



FUNCTIONAL SPECIFICATIONS

SmartH2O

Project FP7-ICT-619172

Deliverable D2.3 WP2

Programme Name: ICT
Project Number: 619172
Project Title: SmarH2O
Partners: Coordinator: SETMOB
Contractors: POLMI, UoM, EIPCM, MOONSUB

Document Number: smarth2o. D2.3.SETMOB.WP2.V1.2
Work-Package: WP2
Deliverable Type: Document
Contractual Date of Delivery:
Actual Date of Delivery:
Title of Document: Functional Specifications
Author(s): Luigi Caldararu, Sever Calit, Isabel Micheel, Jasminko Novak,
Luca Galli, Giorgia Baroffio Andrea Cominola Piero Fraternali
Chiara Pasini, Corrado Valeri, Andrea Emilio Rizzoli

Approval of this report Submitted for review

Summary of this report: Deliverable summary

History: See the document history

Keyword List: Functional specifications, integration specifications, services

Availability This document is public



This work is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License](https://creativecommons.org/licenses/by-nc-sa/3.0/).

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

Document History

Version	Date	Reason	Revised By
0.1	30/06/2015	Set Document Structure	Piero Fraternali
0.2	15/08/2015	SETMOB initial contribution: Platform general Context and Structure chapters. ESB, Portal Exchange Manager	Luigi Caldararu Sever Calit
0.3	20/09/2015	MSM contribution: Games Platform	Luca Galli
0.4	15/09/2015	POLIMI contribution: Social Network Crawler, Data Analyzer	Ilio Catallo
0.5	21/09/2015	POLIMI contribution: Basic Portal, Advanced Portal, Gamification Engine, Customer Portal Admin, Social Connector	Chiara Pasini
0.6	21/09/2015	SUPSI contribution: Agent Based Modeling	Corrado Valeri
0.7	22/09/2015	EIPCM contribution: EMIVASA Case Study and requirements update	Isabel Micheel
0.8	25/09/2015	MSM contribution: Added description of data model and sequence diagrams	Luca Galli
0.9	27/09/2015	SETMOB contribution: Authentication Gateway	Sever Calit
1.0	27/09/2015	POLIMI contribution: Business Dashboard	Chiara Pasini
1.1	28/09/2015	SETMOB contribution: Design Rationale, Conclusions	Sever Calit
1.2	30/09/2015	SUPSI contribution: final quality check	Andrea Emilio Rizzoli

Disclaimer

This document contains confidential information in the form of the SmarH2O project findings, work and products and its use is strictly regulated by the SmarH2O Consortium Agreement and by Contract no. FP7- ICT-619172.

Neither the SmarH2O Consortium nor any of its officers, employees or agents shall be responsible or liable in negligence or otherwise howsoever in respect of any inaccuracy or omission herein.

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

The contents of this document are the sole responsibility of the SmarH2O consortium and can in no way be taken to reflect the views of the European Union.



Table of Contents

1. INTRODUCTION	10
2. CONTEXT	15
3. COMPOSITION	17
4. PORTAL DATA EXCHANGE MANAGER	20
4.1 CONTEXT	20
4.1.1 <i>Processing of water consumption files</i>	20
4.1.2 <i>Retrieval of water raw consumption files</i>	20
4.1.3 <i>File processing</i>	21
4.1.4 <i>Archival of water consumption files</i>	21
4.1.5 <i>Logging</i>	21
4.1.6 <i>Administration</i>	21
4.2 COMPOSITION	21
4.2.1 <i>File retrieval module</i>	21
4.2.2 <i>File processing module</i>	22
4.2.3 <i>Logging module</i>	22
4.2.4 <i>Administration module</i>	22
5. USER AND CONSUMPTION DATABASE	24
5.1 INFORMATION	24
5.2 INTERFACE	29
5.2.1 <i>User and Household Services</i>	30
5.2.2 <i>Consumption Data Services</i>	33
6. BASIC CUSTOMER PORTAL	34
6.1 CONTEXT	34
6.1.1 <i>Consumption-related use cases</i>	34
6.1.2 <i>Learning-related use cases</i>	35
6.1.3 <i>Notification-related use cases</i>	36
6.2 COMPOSITION AND STRUCTURE	36
6.2.1 <i>Summary Sections</i>	37
6.2.2 <i>UserProfile [MasterPage]</i>	38
6.2.3 <i>ProfilePage [Page]</i>	40
6.2.4 <i>Support [Page]</i>	40
6.2.5 <i>Upgrade [Page]</i>	41
6.3 INTERACTION, STATE DYNAMICS	42
6.3.1 <i>Consumption-related use cases</i>	42
6.3.2 <i>Learning-related use cases</i>	43
6.3.3 <i>Notification-related use cases</i>	44
7. ADVANCED CUSTOMER PORTAL	45
7.1 CONTEXT	45
7.1.1 <i>Gamification actions, badges and rewards</i>	45
7.1.2 <i>Gamified water meter use cases</i>	45
7.1.3 <i>User profiling use cases</i>	47
7.1.4 <i>Social use cases</i>	47
7.1.5 <i>User account management use cases</i>	48
7.1.6 <i>Signup use cases</i>	49
7.2 COMPOSITION AND STRUCTURE	50
7.2.1 <i>Summary Sections</i>	50
7.2.2 <i>UserProfile [MasterPage]</i>	52
7.2.3 <i>Support [Page]</i>	54
7.3 INTERACTION, STATE DYNAMICS	54
7.3.1 <i>Gamification actions, badges and rewards</i>	54

7.3.2	<i>Gamified water meter use cases</i>	56
7.3.3	<i>User profiling use cases</i>	58
7.3.4	<i>Social use cases</i>	59
7.3.5	<i>User account management use cases</i>	61
7.3.6	<i>Signup use cases</i>	62
8.	CUSTOMER PORTAL ADMIN	64
8.1	CONTEXT	64
8.2	COMPOSITION AND STRUCTURE	65
8.2.1	<i>Summary Sections</i>	65
8.2.2	<i>Administrator Users Area [Area]</i>	67
8.2.3	<i>Community Users Area [Area]</i>	68
8.2.4	<i>Gamification Area [Area]</i>	69
8.2.5	<i>Rewards Area [Area]</i>	71
8.3	INTERACTION, STATE DYNAMICS	71
8.3.1	<i>Use cases</i>	71
9.	SOCIAL CONNECTOR	74
9.1	CONTEXT	74
9.2	COMPOSITION AND STRUCTURE	74
9.2.1	<i>Invite a friend [Module]</i>	74
9.2.2	<i>Post content [Module]</i>	75
10.	GAMIFICATION ENGINE	77
10.1	COMPOSITION AND STRUCTURE	77
10.2	LOGICAL	77
10.3	INTERACTION & DEPENDENCIES	80
10.3.1	<i>Registration</i>	80
10.3.2	<i>Visualize Consumption Overview</i>	80
10.3.3	<i>External Authentication</i>	81
10.4	STATE DYNAMICS	82
10.4.1	<i>Assign an action to a user</i>	82
10.4.2	<i>Redeem reward</i>	83
10.4.3	<i>Assign a new Badge</i>	83
10.4.4	<i>Reset Score 7 days</i>	84
10.4.5	<i>Check consumption Goal</i>	84
10.4.6	<i>Check podium action</i>	84
10.5	INTERFACES	84
1.1.1	<i>Authentication</i>	85
1.1.2	<i>User Services</i>	89
1.1.3	<i>Thematic Areas</i>	92
1.1.4	<i>Actions</i>	93
1.1.5	<i>Badges</i>	99
1.1.6	<i>Leaderboard</i>	104
1.1.7	<i>Rewards</i>	106
11.	GAMES PLATFORM	112
11.1	STRUCTURE	112
11.2	INTERACTION	113
11.2.1	<i>Sign-up</i>	113
11.2.2	<i>Connecting Player Profile to the Gamification Engine</i>	114
11.2.3	<i>Playing Standard Mobile Game (Drop! The Question)</i>	115
11.2.4	<i>Play the card game and its digital Extension</i>	115
11.2.5	<i>Connecting user from Advanced Customer Portal</i>	116
11.2.6	<i>Content administration & Players Profilation</i>	117
12.	SOCIAL NETWORK CRAWLER AND DATA ANALYZER	120
12.1	SOCIAL NETWORK CRAWLER	120
12.1.1	<i>Logical</i>	120
12.2	DATA ANALYZER	123

12.2.1	<i>Logical</i>	124
12.2.2	<i>Information with data distribution overlay and physical volumetric overlay</i>	130
13.	BUSINESS DASHBOARD	131
13.1	CONTEXT	131
13.2	COMPOSITION AND STRUCTURE	133
13.2.1	<i>Summary Sections</i>	133
13.3	INTERACTION, STATE DYNAMICS	134
13.3.1	<i>Use cases</i>	134
14.	PRICING ENGINE	137
14.1	STRUCTURE	137
15.	AGENT-BASED CONSUMPTION SIMULATOR	138
15.1	CONSUMPTION SIMULATION BASED ON PAST INFORMATION	138
15.1.1	<i>Context</i>	138
15.1.2	<i>Composition</i>	139
15.1.3	<i>Logical</i>	139
15.1.4	<i>Dependency</i>	140
15.1.5	<i>Information</i>	140
15.1.6	<i>Interface</i>	140
15.1.7	<i>Structure</i>	141
15.1.8	<i>Interaction</i>	141
15.1.9	<i>State Dynamics</i>	142
15.1.10	<i>Algorithms</i>	142
15.2	CONSUMPTION SIMULATION BASED ON INCENTIVE RESPONSE	143
15.3	CONSUMPTION SIMULATION BASED ON PRICING SCHEMES	143
16.	ESB	144
16.1	CONTEXT	144
16.1.1	<i>Make Request</i>	144
16.1.2	<i>Get Response</i>	144
16.1.3	<i>Administration</i>	144
16.2	COMPOSITION	145
16.2.1	<i>Consumers</i>	145
16.2.2	<i>Providers</i>	145
16.2.3	<i>Routes</i>	145
17.	AUTHENTICATION GATEWAY	147
17.1	CONTEXT	147
17.1.1	<i>User/Pass Authentication</i>	147
17.1.2	<i>Token based Authentication</i>	147
17.2	INTERACTION	147
18.	DESIGN RATIONALE	149
19.	CONCLUSIONS	150
20.	REFERENCES	151
21.	APPENDIX A: VALENCIA EMIVASA CASE STUDY	152
21.1	VALENCIA CASE STUDY REQUIREMENTS ELICITATION	152
21.1.1	<i>Outline of the approach</i>	152
21.1.2	<i>Requirements workshop with EMIVASA / Aguás de Valencia</i>	152
21.1.3	<i>Adapted personas and user stories</i>	155
21.2	REQUIREMENTS UPDATES	157
22.	APPENDIX B: MAPPING FINAL REQUIREMENTS TO FUNCTIONAL SPECIFICATIONS	158
23.	APPENDIX C: IFML CONCEPTS	163
23.1	OVERVIEW OF IFML MAIN CONCEPTS	163

Executive Summary

This document contains deliverable *D2.3 Functional specification*, which, according to the Description of Work, ...

...contains the functional specifications of the SmarH2O platform, describing clearly and accurately all functionalities and services necessary to meet the identified user requirements and expectations and it provides them to WP6 for their implementation. Functional specification will be formalized according to international best practices in software design, using the most appropriate UML diagrams.

This deliverable has logical connections with other deliverables already produced by the SmarH2O Consortium, most notably with:

- **D2.2 Final requirements:** this deliverable has formalized the second version (after *D2.1 Use cases and early requirements*) of the requirement specifications, and provided a final set of functional and non-functional requirement specifications for the project design and implementation. The deliverable illustrated: the user stories, the mock-ups of the user interfaces, the initial version of the data model and of the user model, and an overview of the main use cases, which included: the basic customer portal: visual water meter; the advanced customer portal: gamified water meter; the customer portal user account management; customer portal admin; the games platform; the business dashboard: customer consumption monitor; and the agent-based customer consumption simulator. D2.2 also contained the success criteria that will be used to verify the project impact, and the software integration requirements, which address the integration between platform components and with external data providers. With respect to D2.2, D2.3 positions as follows
 - It refines the data modeling perspective, specifying the current (i.e., at M18) conceptual and logical specification of the SmarH2O database. This specification refines the data models provided in Section 5 and 6 of D2.2, and their evolution specified in sections and in Section 2 of D6.2 Platform architecture and design.
 - It traces the system design specifications back to the use case requirements that generate them; for convenience we have reported the list of use cases identified in the requirements specifications in [Appendix A](#).
- **D6.2 Platform architecture and design:** this deliverable has presented the architectural design of the SmarH2O platform, describing: the information and data models, the platform components, services, and applications, communication protocols. It also described the integration model that enables various platform modules to interact with each other. With respect to D6.2, D2.3:
 - Represents an extension and an evolution of the platform architecture and design specifications, yielding a System Design Document (SDD) adherent to international standards (specifically, IEEE 1016);
 - Provides the traceability links between the system design specifications (D6.2 and D2.3) and the requirements specifications (use cases of D2.2, recalled in [Appendix A](#)).

In preparing the deliverable, we have chosen the IEEE 1016 standard (IEEE Standard for Information Technology--Systems Design--Software Design Descriptions – Redline), as a blueprint for the document content and outline.

1. Introduction

This document contains the system design specifications of SmartH2O, produced as an evolution of D6.2 *Platform architecture and design*, considering the feedback and requests for change elicited after the first release and test of the SmartH2O software (D6.3 *Platform Implementation and Integration - initial prototype*).

The document is organized according to the 1016-2009 IEEE Standard for Information Technology--Systems Design--Software Design Descriptions [IEEE1016-2009], which describes software designs and establishes the information content and organization of a software design description (SDD).

According to the IEEE 1016-2009, an SDD is a representation of a software design to be used for recording design information and communicating that design information to key design stakeholders. The standard is intended for use in design situations in which an explicit software design description is to be prepared. These situations include traditional software construction activities, when design leads to code, and reverse engineering situations when a design description is recovered from an existing implementation. The standard can be applied to commercial, scientific, or military software that runs on digital computers. Applicability is not restricted by the size, complexity, or criticality of the software. It can be applied to the description of high-level and detailed designs. It does not prescribe specific methodologies for design, configuration management, or quality assurance nor requires the use of any particular design languages, but establishes requirements on the selection of design languages for use in an SDD. It can be applied to the preparation of SDDs captured as paper documents, automated databases, software development tools or other media.

IEEE 1016 organises the SDD around the entities of the conceptual model represented in Figure 1.

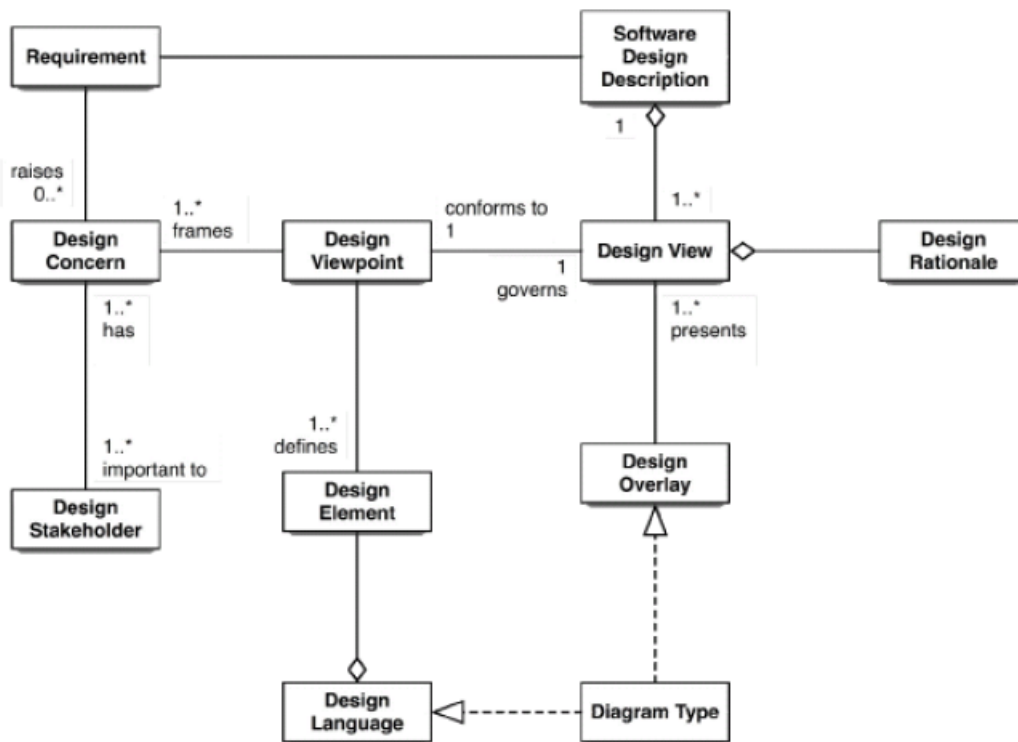


Figure 1 - conceptual model of the IEEE 1016 SDD.

The conceptual model builds upon the following concepts

- **Design attribute:** An element of a design view that names a characteristic or property of a design entity, design relationship, or design constraint. See also: design constraint, design entity, design relationship.
- **Design concern:** An area of interest with respect to a software design.
- **Design constraint:** An element of a design view that names and specifies a rule or restriction

on a design entity, design attribute, or design relationship. See also: design attribute, design entity, design relationship.

