user` (`oid`) ON DELETE CASCADE ON UPDATE CASCADE,
```

```
  CONSTRAINT `fk_notification_type` FOREIGN KEY (`notification_type_oid`) REFERENCES `notification_type` (`oid`) ON DELETE NO ACTION ON UPDATE NO ACTION

) ENGINE=InnoDB AUTO_INCREMENT=1791 DEFAULT CHARSET=utf8;

-- Dumping structure for table encompass_model.notification_type

CREATE TABLE IF NOT EXISTS `notification_type` (

  `oid` int(11) NOT NULL AUTO_INCREMENT,

  `type` varchar(255) DEFAULT NULL,

  `default_priority` varchar(50) DEFAULT NULL,

  `duplicate_control` varchar(255) DEFAULT NULL,

  PRIMARY KEY (`oid`),

  KEY `type` (`type`)

) ENGINE=InnoDB AUTO_INCREMENT=14 DEFAULT CHARSET=utf8;

-- Dumping structure for table encompass_model.occupancy_inference

CREATE TABLE IF NOT EXISTS `occupancy_inference` (

  `oid` int(11) NOT NULL AUTO_INCREMENT,

  `dwelling_oid` int(11) NOT NULL,

  `occupancy` int(11) DEFAULT NULL,

  `timestamp` datetime DEFAULT NULL,

  `timestamp_received` datetime DEFAULT NULL,

  PRIMARY KEY (`oid`),

  KEY `fk_Occupancy_dwelling1_idx` (`dwelling_oid`),

  CONSTRAINT `fk_Occupancy_dwelling1` FOREIGN KEY (`dwelling_oid`) REFERENCES `dwelling` (`oid`) ON DELETE NO ACTION ON UPDATE NO ACTION

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Dumping structure for table encompass_model.on_the_fly_message_localization

CREATE TABLE IF NOT EXISTS `on_the_fly_message_localization` (

  `oid` int(11) NOT NULL AUTO_INCREMENT,

  `message_instance_oid` int(11) NOT NULL,

  `language` varchar(255) NOT NULL,

  `title` varchar(100) DEFAULT NULL,

  `description` varchar(255) DEFAULT NULL,

  `message_type` varchar(255) DEFAULT NULL,

  PRIMARY KEY (`message_instance_oid`,`language`),

  KEY `oid` (`oid`),

  CONSTRAINT `fk_message_localization` FOREIGN KEY (`message_instance_oid`) REFERENCES `message_instance` (`oid`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB AUTO_INCREMENT=139 DEFAULT CHARSET=utf8;
```

```
-- Dumping structure for table encompass_model.outdoor_conditions
CREATE TABLE IF NOT EXISTS `outdoor_conditions` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `district_oid` int(11) DEFAULT NULL,
  `temperature` decimal(19,3) DEFAULT NULL,
  `humidity` decimal(19,3) DEFAULT NULL,
  `timestamp` datetime DEFAULT NULL,
  `timestamp_received` datetime DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_outdoor_condition_district_idx` (`district_oid`),
  CONSTRAINT `fk_outdoor_condition_district` FOREIGN KEY (`district_oid`) REFERENCES `district` (`oid`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=385 DEFAULT CHARSET=utf16;
-- Dumping structure for table encompass_model.room_comfort_inference
CREATE TABLE IF NOT EXISTS `room_comfort_inference` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `dwelling_room_oid` int(11) NOT NULL,
  `thermal_comfort` double DEFAULT NULL,
  `visual_comfort` double DEFAULT NULL,
  `timestamp` datetime DEFAULT NULL,
  `timestamp_received` datetime DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_user_comfort_user10_idx` (`dwelling_room_oid`),
  CONSTRAINT `fk_user_comfort_user10` FOREIGN KEY (`dwelling_room_oid`) REFERENCES `dwelling_room` (`oid`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB AUTO_INCREMENT=21 DEFAULT CHARSET=utf8;
-- Dumping structure for table encompass_model.route_config
CREATE TABLE IF NOT EXISTS `route_config` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `route` varchar(255) NOT NULL DEFAULT '0',
  `source_path` varchar(255) NOT NULL DEFAULT '0',
  `error_path` varchar(255) NOT NULL DEFAULT '0',
  `md5_path` varchar(255) NOT NULL DEFAULT '0',
  `md5_check_timeout` int(11) NOT NULL DEFAULT '25' COMMENT 'time in seconds',
  `outbox_path` varchar(255) NOT NULL DEFAULT '0',
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB AUTO_INCREMENT=9 DEFAULT CHARSET=utf8;
```

```
-- Dumping structure for table encompass_model.schedule_dwelling
CREATE TABLE IF NOT EXISTS `schedule_dwelling` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `dwelling_oid` int(11) NOT NULL,
  `time_open` time DEFAULT NULL,
  `time_close` time DEFAULT NULL,
  `time_open_weekend` time DEFAULT NULL,
  `time_close_weekend` time DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_schedule_dwelling_dwelling1_idx` (`dwelling_oid`),
  CONSTRAINT `fk_schedule_dwelling_dwelling1` FOREIGN KEY (`dwelling_oid`) REFERENCES `dwelling` (`oid`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
-- Dumping structure for table encompass_model.semaphore_log
CREATE TABLE IF NOT EXISTS `semaphore_log` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `timestamp` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `component` varchar(50) NOT NULL,
  `process` varchar(50) NOT NULL,
  `status_code` int(11) NOT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB AUTO_INCREMENT=2621 DEFAULT CHARSET=utf16;
-- Dumping structure for table encompass_model.simple_device_instance
CREATE TABLE IF NOT EXISTS `simple_device_instance` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `device_type_oid` int(11) DEFAULT NULL,
  `dwelling_room_oid` int(11) DEFAULT NULL,
  `number` int(11) DEFAULT NULL,
  `nominal_power` decimal(19,3) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_simple_device_instance_devi` (`device_type_oid`),
  KEY `fk_simple_device_instance_hous_idx` (`dwelling_room_oid`),
  CONSTRAINT `fk_simple_device_instance_devi` FOREIGN KEY (`device_type_oid`) REFERENCES `device_type` (`oid`),
  CONSTRAINT `fk_simple_device_instance_hous` FOREIGN KEY (`dwelling_room_oid`) REFERENCES `dwelling_room` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
-- Dumping structure for table encompass_model.smart_meter
```

