



(FP7 614100)

D6.5 Implementation of Context Modelling Tool and Templates

Published by the IMPReSS Consortium

Dissemination Level: Public



**Project co-funded by the European Commission within the 7th Framework Programme
Objective ICT-2013.10.2 EU-Brazil research and development Cooperation
Target Outcome: b) Sustainable technologies for a Smarter Society**

Document control page

Document file: D6.5 Implementation of Context Modelling Tool and Templates.docx
Document version: 1.0
Document owner: Carlos Kamienski (UFABC)

Work package: WP6 – Software System Engineering and Context Management
Task: Task 6.4 Context Model & Rule Authoring tool
Deliverable type: P

Document status: approved by the document owner for internal review
 approved for submission to the EC

Document history:

Version	Author(s)	Date	Summary of changes made
0.1	Gabriela Oliveira Biondi	07/08/2015	Initial version
0.2	Carlos Alberto Kamienski	19/08/2015	Context Modelling Tool
0.9	Carlos Alberto Kamienski	23/08/2015	First version ready for internal review
1.0	Carlos Alberto Kamienski	26/08/2015	Final version

Internal review history:

Reviewed by	Date	Summary of comments
José Ángel Carvajal Soto	24/08/2015	Small comments and recommendations
Stênio F. L. Fernandes	26/08/2015	Small comments and fixes

Legal Notice

The information in this document is subject to change without notice.

The Members of the Impress Consortium make no warranty of any kind with regard to this document, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Members of the Impress Consortium shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Possible inaccuracies of information are under the responsibility of the project. This report reflects solely the views of its authors. The European Commission is not liable for any use that may be made of the information contained therein.

Index:

- 1. Executive summary 4**
- 1. Introduction 5**
 - 1.1 Purpose and context of this deliverable5
 - 1.2 Scope of this deliverable.....5
 - 1.3 Document Structure.....5
- 2. Context Management Middleware 6**
 - 2.1 Context Entities and Templates6
 - 2.2 Framework Architecture7
 - 2.3 Context Storage8
 - 2.4 Context API9
- 3. Context Modelling Tool 10**
 - 3.1 Place Entity 10
 - 3.2 Resource Entity 11
 - 3.3 Fusion Entity 13
 - 3.4 Rule Entity 14
 - 3.5 Action Entity 15
 - 3.6 Context Entity 16
 - 3.6.1 Context-Type 17
 - 3.6.2 Context Definition Interface 17
 - 3.6.3 Context Graphs 19
 - 3.6.4 Context Log 20
 - 3.6.5 Tracking and Counting Context Execution 21
 - 3.6.6 Tracking Contexts with Nested Fusions 23
 - 3.7 Activity Entity (Schedule) 24
- 4. Conclusion 26**
- References 27**

1. Executive summary

IMPreSS aims at providing a Systems Development Platform (SDP) for enabling rapid development of mixed critical complex systems involving Internet of Things and Services (IoTS). The demonstration and evaluation of the IMPReSS platform focuses on energy efficiency systems addressing the reduction of energy usage and CO² footprint in public buildings. Application developers can develop applications using the SDP for a variety of purposes, including energy efficiency management. In order to provide an efficient use of energy in buildings, the IMPReSS SDP will need to be context aware, which means that it must know what happens inside the buildings so that opportunities to save energy can be identified and effectively fulfilled. Context-aware systems are able to adapt their operations according to the current conditions without any explicit user intervention.

Work package 6 provides context entities and templates for energy efficiency applications as well as the Context Manager, a middleware component. The Context Manager encompasses all background software components that a typical context-aware middleware offers to its users, such as context templates, context models, context reasoning engine, and algorithms for sensor and data fusion. The implementation of the Context Manager was presented in Deliverable D6.4, according to the specification presented previously in Deliverable D6.3.

This document presents the main interface (i.e., the IMPReSS Context Web UI) for accessing the features provided by the Context Manager via a REST API, a Context Modelling Tool. The Web UI allows users to perform CRUD (Create, Read, Update, Delete) operations upon seven entities that may be used to develop a typical context-aware building automation application: Resource (sensors/actuators), Place (rooms, floors), Fusion (data aggregation), Rule (decisions), Action (commands to actuators), Activity (scheduled activities), and Context (combination of Resource, Place, Fusion, Rule and Action). The Context Web UI is one example of many different interfaces that may be developed for interacting with the Context Manager via the Context API, such as a smartphone/table application (mobile app) or even a CLI (Command-Line Interface). Particularly, the Context Web UI is based on AngularJS, an open-source web application framework developed by Google for simple-page Web applications.

The Context Web UI has been developed with two primary goals. Firstly, to explore and demonstrate the features provided by the Context Manager. Secondly, for making it easier for developers or integrators to configure energy efficiency context management applications.

1. Introduction

1.1 Purpose and context of this deliverable

The aim of the IMPReSS project is to provide a Systems Development Platform (SDP) that enables rapid and cost effective development of mixed criticality complex systems involving Internet of Things and Services (IoTS) and at the same time facilitates the interplay with users and external systems. The IMPReSS development platform will be usable for any system intended to embrace a smarter society. The demonstration and evaluation of the IMPReSS platform will focus on energy efficiency systems addressing the reduction of energy usage and CO₂ footprint in public buildings, enhancing the intelligence of monitoring and control systems as well as stimulating user energy awareness.

The present document is an output of Task 6.4 (Context Model & Rule Authoring Tool), which aims at developing a tool for allowing the configuration of entities belonging to the context model of an energy-efficiency management application, including sensors, fusion and rules. Deliverable D6.4 presented the implementation of the IMPReSS Context Manager, the component in charge of adapting system behavior according the changes in the context. The Context Manager offers a REST API that may be used by different client applications implementing user interfaces, such as a typical Graphical User Interface (GUI), a Web User Interface (Web UI), an App User Interface or even a Command Line Interface (CLI). This document presents the IMPReSS Context Web UI that may be used by IMPReSS application developers as well as by IMPReSS solution integrators.

1.2 Scope of this deliverable

In order to allow applications to make efficient use of energy in buildings, the IMPReSS Platform must provide context-aware management features, so that automatic decisions can be made based on existing context information coming from a variety of sources, including physical sensors, timetables, and business rules. The implementation of the IMPReSS Context Web UI is a key achievement for the IMPReSS project, since it allows users to interact with the Context Manager.

This deliverable aims at providing a clear understanding the Context Web UI focusing on its main features. It is not a user guide or manual, but may be used to help users in understanding how the Context Web UI and the Context Manager work.

1.3 Document Structure

The remainder of this document is organized into three chapters.

- Chapter 2 summarizes the key features of the Context Manager, already introduced in Deliverable D6.4, but now with some updates reflecting the new developments.
- Chapter 3 introduces the Context Management Tool, also known as the Context Web UI, presenting the interfaces for dealing with the seven IMPReSS Context Entities, namely Place, Resource, Fusion, Rule, Action, Context and Activity.
- Chapter 4 presents some final remarks and the next steps.

2. Context Management Middleware

2.1 Context Entities and Templates

The use of entities and templates for energy efficiency context management aims at making it easier to understand, model, and program the context-awareness features of the IMPReSS project. This section provides an update of deliverable D6.4 (Kamienski et al. 2015), based on the most recent developments of context model and reasoner.

Through an extensive requirements analysis, we identified seven entities that commonly exist in typical context-aware building automation applications: Resource (sensors/actuators), Place (rooms, floors), Fusion (data aggregation), Rule (decisions), Action (commands to actuators), Activity (scheduled activities), and Context (combination of Resource, Place, Fusion, Rule and Action). Entities have templates with their attributes that are involved in the process of managing energy efficiency context. Figure 1 depicts the relationships among the eight entities, which are:

- Place has Resource: A Resource is always located in a Place;
- Resource influences Rule: Rules are influenced by Resource usage;
- Fusion uses Resource: Fusion criteria combines data coming from sensors, which are classified as Resources;
- Fusion fires Rule: Rules are fired by sensor data that are combined by a Fusion criteria;
- Rule relates to Place: Rules affect and are affected by Places;
- Rule performs Action: when a Rule is fired, its processing results in one or more Actions to be performed;
- Activity fires Rule: the schedule of Activities may fire Rules regardless of data coming from sensors;
- Context contains Entities: the Context entity represents a given context that is considered important to be modeled by any application and it contains Place, Resource, Fusion, Rule, and Action.

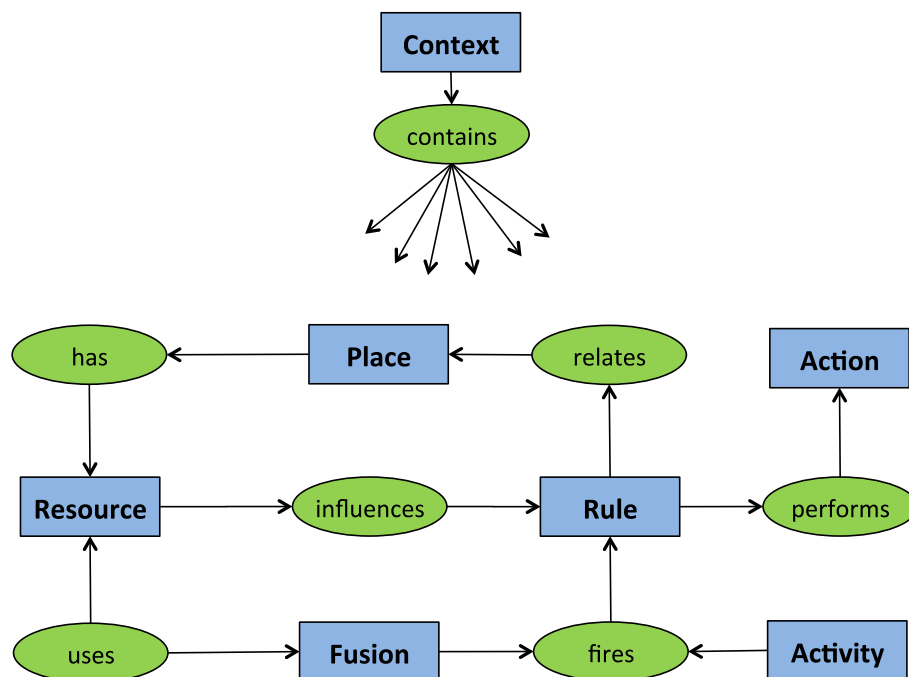


Figure 1 – IMPReSS Context Entities and Relationships

2.2 Framework Architecture

According to the IMPReSS Software Architecture, introduced in IMPReSS Deliverable D2.2.1 (Kamiński 2014a), the Context Manager is a module of the IMPReSS Middleware in charge of providing background software components that a typical context-aware middleware offers to its users, such as context templates, context models, context reasoning engine, and algorithms for sensor and data fusion. This section introduces the Context Management Framework Architecture, also known as Context Manager, introduced in IMPReSS Deliverable D6.3 (Kamiński 2014b). The Context Manager is based on object-oriented context modeling and rule-based context reasoning.