- **Design element:** An item occurring in a design view that may be any of the following: design entity, design relationship, design attribute, or design constraint.
- **Design entity:** An element of a design view that is structurally, functionally, or otherwise distinct from other elements, or plays a different role relative to other design entities. See also: design view.
- **Design overlay:** A representation of additional, detailed, or derived design information organized with reference to an existing design view.
- **Design rationale:** Information capturing the reasoning of the designer that led to the system as designed, including design options, trade-offs considered, decisions made, and the justifications of those decisions.
- **Design relationship:** Element of a design view that names a connection or correspondence between design entities. See also: design entity.
- **Design stakeholder:** An individual, organization, or group (or classes thereof playing the same role) having an interest in, or design concerns relative to, the design of some software item.
- **Design subject:** A software item or system for which an SDD will be prepared. Syn: software under design, system under design.
- **Designer:** The stakeholder responsible for devising and documenting the software design.
- **Design view:** A representation comprised of one or more design elements to address a set of design concerns from a specified design viewpoint. See also: design concern, design element, design viewpoint.
- **Design viewpoint:** The specification of the elements and conventions available for constructing and using a design view. See also: design view.
- **Diagram (type):** A logically coherent fragment of a design view, using selected graphical icons and conventions for visual representation from an associated design language, to be used for representing selected design elements of interest for a system under design from a single viewpoint.

The conceptual model is used to organize the content of the SDD. IEEE 1016 prescribes that the required contents of an SDD are as follows:

- Identification of the SDD.
- Identified design stakeholders.
- Identified design concerns.
- Selected design viewpoints, each with type definitions of its allowed design elements and design.
- Languages.
- Design views.
- Design overlays.
- Design rationale.

The standard provides a non-prescriptive template for the SSD content, which is as follows:

1. Frontispiece.
 - 1.1. Date of issue and status.
 - 1.2. Issuing organization.
 - 1.3. Authorship.
 - 1.4. Change history.
2. Introduction.
3. Purpose.
4. Scope.
5. Context.
6. Summary.
7. References.
8. Glossary.
9. Body.
 - 9.1. Identified stakeholders and design concerns.
 - 9.2. Design viewpoint 1.
 - 9.3. Design view 1.
 - 9.4. ...

- 9.5. Design viewpoint *n*.
- 9.6. Design view *n*.
- 9.7. Design rationale.

Specifically, IEEE 1016 defines the following possible design viewpoints for use in an SDD, depending on the type of system [IEEE 1016, Clause 5], illustrated in Table 1.

Table 1: summary of design viewpoints in IEEE 1016.

Design viewpoint	Design concerns	Example design languages
Context	Systems services and users	IDEF0, UML use case diagram, Structured Analysis context diagram
Composition, which can be refined into new viewpoints, such as: functional (logical) decomposition, and runtime	Composition and modular assembly of systems in terms of subsystems and (pluggable) components, buy vs. build, reuse of components	Logical: UML package diagram, UML component diagram, Architecture Description Languages, IDEF0, Structure chart, HIPO Physical: UML deployment diagram
Logical	Static structure (classes, interfaces, and their relationships) Reuse of types and implementations (classes, data types)	UML class diagram, UML object diagram
Dependency	Interconnection, sharing, and parameterization	UML package diagram and component diagram
Information with data distribution overlay and physical volumetric overlay	Persistent information	IDEF1X, entity-relation diagram, UML class diagram
Patterns	Reuse of patterns and available Framework template	UML composite structure diagram
Interface	Service definition, service access	Interface definition languages (IDL), UML component diagram
Structure	Internal constituents and organization of design subjects, components and classes	UML structure diagram, class diagram
Interaction	Object communication, messaging	UML sequence diagram, UML communication diagram
State dynamics	Dynamic state transformation	UML state machine diagram, Statechart (Harel's), state transition table (matrix), automata, Petri net
Algorithm	Procedural logic	Decision table, Warnier diagram, JSP, PDL
Resources May be refined into resource based viewpoints with possible	Resource utilization	UML Real-time Profile, UML class diagram, UML Object Constraint Language (OCL)

overlays		
----------	--	--

The choice of the relevant design viewpoints depends on the class of system, or of sub-system, under description.

Given the nature of the SmartH2O components, the choice of the pertinent viewpoints and design languages for each of the architecture components identified in D6.2 is reported in Table 2.

Component	Viewpoints	Languages
SmartH2O Database	Information	Entity Relationship diagram SQL DDL logical schema
Enterprise Service Bus	Context, Interface	Interface Description Language, UML component diagram
Smart Meter Data Manager	Context, Logical, Dependency, Interface, Information	UML use case diagrams UML class/component diagrams Json data representation format
Water Utility Consumer Portal (basic, advanced)	Context, Composition, Structure, Interaction, State dynamics	UML use case diagrams IFML interaction flow diagrams
Water Utility Admin Portal	Context, Composition, Interface, Structure, Interaction, State dynamics	UML use case diagrams IFML interaction flow diagrams
Portal Data Exchange Manager	Composition, Logical, Dependency, Information Interface, Structure, Interaction	UML use case diagrams UML class/component diagram UML sequence diagrams
Gamification Engine	Context, Composition, Logical, Dependency, Interface, Structure, Interaction	UML use case diagrams UML class diagram UML sequence diagrams IFML diagrams
Games Platform (Drop! game)	Context , Composition, Logical, Dependency, Information, Patterns, Interface, Structure, Interaction, Resources	UML use case diagrams UML class diagrams UML sequence diagrams
Pricing Engine		
Models of User Behaviour	Context, Composition, Logical, Algorithm, Resources	UML use case diagrams UML class/component diagrams Structured English
Agent Based Modelling	Context , Composition, Logical, Dependency, Information , Interface, Structure, Interaction, State dynamics, Algorithm	UML use case diagrams UML class/component diagrams UML sequence diagrams UML Statecharts Structured English
Authentication Gateway	Context, Logical, Dependency, Information, Interface, Structure, Interaction	UML use case diagrams UML class/component diagrams UML sequence diagrams
Social Network Crawler and Data Analyser	Context, Logical, Dependency, Information, Interface, Structure, Interaction, State dynamics, Algorithm, Resources	UML use case diagrams
Social Network Connector	Context , Logical, Dependency, Information , Patterns, Interface, Structure, Interaction, State dynamics, Algorithm, Resources	UML use case diagrams

Table 2: relevant design viewpoints and languages for SmartH2O components.

The design viewpoints will be presented both at Smart H2O Platform level and at Component level.

The design viewpoints at Platform level will present the Platform as a system of interconnected Components as they were identified in *D6.2 – Architecture and design*.. Functional specification of each Component will be further detailed in distinct chapters using the proposed methodology.

2. Context

D2.2 – Final Requirements and further refined in *D6.2 – Platform architecture and design* deliverable identified the user roles that interact with Smart H2O Platform:

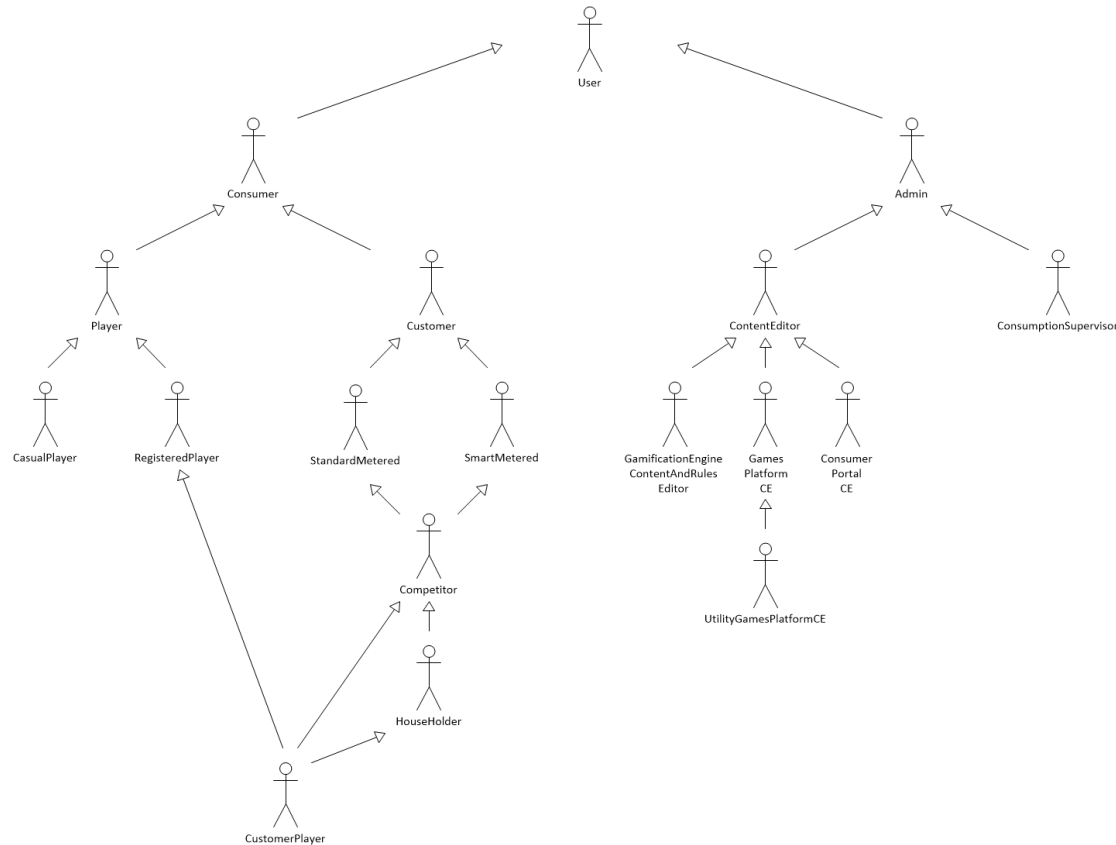


Figure 2 - Smart H2O User Types.

The **Consumer** branch of user roles will interact with the Platform via:

- A User Portal that comes in 2 versions:
 - Basic Portal with following important use cases:
 - Visualization and exploration of consumption data.
 - Provide feedback on disaggregated data.
 - Receive notification, water saving tips and alerts.
 - Advanced Portal which adds to Basic version Gamification and Social use cases:
 - Set consumption goals, explore results, earn points, get badges, awards and physical rewards.
 - Declare profiling information and end-use water events.
 - Compare achievements with other participants.
 - Join teams, play as a team and invite others to join a team.
 - Interact on social networks by sharing results and inviting friends from social network to join the Platform.
- A mobile trivia game
 - Play trivia Game on water saving topics and earn points that can be added to its Advanced Portal account.

The **Admin** branch of user roles will interact with the Platform via specialized content editors for each component:

- Basic Portal: water saving tips, videos and info.

- Advanced Portal: game rules, badges, awards, physical rewards, point redeeming rules
- Mobile Game: trivia questions and points awarded.

The ConsumptionSupervisors role will also have access to a specialized component water visualization and consumption simulation. They can also enter notifications, water related alerts that will be displayed in **User** portals.

Each user interaction with the Platform will be specified in chapters that details the specific components.

3. Composition

Figure 3 recalls from D6.2 *Platform architecture and design* the main components that constitute the SmartH2O platform.

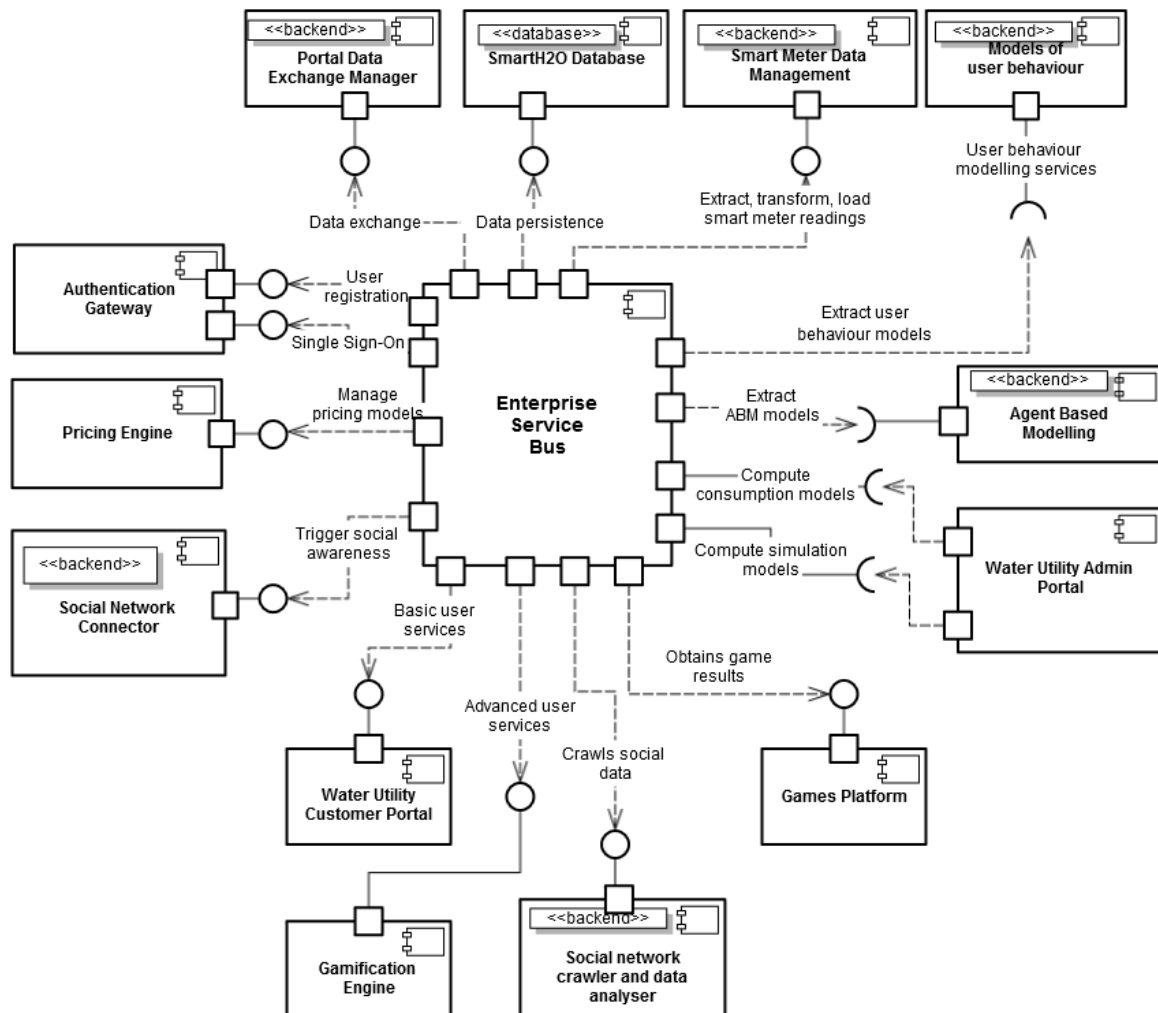


Figure 3 - Main components of the SmartH2O architecture.

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

The **Enterprise Service Bus** (from now on, **ESB**) is a middleware layer that supports the loose coupling of the SmartH2O components; it permits the publication of their interfaces (Application Programming Interfaces, APIs) and the synchronous and asynchronous communication among components. The goal of the ESB is to decouple the heterogeneous components of the SmartH2O platform as much as possible, to support the extension with new services and functions and the rapid adaptation of the platform to new contexts (e.g., other utility companies with different IT standards and information systems).

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

stored within the platform database.

The **Water Utility Consumer Portal** is a component, typically embedded within the proprietary portal services of the utility company, which supports the interaction between the utility customers and the SmartH2O awareness functionality. The integration is lightweight: the consumer will navigate from the standard GUI of the utility company to what she sees as a special section of the portal, where she can access the awareness tools and interfaces developed by SmartH2O.

The **Water Utility Admin Portal (Business Dashboard)** is a component, integrated within the proprietary portal services of the utility company, which supports the work of the supervisor in the analysis of the water consumption data and of the outcome of the gamification rules; it also supports the work of the content editor, who administers the content (e.g., tips, articles, news, etc.) published to the customers. The portal also offers interfaces to the water utility operators to run simulations, based on the models embodied in the Models of User Behaviour component and on the algorithms implemented in the Pricing Engine and in the Agent Based Modelling component.

The **Portal Data Exchange Manager includes Smart Meter Data Management Component** deals with the data exchange communication that occurs “behind the scenes” among the SmartH2O platform and the third party applications already supporting the interaction with the various types of users. Such applications may comprise: raw water consumption data, “standard” customers’ portal of the utility company, or a B2E application for the utility company’s supervisors.

The **Gamification Engine** is a back-end component that embodies rules for transforming users’ actions into gamification scores and achievements. It is exploited in order to “gamify” the water consumption of the users, according to the awareness approach implemented by SmartH2O. It has an interface for the end-user, who sees the results of her water consumption actions; and administrative interfaces for the utility company’s managers and operators, who can supervise the outcome of the awareness policies and define the rules that reward the actions of water consumers.

The **Games Platform** supports the execution of all the digital games of SmartH2O, including the games that are played as part of the interaction with the Drop! board game (for a description of the current status of the social awareness applications, including the SmartH2O games, see deliverable *D4.1: First social game and implicit user information techniques*). The Games Platform must also support casual players, and thus has an independent users’ registration procedure, as well as a procedure for enrolling users that are already registered in the Utility Portal. The Games Platform exposes two kinds of interfaces: one or more digital games directed to the end users; an administrative interface, directed to the content editors of the game platform. The GUIs are served by a local database (the Games DB), which stores information that is pertinent only to the game play (e.g., the gaming history of players not registered in the Utility Portal).