```
CREATE TABLE IF NOT EXISTS `smart_meter` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `dwelling_oid` int(11) DEFAULT NULL,
  `smart_meter_id` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_smart_meter_building_idx` (`dwelling_oid`),
  CONSTRAINT `fk_smart_meter_building` FOREIGN KEY (`dwelling_oid`) REFERENCES `dwelling` (`oid`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=177 DEFAULT CHARSET=utf8;

-- Dumping structure for table encompass_model.smart_plug
CREATE TABLE IF NOT EXISTS `smart_plug` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `dwelling_room_oid` int(11) DEFAULT NULL,
  `smart_plug_id` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  CONSTRAINT `fk_smart_plug_dwelling_room` FOREIGN KEY (`oid`) REFERENCES `dwelling_room` (`oid`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf16;

-- Dumping structure for table encompass_model.smart_plug_consumption
CREATE TABLE IF NOT EXISTS `smart_plug_consumption` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `smart_plug_oid` int(11) NOT NULL,
  `datetime` datetime DEFAULT NULL,
  `datetime_received` datetime DEFAULT NULL,
  `instant_consumption` decimal(19,3) DEFAULT NULL,
  `accumulative_consumption` decimal(19,3) DEFAULT NULL,
  `unit_of_measurement` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_smart_consumption_smart_plug` (`smart_plug_oid`),
  CONSTRAINT `fk_smart_consumption_smart_plug` FOREIGN KEY (`smart_plug_oid`) REFERENCES `smart_plug` (`oid`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB AUTO_INCREMENT=129 DEFAULT CHARSET=utf16;

-- Dumping structure for table encompass_model.user
CREATE TABLE IF NOT EXISTS `user` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `dwelling_oid` int(11) NOT NULL,
  `username` varchar(255) DEFAULT NULL,
```

```
  `password` varchar(255) DEFAULT NULL,
  `email` varchar(255) DEFAULT NULL,
  `birth_date` datetime DEFAULT NULL,
  `internal` bit(1) DEFAULT NULL,
  `type` varchar(255) DEFAULT NULL,
  `language` varchar(255) DEFAULT NULL,
  `title` varchar(45) DEFAULT NULL,
  `saving_goal_money` int(11) DEFAULT NULL,
  `saving_goal_fun` int(11) DEFAULT NULL,
  `saving_goal_environmental` int(11) DEFAULT NULL,
  `firebasegroup_id` varchar(255) DEFAULT NULL,
  `firebasegroup_name` varchar(255) DEFAULT NULL,
  `motivation` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_user_dwelling1_idx` (`dwelling_oid`),
  CONSTRAINT `fk_user_dwelling1` FOREIGN KEY (`dwelling_oid`) REFERENCES `dwelling` (`oid`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=186 DEFAULT CHARSET=utf8;

-- Dumping structure for table encompass_model.user_comfort_feedback
CREATE TABLE IF NOT EXISTS `user_comfort_feedback` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `user_oid` int(11) NOT NULL,
  `thermal_comfort` double DEFAULT NULL,
  `visual_comfort` double DEFAULT NULL,
  `timestamp` datetime DEFAULT NULL,
  `timestamp_received` datetime DEFAULT NULL,
  `feedback_type` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_user_comfort_user1_idx` (`user_oid`),
  CONSTRAINT `fk_user_comfort_user1` FOREIGN KEY (`user_oid`) REFERENCES `user` (`oid`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB AUTO_INCREMENT=91 DEFAULT CHARSET=utf8;

-- Dumping structure for table encompass_model.user_group
CREATE TABLE IF NOT EXISTS `user_group` (
  `group_oid` int(11) NOT NULL,
  `user_oid` int(11) NOT NULL,
  KEY `fk_user_group_group` (`group_oid`),
```

```
  KEY `fk_user_group_user1_idx` (`user_oid`),
  CONSTRAINT `fk_user_group_group` FOREIGN KEY (`group_oid`) REFERENCES `group` (`oid`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `fk_user_group_user1` FOREIGN KEY (`user_oid`) REFERENCES `user` (`oid`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
-- Dumping structure for table encompass_model.user_preference
CREATE TABLE IF NOT EXISTS `user_preference` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `user_oid` int(11) DEFAULT NULL,
  `language` varchar(255) DEFAULT NULL,
  `timezone` int(11) DEFAULT NULL,
  `receive_notification` tinyint(1) DEFAULT NULL,
  `notification_sent_at_once` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  UNIQUE KEY `oid_UNIQUE` (`oid`),
  KEY `fk_pref_users_idx` (`user_oid`),
  CONSTRAINT `fk_pref_users` FOREIGN KEY (`user_oid`) REFERENCES `user` (`oid`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=182 DEFAULT CHARSET=utf8;
-- Dumping structure for table encompass_model.user_preference_daytime
CREATE TABLE IF NOT EXISTS `user_preference_daytime` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `user_preference_oid` int(11) DEFAULT NULL,
  `preferred_day` varchar(50) DEFAULT NULL,
  `preferred_time_start` varchar(50) DEFAULT NULL,
  `preferred_time_end` varchar(50) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  UNIQUE KEY `oid_UNIQUE` (`oid`),
  KEY `fk_pref_daytime_idx` (`user_preference_oid`),
  CONSTRAINT `fk_pref_daytime` FOREIGN KEY (`user_preference_oid`) REFERENCES `user_preference` (`oid`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=438 DEFAULT CHARSET=utf8;
-- Dumping structure for table encompass_model.user_profile
CREATE TABLE IF NOT EXISTS `user_profile` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `user_oid` int(11) DEFAULT NULL,
  `no_of_adults_older_than_16` int(11) DEFAULT NULL,
```