Figure 2 depicts the Context Manager Architecture, which contains the following modules:

- Context API: part of the IMPReSS Middleware API. It provides a REST interface, allowing other modules to interact with the Context Manager. CRUD operations are executed upon context entities through the Context API by a variety of different applications, such as the Context Web UI presented in this document.
- Context Storage: deals with storage and retrieval of context entity templates, via the Context API. Any Relational Database Management System (RDBMS) with an Object-Relational Mapping (ORM) system may be used. We used EclipseLink as ORM, along with PostgreSQL RDBMS.
- Reasoner: infers logical consequences from a set of facts. The Reasoner is invoked by the Fuser and reads entities from the Context Storage. When it is invoked with a set of parameters it searches the entire set of rules for a match, i.e., a rule that matches the parameters. In case of rule conflicts, the Reasoner must select only one rule to be executed based on some resolution mechanism. As a result of firing a rule, one or more actions are performed and they usually refer to changing the configuration of devices or equipments for dynamically adapting behavior, e.g. turning off an elevator or lowering the temperature of an air conditioner. The Reasoner performs this task by sending command messages to actuators through the Communication Proxy. Our implementation is based on Drools¹ (Expert and Workbench).
- Fuser: responsible for data fusion, which means the use of a set of techniques for combining data from multiple sources or computing statistics. The Fuser is directly connected to the Communication Proxy for receiving real-time sensor data and when fusion criteria are met it activates the Reasoner and stores the fused results. Also, fused data may become a virtual sensor and be redirected back to the Fuser. Multiple fusion criteria may be active concurrently and therefore this module plays a key role for the performance of the Context Manager. In our implementation, fusion is performed by Esper², which can perform Complex Event Processing (CEP) by filtering, analyzing, and fusing events in various ways, configurable through an SQL-like language.
- Scheduler: Manages the agenda for prescheduled events (such as classes in a university) and fires the Reasoner for taking appropriate actions.
- Communication Proxy: encapsulates communication with resources, interfacing with the Communication Manager and with a MQTT³ broker.
- Local Data Storage: implements internal data storage, for sensor data, fused data, and event logging.

¹ www.drools.org

² www.espertech.com

³ mqtt.org

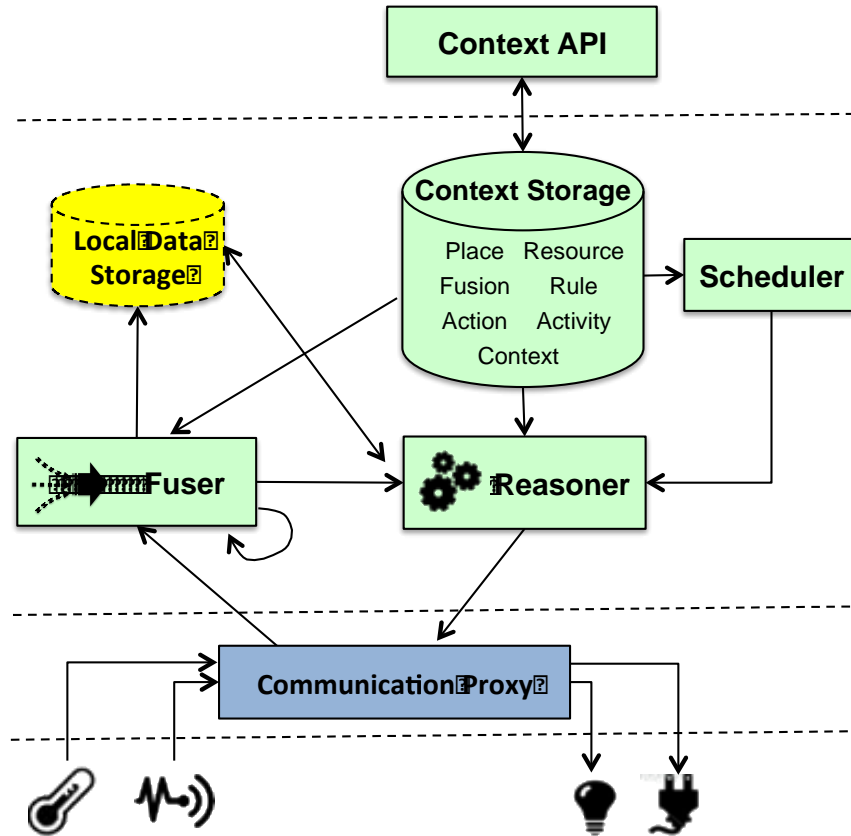


Figure 2 – IMPReSS Context Manager Architecture

2.3 Context Storage

The Context Manager is based on an object-oriented modelling, since it easily integrates with current programming languages, such as Java. However, when it comes to storing the objects in secondary storage, the most common way currently in use is by converting them into a relational database. This solution requires: a) a relational database (PostgreSQL, in the current version); b) a mapping between objects represented in the programming language and tables in the database; c) an efficient modeling of the database for representing the context entities templates.

We developed a relational modelling for storing entity templates, which operates in two categories: Operation and Log, which are represented in blue and orange in Figure 3 respectively.

Tables belonging to the “how to” category are aimed at storing the entity templates, such as places, sensors, actuators and other preferences. On the other hand, tables belonging to the “operation log” category store log information of all actions executed by the Context Manager, such as actions executed by actuators, fusion outcomes, etc.

Figure 3 depicts the diagram that represents a relational model developed for the Context Manager. In a relational model, each entity (table) is represented as a rectangle. Relationships between entities are represented as continuous lines with symbols in their ends representing the type of relationship, which in our case may be one-to-one/many or one-to-zero/one/many. Please notice that entity PLACE, which stores all “places” of importance to the system, has self-relationship, because a place may be located inside another place. For example, a “classroom” place is within a “corridor” place, which in turn is within a “floor” place, which finally is within a “building” place. A self-relationship is implemented by using a foreign key filled in with a value coming from a primary key from the same table. Some tables may play a temporary and internal role and may be substituted when all modules of the IMPReSS architecture are integrated into the platform, because some data types (entities) will be stored inside other modules of the IMPReSS platform. This

relational model has been implemented in PostgreSQL and the creation script can be found in Annex A.

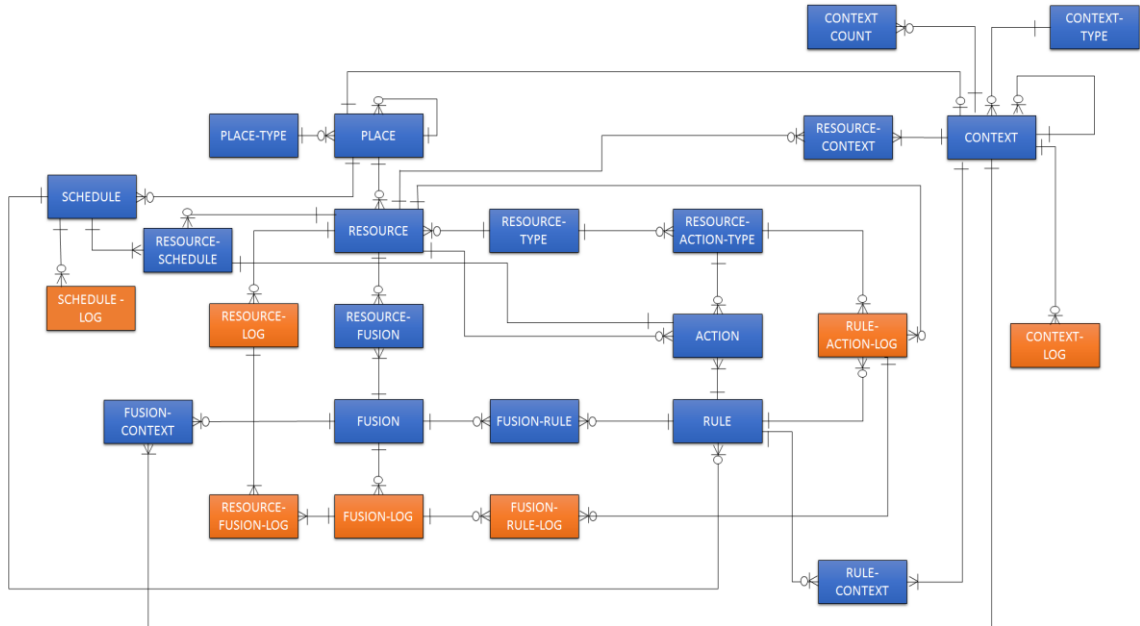


Figure 3 – Relational database of the IMPReSS Context Manager

2.4 Context API

In order to provide a uniform and standard interface for external access as well as to provide data integrity to the Context Manager, all CRUD operations are performed through a REST API. Basically, a REST API uses four methods: GET, DELETE, POST, and PUT. Appendix B contains a detailed list of all methods belonging to the Context Manager API.