The **Pricing Engine** allows water utility companies to assess the impact for various dynamic pricing algorithms on their customer behaviour. The pricing engine will use consumption data, user profile data, external input like meteorological data, water supply forecast. The Pricing Engine will model the user elasticity to different pricing schemes and it will be able to report how pricing stimuli can impact aggregate customer behaviour. The Pricing Engine also allows the Consumer to estimate the cost of its water use according to different pricing schemas, ranging from the actual tariff, to various simulated tariffs, which are evaluated in the SmartH2O project.

The **Models of User Behaviour** component contains models and algorithms for profiling the behaviour of water consumers. It contains a classification algorithm that creates user segments (classes of users with similar behaviour) on the basis of their features. It also contains a disaggregation algorithm that can attribute the end uses of the total amount of water used by a household during one day, with a certain degree of approximation. This algorithm is also used to identify the relevant features to be used in classification. Through the use of the SmartH2O platform supplemental features will be generated, such as the influence of social awareness (obtained by the Gamification component) or the sensibility to price changes (obtained by the pricing engine).

In summary, through this component, the water utility can visualize the water consumption of each customer at a fixture/appliance level, in order to identify consumption patterns and trends, and thus identifying the most promising areas where conservation efforts may be polarized. For a description of the algorithms exploited to model the user behaviour, see deliverable *D3.2: First user behaviour models*.

The **Agent Based Modelling** component allows the water utility to simulate whole districts of users,

thus extrapolating user models provided by the Models of User Behaviour component at a larger scale and also extrapolating the impact of network effects due to users' interactions, both in the physical and in the virtual world. The agent based model includes influence/mimicking mechanisms and social interaction among the consumers, and thus will be employed by the water utility to understand how some user types (leaders/influencers) can stimulate a behavioural change on other users.

The **Authentication Gateway** component centralizes user registration into the SmartH2O platform database for the users registered in the components having own user database.

Also, this component provides an unique point for authentication to all the users primarily registered at a component level, in order to allow logging in to other components by using the original set of credentials without the need to perform another registration.

The **Social Network Crawler and Data Analyser** component allows the platform, where deemed appropriate by the water utility portal, to launch social data analysis campaigns to identify relevant users and content in the area of sustainable water consumption. For example, this component supports the crawling of Twitter data in order to automatically find people and content relevant for a thematic area, such as water consumption.

The **Social Network Connector** component has a dual role with respect to the Social Network Crawler and Data Analyser; it allows the Consumer, Player, and Competitor users to post their achievements from the SmartH2O Water Utility Portal and Games Platform to the social network of their preference, in order to engage people from their social circle to the water consumption and sustainability campaigns of the water utility company.

4. Portal data exchange manager

4.1 Context

This component processes raw consumption data received from Water Utility. It matches it with user and household information and aggregates raw data.

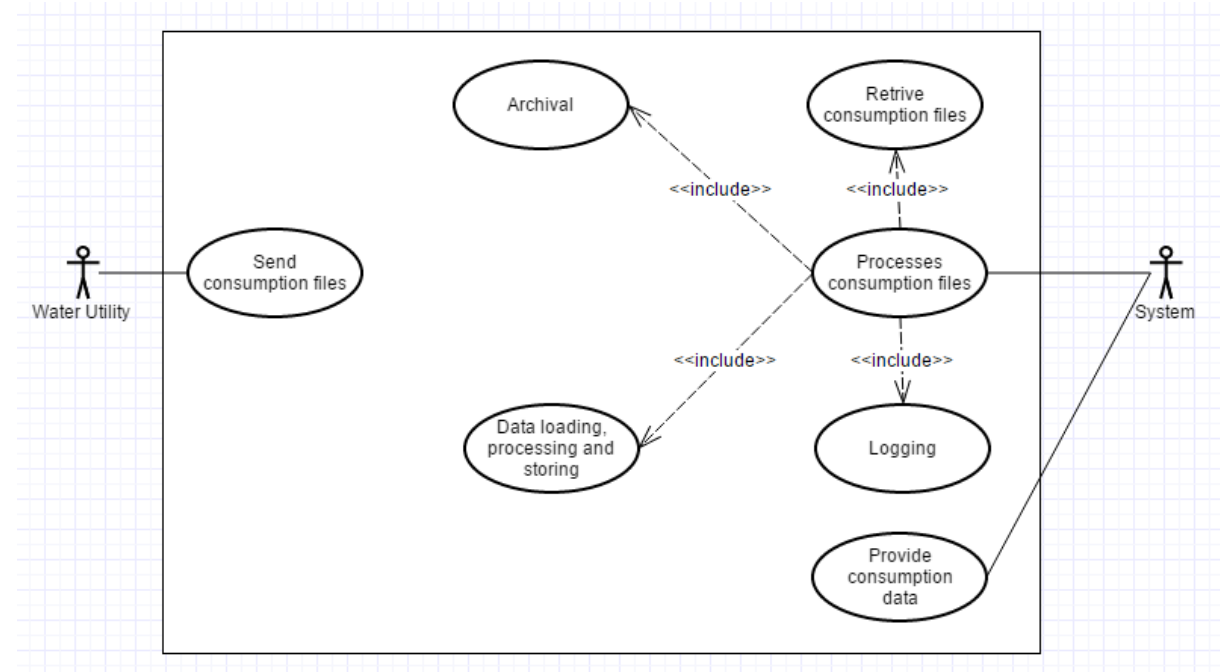


Figure 4 - Use Cases Smart Metered Data Manager.

This component assumes that Water Utility will upload consumption files to a specified location that is available through FTP to Smarth2O Platform. The functionality of this component assumes that consumption files will be available in the specified location.

4.1.1 Processing of water consumption files

The processing of input files will be staged in these sub-steps:

- Retrieval of raw consumption data files.
- Raw data loading, parsing and aggregation.
- Aggregated data storage in Smarth2O database.
- Archival of processed files.
- Logging of relevant information for each of the steps mentioned above.

4.1.2 Retrieval of water raw consumption files

The system assumes that Water Utility will send raw consumption files via SFTP in a specified location. The Water Utility partner will also upload MD5 signature file to ensure validation of transferred data file.

The system will periodically monitor the location for new files. When a new data file is identified:

- it is copied in a staging area.
- its MD5 signature is verified.
- if file is valid, it is moved to processing area.

4.1.3 File processing

The system parses the file and raw data will be aggregated from meter level at household level. Each household can have one or more meters.

Aggregated consumption data is stored by the system in Platform database. Initial data is also stored in the database for reference purposes.

4.1.4 Archival of water consumption files

After a file is processed, the system will encrypt the file and it will copy it into the archive location of the platform File System. The archival location will be set by Admin User set administration module.

4.1.5 Logging

The system will log errors, warnings and successful completion information of the relevant events during the entire process:

- File retrieval -> File processing -> File archival

The logging information will be stored by the system in a file on the Platform FileSystem. File location, file naming, maximum size, rolling policy will be configured by Admin User.

4.1.6 Administration

The Admin User sets different parameters that have an impact in this component functionality.

The Admin User monitors in the log file details of any execution step.

The Admin User monitors in real-time the status and utilization of each resource of the component

4.2 Composition

This component will be organized in modules with distinct roles as in Figure 5.

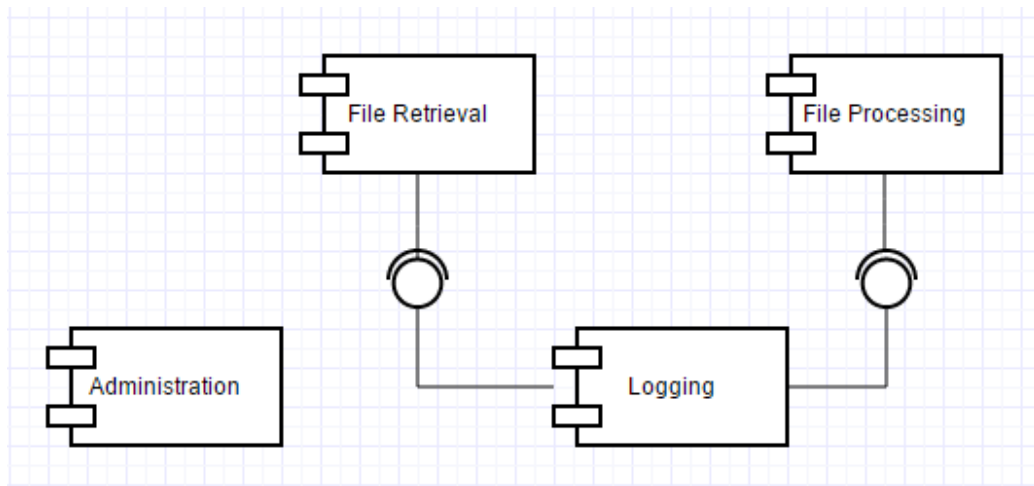


Figure 5 - Structure diagram of SDMC.

4.2.1 File retrieval module

This subcomponent handles the retrieval of data and MD5 files. Data files are verified against MD5 signature files. If the file is validated against its signature then the file is copied to processing area.

4.2.2 File processing module

In order to ensure availability and scalability of the system data files will be processed using the Apache Hadoop distributed cluster. The structure of a typical processing cluster is presented in Figure 6.

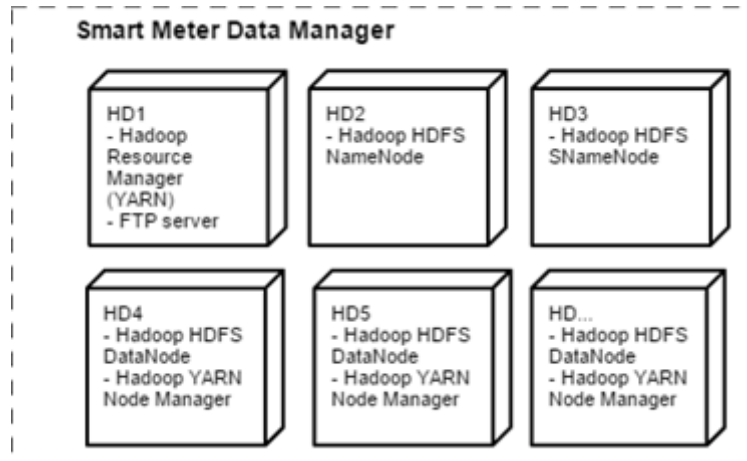


Figure 6 - Components of Apache Hadoop distributed processing.

Note: Details of the Apache Hadoop distributed infrastructure can be found in “D6.2 Platform Architecture and Design” document.

A typical file processing flow in the Apache Hadoop cluster takes following steps:

- File content is parsed by one or more processing nodes. Nodes of type “DataNode” as in Figure 6.
- In each processing node, parsed content is processed by a map-reduce engine; more readings of the same metered are aggregated at meter level.
- The result of each processing node is aggregated in a final result.
- Final result is saved to Consumption database.

4.2.3 Logging module

This module will output relevant and execution information during the entire process. The logging of information will be triggered by events like:

- Resource initialization.
- Resource availability.
- Resource unavailability.
- Start of an operation.
- Outcome of an operation: success, warning, error.
- User generated events.

A log entry will contain relevant details for the event that triggered the log like: timestamp, event initiator, event type, message, the execution context and the error message in case of warnings and errors.

4.2.4 Administration module

This Admin User will set in a configuration file execution parameters like:

- file retrieval location
- file archival location
- log level

- authentication credentials

This module will also allow Admin User to monitor resource utilization of the distributed processing infrastructure. The Admin User will use the Apache Ambari (ambari.apache.org) to monitor execution and resource utilization. This administration console allows the Admin User to provision, manage and monitor Apache Hadoop clusters.

Note: The deployment architecture of the Apache Hadoop cluster will be specific for each Water Utility partner. This document does not detail the initial deployment architecture nor the updates needed to accommodate up scaling or downscaling requirements of Water Utility partner.

5. User and Consumption Database

5.1 Information

This component will store most of data of the Smart H2O Platform. The database will store following main categories of data:

- **User and Household data.** These data are of type master data. This database mainly consist of sociographic user information and information on the household. Part of the data are provided by Water Utility and part of the data are provided by User when using the platform.
- **Consumption data.** These data are transactional data of water consumption. The data are provided by Water Utility in raw files, which are subsequently processed by the platform and stored in this database.
- **Administration data.** These data will contain configuration and administrative information like: Admin users, SFTP parameters, authentication parameters, logging parameters

Note: Gamification data will be stored in a distinct database in the Games Platform. Details in [Games Platform](#) chapter

The data model as from *D6.2 – Architecture and design* is presented in Figure 7, Figure 8, Figure 9, Figure 10, and Figure 11.

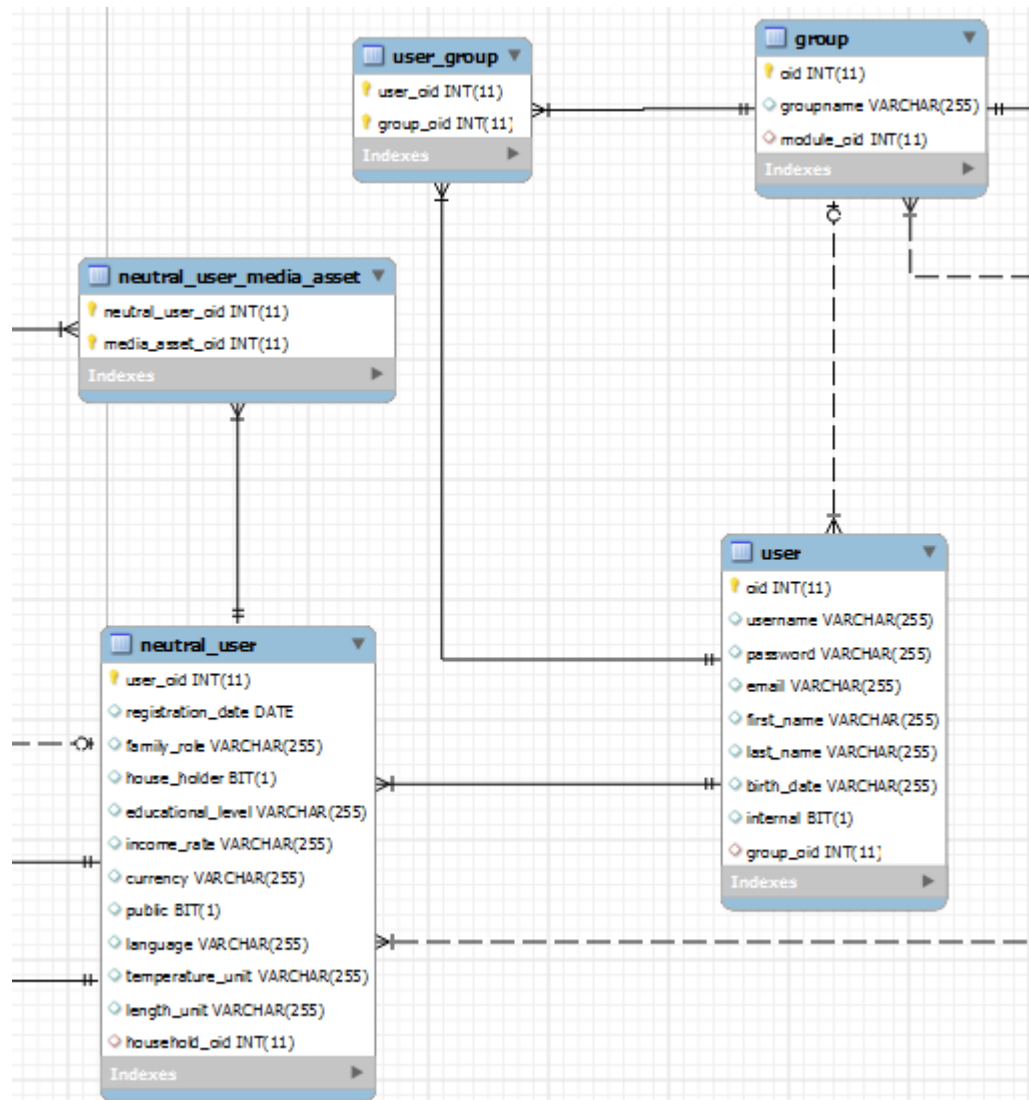


Figure 7 - User related entities.