```
`no_of_kids_younger_than_16` int(11) DEFAULT NULL,
`no_of_pets` int(11) DEFAULT NULL,
`living_since_may_2017` tinyint(1) DEFAULT NULL,
`dwelling_type_oid` int(11) DEFAULT NULL,
`number_of_rooms` int(11) DEFAULT NULL,
`heating_type_oid` int(11) DEFAULT NULL,
`heating_source_type_oid` int(11) DEFAULT NULL,
`heat_pump` tinyint(1) DEFAULT NULL,
`water_boiler` tinyint(1) DEFAULT NULL,
`electric_car` tinyint(1) DEFAULT NULL,
`main_lighting_type_oid` int(11) DEFAULT NULL,
`AC` tinyint(1) DEFAULT NULL,
`electric_oven` tinyint(1) DEFAULT NULL,
`microwave` tinyint(1) DEFAULT NULL,
`electric_hot_plates` tinyint(1) DEFAULT NULL,
`electric_kettle` tinyint(1) DEFAULT NULL,
`coffee_machine` tinyint(1) DEFAULT NULL,
`vacuum_cleaner` tinyint(1) DEFAULT NULL,
`dishwasher` tinyint(1) DEFAULT NULL,
`washing_machine_existence` tinyint(1) DEFAULT NULL,
`washing_machine_energy_info` varchar(255) DEFAULT NULL,
`washing_machine_shared` tinyint(1) DEFAULT NULL,
`tumble_dryer_existence` tinyint(1) DEFAULT NULL,
`tumble_dryer_energy_info` varchar(255) DEFAULT NULL,
`tumble_dryer_shared` tinyint(1) DEFAULT NULL,
`dehumidifier` tinyint(1) DEFAULT NULL,
`fridge` int(11) DEFAULT NULL,
`freezer` int(11) DEFAULT NULL,
`tv_set` int(11) DEFAULT NULL,
`hi_fi` int(11) DEFAULT NULL,
`desktop_computer` int(11) DEFAULT NULL,
`laptop_computer` int(11) DEFAULT NULL,
`gaming_set` int(11) DEFAULT NULL,
`energy_saving_motivation` varchar(255) DEFAULT NULL,
`occupancy_during_day_weekdays` tinyint(1) DEFAULT NULL,
`occupancy_during_night_weekdays` tinyint(1) DEFAULT NULL,
```

```
  `occupancy_during_day_weekend` tinyint(1) DEFAULT NULL,
  `occupancy_during_night_weekend` tinyint(1) DEFAULT NULL,
  `desired_thermal_comfort` int(11) DEFAULT NULL,
  `desired_visual_comfort` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_user_recom_settings_idx` (`user_oid`),
  KEY `fk_dwelling_type_idx` (`dwelling_type_oid`),
  KEY `fk_heating_type_idx` (`heating_type_oid`),
  KEY `fk_heating_source_type_idx` (`heating_source_type_oid`),
  KEY `fk_main_lighting_type_idx` (`main_lighting_type_oid`),
  CONSTRAINT `fk_dwelling_type` FOREIGN KEY (`dwelling_type_oid`) REFERENCES `dwelling_type` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `fk_heating_source` FOREIGN KEY (`heating_source_type_oid`) REFERENCES `heating_source_type` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `fk_heating_type` FOREIGN KEY (`heating_type_oid`) REFERENCES `heating_type` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `fk_main_lighting_type` FOREIGN KEY (`main_lighting_type_oid`) REFERENCES `main_lighting_type` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `fk_user_recom_settings` FOREIGN KEY (`user_oid`) REFERENCES `user` (`oid`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=179 DEFAULT CHARSET=utf8;

-- Dumping structure for table encompass_model.weather_condition
CREATE TABLE IF NOT EXISTS `weather_condition` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `district_oid` int(11) DEFAULT NULL,
  `date_start` date DEFAULT NULL,
  `date_end` date DEFAULT NULL,
  `rain_fall` decimal(19,2) DEFAULT NULL,
  `avg_temperature` decimal(19,2) DEFAULT NULL,
  `avg_co2` decimal(19,2) DEFAULT NULL,
  `avg_humidity` decimal(19,2) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_weather_condition_district` (`district_oid`),
  CONSTRAINT `fk_weather_condition_district` FOREIGN KEY (`district_oid`) REFERENCES `district` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Dumping structure for view encompass_model.get_tips
-- Removing temporary table and create final VIEW structure
DROP TABLE IF EXISTS `get_tips`;
```

CREATE ALGORITHM=UNDEFINED DEFINER=`puser`@`%` SQL SECURITY DEFINER VIEW `get_tips` AS select `mi`.`user_oid` AS `user_oid`,`gm`.`oid` AS `id`,`ml`.`title` AS `name`,'' AS `header`,`ml`.`description` AS `body`,`ml`.`image` AS `image`,`ml`.`video` AS `video`,`mi`.`oid` AS `instance`,`mf2`.`feedback` AS `feedback`,`mi`.`is_static` AS `static`,`mi`.`timestamp_creation` AS `timestamp_creation` from ((((`generic_message` `gm` join `message_instance` `mi` on((`mi`.`generic_message_oid` = `gm`.`oid`))) join `message_localization` `ml` on(((`ml`.`generic_message_oid` = `gm`.`oid`) and (`mi`.`is_static` = 1)))) join `user_preference` `up` on(((`up`.`user_oid` = `mi`.`user_oid`) and (`up`.`language` = `ml`.`language`)))) left join `get_tips_with_feedback` `mf2` on((`mf2`.`message_instance_oid` = `mi`.`oid`))) where (`mi`.`hidden` = 0) union select `mi`.`user_oid` AS `user_oid`,`gm`.`oid` AS `id`,`ofml`.`title` AS `name`,'' AS `header`,`ofml`.`description` AS `body`,NULL AS `image`,NULL AS `video`,`mi`.`oid` AS `instance`,`mf2`.`feedback` AS `feedback`,`mi`.`is_static` AS `static`,`mi`.`timestamp_creation` AS `timestamp_creation` from ((((`generic_message` `gm` join `message_instance` `mi` on((`mi`.`generic_message_oid` = `gm`.`oid`))) join `on_the_fly_message_localization` `ofml` on(((`ofml`.`message_instance_oid` = `gm`.`oid`) and (`mi`.`is_static` = 0)))) join `user_preference` `up` on(((`up`.`user_oid` = `mi`.`user_oid`) and (`up`.`language` = `ofml`.`language`)))) left join `get_tips_with_feedback` `mf2` on((`mf2`.`message_instance_oid` = `mi`.`oid`))) where (`mi`.`hidden` = 0);

-- Dumping structure for view encompass_model.get_tips_with_feedback

-- Removing temporary table and create final VIEW structure

DROP TABLE IF EXISTS `get_tips_with_feedback`;

CREATE ALGORITHM=UNDEFINED DEFINER=`puser`@`%` SQL SECURITY DEFINER VIEW `get_tips_with_feedback` AS select `mf`.`oid` AS `oid`,`mf`.`feedback` AS `feedback`,`mf`.`message_instance_oid` AS `message_instance_oid` from (`message_feedback` `mf` join `message_instance` `mi` on(((`mf`.`message_instance_oid` = `mi`.`oid`) and (`mf`.`message_instance_oid` = `mi`.`oid`)))) where `mf`.`oid` in (select `mf`.`oid` from `message_feedback` `mf` where `mf`.`timestamp` in (select max(`mf`.`timestamp`) from `message_feedback` `mf` group by `mf`.`message_instance_oid`)) order by `mf`.`timestamp` desc;

-- Dumping structure for view encompass_model.meter_consumption_daily_avg

-- Removing temporary table and create final VIEW structure

DROP TABLE IF EXISTS `meter_consumption_daily_avg`;

CREATE ALGORITHM=UNDEFINED DEFINER=`puser`@`%` SQL SECURITY DEFINER VIEW `meter_consumption_daily_avg` AS select `u`.`oid` AS `user_oid`,sum(`mc`.`consumption`) AS `daily_consumption`,dayofmonth(`mc`.`datetime`) AS `day`,month(`mc`.`datetime`) AS `month`,year(`mc`.`datetime`) AS `year` from (((`meter_consumption` `mc` join `smart_meter` `sm` on((`mc`.`smart_meter_oid` = `sm`.`oid`))) join `dwelling` `d` on((`sm`.`dwelling_oid` = `d`.`oid`))) join `user` `u` on((`d`.`oid` = `u`.`dwelling_oid`))) where (cast(`mc`.`datetime` as date) <= cast((now() + interval -(1) day) as date)) group by `u`.`oid`,dayofmonth(`mc`.`datetime`),month(`mc`.`datetime`),year(`mc`.`datetime`) order by `u`.`oid`,year(`mc`.`datetime`),month(`mc`.`datetime`),dayofmonth(`mc`.`datetime`);

-- Dumping structure for view encompass_model.meter_consumption_from_to

-- Removing temporary table and create final VIEW structure

DROP TABLE IF EXISTS `meter_consumption_from_to`;

CREATE ALGORITHM=UNDEFINED DEFINER=`puser`@`%` SQL SECURITY DEFINER VIEW `meter_consumption_from_to` AS select `u`.`oid` AS `user_oid`,min(`mc`.`datetime`) AS `from_`,max(`mc`.`datetime`) AS `to_` from (((`meter_consumption` `mc` join `smart_meter` `sm` on((`mc`.`smart_meter_oid` = `sm`.`oid`))) join `dwelling` `d` on((`sm`.`dwelling_oid` = `d`.`oid`))) join `user` `u` on((`d`.`oid` = `u`.`dwelling_oid`))) where (cast(`mc`.`datetime` as date) <= cast((now() + interval -(1) day) as date)) group by `u`.`oid`;

-- Dumping structure for view encompass_model.meter_consumption_monthly_avg

-- Removing temporary table and create final VIEW structure

DROP TABLE IF EXISTS `meter_consumption_monthly_avg`;

CREATE ALGORITHM=UNDEFINED DEFINER=`admin`@`%` SQL SECURITY DEFINER VIEW `meter_consumption_monthly_avg` AS select uuid() AS `oid`,`u`.`oid` AS `user_oid`,truncate(sum(`mc`.`consumption`),2) AS `monthly_consumption`,month(`mc`.`datetime`) AS `month`,year(`mc`.`datetime`) AS `year` from (((`meter_consumption` `mc` join `smart_meter` `sm` on((`mc`.`smart_meter_oid` = `sm`.`oid`))) join `dwelling` `d` on((`sm`.`dwelling_oid` = `d`.`oid`))) join `user` `u` on((`d`.`oid` = `u`.`dwelling_oid`))) where (cast(`mc`.`datetime` as date) <= cast((now() + interval -(1) day) as date)) group by `u`.`oid`,month(`mc`.`datetime`),year(`mc`.`datetime`);

-- Dumping structure for view encompass_model.meter_consumption_this_month_total