Table 1 – Mapping between CRUD operation and REST Method

Operation	Method
Create	POST
Read	GET
Update	PUT
Delete	DELETE

The Swagger framework is used to generate automatic documentation of the REST API developed for the Context Manager of the IMPReSS platform. This framework is based on a Graphical User Interface and provides searching operations and REST URLs.

3. Context Modelling Tool

This section introduces the Context Modelling Tool, also known as Context Web User Interface (Web UI), used for performing CRUD (Create, Read, Update, Delete) operations on the seven context entities presented in section 2.1, namely Place, Resource, Fusion, Rule, Action, Context, and Activity. In fact, the Context Web UI is one example of many different interfaces that may be developed for interacting with the Context Manager via the Context API (section 2.4), such as a smartphone/table application (mobile app) or even a CLI (Command-Line Interface). Particularly, the Context Web UI is based on AngularJS, an open-source web application framework developed by Google for simple-page Web applications⁴.

The Context Web UI has been developed with two primary goals:

1. Exploring and showing the features provided by the Context Manager;
2. Making it easier for developers or integrators to configure energy efficiency context management applications.

Therefore, functionality is the main purpose of the Web UI so that items purely related to design were not included in this version, being considered out of the scope. Figure 4 depicts the main screen for the Context Web UI, with the lateral menu where users can access configuration options for all entities.



Figure 4 – Context Web UI - Home

3.1 Place Entity

The Place Context Entity is a container for a series of events that happen in physical places, open or close, such as rooms. Particularly, the specification of places can be nested inside other places, thus making it possible a recursive modeling style. In a university, examples of places are classroom, office, amphitheater, hall, building and floor.

Places have multiple attributes and one of them is type, which given its importance to the implementation of the Context Manager is modeled as an auxiliary entity called Place-Type. Figure 5 depicts the screen for configuring Place-Type entities.

⁴ <https://angularjs.org>

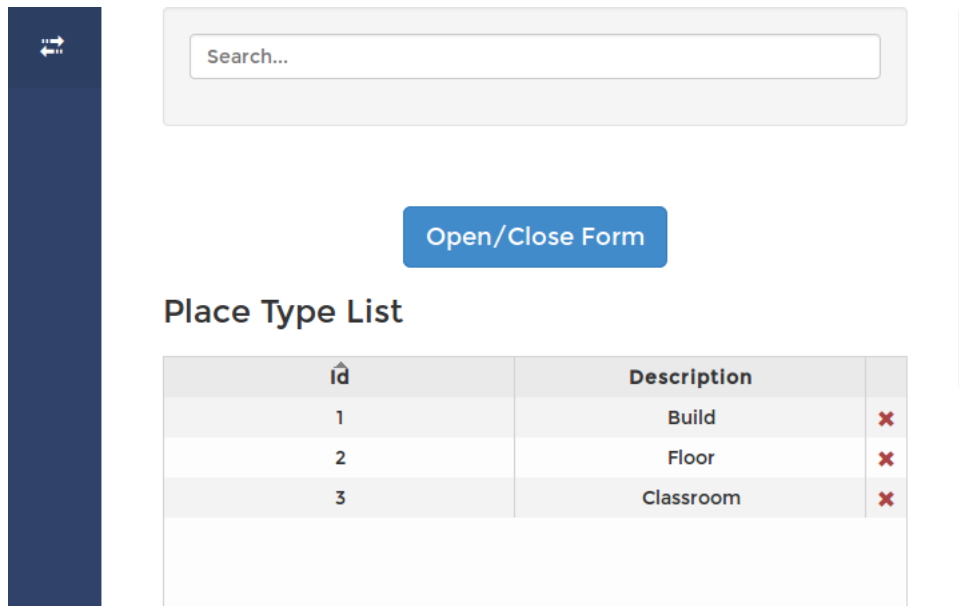


Figure 5 – Context Web UI – Place Type

Figure 6 depicts the screen for creating, reading, updating, and deleting Place entries. Some attributes of Place are ID, description, type, location, and dependence. The latter is used to for stating whether the existence of this place is dependent or independent of a Context Entity. Dependent places will be automatically deleted whenever the linked context is deleted and independent ones will be preserved.

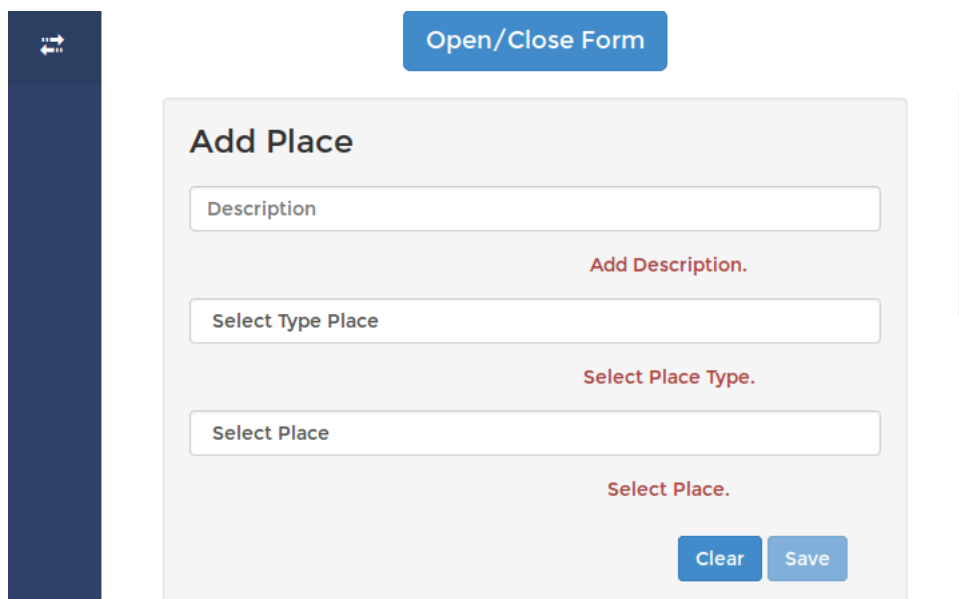


Figure 6 – Context Web UI – Place

3.2 Resource Entity

Resources are used mainly to represent sensors and actuators, but also to represent any equipment that is worth to be identified in an energy efficiency management application, such as air conditioners, elevators, and water pumps. As for Places, Resources have multiple attributes and one of them is type, which given its importance to the implementation of the Context Manager is modeled as an auxiliary entity called Resource-Type, which can be lighting or temperature sensor, for instance. Figure 7 depicts the screen for configuring Resource-Type entities.

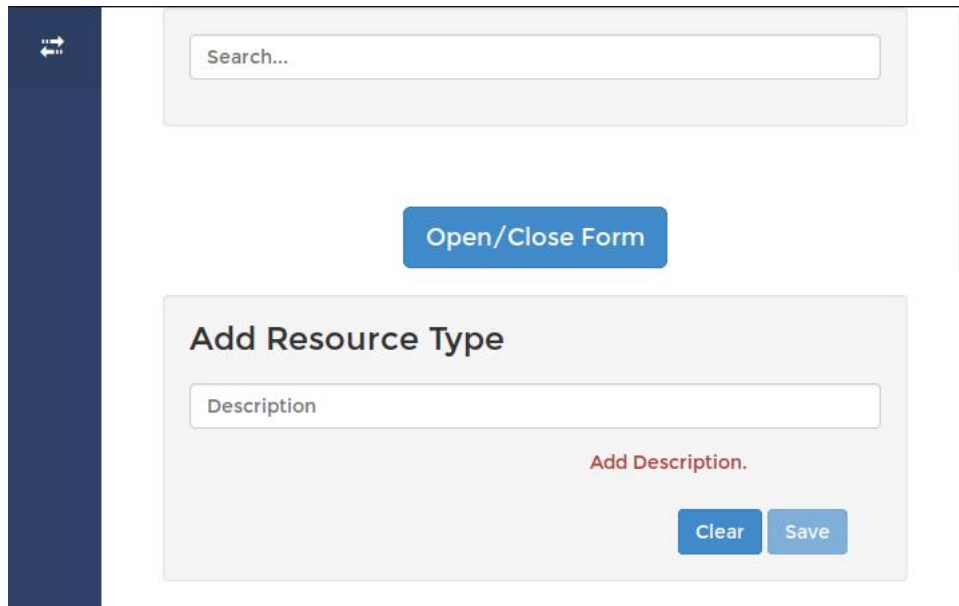


Figure 7 – Context Web UI – Resource Type

Also, Figure 8 depicts the screen for creating, reading, updating, and deleting Resource entries. Some attributes of Place are ID, description, type, location, and dependence. The latter is used to for stating whether the existence of this resource is dependent or independent of a Context Entity. Dependent resources will be automatically deleted whenever the linked context is deleted and independent ones will be preserved. Some examples of resources could be a temperature sensor T1310 or an air-conditioner actuator AC1320.