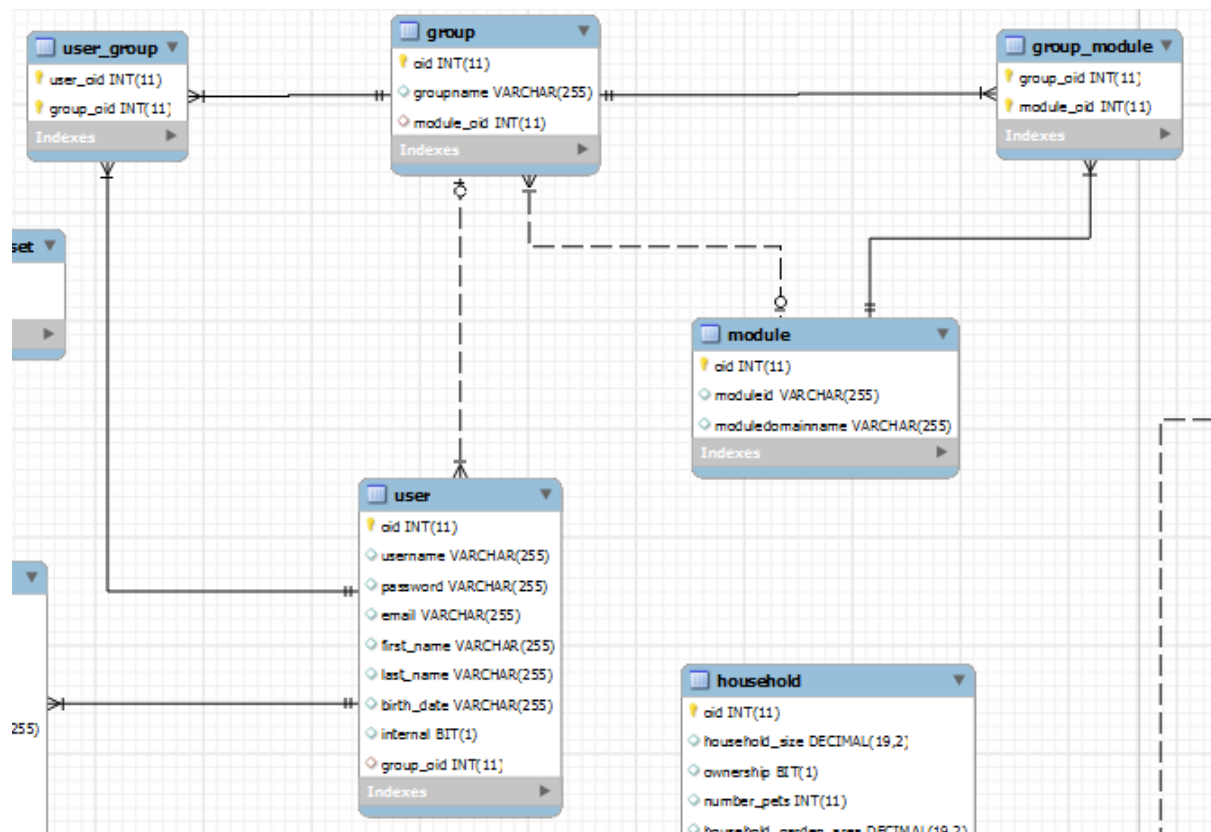


Figure 8 - Group related entities.

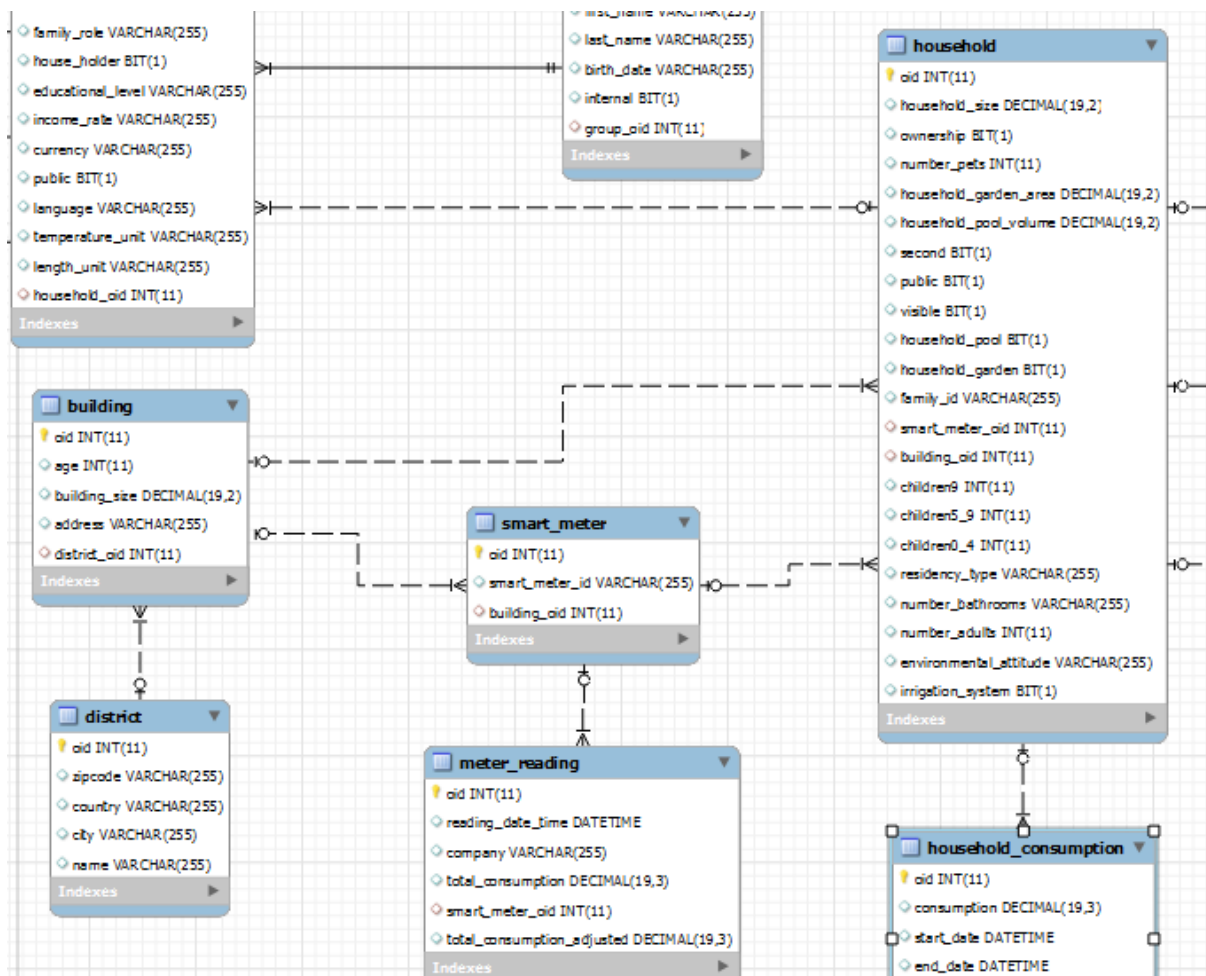


Figure 9 - Household and smart meters related entities.

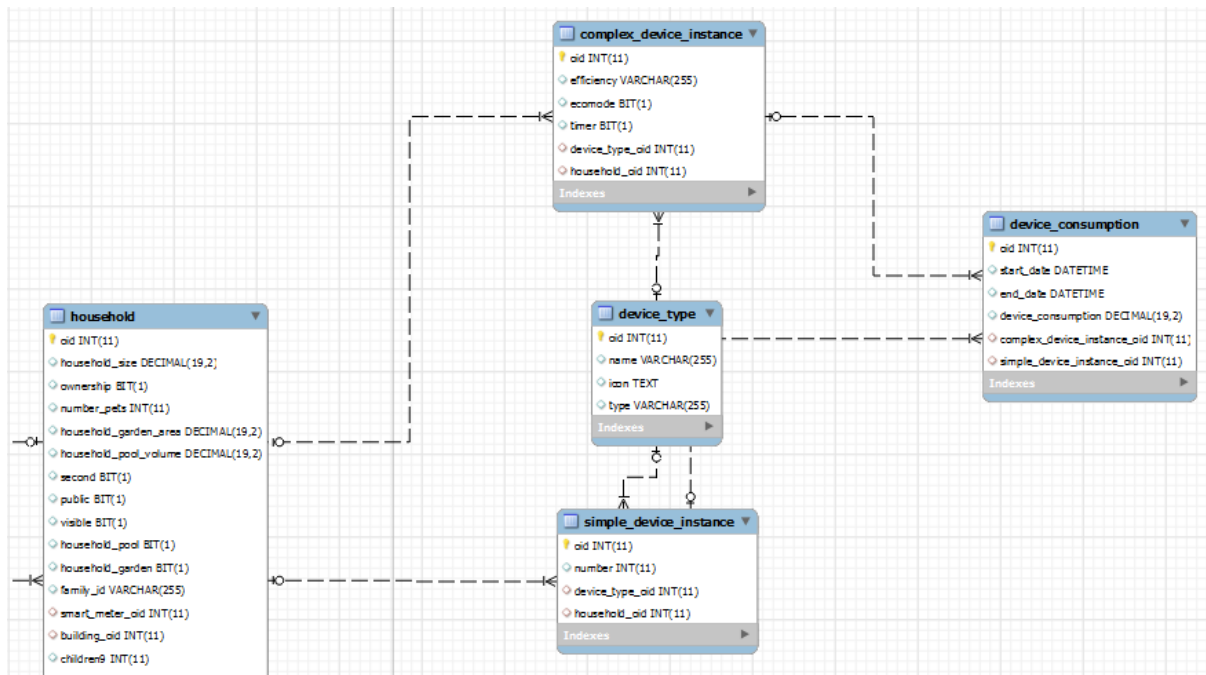


Figure 10 - Household and device related entities.

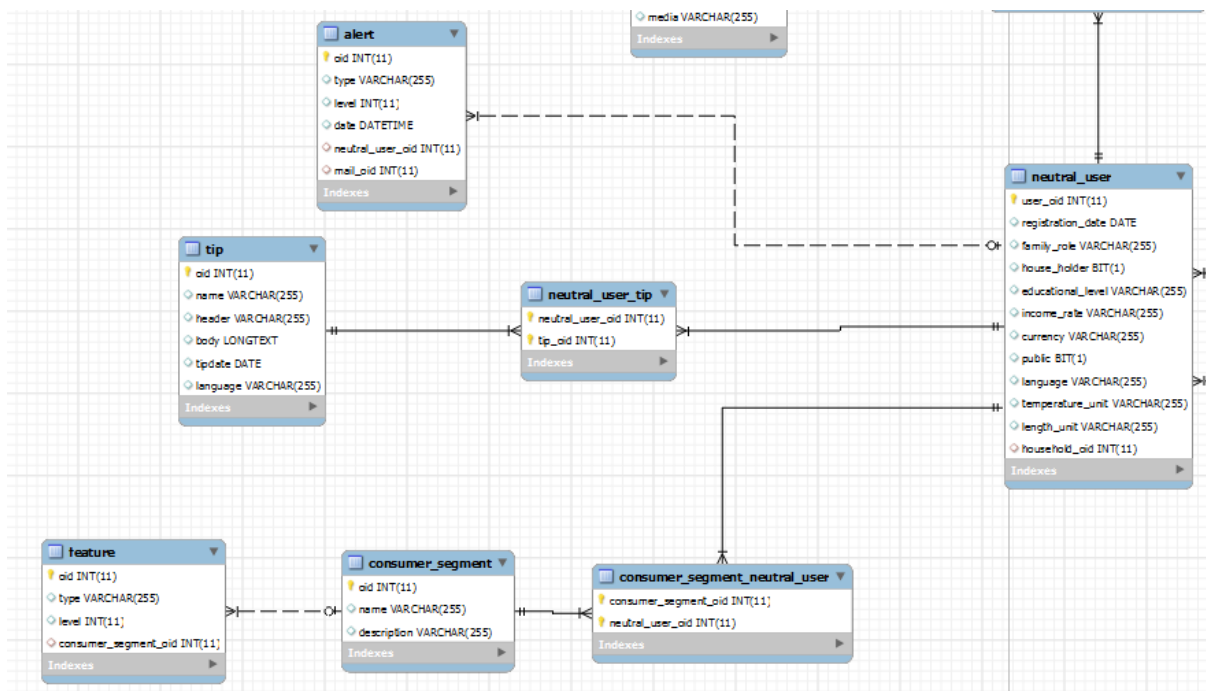


Figure 11 - Utility Customer (neutral user) related entities.

The database will store the entities listed in Table 3.

Table 3. Entities of the user and consumption database.

Entity name	Attributes	Description
Group	<TBD>	Defines user groups in the Platform
Group module		Define relationships between groups and modules
User		Define platform users
User group		Define what groups a user belongs to
Neutral user		Define profile information of utility customers and users. Neutral users may or may not be users of the platform.
Consumer segment		Define the customer segments that can be created by UtilityConsumerSupervisors
Consumer segment neutral user		Links a customer to a segment
Household		Define the living unit space as defined by utility company
Building		Define the building that a household belong
District		Define the districts of areas covered utility company. A building is included in a district
Smart meter		Define the smart meter device installed in a household for individual metering or at building level for collective metering
Meter reading		Raw metered data read at a specified timestamp.
Simple device instance		Records the simple devices installed in a household
Complex device instance		Records the complex devices installed in a household
Device consumption		Records consumption recorded at device level

5.2 Interface

This component will implement a Data Service layer that will provide retrieval, creation and update of the information in this database. This DataService layer will be implemented as RESTful WebServices. These WebServices will be enlisted as Publishers in the integration component (ESB) to be further consumed by different Platform components.

This DataService layer will be structured in following main groups of services:

- User and Household
- Water consumption

5.2.1 User and Household Services

This group will expose the consumption data to other components. Based on the requirements document D2.2, various other components of the platform will request consumption data aggregated on different criteria:

- User
- Household
- Neighbourhood
- Timeframe

This group will implement WebServices that provide User and Household data operations, as listed in Table 4.

Table 4. Web services providing operations related to the user and household entities.

Service Name	Input	Output
CheckExistingEmail	Email	Json
CheckExistingUsername	username	Json
CustomerPortalSignup	customerDetails{ "username" : "fred", "email" : "fred", "smartMeterID" : "fred", "familyID" : "fred", "zipcode" : "fred", "country" : "fred", "password" : "fred" }	Json
GetAlerts	User_id	Json
GetBill	User_id	Json
GetBuildingInfo	User_id	Json Object
GetDevicesInfo	User_id	Json Object { { "simple_devices" : [{ "name" : "fred", "number" : 1968893679, "icon" : "fred" }], { "name" : "fred", "number" : 1968893679, "icon" : "fred" }], "complex_devices" : [{ "name" : "fred", "icon" : "fred", "efficiency" : "fred",

		<pre> "timer" : true, "ecomode" : true }, { "name" : "fred", "icon" : "fred", "efficiency" : "fred", "timer" : true, "ecomode" : true }], "completion_percentage" : "fred" } } </pre>
GetHomeInfo	User_id	Json Object
GetNeighbourhood	User_id	Json Object
GetTips	<pre> { User_id:fred, Lang:"en" } </pre>	Json Object
GetVideos	User_id	Json Object
SetBuildingInfo	<pre> { "address" : "fred", "zipcode" : "fred", "country" : "fred", "residence_type" : "fred", "age" : 1968893679, "building_size" : 1968893679, "pool" : true, "pool_size" : 1968893679, "garden" : true, "garden_size" : 1968893679, "userID" : 1968893679 } </pre>	<pre> Json { "result" : true } </pre>
SetDevicesInfo	<pre> { "simple_devices" : [{ "name" : "fred", "number" : 1968893679 }, { "name" : "fred", "number" : 1968893679 }], "complex_devices" : [{ "name" : "fred", </pre>	<pre> Json { "result" : "fred" } </pre>

	<pre> "efficiency" : "fred", "timer" : true, "ecomode" : true }, { "name" : "fred", "efficiency" : "fred", "timer" : true, "ecomode" : true }], "userID" : 1968893679 } </pre>	
SetHomeInfo	<pre> { "address" : "fred", "zipcode" : "fred", "country" : "fred", "residency_type" : "fred", "owner" : true, "second_home" : true, "residency_size" : "30.47225414130857501504578976891934871673583984375", "building_size" : "30.47225414130857501504578976891934871673583984375", "age" : 1968893679, "number_bathrooms" : 1968893679, "pool" : true, "pool_size" : "30.47225414130857501504578976891934871673583984375", "garden" : true, "garden_size" : "30.47225414130857501504578976891934871673583984375", "irrigation" : true, "userID" : "fred" } </pre>	<pre> { "result" : true } </pre>
SetHouseholdInfo	<pre> { "number_adults" : 1968893679, "number_children_0_4" : 1968893679, "number_children_4_9" : 1968893679, "number_children_9" : 1968893679, "number_pets" : 1968893679, "userID" : 1968893679, "environmental_attitude" : "fred" } </pre>	<pre> { "result" : true } </pre>
GetDevices		<pre> { "devices" : [{ "name" : "fred", "icon" : "fred", </pre>

		<pre>"type" : "fred" }, { "name" : "fred", "icon" : "fred", "type" : "fred" }] }</pre>
UserValidation	<pre>{ User_email:"fred@gred.com", User_pasword:"*****" }</pre>	Json

5.2.2 Consumption Data Services

This group will implement the WebServices related to consumption data, as listed in Table 5.

Table 5. Web services providing access to consumption data.

Service Name	Input	Output
GetConsumptionData	User_id	Json
GetConsumptionDataHourly	User_id	Json Array
GetConsumptionSummary	User_id	Json
GetCurrentMeterReading	User_id	Json Array
GetLastReadingDate	User_id	Json Array
GetNeighbourhoodConsumption	User_id	Json Object
GetThisMonthAverageConsumption	User_id	Json Object
GetThisMonthTotalConsumption	User_id	Json Object
GetThisWeekAverageConsumption	User_id	Json Object
CheckValidSmartMeterID	smartmeterid	Json Object

6. Basic Customer Portal

6.1 Context

6.1.1 Consumption-related use cases

This set of use cases describes the basic interaction of the customer with the consumption visualization functionalities in the basic portal, including the verification of aggregated data.