```
-- Removing temporary table and create final VIEW structure

DROP TABLE IF EXISTS `meter_consumption_this_month_total`;

CREATE ALGORITHM=UNDEFINED DEFINER=`puser`@`%` SQL SECURITY DEFINER VIEW `meter_consumption_this_month_total` AS select `u`.`oid` AS `user_oid`,sum(`mc`.`consumption`) AS `this_month_total`,month((now() - interval 1 day)) AS `month`,year(now()) AS `year` from (((`meter_consumption` `mc` join `smart_meter` `sm` on((`mc`.`smart_meter_oid` = `sm`.`oid`))) join `dwelling` `d` on((`sm`.`dwelling_oid` = `d`.`oid`))) join `user` `u` on((`d`.`oid` = `u`.`dwelling_oid`))) where ((month(`mc`.`datetime`) = month((now() - interval 1 day))) and (year(`mc`.`datetime`) = year((now() - interval 1 day))) and (cast(`mc`.`datetime` as date) <= cast((now() + interval -(1) day) as date))) group by `u`.`oid`;

-- Dumping structure for view encompass_model.meter_consumption_this_week_total

-- Removing temporary table and create final VIEW structure

DROP TABLE IF EXISTS `meter_consumption_this_week_total`;

CREATE ALGORITHM=UNDEFINED DEFINER=`puser`@`%` SQL SECURITY DEFINER VIEW `meter_consumption_this_week_total` AS select `u`.`oid` AS `user_oid`,sum(`mc`.`consumption`) AS `this_week_total`,week((now() - interval 1 day),0) AS `week`,year((now() - interval 1 day)) AS `year` from (((`meter_consumption` `mc` join `smart_meter` `sm` on((`mc`.`smart_meter_oid` = `sm`.`oid`))) join `dwelling` `d` on((`sm`.`dwelling_oid` = `d`.`oid`))) join `user` `u` on((`d`.`oid` = `u`.`dwelling_oid`))) where ((week(`mc`.`datetime`,7) = week(now(),7)) and (year(`mc`.`datetime`) = year(now())) and (cast(`mc`.`datetime` as date) <= cast((now() + interval -(1) day) as date))) group by `u`.`oid`;

-- Dumping structure for view encompass_model.meter_consumption_today_total

-- Removing temporary table and create final VIEW structure

DROP TABLE IF EXISTS `meter_consumption_today_total`;

CREATE ALGORITHM=UNDEFINED DEFINER=`admin`@`%` SQL SECURITY DEFINER VIEW `meter_consumption_today_total` AS select `u`.`oid` AS `user_oid`,sum(`mc`.`consumption`) AS `today_total`,dayofmonth((now() - interval 1 day)) AS `day`,month((now() - interval 1 day)) AS `month`,year((now() - interval 1 day)) AS `year` from (((`meter_consumption` `mc` join `smart_meter` `sm` on((`mc`.`smart_meter_oid` = `sm`.`oid`))) join `dwelling` `d` on((`sm`.`dwelling_oid` = `d`.`oid`))) join `user` `u` on((`d`.`oid` = `u`.`dwelling_oid`))) where (cast(`mc`.`datetime` as date) = cast((now() + interval -(1) day) as date)) group by `u`.`oid`;