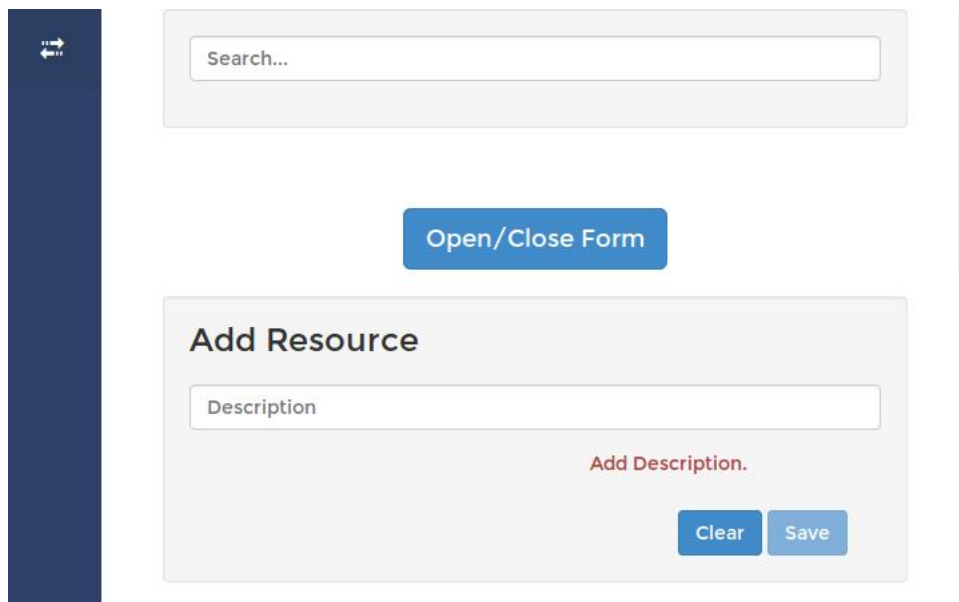


Figure 8 – Context Web UI – Resource

The Context Web UI also allows the visualization of resource logs, such as data coming from sensors or commands sent to actuators, as shown by Figure 9.

The screenshot shows a web interface with a dark blue sidebar on the left containing navigation icons. At the top right is a search bar with the placeholder text 'Search...'. Below the search bar is the heading 'Resource Log' followed by a table with the following data:

Id	Resource	Value	Timestamp
1	LS0088	60%	1886686544000
2	T1310	23.4	1886686547000
3	LS0088	55%	1886686554000
4	T1310	22.9	1886686557000
5	L8001	Off	1886686614000
6	L8002	Off	1886686667000
7	AC1320	On	1886687207000
8	AC1320	On	1886687207000

Figure 9 – Context Web UI – Resource Log

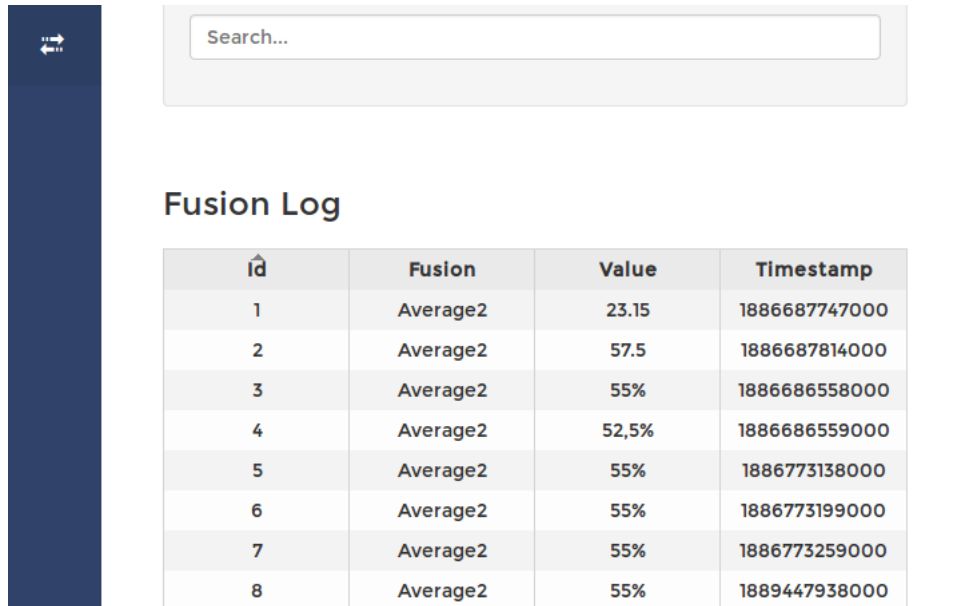
3.3 Fusion Entity

Data Fusion in the IMPReSS Context Manager is represented by Esper Streams (called fusion criteria) and they are used to prepare and combine data in a way that is more adequate for context reasoning. Figure 10 depicts the interface for configuring fusion criteria. Some examples might be to compute the 10-minute temperature average of a classroom or the light intensity of the front part of the classroom.

The screenshot shows the 'Add Fusion' configuration interface. It features a search bar at the top with the placeholder 'Search...'. Below the search bar is a blue button labeled 'Open/Close Form'. The main form area is titled 'Add Fusion' and contains a text input field labeled 'Description'. Below the input field is a red text prompt 'Add Description.'. At the bottom right of the form are two blue buttons: 'Clear' and 'Save'.

Figure 10 – Context Web UI – Fusion

Whenever a fusion criterion is executed, the execution context and results are logged in the local database, which are available for query and visualization, as shown by Figure 11.

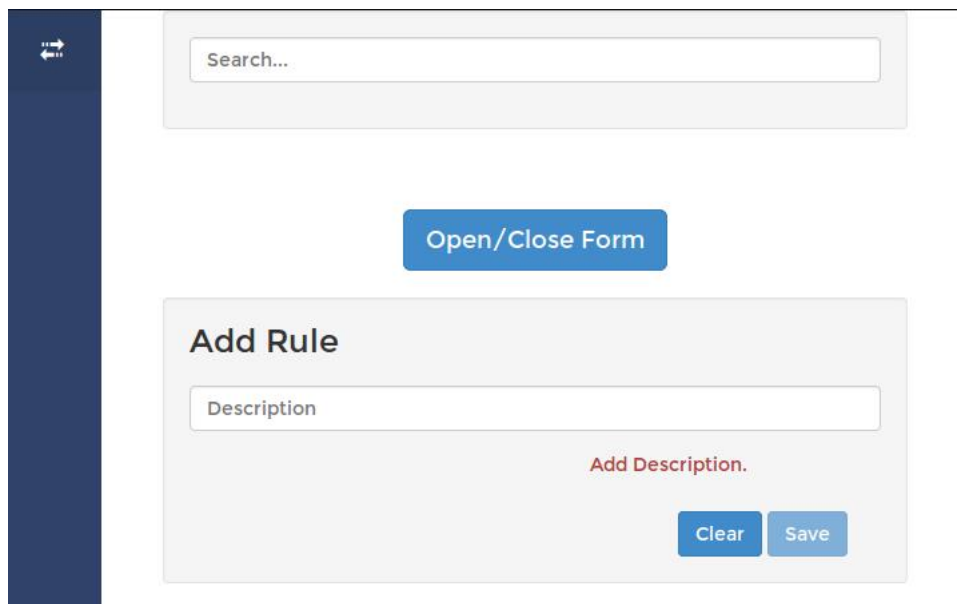


Id	Fusion	Value	Timestamp
1	Average2	23.15	1886687747000
2	Average2	57.5	1886687814000
3	Average2	55%	1886686558000
4	Average2	52,5%	1886686559000
5	Average2	55%	1886773138000
6	Average2	55%	1886773199000
7	Average2	55%	1886773259000
8	Average2	55%	1889447938000

Figure 11 – Context Web UI – Fusion Log

3.4 Rule Entity

Rules perform context reasoning in the IMPReSS Context Manager, which allows system behavior to change automatically, without any human intervention. The key player for on-the-fly changing behaviour is the rule engine (Drools) that receives data coming from the fusion engine (Esper), analyzes a set of rules and decides whether actions should be taken. Figure 12 depicts the interface for rules configuration. Some examples are to turn on or off the air conditioner when temperature is above or below a given threshold or to switch lights on or off in certain parts of a classroom depending on its occupancy.



Search...

Open/Close Form

Add Rule

Description

Add Description.

Clear Save

Figure 12 – Context Web UI – Rule

Whenever a rule is executed, the execution context and results are logged in the local database, which are available for query and visualization, as shown in Figure 13.

The screenshot shows a web interface with a dark blue sidebar on the left containing a home icon. At the top right is a search bar with the placeholder text 'Search...'. Below the search bar is the heading 'Rule Log' followed by a table with three columns: 'Id', 'Rule', and 'Timestamp'.

Id	Rule	Timestamp
1	TempHot	1886690754000
2	TempHot	1886690754000
3	LightClearly	1886693214000
4	LightClearly	1886693214000
5	LightClearly	1886700414000
6	LightClearly	1886700414000
7	LightClearly	1886614014000
8	LightClearly	1886614014000

Figure 13 – Context Web UI – Rule Log

3.5 Action Entity

Actions implement the concept of changing behavior automatically as an outcome of executing a rule and they do this by sending commands to actuators. As for other entities, Actions have multiple attributes and one of them is type, which given its importance to the implementation of the Context Manager is modeled as an auxiliary entity called Resource-Action-Type. This entity relates actions with resources, for example, switch on lights or turn off air conditioner. Figure 14 depicts the interface for configuring Resource-Action-Type entities.

The screenshot shows a web interface with a dark blue sidebar on the left containing a home icon. At the top right is a search bar with the placeholder text 'Search...'. Below the search bar is a blue button labeled 'Open/Close Form'. Below the button is the heading 'Resource Action Type List' followed by a table with four columns: 'Id', 'Value', 'Resource Type', and an action icon.

Id	Value	Resource Type	
1	(1,1,On)	Air Conditioning	✘
2	(2,1,Off)	Air Conditioning	✘
3	(3,1,Increase)	Air Conditioning	✘
4	(4,1,Decrease)	Air Conditioning	✘
5	(5,4,On)	Lamp	⌵

Figure 14 – Context Web UI – Resource-Action Type

In addition, Figure 15 depicts the screen for creating, reading, updating, and deleting Action entries. Some attributes of Action are ID, rule, resource, and resource-action-type. An example of action is Turn ON AC1320 (a resource of the type air conditioner actuator).