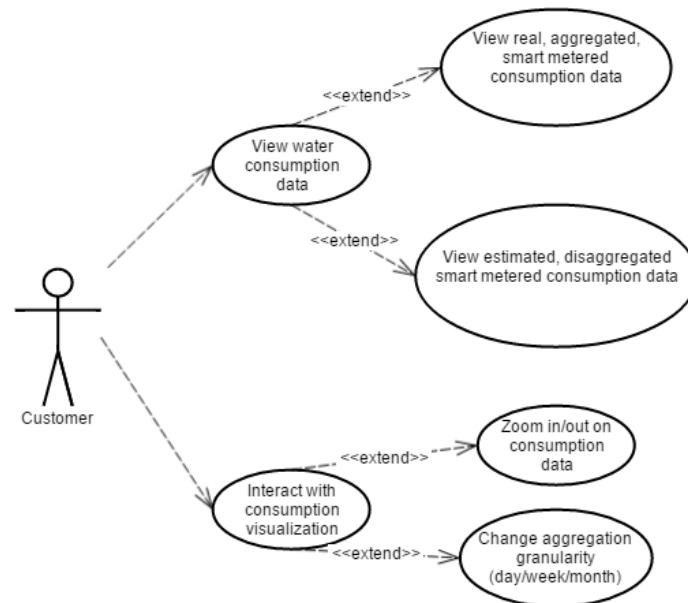


Figure 12 - USE CASE 8.3: Visual exploration of water consumption information.

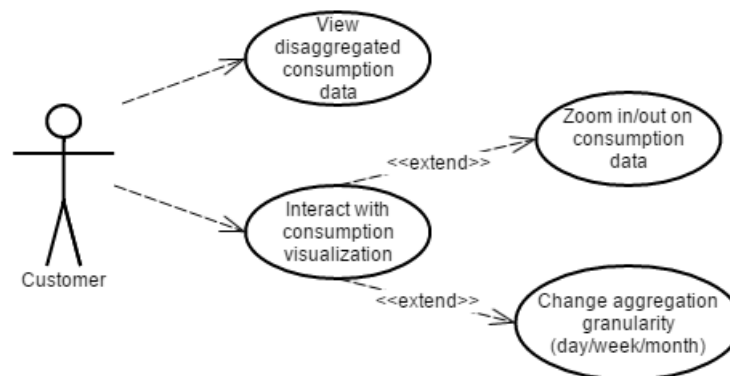


Figure 13 - USE CASE 8.4: Visual exploration of water consumption at fixture/appliance level.

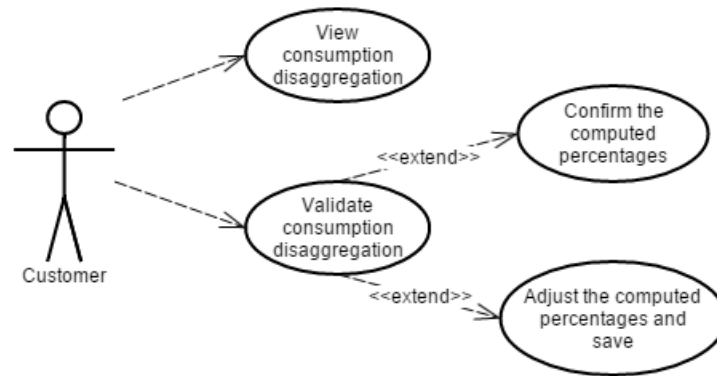


Figure 14 - USE CASE 8.5: Providing feedback on disaggregated consumption data.

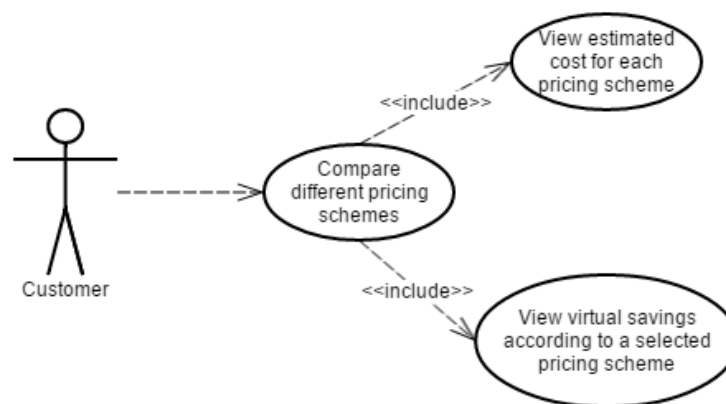


Figure 15 - USE CASE 8.9 Learning interactively about innovative pricing schemes.

6.1.2 Learning-related use cases

This set of use cases describes the learning activities and educational content provided to the customers.

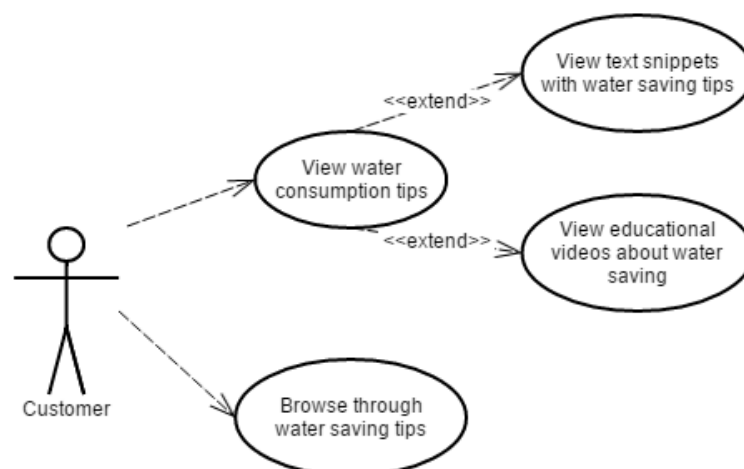


Figure 16 - USE CASE 8.7: Getting water consumption tips.

6.1.3 Notification-related use cases

This set of use cases describes the process of notifications used in the interface, for example in case of water consumption distress.

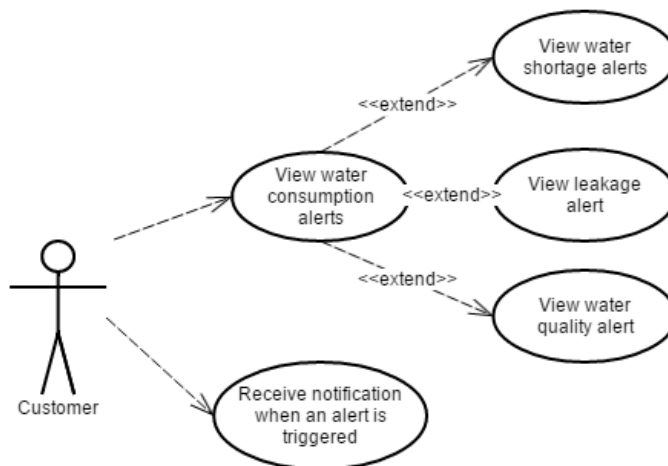


Figure 17 - USE CASE 8.6: Getting water consumption alerts.

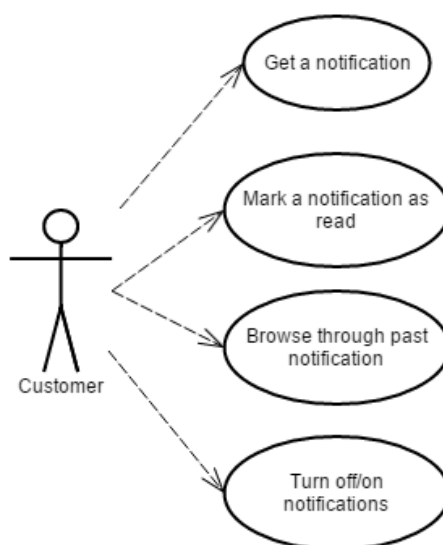


Figure 18 - USE CASE 8.8: Getting system notifications.

6.2 Composition and structure

According to the IFML ¹terminology, the Basic Customer Portal represents a Site View, accessible to all the customers belonging to the group “Neutral User”.

¹ <http://www.ifml.org>

6.2.1 Summary Sections

In this section a description of the main sections of the site view is presented. Figure 19 represents a graphical overview of the Basic Customer site view.

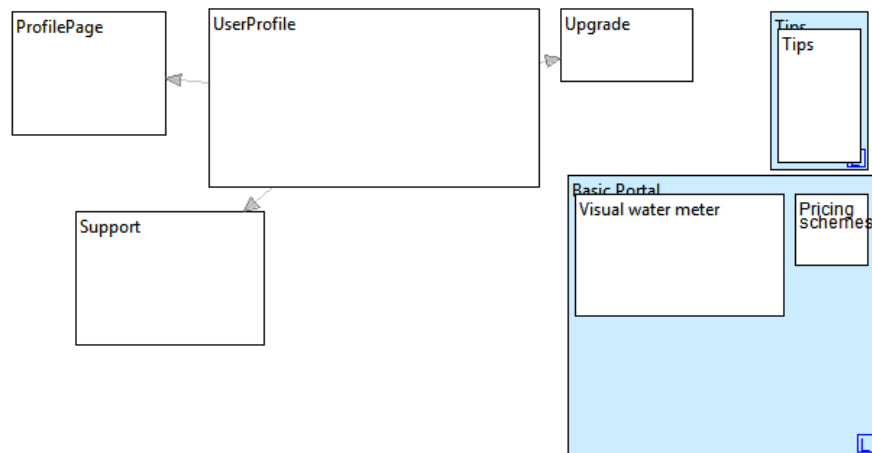


Figure 19 - Basic Customer Site View – Overview.

Area Summary

Basic Portal

The Basic Portal area is related to consumption. In this area consumption visualization is presented at different levels of detail and abstraction. The Basic Portal area is the default area a user accesses after the login.

Tips

The Tips area is related to learning. In this area consumption tips and educational videos are offered to users, in order to enhance their awareness.

Page Summary

ProfilePage

The ProfilePage shows the current user profile details (e.g username, email, profile image) and contains a form to update profile information.

Support

The SupportPage allows the user to notify a problem identified on the portal.

Upgrade

The Upgrade page shows a preview of the features composing the advanced version of the portal, in order to persuade users to upgrade.

UserProfile

The master page UserProfile contains all the elements which are constant in all the pages of the view. It consists of the top bar elements (username box, logout and profile dropdown, localization dropdown, last reading box and upgrade button).

Component Summary

[ModuleInstanceUnit] Change Language

This action contains the operations needed to change the current language of the portal.

[ModuleInstanceUnit] Logout

This action contains the operations needed to perform the logout.

[ModuleInstanceUnit] SendBugEmail

This action contains the operations needed to send a notification email to the administrator, when a user notifies a problem on the portal.

[ModuleInstanceUnit] Update Profile

This action contains the operations needed to update the user profile. The operations include the update on the local database and the invocation of the external service to update user data also in the central database of the SmarH2O system.

[ModuleInstanceUnit] Upgrade

This action contains the operations needed to upgrade a customer to the Advanced Portal. The operations include the update of the default group of the user (from Basic User to Advanced User) and the assignment of an action related to the Upgrade event.

Incoming Flow Summary

→ English	from  UserProfile /  [NoOpContentUnit] Change Language	to  [ModuleInstanceUnit] Change Language
→ Italian	from  UserProfile /  [NoOpContentUnit] Change Language	to  [ModuleInstanceUnit] Change Language
→ Spanish	from  UserProfile /  [NoOpContentUnit] Change Language	to  [ModuleInstanceUnit] Change Language
→ Logout	from  UserProfile /  [NoOpContentUnit] Logout&Profile	to  [ModuleInstanceUnit] Logout
→ SaveButton	from  ProfilePage /  [EntryUnit] Update Profile	to  [ModuleInstanceUnit] Update Profile
→ SendButton	from  Support /  [EntryUnit] Report a bug	to  [ModuleInstanceUnit] SendBugEmail
→ Upgrade	from  Upgrade /  [NoOpContentUnit] Upgrade	to  [ModuleInstanceUnit] Upgrade

6.2.2 UserProfile [MasterPage]

In this section a description of the UserProfile master page is presented. Figure 20 represents the structure of the page, in terms of components and links.

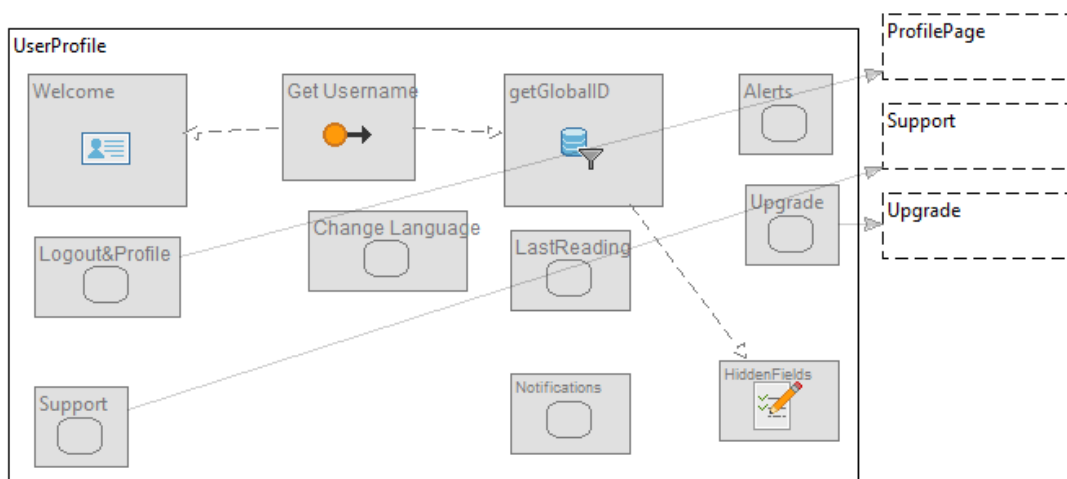


Figure 20 - UserProfile Master Page.

The master page UserProfile contains all the elements, which are constant in all the pages of the view. It consists of the top bar elements (username box, logout and profile dropdown, alerts, localization dropdown, last reading box and upgrade button).

Components:

☐ [NoOpContentUnit] Alerts

This component shows the alerts, e.g. warning the user about possible leaks or bad water quality.

☐ [NoOpContentUnit] Notifications

This component shows the notifications, e.g. new user achievements.

This component shows the alerts, e.g. warning the user about possible leaks or bad water quality.

☐ [NoOpContentUnit] Change Language

This component represents the localization feature, a dropdown list of all the available languages for the portal, allowing the user to change the current language.

➡ [GetUnit] Get Username

This component retrieves the identity of the logged user from the session.

🔍 [SelectorUnit] getGlobalID

This component retrieves the global identity (the user identifier in the central database of the SH2O System) of the logged user.

☐ [NoOpContentUnit] LastReading

This component shows the last smart meter reading of the logged user.

☐ [NoOpContentUnit] Logout&Profile

This component represents a dropdown with the links to the user profile and the link to perform the logout.

☐ [NoOpContentUnit] Support

This component represents a link to the Support page, in which the user can notify issues related to the portal.

☐ [NoOpContentUnit] Upgrade

This component represents a box in the top bar persuading the user to upgrade to the Advanced

version.

 [DataUnit] Welcome

This component represents a box with logged user details (e.g. username). It receives as input the user identity.

6.2.3 ProfilePage [Page]

In this section a description of the Profile page is presented. Figure 21 represents the structure of the page, in terms of components and links.

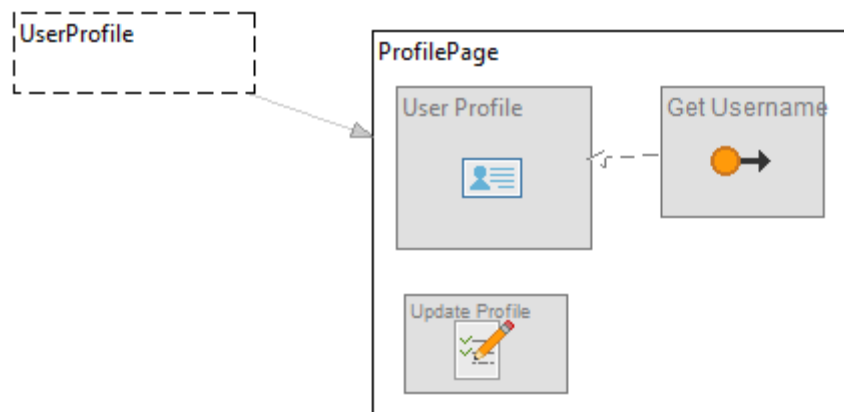


Figure 21 - ProfilePage overview.

The ProfilePage shows the current user profile details (e.g username, email, profile image) and contains a form to update profile information.

Components:

➡ [GetUnit] Get Username

This component retrieves the identity of the logged user from the session.

 [EntryUnit] Update Profile

This component is a form to update profile information.

 [DataUnit] User Profile

This component shows the current profile information provided by the user (e.g. username, email, profile image). It receives as input the user identity.

6.2.4 Support [Page]

In this section a description of the Support page is presented. Figure 22 represents the structure of the page, in terms of components and links.

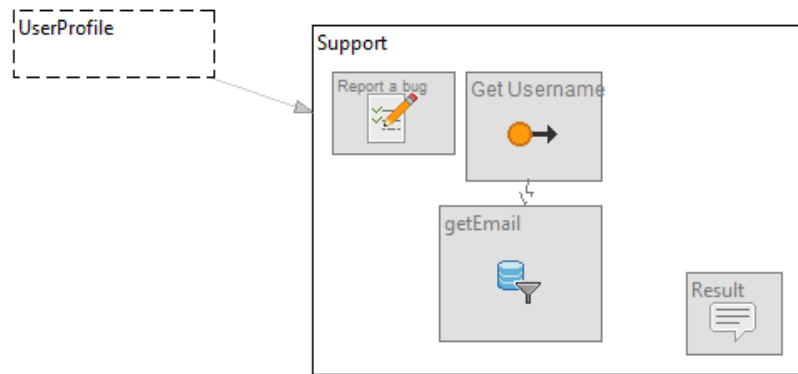


Figure 22 - Support Page.