-- Dumping structure for view encompass_model.meter_consumption_weekly_avg

-- Removing temporary table and create final VIEW structure

DROP TABLE IF EXISTS `meter_consumption_weekly_avg`;

CREATE ALGORITHM=UNDEFINED DEFINER=`puser`@`%` SQL SECURITY DEFINER VIEW `meter_consumption_weekly_avg` AS select `u`.`oid` AS `user_oid`,sum(`mc`.`consumption`) AS `weekly_consumption`,week(`mc`.`datetime`,0) AS `week`,year(`mc`.`datetime`) AS `year` from (((`meter_consumption` `mc` join `smart_meter` `sm` on((`mc`.`smart_meter_oid` = `sm`.`oid`))) join `dwelling` `d` on((`sm`.`dwelling_oid` = `d`.`oid`))) join `user` `u` on((`d`.`oid` = `u`.`dwelling_oid`))) where (cast(`mc`.`datetime` as date) <= cast((now() + interval -(1) day) as date)) group by `u`.`oid`,week(`mc`.`datetime`,0),year(`mc`.`datetime`);

/*!40101 SET SQL_MODE=IFNULL(@OLD_SQL_MODE, '') */;

/*!40014 SET FOREIGN_KEY_CHECKS=IF(@OLD_FOREIGN_KEY_CHECKS IS NULL, 1, @OLD_FOREIGN_KEY_CHECKS) */;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
```

3.2 DISAGGREGATION ENGINE COMPONENT

4.2.1 Producer

```
@ApiOperation(notes = "Run disaggregation ", value = "runDisaggregation", response = User.class)
@GetMapping(value = "/runDisaggregation")
        private ResponseEntity<Object> startExternal(
                                        @RequestParam(name = "date", required = true) @DateTimeFormat(pattern="yyyy-MM-dd") Date date,
                                        @RequestParam(name = "user_oid", required = false) Integer user_oid)
                                throws ParseException, JsonParseException, JsonMappingException, IOException {
```

```
		/* Get DE inputs */
		JSONArray jArray = getDEInputs(date, user_oid);
		System.out.println(jArray.toJSONString());

		/* Call external service */
		String data = basicAuthUsername + ":" + basicAuthPassword;
		String basicAuthStr = new String(Base64.getEncoder().encode(data.getBytes()));
		HttpHeaders httpHeaders = new HttpHeaders();
		httpHeaders.set("Content-Type", "application/json");
		httpHeaders.set("Accept", "application/json");
		httpHeaders.set("Authorization", "Basic " + basicAuthStr);

		RestTemplate restTemplate = new RestTemplate();
		String response = restTemplate.postForObject(url, new HttpEntity <String> (jArray.toJSONString(), httpHeaders), String.class);
		System.out.println(response);

		JSONArray DEOutput = (JSONArray)JSONValue.parse(response);
		System.out.println(DEOutput);

		/* Get dateTimeReceived */
		Date datetimeReceived = Calendar.getInstance().getTime();

		/* Parse result */
		ObjectMapper objectMapper = new ObjectMapper();
		JsonNode disaggregationDataList = objectMapper.readTree(response);

		/* Fill the disaggregation_data table */
	  for (Iterator<JsonNode> iter = disaggregationDataList.elements(); iter.hasNext();) {
		JsonNode node = iter.next();
		DisaggregationData disaggregationData = createNewDisaggregationData(node, datetimeReceived);

		disaggregationDataService.saveDisaggregationData(disaggregationData);
	  }

		Boolean result = true; // Errors are logged and reported in the external service
		if(result) {
			return Response.data(DEOutput);
		} else
			return Response.error(ServiceError.UNEXPECTED_DE_ERROR, req);
	}


private JSONArray getDEInputs (Date date, Integer user_oid) throws ParseException {

	DateFormat df = new SimpleDateFormat("yyyy-MM-dd");

	Date dateFrom = date;
	Date dateTo = addDays(dateFrom, 1);

	List<Dwelling> dwellingList = dwellingService.getAllDwellings();
	JSONArray jArray = new JSONArray();
	for (Dwelling dwelling : dwellingList) {
		// Get user only if building_type must be different from 4(public building) and 5(school)
		Integer building_type_oid = dwelling.getBuilding().getBuildingTypeOid();
		if ( (building_type_oid != 4) && (building_type_oid != 5) ) {
			for (User user : dwelling.getUsers()) {
				/* Check user_oid param */
				if ( (user_oid == null) || (user.getId().intValue() == user_oid.intValue()) ) {
					for (SmartMeter meter : dwelling.getMeters()) {
```

```
if (meter.getDwelling().getId().intValue() == user.getDwelling().getId().intValue()) {

                                                    JSONObject jObject = new JSONObject();
                                                    jObject.put("dwelling_oid", dwelling.getId());
                                                    jObject.put("user_oid", user.getId());
                                                    jObject.put("smart_meter_oid", meter.getId());
                                                    jObject.put("smart_meter_id", meter.getSmartMeterId());
                                                    jObject.put("date", df.format(date));

                                                    // Get meter_consumption
                                                    List<MeterConsumption> meterConsumptionData= meterService.getMeterConsumptionsBySmartMeterAndDateBetween(meter, dateFrom, dateTo);
                                                    JSONArray jArrayMc = new JSONArray();
                                                    for (MeterConsumption meterConsumption : meterConsumptionData) {
                                                        JSONObject jObjectMc= new JSONObject();
                                                        jObjectMc.put("datetime", meterConsumption.getDatetime().toString());
                                                        jObjectMc.put("consumption", meterConsumption.getConsumption());
                                                        jObjectMc.put("datetime_received", meterConsumption.getDatetimeReceived().toString());
                                                        jArrayMc.add(jObjectMc);
                                                    }
                                                    jObject.put("meter_consumption", jArrayMc);

                                                    JSONObject jObjectApp = new JSONObject();