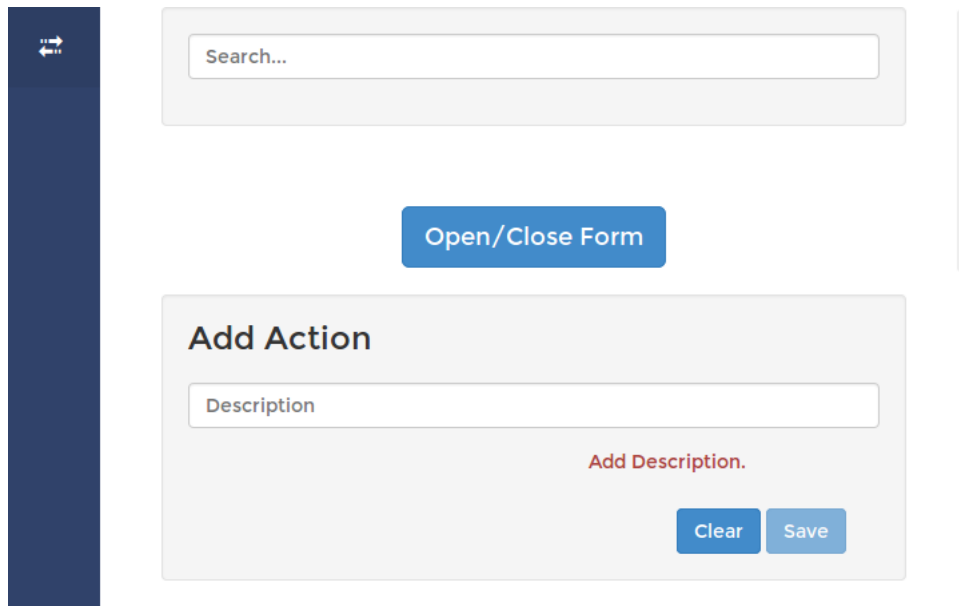


Figure 15 – Context Web UI – Action

Whenever a certain action is executed, the execution context is logged in the local database, which is available for query and visualization, as shown in Figure 16.

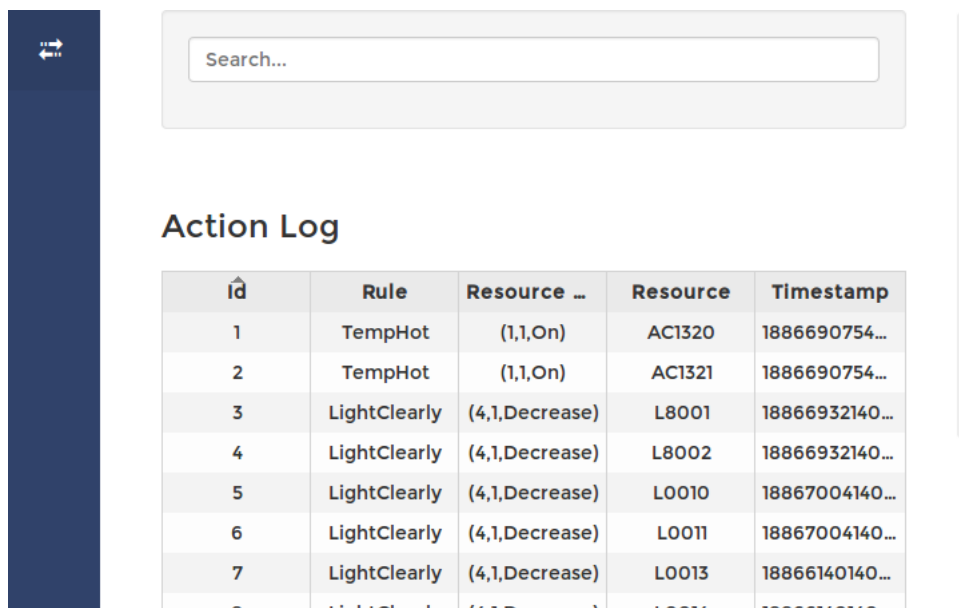


Figure 16 – Context Web UI – Action Log

3.6 Context Entity

Context is a container entity, i.e., it contains a set of instances of other five entities: Place, Resource, Fusion, Rule, and Action. Some examples may be classroom is empty, elevator is idle or university is closed. In fact, the specification of context is important for modelling or documentation purposes, which means that developers, integrators, or end-users might be interested in specifying changes of system behavior according to a broader concept of context, instead of thinking about individual sensors, fusion, or rules. However, contexts are non-binding when it comes to the execution of particular fusion criteria, rules, and actions. In other words, even though a particular context specifies that a set of sensors have their data combined by a fusion, which in turn fires a rule that perform some actions, in practice that might happen in different ways.

3.6.1 Context-Type

Contexts have a type, which follow an auxiliary entity called Context-Type, as depicted by Figure 17.

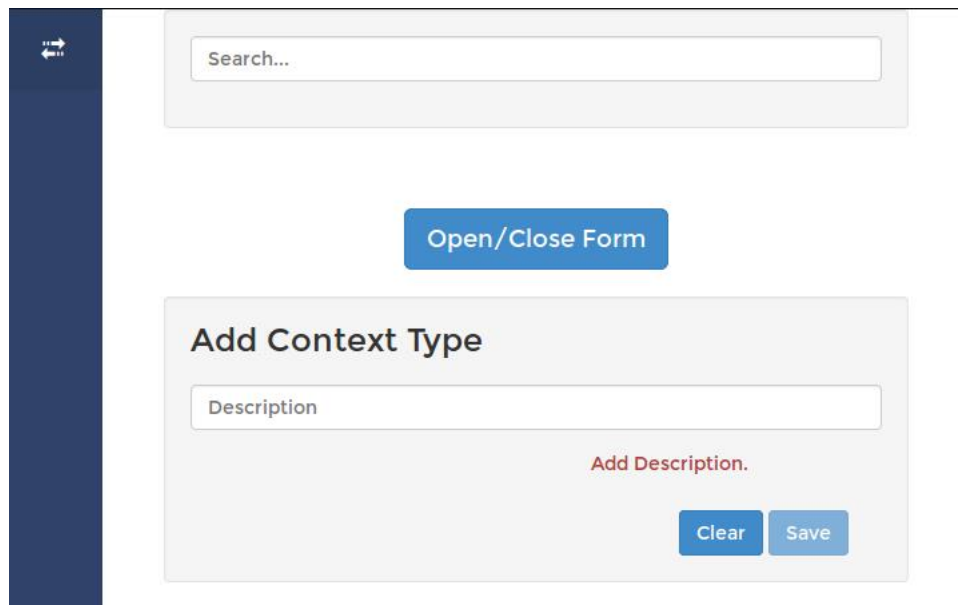
The image shows a web interface for managing context types. On the left is a dark blue vertical sidebar with a small icon at the top. The main content area has a light gray background. At the top, there is a search bar with the placeholder text "Search...". Below the search bar is a blue button labeled "Open/Close Form". Underneath that is a form titled "Add Context Type". The form contains a text input field with the placeholder "Description". Below the input field, the text "Add Description." is displayed in red. At the bottom right of the form are two blue buttons: "Clear" and "Save".

Figure 17 – Context Web UI – Context Type

3.6.2 Context Definition Interface

The interface for configuring Context entries is depicted by Figure 18. For example, a university may determine that whenever a classroom is empty, its lights must be switched off and the air conditioners (or heaters) must be turned off. Determining that a classroom is empty may involve different presence sensors placed in strategic positions within the classroom. The values measured by these sensors ideally would not be considered individually, but an average of their measurements in the last, say, 30 seconds may be considered. Also, when all presence sensors indicate that no one is within the classroom, a rule will be fired that in turn executes two actions for switching off lights and turning off air conditioners.

Also, users may be interested in testing different ways of implementing a particular context and comparing their results. Therefore, the context interface allows contexts to have versions where only one version will be active in a particular point in time. Such approach allows history of previous context definitions to be maintained while new ones are tested.

Search...

Open/Close Form

Add Context

Description

Add Description.

Clear Save

a)

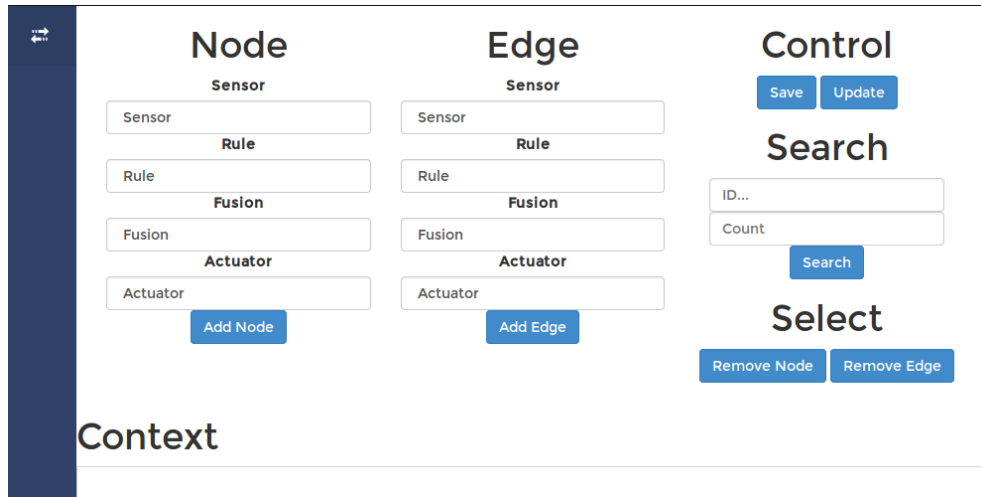
Search...