The SupportPage allows the user to notify a problem identified on the portal to the administrator and technical staff.

Components:

➡ [GetUnit] Get Username

This component retrieves the identity of the logged user from the session.

📧 [SelectorUnit] getEmail

This component retrieves the email address of the user, sent together with the mail message to the administration for future communication between user and customer about the notified problem.

📝 [EntryUnit] Report a bug

This component is a form to allow the user to describe the identified problem. The user can specify the object and the body of the message.

💬 [MultiMessageUnit] Result

A message is shown on the portal after the completion of the SendBugMail action. It can be either an error or a success message.

6.2.5 Upgrade [Page]

In this section a description of the Upgrade page is presented. Figure 23 represents the structure of the page, in terms of components and links.

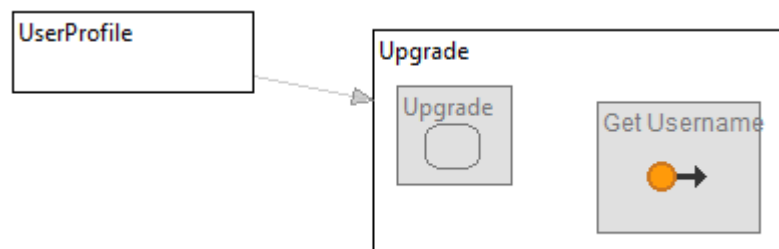


Figure 23 - Upgrade Page.

The Upgrade page shows a preview of the features composing the Advanced version of the portal, in order to persuade users to upgrade.

Components:

➡ [GetUnit] Get Username

This component retrieves the identity of the logged user from the session.

☐ [NoOpContentUnit] Upgrade

This component is a carousel of images showing the main features of the Advanced Portal.

6.3 Interaction, State dynamics

6.3.1 Consumption-related use cases

Consumption visualization is presented at different levels of detail and abstraction in the Basic Portal area. In this section a description of the Basic Portal area is presented, highlighting the relation between the components and the use cases. **Figure 24** represents the structure of the area, in terms of pages and components.

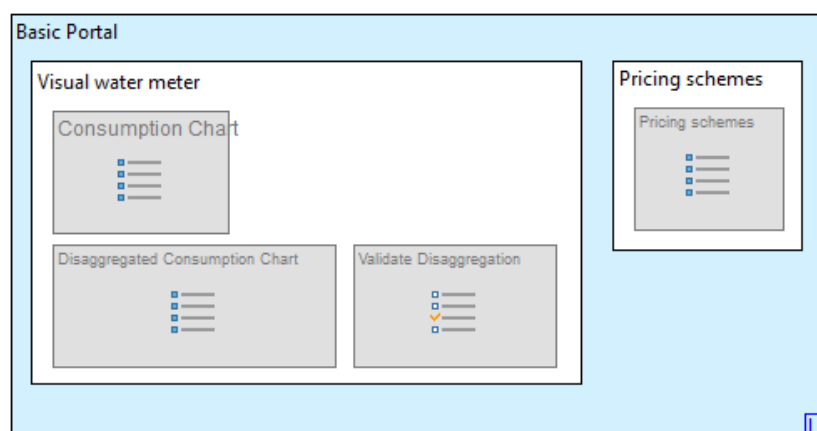


Figure 24 - Basic Portal Area.

USE CASE 8.3: Visual exploration of water consumption information

Context:

Visual water meter [Page]

Component:

☰ [PowerIndexUnit] Consumption Chart

This component shows the user personal consumption. It is composed of two tabs presenting water consumption with different approaches. The first tab represents the Detailed consumption, showing a detailed chart with consumption data. The second tab represents an Overview of the user average consumption.

USE CASE 8.4: Visual exploration of water consumption at fixture/appliance level

Context:

Visual water meter [Page]

☰ [PowerIndexUnit] Disaggregated Consumption Chart

This component visualizes the disaggregated water consumption data of the user at a fixture/appliance level.

USE CASE 8.5: Providing feedback on disaggregated consumption data

Context:

Visual water meter [Page]

Component:

 [MultiChoiceIndexUnit] Validate Disaggregation

This component shows the user the disaggregation percentages computed on the consumption data, that the user can validate.

USE CASE 8.9 Learning interactively about innovative pricing schemes

Context:

Pricing schemes [Page]

Component:

 [PowerIndexUnit] Pricing schemes

This component shows different pricing schemes computed according to different patterns of consumption.

6.3.2 Learning-related use cases

Some educational content, related to water consumption, is presented to the user in the Tips area. The area consists of a single page with two main components: tips and videos.

In this section a description of the Tips area is presented. Figure 25 represents the structure of the area, in terms of pages and components.

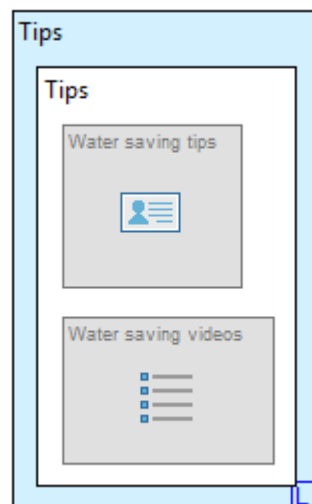


Figure 25 - Tips Area.

USE CASE 8.7: Getting water consumption tips

Context:

Tips [Page]

Components:

 [DataUnit] Water saving tips

This component is a list of water saving tips, each characterized by a Title and a Body.

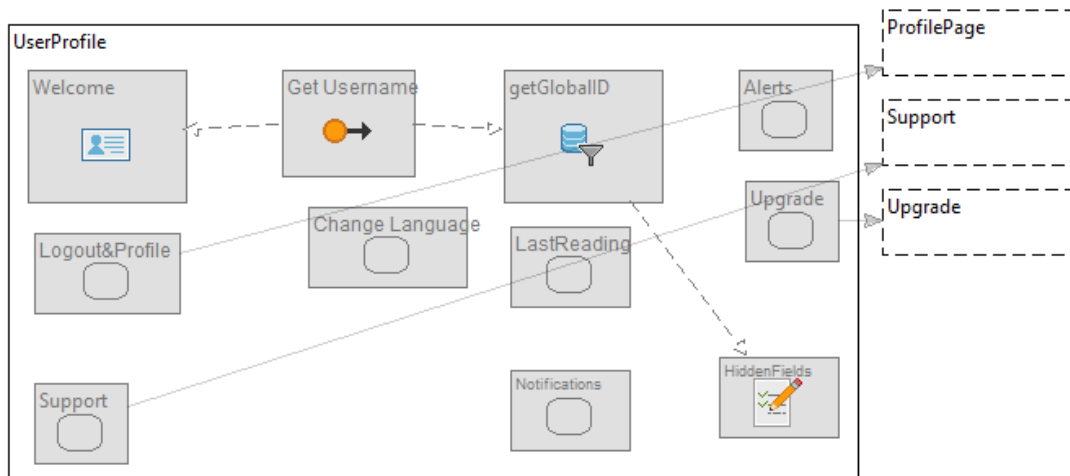
☰ [PowerIndexUnit] Water saving videos

This component is a list of water saving videos, each characterized by a title and a url of the media content.

6.3.3 Notification-related use cases

A notification system is needed inside the portal to inform the user in case of water consumption distress or simply to update users about news, portal activity, events, and problems.

Notification and alerts are always visible to the user and are visualized in all the pages, as shown in



USE CASE 8.6: Getting water consumption alerts

Context:

UserProfile [MasterPage]

Component:

☐ [NoOpContentUnit] Alerts

This component shows the alerts, e.g. warning the user about possible leaks or bad water quality.

USE CASE 8.8: Getting system notifications

Context:

UserProfile [MasterPage]

Component:

☐ [NoOpContentUnit] Notifications

This component shows the notifications, e.g. new user achievements.

7. Advanced Customer Portal

7.1 Context

7.1.1 Gamification actions, badges and rewards

This set of use cases describes the interactions of users with the gamification features.

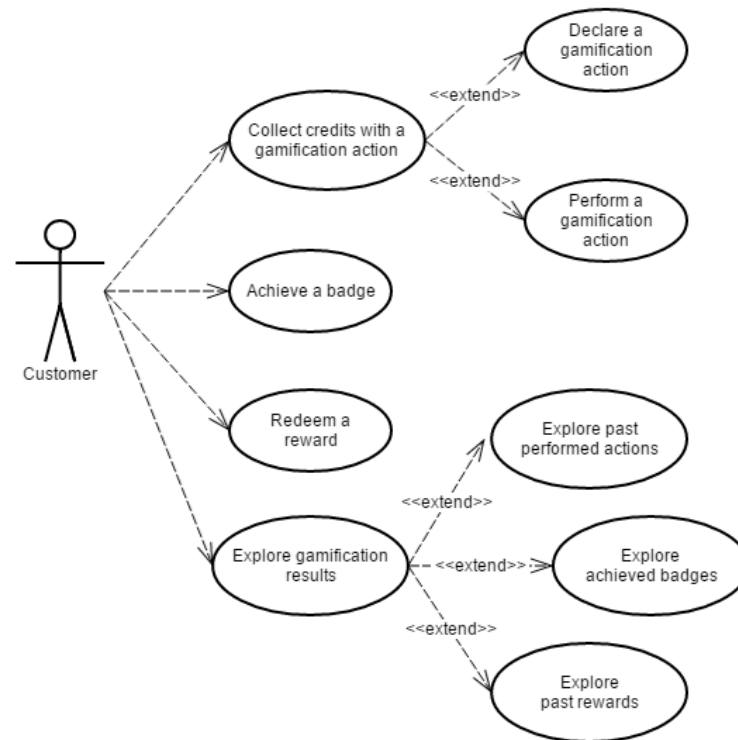


Figure 26 - USE CASE 9.1: Making gamification actions and exploring results.

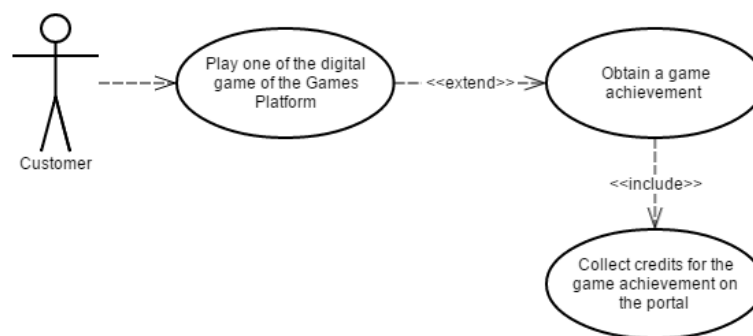


Figure 27 - USE CASE 9.9: Making actions and earning digital credits with the Games Platform.

7.1.2 Gamified water meter use cases

This set of use cases describes the advanced interaction of the customer with the consumption functionalities in the advanced portal, including declaring water use events or performed actions and consumption goals.

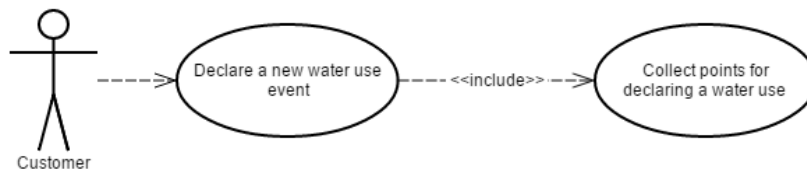


Figure 28 - USE CASE 9.7: Declaring water end-use events.

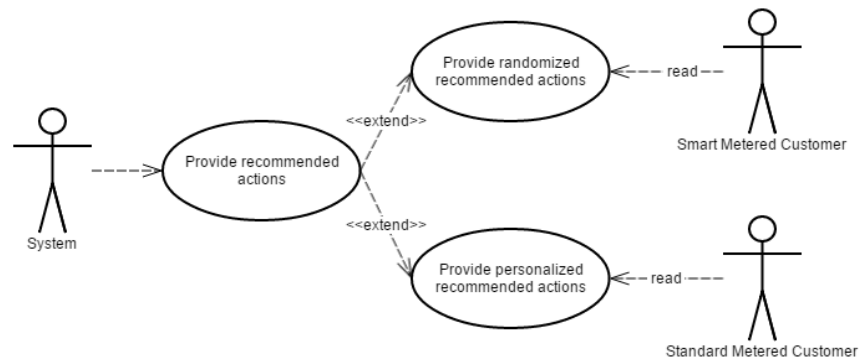


Figure 29 - USE CASE 9.4: Recommending water saving actions.

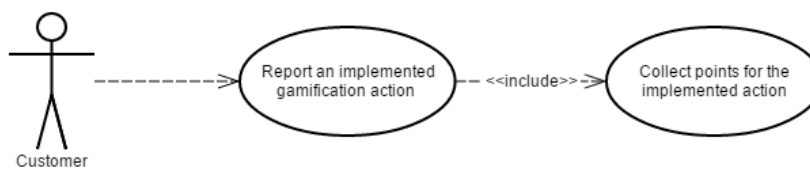


Figure 30 - USE CASE 9.5: Declaring water saving actions.

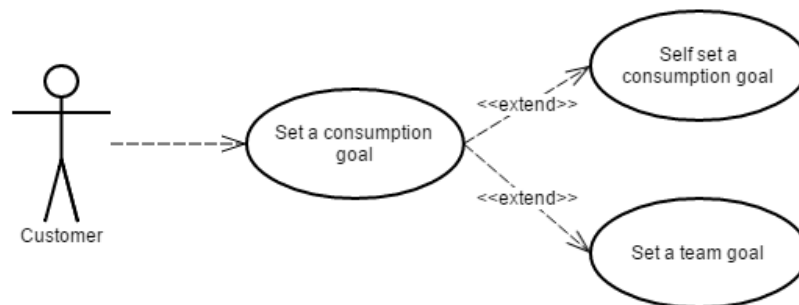


Figure 31 - USE CASE 9.2: Self setting consumption goals.

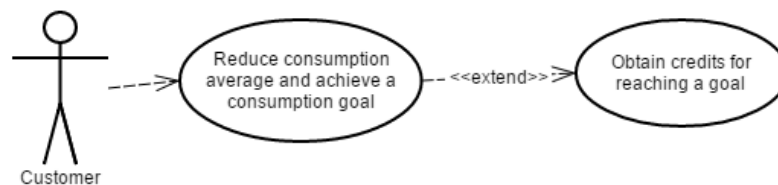


Figure 32 - USE CASE 9.3: Fulfilling consumption goals.

7.1.3 User profiling use cases

This set of use cases describes the profiling features, meant to collect a subset of psychographic variables about users and their households.

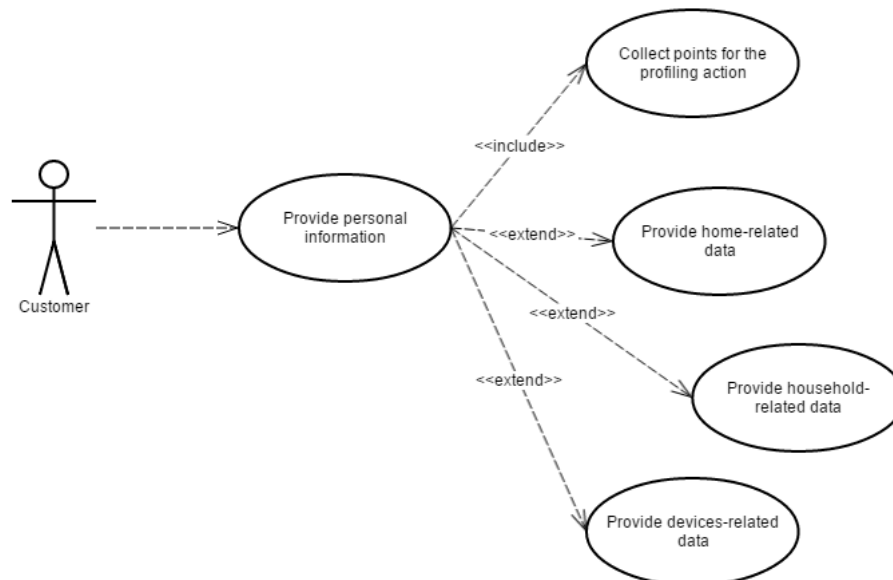


Figure 33 - USE CASE 9.6: Contributing household and user profiling information.

7.1.4 Social use cases

This set of use cases describes the social features, including leaderboards and teams.

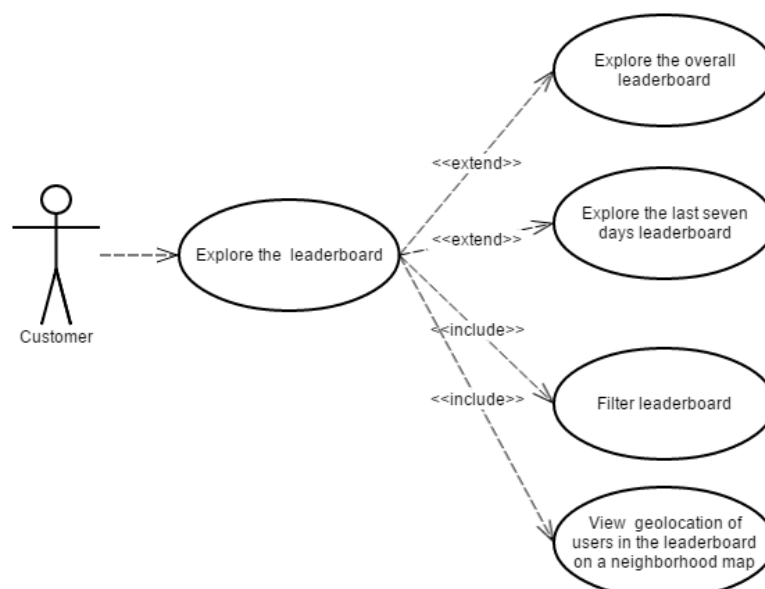


Figure 34 - USE CASE 10.1: Comparing achievements with other households.