                                                    // Get user appliances
                                                    UserProfile user_profile = user.getUserProfile();
                                                    //System.out.println(user.getId());
                                                    // Set default values for user appliances (in case they are missing)
                                                    user_profile.setFridge(user_profile.getFridge() == null ? FRIDGE_DEF_VAL : user_profile.getFridge());
                                                    user_profile.setWashingMachineExistence((Boolean) (user_profile.getWashingMachineExistence() == null ? WASHING_MACHINE_DEF_VAL : user_profile.getWashingMachineExistence()));
                                                    user_profile.setTumbleDryerExistence((Boolean) (user_profile.getTumbleDryerExistence() == null ? TUMBLE_DRYER_DEF_VAL : user_profile.getTumbleDryerExistence()));
                                                    user_profile.setDishwasher((Boolean) (user_profile.getDishwasher() == null ? DISHWASHER_DEF_VAL : user_profile.getDishwasher()));
                                                    user_profile.setAC((Boolean) (user_profile.getAC() == null ? AC_DEF_VAL : user_profile.getAC()));
                                                    user_profile.setElectricCar((Boolean) (user_profile.getElectricCar() == null ? ELECTRIC_CAR_DEF_VAL: user_profile.getElectricCar()));
                                                    user_profile.setElectricOven((Boolean) (user_profile.getElectricOven() == null ? ELECTRIC_OVEN_DEF_VAL : user_profile.getElectricOven()));
                                                    user_profile.setHeatPump((Boolean) (user_profile.getHeatPump() == null ? HEAT_PUMP_DEF_VAL : user_profile.getHeatPump()));

                                                    jObjectApp.put("fridge", user_profile.getFridge() == 1 ? true : false);
                                                    jObjectApp.put("washing_machine", user_profile.getWashingMachineExistence());
                                                    jObjectApp.put("tumble_dryer", user_profile.getTumbleDryerExistence());
                                                    jObjectApp.put("dishwasher", user_profile.getDishwasher());
                                                    jObjectApp.put("AC", user_profile.getAC());
```

```
                                                    jObjectApp.put("electric_car",
user.getUserProfile().getElectricCar());
                                                    jObjectApp.put("electric_oven",
user.getUserProfile().getElectricOven());
                                                    jObjectApp.put("heat_pump",
user.getUserProfile().getHeatPump());

                                                    jObject.put("appliances", jObjectApp);

                                                    jArray.add(jObject);
                                            }
                                    }
                            }
                    }
            }
    }

private DisaggregationData createNewDisaggregationData(JsonNode node, Date datetimeReceived) throws ParseException {
        DisaggregationData disaggregationData = new DisaggregationData();
        User user = userService.getUser(node.get("user_oid").asInt());
        disaggregationData.setUser(user);
        disaggregationData.setDate(new SimpleDateFormat("yyyy-MM-dd").parse(node.get("date").asText()));
        disaggregationData.setDatetime_received(datetimeReceived);
        disaggregationData.setDatetime_proc(new                                    SimpleDateFormat("yyyy-MM-dd
H:m:s").parse(node.get("datetime_proc").asText()));
        disaggregationData.setFridge(BigDecimal.valueOf(node.get("fridge").asDouble()));
        disaggregationData.setWashing_machine(BigDecimal.valueOf(node.get("washing_machine").asDouble()));
        disaggregationData.setTumble_dryer(BigDecimal.valueOf(node.get("tumble_dryer").asDouble()));
        disaggregationData.setDishwasher(BigDecimal.valueOf(node.get("dishwasher").asDouble()));
        disaggregationData.setAC(BigDecimal.valueOf(node.get("AC").asDouble()));
        disaggregationData.setElectric_car(BigDecimal.valueOf(node.get("electric_car").asDouble()));
        disaggregationData.setElectric_oven(BigDecimal.valueOf(node.get("electric_oven").asDouble()));
        disaggregationData.setHeat_pump(BigDecimal.valueOf(node.get("heat_pump").asDouble()));
        disaggregationData.setOther(BigDecimal.valueOf(node.get("other").asDouble()));
        disaggregationData.setTotal_consumption(BigDecimal.valueOf(node.get("total_consumption").asDouble()));
        disaggregationData.setV_mod(Integer.valueOf(node.get("v_mod").asInt()));
        disaggregationData.setV_dat(Integer.valueOf(node.get("v_dat").asInt()));
        return disaggregationData;
        }
```

4.2.2 Consumer

```
        @ApiOperation(notes = "Get disaggregated data mean ", value = "getDisaggregatedDataMean", response =
DisaggregationDataMeanDto.class)
        @GetMapping(value = "/getDisaggregatedDataMean")
        private ResponseEntity<Object> getDisaggregatedDataMean (
                        @RequestParam(name = "user_oid", required = true) Integer user_oid,
                        @RequestParam(name = "datefrom", required = false) @DateTimeFormat(pattern="yyyy-MM-dd")
Date dateFrom,
                        @RequestParam(name = "dateto", required = false) @DateTimeFormat(pattern="yyyy-MM-dd") Date
dateTo ) {

                if ( (dateFrom == null) || (dateTo == null) ) {
                        dateFrom = getDate(DE_RES_DATEFROM);
                        dateTo = getDate(DE_RES_DATETO);
                }

                List<DisaggregationDataDto> de_data = getDisaggregatedData(user_oid, dateFrom, dateTo);
                DisaggregationDataMeanDto de_mean = new DisaggregationDataMeanDto();

                de_mean.setDate_start(new SimpleDateFormat("yyyy-MM-dd").format(dateFrom));
```

```
                de_mean.setDate_end(new SimpleDateFormat("yyyy-MM-dd").format(dateTo));

                Integer num_days = de_data.size();

                if (num_days >= DAYS_THRESHOLD) {

                        Double sum_fridge=0.0, sum_washing_machine=0.0, sum_tumble_dryer=0.0, sum_dishwasher=0.0,
sum_AC=0.0,
                                sum_electric_car=0.0, sum_electric_oven=0.0, sum_heatpump=0.0, sum_other=0.0,
sum_total_consumption=0.0;

                        for (DisaggregationDataDto de : de_data) {
                                sum_fridge += de.getFridge() == null ? 0.0 : de.getFridge().doubleValue();
                                sum_washing_machine += de.getWashing_machine() == null ? 0.0 :
de.getWashing_machine().doubleValue();
                                sum_tumble_dryer += de.getTumble_dryer() == null ? 0.0 :
de.getTumble_dryer().doubleValue();
                                sum_dishwasher += de.getDishwasher() == null ? 0.0 : de.getDishwasher().doubleValue();
                                sum_AC += de.getAC() == null ? 0.0 : de.getAC().doubleValue();
                                sum_electric_car += de.getElectric_car() == null ? 0.0 : de.getElectric_car().doubleValue();
                                sum_electric_oven += de.getElectric_oven() == null ? 0.0 :
de.getElectric_oven().doubleValue();
                                sum_heatpump += de.getHeat_pump() == null ? 0.0 : de.getHeat_pump().doubleValue();
                                sum_other += de.getOther() == null ? 0.0 : de.getOther().doubleValue();
                                sum_total_consumption += de.getTotal_consumption() == null ? 0.0 :
de.getTotal_consumption().doubleValue();
                        }