Open/Close Form

Context List

id	context Name	context Type	
1	Hot R808 Classroom	Hot Classroom	✘
2	Clearly R300 Classroo...	Clearly Classroom	✘

b)

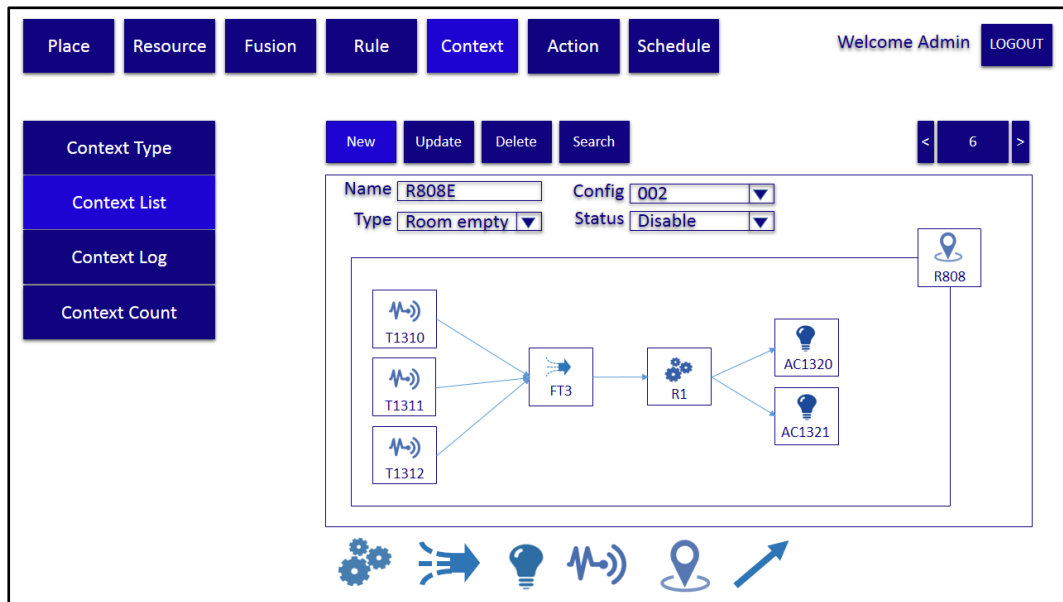


c)

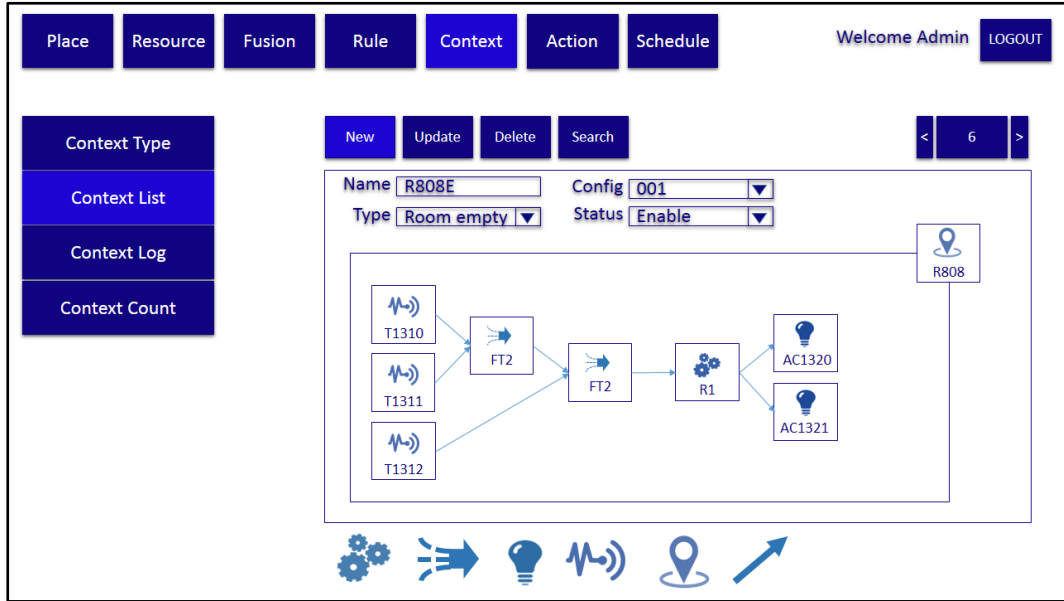
Figure 18 – Context Web UI – Context

3.6.3 Context Graphs

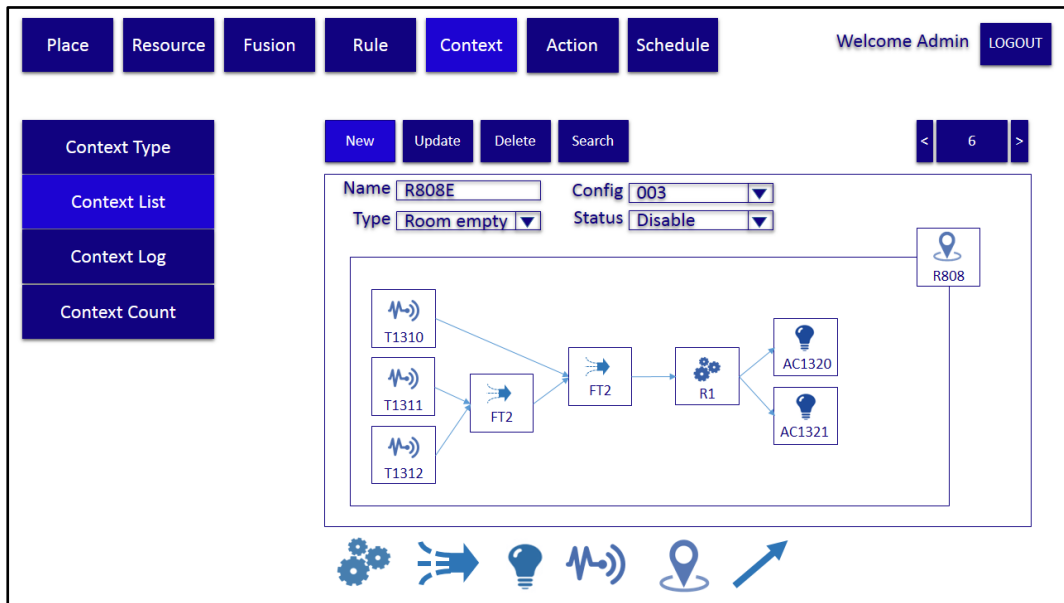
As an additional feature, contexts can be visualized in a graph-based interface. This feature is aimed at making it easier to users to model Context entities as a combination of Place, Resource, Fusion, Rule, and Action entities. In a context graph, the set of vertices is represented by entity instances and the set of edges is represented by data flow or control flow among entities. Figure 19 presents three examples of context graphs in a preliminary design, since this feature is currently under development.



a)



b)



c)

Figure 19 – Context Web UI – Context Graph

3.6.4 Context Log

Whenever a context is executed, a log entry is generated in the local database, which is available for query and visualization, as shown by Figure 20.

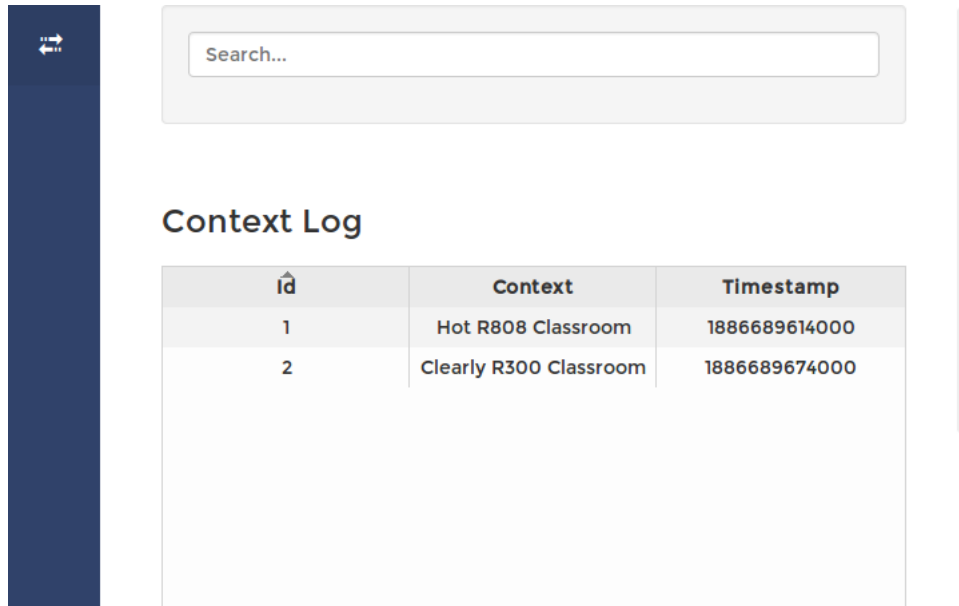
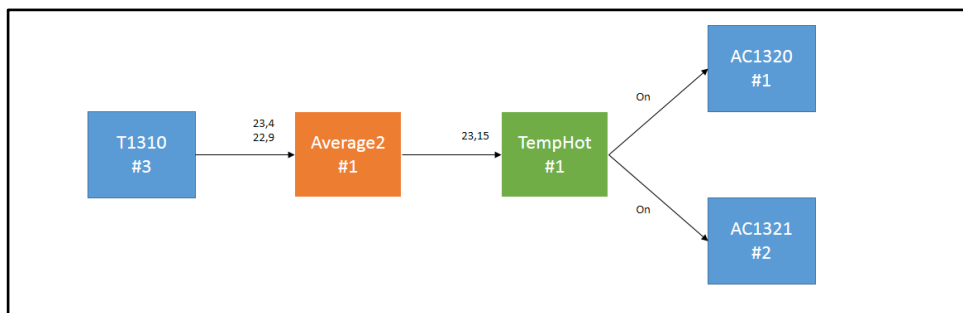