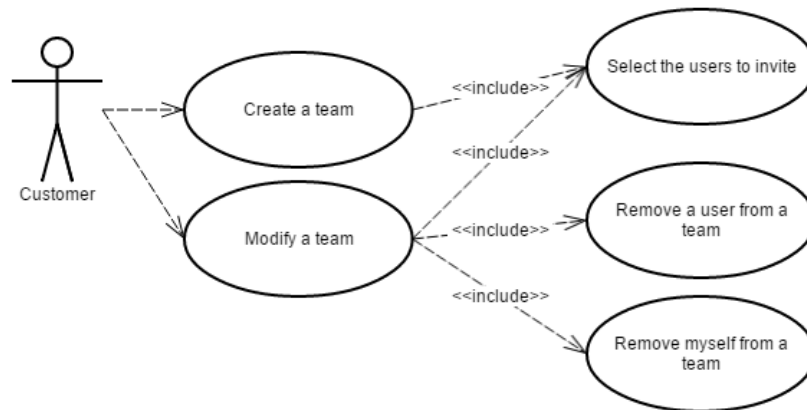


Figure 35 - USE CASE 10.2: Inviting another user to join a team.



Figure 36 - USE CASE 10.3: Achieving goals collaboratively as a team.

7.1.5 User account management use cases

This set of use cases describes the user account management features, including the modification of the personal profile and settings, the possibility to upgrade/downgrade to a different version, the unsubscription and the opt-in/out to appear in public leaderboards and map.

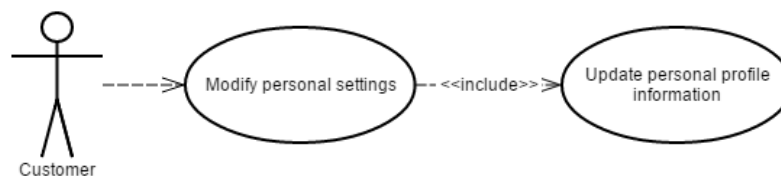


Figure 37 - USE CASE 11.3: Modifying User Settings.

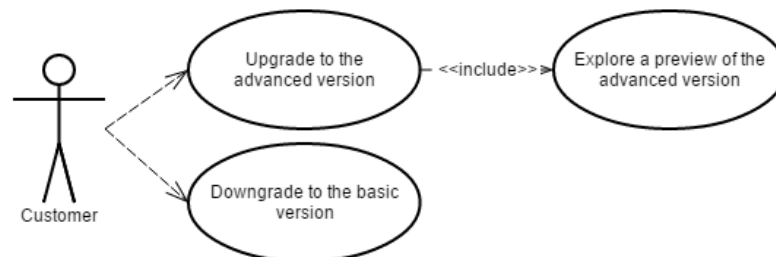


Figure 38 - USE CASE 11.4: Upgrading to the advanced gamified version / downgrading to the basic version of Customer Portal.

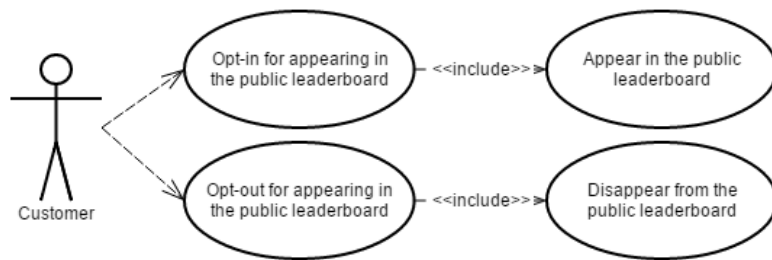


Figure 39 - USE CASE 11.5: Leaderboard opt-in / opt-out.

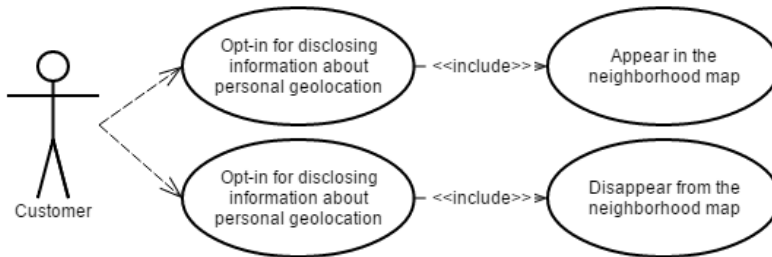


Figure 40 - USE CASE 11.6: Geolocation opt-in / opt-out.

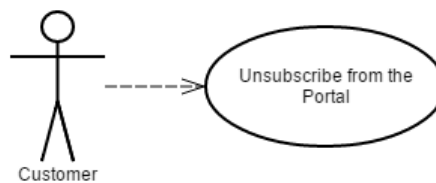


Figure 41 - USE CASE 11.7: Customer Portal Unsubscription.

7.1.6 Signup use cases

This set of use cases describes the signup procedure, for the Basic and the Advanced customer portal.

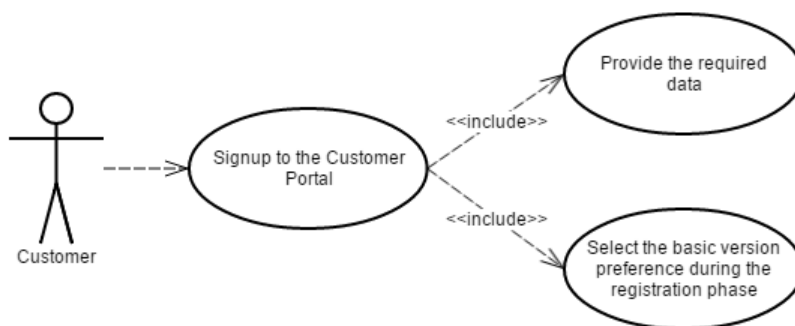


Figure 42 - USE CASE 11.1 Customer Portal Sign-up.

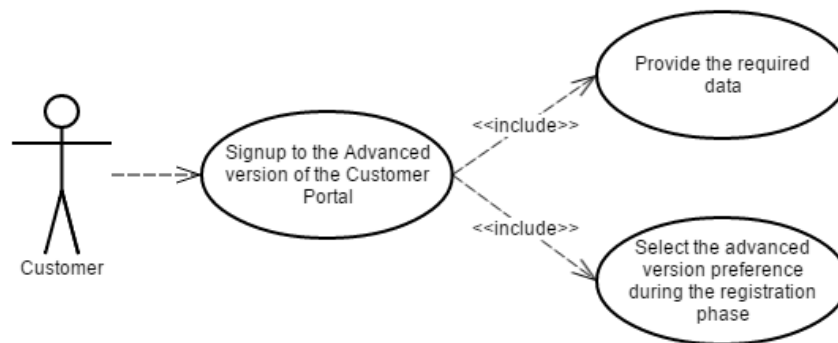


Figure 43 - USE CASE 11.2: Gamification Engine Sign-up.

7.2 Composition and structure

According to the IFML terminology, the Advanced Customer Portal represents a Site View, accessible to all the customers belonging to the group “Competitor User”.

7.2.1 Summary Sections

In this section a description of the main sections of the site view is presented. Figure 44 represents a graphical overview of the Advanced Customer site view.

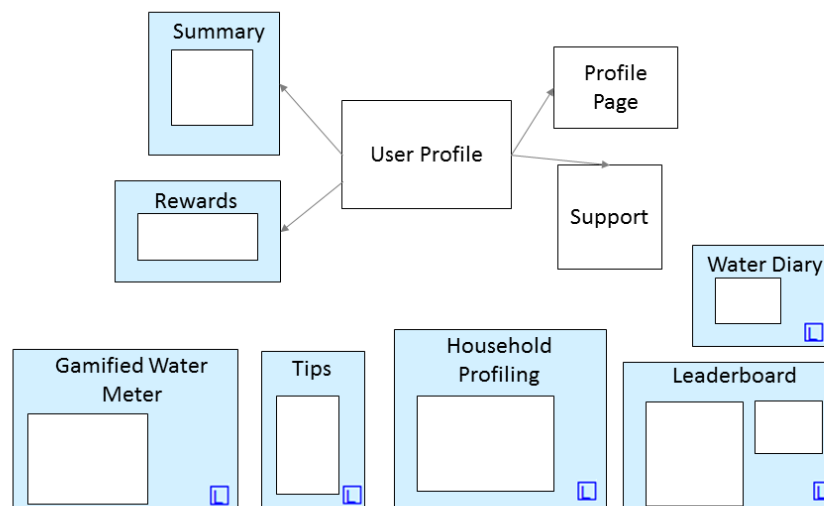


Figure 44 - Advanced Customer Site View – Overview.

Area Summary

Gamified Water Meter

The Gamified Water Meter area is related to consumption. In this area consumption visualization is presented at different levels of detail and abstraction. The Gamified Water Meter area is the default area a user accesses after the login.

Household profile

The Household profile area is meant to collect a subset of psychographic variables as basic profile information about users and their households.

Leaderboard

The Leaderboard area shows the results of the gamified water saving competition.

Rewards

The Rewards area shows the details of the available rewards on the gamified platform.

Summary

The Summary area contains the history of past actions and achievements related to the gamified competition.

Tips

The Tips area is related to learning. In this area consumption tips and educational videos are offered to users, in order to enhance their awareness.

Water Diary

The Water Diary area is related to water end-use events declared by users in a sort of water diary.

Page Summary

ProfilePage

The ProfilePage shows the current user profile details (e.g username, email, profile image) and contains a form to update profile information.

Support

The SupportPage allows the user to notify a problem identified on the portal.

UserProfile

The master page UserProfile contains all the elements which are constant in all the pages of the view. It consists of the top bar elements (username box, logout and profile dropdown, localization dropdown, last reading box and upgrade button) and the right side section containing all the gamification information (performed actions and achievements, available rewards, short leaderboard, recommended actions).

Component Summary

[ModuleInstanceUnit] Change Language

This action contains the operations needed to change the current language of the portal.

[ModuleInstanceUnit] Logout

This action contains the operations needed to perform the logout.

[ModuleInstanceUnit] SendBugEmail

This action contains the operations needed to send a notification email to the administrator, when a user notifies a problem on the portal.

[ModuleInstanceUnit] Update Profile

This action contains the operations needed to update the user profile. The operations include the update on the local database and the invocation of the external service to update user data also in the central database of the SmarH2O system.

[ParameterCollectorUnit] FromBasicVersion

This component represents the access point for users who upgrade from the basic version to the advanced gamified version.

Incoming Flow Summary

→ English	from  UserProfile /  [NoOpContentUnit] Change Language	to  [ModuleInstanceUnit] Change Language
→ Italian	from  UserProfile /  [NoOpContentUnit] Change Language	to  [ModuleInstanceUnit] Change Language
→ Spanish	from  UserProfile /  [NoOpContentUnit] Change Language	to  [ModuleInstanceUnit] Change Language
→ Logout	from  UserProfile /  [NoOpContentUnit] Logout&Profile	to  [ModuleInstanceUnit] Logout
→ SaveButton	from  ProfilePage /  [EntryUnit] Update Profile	to  [ModuleInstanceUnit] Update Profile
→ SendButton	from  Support /  [EntryUnit] Report a bug	to  [ModuleInstanceUnit] SendBugEmail

7.2.2 UserProfile [MasterPage]

In this section a description of the UserProfile master page is presented. Figure 45 represents the structure of the page, in terms of components and links.

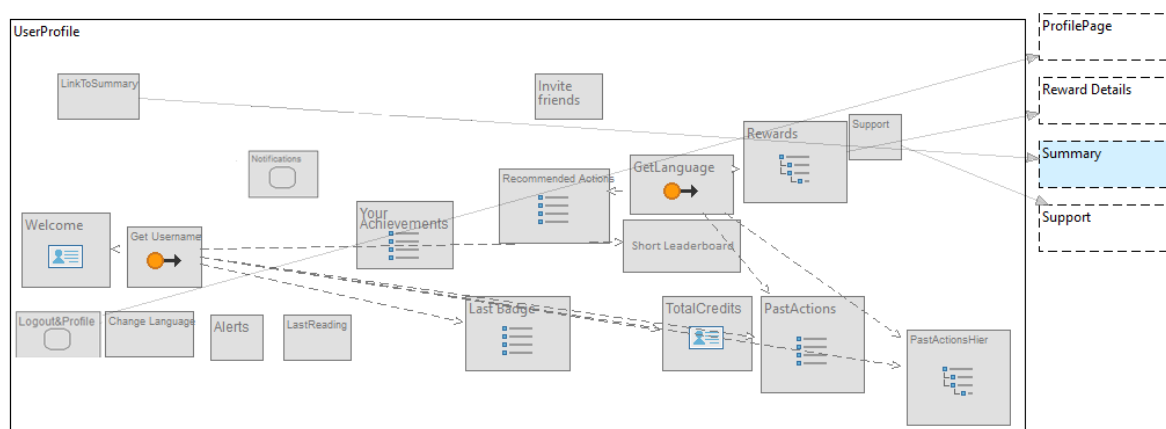


Figure 45 - UserProfile Master Page.

The master page UserProfile contains all the elements which are constant in all the pages of the view. It consists of the top bar elements (username box, logout and profile dropdown, localization dropdown, last reading box and upgrade button) and the right side section containing all the gamification information (past actions and badges, available rewards, short leaderboard, recommended actions).

Components:

-  [NoOpContentUnit] Alerts

This component shows the alerts, e.g. warning the user about possible leaks or bad water quality.

-  [NoOpContentUnit] Notifications

This component shows the notifications, e.g. new user achievements.

This component shows the alerts, e.g. warning the user about possible leaks or bad water quality.

-  [NoOpContentUnit] Change Language

This component represents the localization feature, a dropdown list of all the available languages for the portal, allowing the user to change the current language.

-  [GetUnit] Get Username

This component retrieves the identity of the logged user from the session.

[SelectorUnit] getGlobalID

This component retrieves the global identity (the user identifier in the central database of the SH2O System) of the logged user.

[NoOpContentUnit] LastReading

This component shows the last smart meter reading of the logged user.

[NoOpContentUnit] Logout&Profile

This component represents a dropdown with the links to the user profile and the link to perform the logout.

[NoOpContentUnit] Support

This component represents a link to the Support page, in which the user can notify issues related to the portal.

[DataUnit] Welcome

This component represents a box with logged user details (e.g. username). It receives as input the user identity.

[GetUnit] GetLanguage

This component retrieves the current selected language, stored in the language context parameter.

[NoOpContentUnit] Invite friends

This component manages invitations of contacts from social network to join the portal.

[PowerIndexUnit] Last Badge

This component shows the last badge obtained by the logged user.

[NoOpContentUnit] LinkToSummary

This component shows a link (a button) pointing to the gamification summary page.

[PowerIndexUnit] PastActions

This component shows a list of the actions performed by the users, with some details about actions names, dates and scores.

[PowerIndexUnit] Recommended Actions

This component shows a list of available actions recommended to the user to earn more points on the portal.

[HierarchicalIndexUnit] Rewards

This component shows a list of available rewards, with some details about names and needed credits.

[DataUnit] TotalCredits

This component shows the amount of credits available for the logged user on the gamified portal, considering the positive points collected performing actions and the negative points spent redeeming rewards.

[PowerIndexUnit] Your Achievements

This component shows user progress in the gamified competition in terms of badges. It displays the available badges and the current status of achievements obtained by the user.

7.2.3 Support [Page]

In this section a description of the Support page is presented. Figure 46 represents the structure of the page, in terms of components and links.

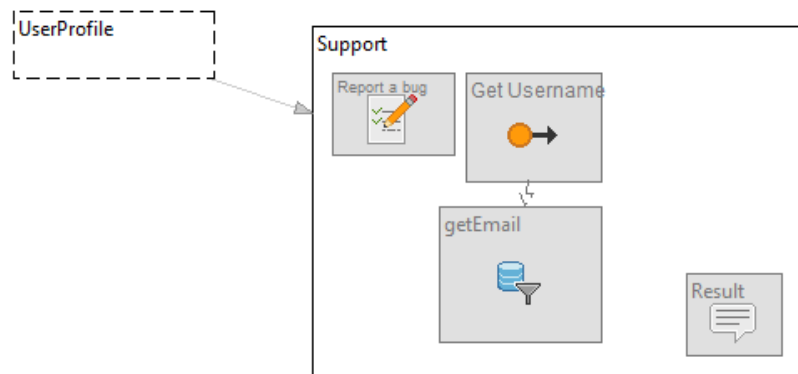


Figure 46 - Support page overview.

The SupportPage allows the user to notify a problem identified on the portal to the administrator and technical staff.

Components:

→ [GetUnit] Get Username

This component retrieves the identity of the logged user from the session.

📧 [SelectorUnit] getEmail

This component retrieves the email address of the user, sent together with the mail message to the administration for future communication between user and customer about the notified problem.

📝 [EntryUnit] Report a bug

This component is a form to allow the user to describe the identified problem. The user can specify the object and the body of the message.