                        de_mean.setFridge(new BigDecimal(df.format(sum_fridge)));
                        de_mean.setWashing_machine(new BigDecimal(df.format(sum_washing_machine)));
                        de_mean.setTumble_dryer(new BigDecimal(df.format(sum_tumble_dryer)));
                        de_mean.setDishwasher(new BigDecimal(df.format(sum_dishwasher)));
                        de_mean.setAC(new BigDecimal(df.format(sum_AC)));
                        de_mean.setElectric_car(new BigDecimal(df.format(sum_electric_car)));
                        de_mean.setElectric_oven(new BigDecimal(df.format(sum_electric_oven)));
                        de_mean.setHeat_pump(new BigDecimal(df.format(sum_heatpump)));
                        de_mean.setOther(new BigDecimal(df.format(sum_other)));
                        de_mean.setTotal_consumption(new BigDecimal(df.format(sum_total_consumption)));
                }
                else {
                        return Response.error(ServiceError.INVALID_DE_INPUT, req);
                }

                return Response.data(de_mean);
        }
        @ApiOperation(notes = "Get disaggregated data by user_id and datetime between ", value = "getDisaggregatedData",
response = DisaggregationDataDto.class)
        @GetMapping(value = "/getDisaggregatedData")
        private List<DisaggregationDataDto> getDisaggregatedData(
                        @RequestParam(name = "user_oid", required = true) Integer user_oid,
                        @RequestParam(name = "datefrom", required = true) @DateTimeFormat(pattern="yyyy-MM-dd")
Date dateFrom,
                        @RequestParam(name = "dateto", required = true) @DateTimeFormat(pattern="yyyy-MM-dd") Date
dateTo) {

                List<DisaggregationDataDto> res =
disaggregationDataRepository.getDisaggregationDataByUserIdAndDateBetweenLastReceived(user_oid, dateFrom, dateTo);

                return res;
        }
```

3.3 JAVADOC OF SERVICE INTEGRATION AND COMPONENT ORCHESTRATION

getUserConsumption

```
@GetMapping(value="/users/{id}/consumption")
public org.springframework.http.ResponseEntity<java.lang.Object> getUserConsumption(
@PathVariable(value="id")
java.lang.Integer userId,
 @RequestParam(name="dateStart",required=false)
java.sql.Date dateStart,
@RequestParam(name="dateEnd",required=false)
java.sql.Date dateEnd)
throws java.text.ParseException
```

Calculates the energy consumption of a user identified by ID for a specific interval of time or an INVALID_USER error if the user ID is uknown.

Parameters:

userId - - user id

dateStart - - starting date of the calculation interval

dateEnd - - ending date of the calculation interval

Returns:

energy consumption json

Throws:

java.text.ParseException

See Also:

PathVariable, RequestParam

getUserBaseline

```
@GetMapping(value="/users/{id}/baseline")
public org.springframework.http.ResponseEntity<java.lang.Object> getUserBaseline(
@PathVariable(value="id")
java.lang.Integer userId,
@RequestParam(name="month",required=false)
java.lang.String month)
throws java.text.ParseException
```

Retrives a user baseline energy consumption for a specified month.

Parameters:

userId - - user id

month - - month of the baseline interval

Returns:

month baseline json

Throws:

java.text.ParseException

See Also:

PathVariable, RequestParam

getUserConsumptionSummary

@GetMapping(value="/users/{id}/consumption/summary")
public org.springframework.http.ResponseEntity<java.lang.Object> getUserConsumptionSummary(
@PathVariable(value="id")
java.lang.Integer userId)
throws java.text.ParseException

Retrives a user energy consumption summary.

Parameters:

userId - - user id

Returns:

consumption summary json

Throws:

java.text.ParseException

See Also:

PathVariable, RequestParam

getUserTips

@GetMapping(value="/users/{id}/tips")
public org.springframework.http.ResponseEntity<java.lang.Object> getUserTips(
@PathVariable(value="id")
java.lang.Integer userId)

Retrives energy saving tips recommended for a user.

Parameters:

userId - - user id

Returns:

saving tips json

See Also:

PathVariable, RequestParam

getUserSavings

@GetMapping(value="/users/{id}/savings")
public org.springframework.http.ResponseEntity<java.lang.Object> getUserSavings(
@PathVariable(value="id")
java.lang.Integer userId)

Calculates energy saving of a user.

Parameters:

userId - - user id

Returns:

user saving json

See Also:

PathVariable, RequestParam

putMessageInstanceTimestampRead

@PutMapping(value="/user/message/instance/read")
public org.springframework.http.ResponseEntity<java.lang.Object> putMessageInstanceTimestampRead(
@RequestBody
eu.encompass.common.dto.MessageInstanceTimestampDto messageInstanceTimestamp)
throws java.text.ParseException

Update timestamp read for a notification instance.

Parameters:

messageInstanceTimestamp - - message instance timestamp

Returns:

message instance timestamp json

Throws:

java.text.ParseException

See Also:

MessageInstanceTimestampDto, RequestBody

postMessageInstance

@RequestMapping(value="/user/message/instance", method=POST)
public org.springframework.http.ResponseEntity<java.lang.Object> postMessageInstance(
@RequestBody
eu.encompass.common.dto.MessageInstanceDto messageInstance)
throws java.text.ParseException

Post a message instance for generating a notification.

Parameters:

messageInstance - - message instance

Returns:

message instance json

Throws:

java.text.ParseException

See Also:

MessageInstanceDto, RequestBody

postMessageInstance

@RequestMapping(value="/user/message/instance/list",method=POST)
public org.springframework.http.ResponseEntity<java.lang.Object> postMessageInstance(
@RequestBody
java.util.List<eu.encompass.common.dto.MessageInstanceDto> messageInstanceList)
throws java.text.ParseException

Post a list of message instances for generating a notification list.

Parameters:

messageInstanceList - - message instance list

Returns:

message instance json

Throws:

java.text.ParseException

See Also:

MessageInstanceDto, RequestBody

postRecommendationList

@RequestMapping(value="/user/recommendation/list", method=POST)
public org.springframework.http.ResponseEntity<java.lang.Object> postRecommendationList(
@RequestBody
java.util.List<eu.encompass.common.dto.MessageInstanceDto> messageInstanceList)
throws java.text.ParseException, java.io.IOException

Post a list of recommendation instances for generating a notification list.

Parameters:

messageInstanceList - - message instance list

Returns:

message instance json

Throws:

java.text.ParseException

java.io.IOException

See Also:

MessageInstanceDto, RequestBody

getAverageTemperature

@GetMapping(value="/users/{id}/temperature/average")
public org.springframework.http.ResponseEntity<java.lang.Object> getAverageTemperature(
@PathVariable(value="id")
java.lang.Integer userId,
@RequestParam(name="dateStart",required=false)
java.sql.Date dateStart,
@RequestParam(name="dateEnd",required=false)
java.sql.Date dateEnd)
throws java.text.ParseException

Calculates the temperature average registered in a user dwelling for a specific interval of time.

Parameters:

userId - - user id

dateStart - - starting date of the calculation interval

dateEnd - - ending date of the calculation interval

Returns:

temperature average json

Throws:

java.text.ParseException

See Also:

PathVariable, RequestParam

getAverageHumidity