Figure 20 – Context Web UI – Context Log

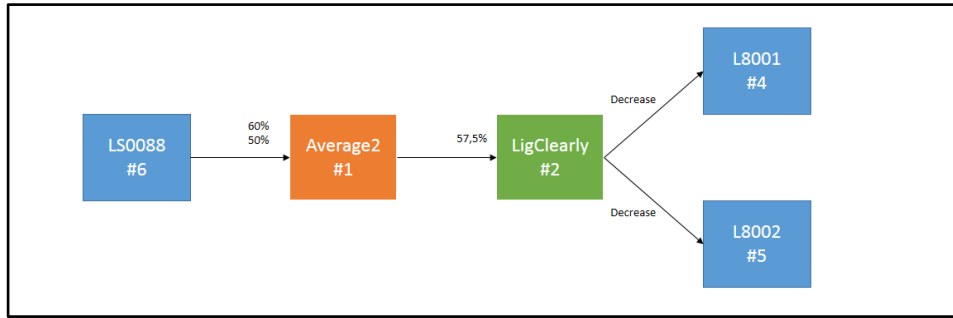
3.6.5 Tracking and Counting Context Execution

Context logging does not only play an important auditing role, but also it helps users to understand which contexts are being executed, given a set of places, resources, fusion, rules, and actions. Also, context logging is not simply to track the execution of a single entity, since they contain multiple entities of different classes. Therefore, the Context Web UI has an additional feature of context tracking and counting, for helping users to understand and control which contexts are begin executed in practice. That happens, because as mentioned above, it is not possible to enforce the execution of a give context graph as defined by developers or integrators. Rather, contexts may be executed in unpredictable ways. Please notice that some contexts happen in the exact way they were defined, whereas other contexts may happen in unexpected ways, i.e. without administrators being aware of them.

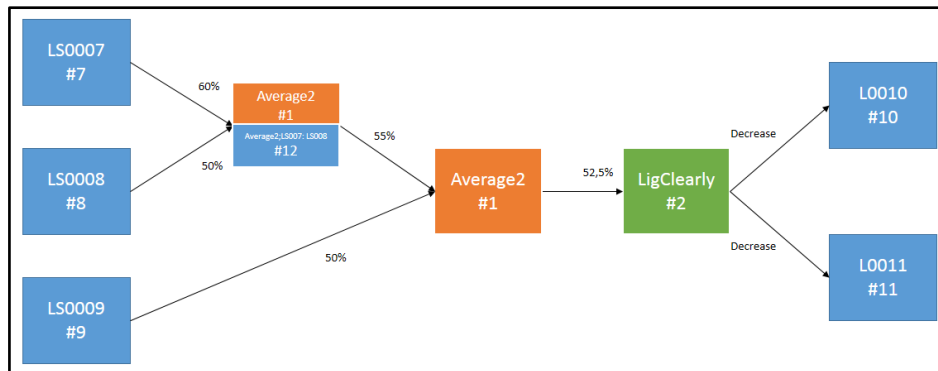
A new syntax was defined in order to track and count the sequence of entities involved in the execution of a particular context. It uses a letter for identifying each entity (S = sensor; F = fusion; R = Rule; A = action/actuator) followed by a virtual ID (a number). In the sequence there are a "*" (star) for relating the virtual ID with the real ID in the context storage. Virtual IDs are necessary because the same fusion may be used more than once in the same context, as shown by Figure 24, so that the real ID of a fusion would not be enough for uniquely identifying each element of a graph. The real ID relates each element with its function, whereas the virtual ID relates each element with its position in a graph. Entities of different classes are connected via a ":" (colon), whereas different connections are separated by a ";" (semicolon). Entities of the same class, such as two actions, are separated by a "," (comma). Figure 21 illustrates different graphs and their representation in this new syntax.



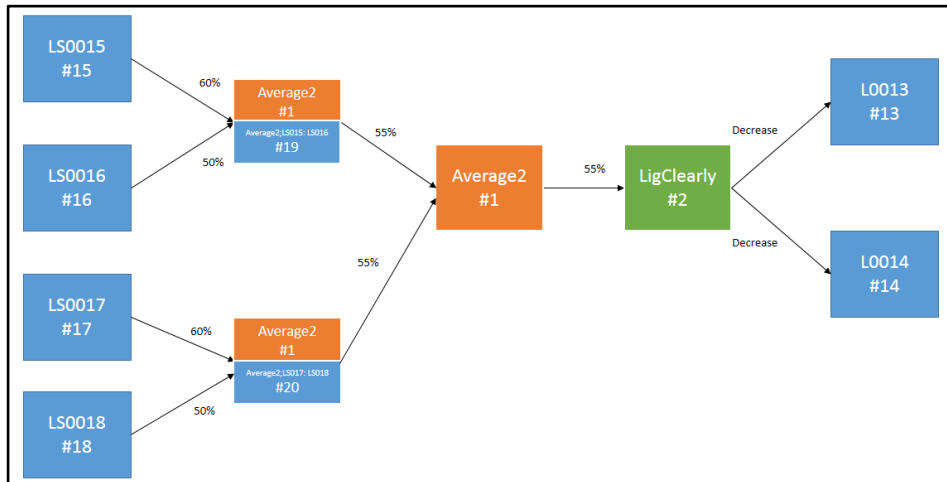
a) S1*3:F1*1;F1*1:R1*1;R1*1:A1*1,R1*1:A2*2;



b) S1*6:F1*1;F1*1:R1*2;R1*2:A1*4,R1*2:A2*5;



c) S1*7:F1*1,S2*8:F1*1,S3*9:F2*1;F1*1:F2*1,F2*1:R1*2;R1*2:A1*10,R1*2:A2*11;



d) S1*15:F1*1,S2*16:F1*1,S3*17:F2*1,S4*18:F2*1;F1*1:F3*1,F2*1:F3*1,F3*1:R1*2;R1*2:A1*13,R1*2:A2*14;

Figure 21 – Tracking and Counting Context Executions

Each Context entry registered in the application is included in the Context Count Table starting with 0 executions, as shown by Figure 22 as a preliminary interface, since this feature is not finished yet. For the IMPRESS Context Manager, Context tracking is performed backwards, which means that it starts with the analyses of actions send to actuators, going back to a rule that performed that action, and back to a fusion that fired that rule and to the sensors that generated the initial data. This tracking yields the executed context graphs, i.e., the sequences of actions and transitions that occurred in practice. As mentioned before, some sequences (graphs) may be registered in the system and other may not. In that case, users may choose to add the graph generated as a result of the tracking process. Whenever a context execution is identified, it is lookup up in the table. If it is already there, its count is incremented. Otherwise, they are added to the Context Count Table with the count set to 1.

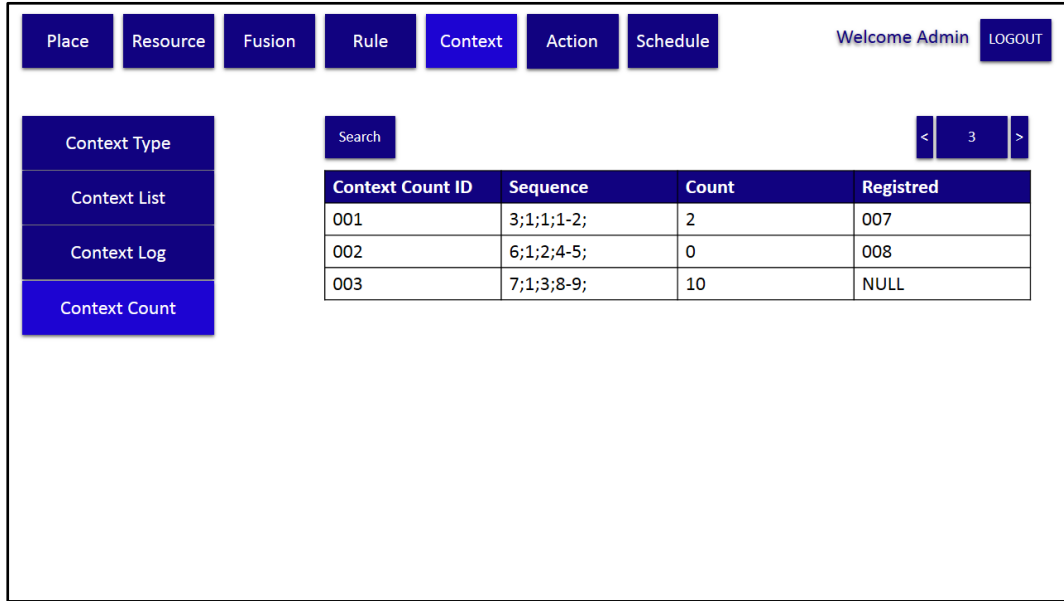


Figure 22 – Context Web UI – Context Count Table

As presented in section 2.3, Context Storage is implemented as a relational database. Therefore, context tracking is performed by database queries (Figure 23), actually using sub-queries. Each sub-query tracks backwards, i.e., from the action in an actuator back to the sensor(s) that started it. For example, suppose that an air conditioner was turned off because a rule fired an action, a fusion fired a rule and data coming from sensors fired a fusion. That is exactly the backtrack function that the sub-queries from Figure 23 performs.

```

select id_dependence_fusion_log_fk
from resource
where id_resource in
  (select id_resource_fk
   from resource_log
   where id_resource_log in
     (select id_resource_log_fk
      from rsc_fusion_log
      where id_fusion_log_fk in
        (select id_fusion_log_fk
         from fusion_rule_log
         where id_rule_action_log_fk in
           (select id_rule_action_log
            from rule_action_log
            where creation_date = '99/99/99 99:99:99'))));

```

Figure 23 – Context Web UI – Context Tracking Query

3.6.6 Tracking Contexts with Nested Fusions

For the IMPReSS Context Manager, the result of a Fusion might become a virtual sensor, which in turn may be fed into another Fusion, as shown by Figure 24. Tracking such nested fusions is a

challenge, which required the development of a recursive function, presented in Appendix B. Using that function, no matter how many nested fusions are used, they all will be tracked back reaching the source sensors.