```
@GetMapping(value="/users/{id}/humidity/average")
public org.springframework.http.ResponseEntity<java.lang.Object> getAverageHumidity(
@PathVariable(value="id")
java.lang.Integer userId,
@RequestParam(name="dateStart",required=false)
java.sql.Date dateStart,
@RequestParam(name="dateEnd",required=false)
java.sql.Date dateEnd)
throws java.text.ParseException
```

Calculates the humidity average registered in a user dwelling for a specific interval of time.

Parameters:

userId - - user id

dateStart - - starting date of the calculation interval

dateEnd - - ending date of the calculation interval

Returns:

humidity average json

Throws:

java.text.ParseException

See Also:

PathVariable, RequestParam

getAverageLuminance

```
@GetMapping(value="/users/{id}/luminance/average")
public org.springframework.http.ResponseEntity<java.lang.Object> getAverageLuminance(
@PathVariable(value="id")
java.lang.Integer userId,
@RequestParam(name="dateStart", required=false)
java.sql.Date dateStart,
@RequestParam(name="dateEnd", required=false)
java.sql.Date dateEnd)
throws java.text.ParseException
```

Calculates the luminance average registered in a user dwelling for a specific interval of time.

Parameters:

userId - - user id

dateStart - - starting date of the calculation interval

dateEnd - - ending date of the calculation interval

Returns:

luminance average json

Throws:

java.text.ParseException

See Also:

PathVariable, RequestParam

postSettings

```
@PostMapping(value="/users/{id}/settings")
public org.springframework.http.ResponseEntity<java.lang.Object> postSettings(
@PathVariable(value="id")
java.lang.Integer userId,
@RequestBody
eu.encompass.common.dto.UserPreferenceDto userPref)
throws java.text.ParseException
```

Posts the user settings.

Parameters:

userId - - user id

userPref - - user preferences

Returns:

user settings json

Throws:

java.text.ParseException

See Also:

RequestBody, Response

getSettings

```
@GetMapping(value="/users/{id}/settings")
public org.springframework.http.ResponseEntity<java.lang.Object> getSettings(
@PathVariable(value="id")
java.lang.Integer userId)
throws java.text.ParseException
```

Retrieves the user settings.

Parameters:

userId - - user id

Returns:

user settings json

Throws:

java.text.ParseException

See Also:

RequestBody, Response

setProfile

@PostMapping(value="/users/{id}/profile")
public org.springframework.http.ResponseEntity<java.lang.Object> setProfile(
@PathVariable(value="id")
java.lang.Integer userId,
@RequestBody
eu.encompass.common.dto.UserProfilingDto userProfiling)
throws java.text.ParseException

Updates the user profile information.

Parameters:

userId - - user id

userProfiling - - user profiling information

Returns:

user profiling information json

Throws:

java.text.ParseException

See Also:

RequestBody, Response

getProfile

@GetMapping(value="/users/{id}/profile")
public org.springframework.http.ResponseEntity<java.lang.Object> getProfile(
@PathVariable(value="id")
java.lang.Integer userId)
throws java.text.ParseException

Retrieves user profile information.

Parameters:

userId - - user id

Returns:

user profiling information json

Throws:

java.text.ParseException

See Also:

RequestBody, Response

postComponentSubprocessResult

@PostMapping(value="/semaphore/subprocess/result")

public org.springframework.http.ResponseEntity<java.lang.Object> postComponentSubprocessResult(
@RequestBody
java.util.List<eu.encompass.common.domain.semaphore.SemaphoreLog> semaphoreLogs)
throws java.text.ParseException,
com.fasterxml.jackson.core.JsonParseException,
com.fasterxml.jackson.databind.JsonMappingException,
java.io.IOException

Posts the result returned by a process component after execution when called by the orchestration logic.

Parameters:

semaphoreLogs - - semaphore state

Returns:

state of the component after execution when called by the orchestration logic json

Throws:

java.text.ParseException

com.fasterxml.jackson.core.JsonParseException

com.fasterxml.jackson.databind.JsonMappingException

java.io.IOException

See Also:

SemaphoreLog, Response

postComponentProcessingFinalization

@PostMapping(value="/semaphore/result")
public org.springframework.http.ResponseEntity<java.lang.Object> postComponentProcessingFinalization(
@RequestBody
java.util.List<eu.encompass.common.domain.semaphore.SemaphoreLog> semaphoreLogs)
throws java.text.ParseException,
com.fasterxml.jackson.core.JsonParseException,
com.fasterxml.jackson.databind.JsonMappingException,
java.io.IOException

Posts component finalization status after execution when called by the orchestration logic.

Parameters:

semaphoreLogs - - semaphore state

Returns:

finalization state of the component after orchestration execution json

Throws:

java.text.ParseException

com.fasterxml.jackson.core.JsonParseException

com.fasterxml.jackson.databind.JsonMappingException

java.io.IOException

See Also:

RequestBody, Response

4 REFERENCES

- Git for Windows, Retrieved from https://gitforwindows.org/
- Apache, Maven. (n.d.). Retrieved from https://maven.apache.org/
- IFML Edit, Retrieved from http://ifmledit.org/
- IBM, Service-Oriented Analysis and Design. (n.d.). Retrieved from https://www.ibm.com/developerworks/library/ws-soad1/index.html
- Pivotal Software, SpringBoot. (n.d.). Retrieved from Spring: https://projects.spring.io/spring-boot/
- RedHat, Hibernate ORM. (n.d.). Retrieved from http://hibernate.org/orm/
- RM-ODP. (n.d.). Retrieved from http://www.rm-odp.net/
- SmartBear Software, Swagger. (n.d.). Retrieved from https://swagger.io/
- Zachman, J. A. (n.d.). Retrieved from https://www.zachman.com/about-the-zachman-framework