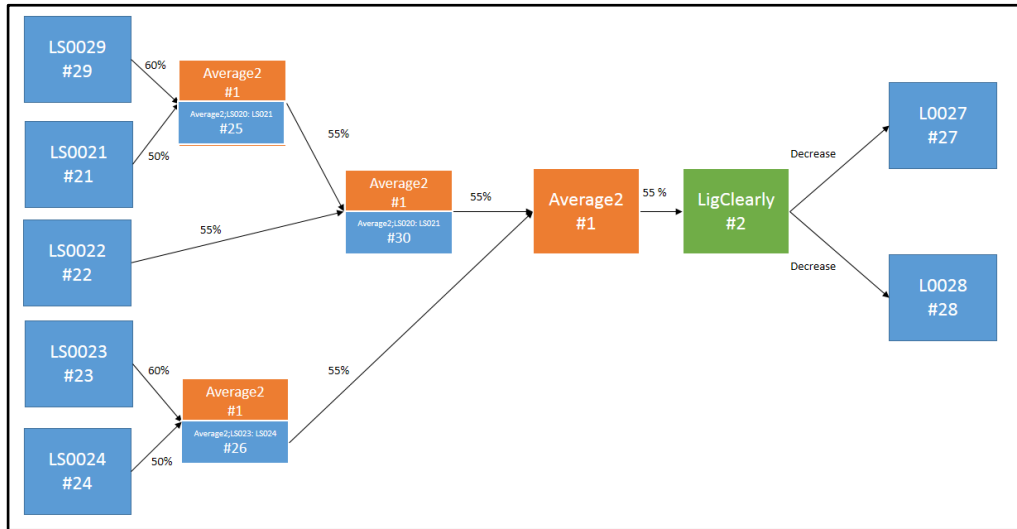


Figure 24 – Tracking and Counting Context Execution – Nested Fusions
 (S1*29:F1*1,S2*21:F1*1,S3*22:F2*1,S4*23:F3*1,S5*24:F3*1;F1*1
 :F2*1,F2*1:F4*1,F3*1:F4*1,F4*1:R1*2;R1*2:A1*27,R1*2:A2*28;)

3.7 Activity Entity (Schedule)

Typically some activities are pre-scheduled such are classes in a university. In those cases, whenever a class is about to start, some preparations should be made in the previous minutes, such as opening the door, switching on lights and turning on air conditioners or heaters. Activities can be configured in the schedule via the Schedule interface of the Context Web UI, as shown by Figure 25.

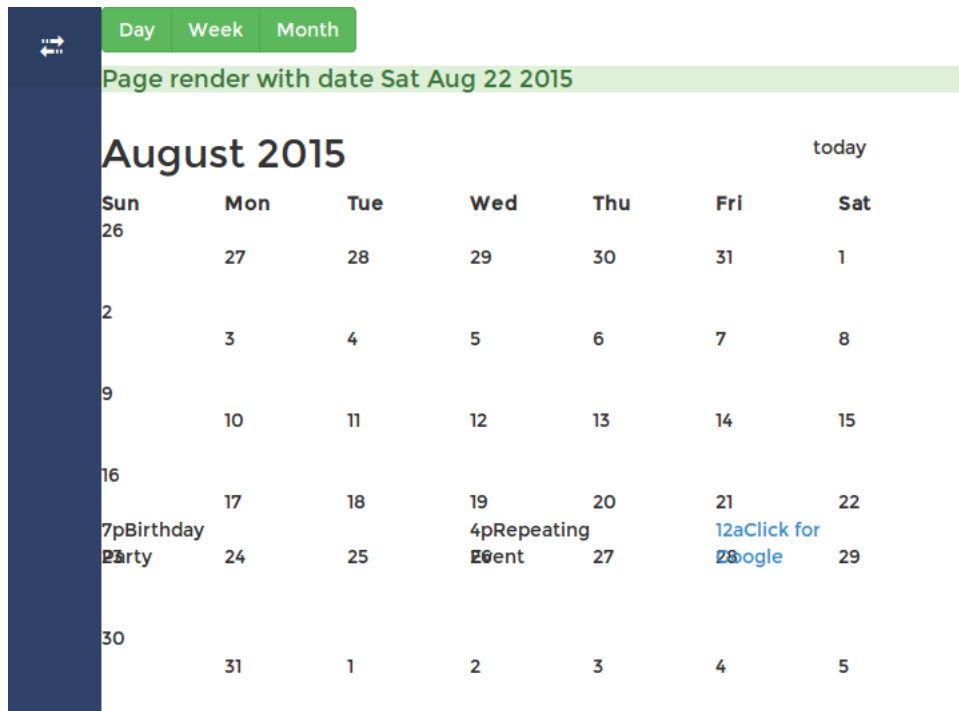


Figure 25 – Context Web UI – Schedule

Whenever a scheduled activity is executed, the execution context is logged in the local database, which is available for query and visualization, as shown by Figure 26.

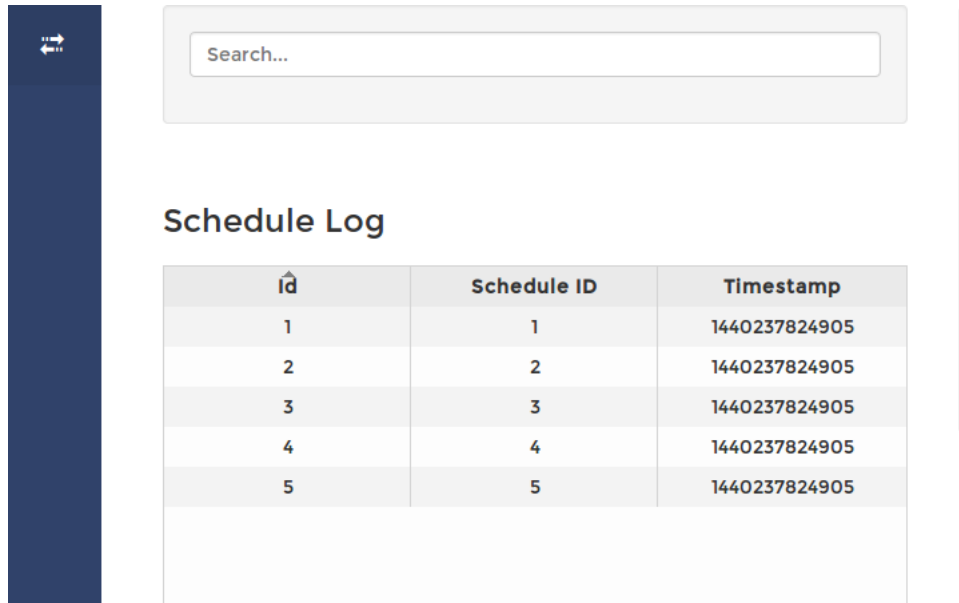


Figure 26 – Context Web UI – Schedule Log

4. Conclusion

The IMPReSS System Development Platform (SDP) needs to be context aware in order to provide an efficient use of energy in buildings, in such a way to adapt its operations to the current context conditions without explicit user intervention. The implementation of the IMPReSS Context Manager was presented in Deliverable D6.4, which exports a REST API for allowing the development of different client applications.

This deliverable presents the implementation of the IMPReSS Context Web UI, which is a Context Modelling Tool that allows users to perform CRUD (Create, Read, Update, Delete) operations on seven entities that may be used to develop a typical context-aware building automation application: Resource (sensors/actuators), Place (rooms, floors), Fusion (data aggregation), Rule (decisions), Action (commands to actuators), Activity (scheduled activities), and Context (combination of Resource, Place, Fusion, Rule, and Action). The Context Web UI is based on AngularJS, an open-source web application framework developed by Google for simple-page Web applications.

This deliverable is an important output of Task 6.4 (Context Model & Rule Authoring Tool), which complements that development of the Context Manager, which is in turn an output of Task 6.3. The Context Web UI aims at exploring and demonstrating the features provided by the Context Manager, as well as making it easier for developers or integrators to configure energy-efficiency context management applications. The next steps are testing and debugging the Context Web UI for making it suitable to be used in the development of energy efficiency management applications. Also, it will be used for demonstration purposes.

References

- (Kamienski et. al 2015) Kamienski, C., Borelli, F., Oliveira, G., Moretti, W., Pinheiro, I., (2015), Implementation of Context Reasoning Engine, IMPReSS Consortium, Deliverable D6.4, March 2015.
- (Kamienski et. al 2014a) Kamienski, C., Borelli, F., Oliveira, G., Moretti, W., Pinheiro, I., Belati, E. (2014), SDP Initial Architecture Report, IMPReSS Consortium, Deliverable D2.2.1, February 2014.
- (Kamienski et. al 2014b) Kamienski, C., Borelli, F., Oliveira, G., Moretti, W., Pinheiro, I., Belati, E. (2014), Context Management Framework Architecture and Design of Context Templates, IMPReSS Consortium, Deliverable D6.4, November 2014.