💬 [MultiMessageUnit] Result

A message is shown on the portal after the completion of the SendBugMail action. It can be either an error or a success message.

7.3 Interaction, State dynamics

7.3.1 Gamification actions, badges and rewards

In this section a description of the components in charge of the Gamification mechanisms is presented.

Gamification results are presented to the user in different sections. In the right side bar section the user can find a short presentation of past actions and achievements and browse the available actions, badges and rewards (see UserProfile [MasterPage]). A link to a summary page is provided, where the user can explore the gamified activities in more details; gifts details are available in the Reward Details page, where the user can also redeem a reward, as shown in Figure 47 - Summary and Rewards areas overview.

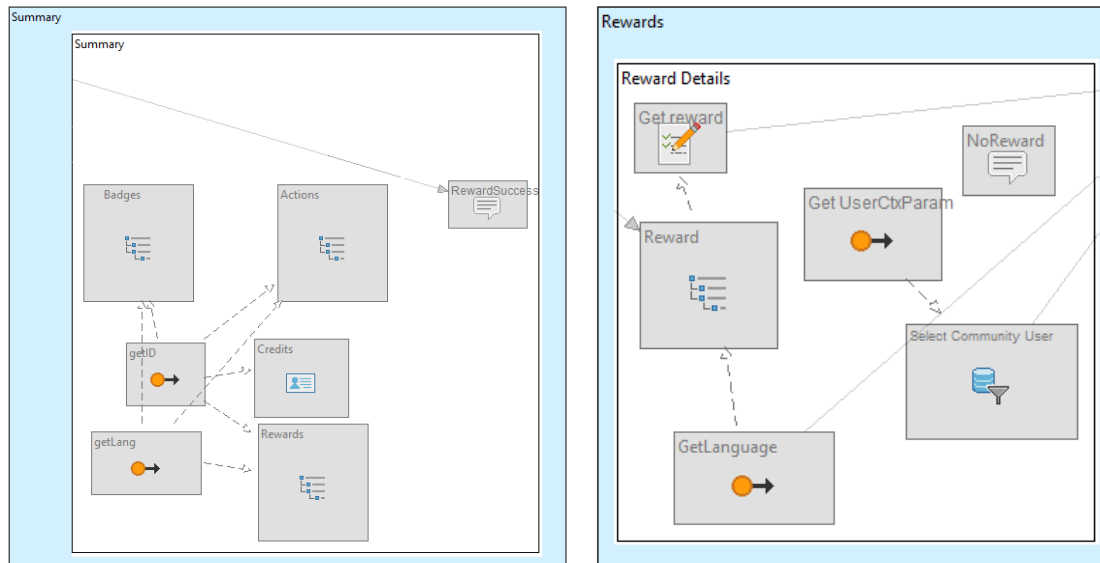


Figure 47 - Summary and Rewards areas overview.

USE CASE 9.1: Making gamification actions and exploring results

Context: Summary [Page]

Components:

[HierarchicalIndexUnit] Actions

This component shows the past actions performed by the user, ordered by date. Actions details provided are name, category, date and related score.

[HierarchicalIndexUnit] Badges

This component shows the past badges obtained by the user, ordered by date. Badges details provided are name, icon and date.

[DataUnit] Credits

This component shows the current status of the gamified user profile: current credits, credits earned during the last week (weekly credits) and credits spent with rewards.

➡ [GetUnit] getID

This component retrieves the identity of the logged user from the session.

➡ [GetUnit] getLang

This component retrieves the current selected language, stored in the language context parameter.

[HierarchicalIndexUnit] Rewards

This component shows the past rewards redeemed by the user, ordered by date. Rewards details provided are name, image, date and shipment details.

[MultiMessageUnit] RewardSuccess

A message is shown on the portal after the completion of the reward redemption action.

Context: Reward Details [Page]

Components:

[EntryUnit] Get reward

This is a form component to input reward shipment information (recipient address and name) and get the reward.

[GetUnit] Get UserCtxParam

This component retrieves the identity of the logged user from the session.

[GetUnit] GetLanguage

This component retrieves the current selected language, stored in the language context parameter.

[HierarchicalIndexUnit] Reward

This components shows the reward details: name, description, image, needed points.

[SelectorUnit] Select Community User

This component retrieves the available credits of the logged user, to check if the reward is redeemable

( isRedeemable ConditionExpression)

USE CASE 9.9: Making actions and earning digital credits with the Games Platform

Context: Summary [Page]

Components:

[HierarchicalIndexUnit] Actions

Actions performed on the Games Platform can be notified to the Advanced Customer Portal. They can be considered standard gamification actions, visible in the action history and contributing to the total credits of the competitor customer .

7.3.2 Gamified water meter use cases

In this section a description of the components in charge of the gamification mechanisms related to consumption are presented: in the advanced version the user can declare water consumption events, perform a recommended saving action and fulfill some consumption goals.

USE CASE 9.7: Declaring water end-use events

In the Water Diary area the user can view the personal water diary and enter a new water use event, as shown in Figure 48 - Water Diary area overview.

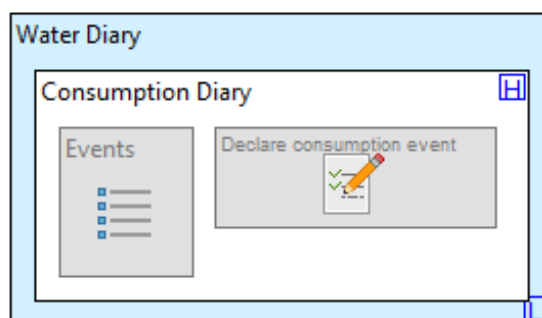


Figure 48 - Water Diary area overview.

Context: Consumption Diary [Page]

Components:

[EntryUnit] Declare consumption event

This is a form component to input a new water use event.

☰ [PowerIndexUnit] Events

This is a list showing all water use events, ordered by date as for a real water diary.

USE CASE 9.4: Recommending water saving actions

Context: UserProfile [Master Page]

Components:

☰ [PowerIndexUnit] Recommended Actions

This component shows a list of available actions recommended to the user to earn more points on the portal.

USE CASE 9.5: Declaring water saving actions

Context: UserProfile [Master Page]

Components:

☰ [PowerIndexUnit] Recommended Actions

The user has the possibility to report that he implemented one of the recommended water saving actions, e.g. installing more efficient appliances, or one of the available pre-defined actions. For some actions the system could also verify and validate that an action has been performed before assigning points to the user.

USE CASE 9.2: Self setting consumption goals

In the Advanced customer portal the same consumption visualization as in the Basic Portal is adopted. In addition some gamification tools are available for competitor customers. Users can self set a consumption goal, which will be evaluated according to her possible consumption reduction.

Goals will be displayed in comparison with the current consumption average of the user, in the Consumption page. Figure 49 represents the structure of the page, in terms of components and links.

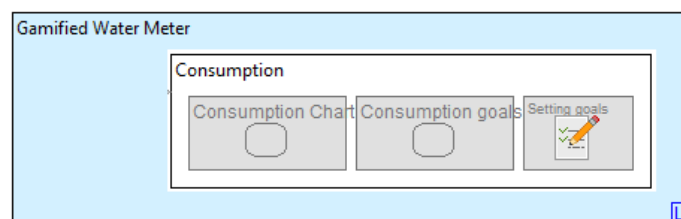


Figure 49 - Consumption area overview.

Context: Consumption [Page]

Components:

📝 [EntryUnit] Setting goals

This is a form component to allow the user to set a consumption goal.

USE CASE 9.3: Fulfilling consumption goals

Periodically the system validate consumption goals with respect to customer averages and assign credits to users who achieve a consumption goal reducing consumption.

Context: Consumption page

Components:

☐ [NoOpContentUnit] Consumption Chart

This component shows the user personal consumption. It represents the Detailed consumption, showing a bar chart with consumption data.

☐ [NoOpContentUnit] Consumption goals

This component shows an Overview of the user average consumption, compared with consumption goals.

7.3.3 User profiling use cases

In the advanced portal users can gain points by providing some personal and household information.

In this section a description of the Household profile area is presented, highlighting the relation between the components and the use cases. Figure 50 represents the structure of the area, in terms of pages and components.

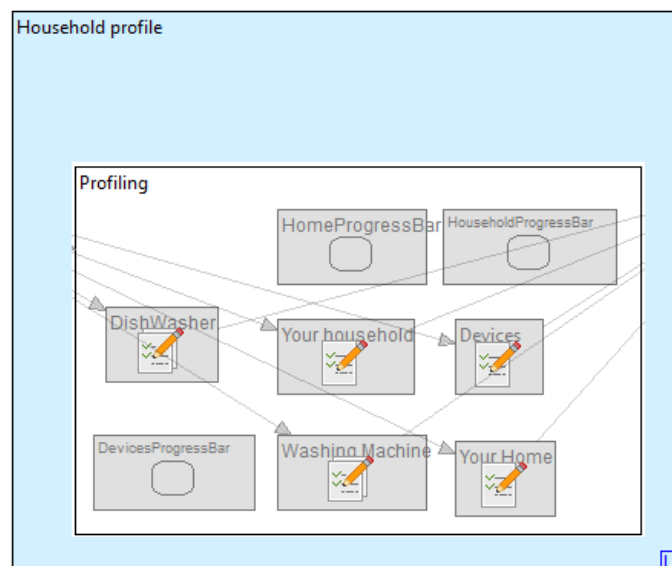


Figure 50 - Household profile area overview.

USE CASE 9.6: Contributing household and user profiling information

Context: Profiling [Page]

Components:

☒ [EntryUnit] Devices

This is a form component to input information about household devices.

☐ [NoOpContentUnit] DevicesProgressBar

This component represents a progress bar, showing the percentage of completion of the appliances profile.

☒ [MultiEntryUnit] DishWasher

This is a form component to input information about dishwasher devices in the household.

☐ [NoOpContentUnit] HomeProgressBar

This component represents a progress bar, showing the percentage of completion of the home profile.

☐ [NoOpContentUnit] HouseholdProgressBar

This component represents a progress bar, showing the percentage of completion of the household profile.

📝 [MultiEntryUnit] Washing Machine

This is a form component to input information about washing machine devices in the household.

📝 [EntryUnit] Your Home

This is a form component to input information about customer home.

📝 [EntryUnit] Your household

This is a form component to input information about customer household.

7.3.4 Social use cases

On the advanced portal users are required to perform also some social actions: compete with other users, create teams, set common goals, invite friends or post content on social networks.

In this section a description of the Leadboard area is presented. Figure 51 represents the structure of the area, in terms of pages and components.

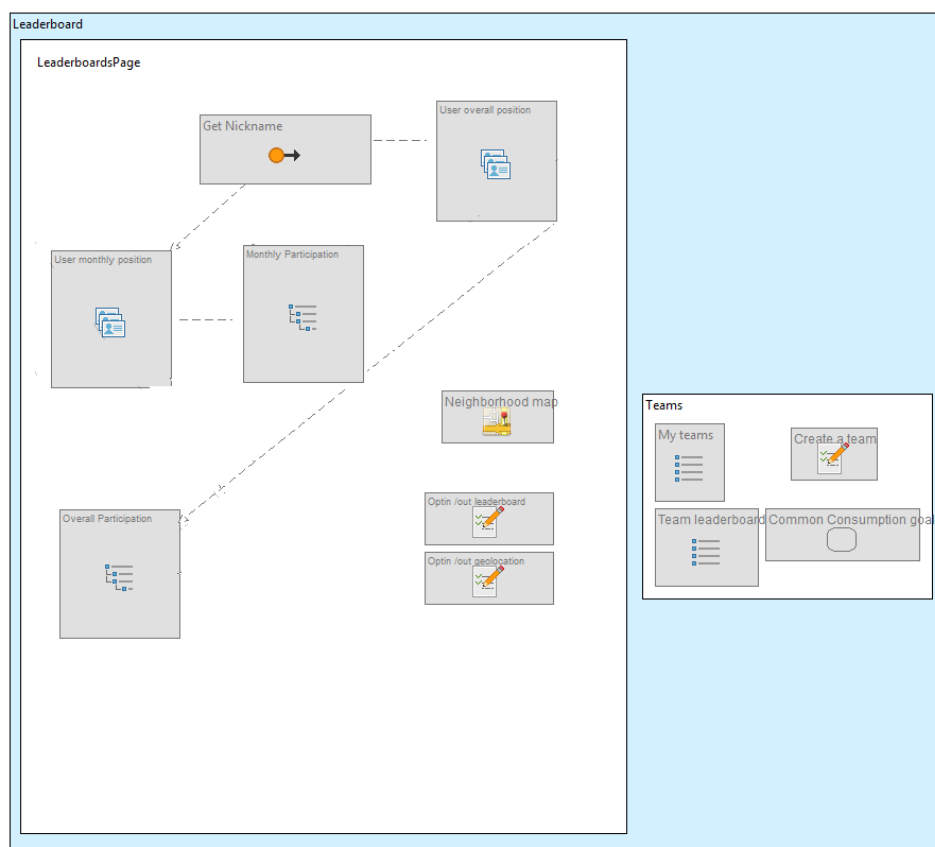


Figure 51 - Leaderboard area overview.

Page Summary

📄 LeaderboardsPage

The Leaderboard page shows to the user her ranking position in the leaderboard of the gamified water saving competition.

Teams

The Teams page allows the customer to form teams with other users to benefit from each others' water saving.

USE CASE 10.1: Comparing achievements with other households

Context: Leaderboard [Page]

Components:

 [HierarchicalIndexUnit] Monthly Participation

This component represents the last seven days leaderboard. The leaderboard is centered according to the user ranking position, received as input.

 [GoogleMapUnit] Neighborhood map

This component plots users belonging to the same district of the current customer on a map.

 [HierarchicalIndexUnit] Overall Participation

This component represents the overall leaderboard. The leaderboard is centered according to the user ranking position, received as input.

 [MultiDataUnit] User monthly position

This component provides the ranking position of the user considering the last seven days.

 [MultiDataUnit] User overall position

This component provides the overall ranking position of the user.

USE CASE 10.2: Inviting another user to join a team

The Advanced Portal employs social cooperation with collaborative tasks: users can form teams to reach a goal together and compete on a teamlevel.

Context: Teams [Page]

Components:

 [PowerIndexUnit] My teams

This component shows a list of the teams of the user.

 [EntryUnit] Create a team

This is a form component to create a new team, selecting members and inserting team information (e.g. name).

 [PowerIndexUnit] Team leaderboard

This component represents the leaderboard for the teams competition.

USE CASE 10.3: Achieving goals collaboratively as a team

Context: Teams [Page]

Components:

 [NoOpContentUnit] Common Consumption goals

This component shows common consumption goals, considering the aggregated consumption of all the members of the team.

7.3.5 User account management use cases

In this section a description of the components in charge of managing the user account is presented, highlighting the relation with the use cases.

USE CASE 11.3: Modifying User Settings

Figure 52 represents the structure of the Profile page, in terms of components and links.

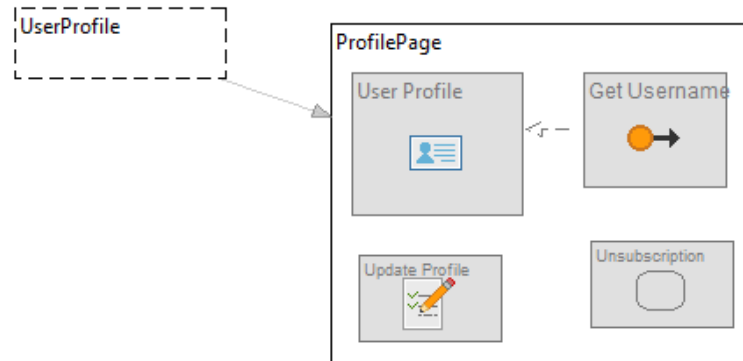


Figure 52 - Profile Page overview.

The ProfilePage shows the current user profile details (e.g username, email, profile image) and contains a form to update profile information.

Context: ProfilePage [Page]

Components:

➡ [GetUnit] Get Username

This component retrieves the identity of the logged user from the session.

📝 [EntryUnit] Update Profile

This component is a form to update profile information.

👤 [DataUnit] User Profile

This component shows the current profile information provided by the user (e.g. username, email, profile image). It receives as input the user identity.

USE CASE 11.4: Upgrading to the advanced gamified version / downgrading to the basic version of Customer Portal

It is possible for the user to upgrade from the basic version to the advanced version. Section Upgrade [Page] describes in details the flow from the Basic Portal prospective.

Context: Advanced Customer [Site View]

Components:

🔗 [ParameterCollectorUnit] FromBasicVersion

This component represents the access point when a user upgrades from the basic to the advanced version.

USE CASE 11.5: Leaderboard opt-in / opt-out

Context: Leaderboard [Page]

Component:

📝 [EntryUnit] Optin /out leaderboard

This is a form component to accept to appear in the public leaderboard.

USE CASE 11.6: Geolocation opt-in / opt-out

Context: Leaderboard [Page]

Component:

 [EntryUnit] Optin /out geolocation

This is a form component to accept to appear on the neighbourhood map.

USE CASE 11.7: Customer Portal Unsubscription

Context: ProfilePage [Page]

Component:

 [NoOpContentUnit] Unsubscription

This is a form component to perform the unsubscription from the Customer Portal.

7.3.6 Signup use cases

In this section a description of the public Homepage area is provided. This area is accessible by all the users and contains the Home page. Figure 53 represents the structure of the area, in terms of pages and components.

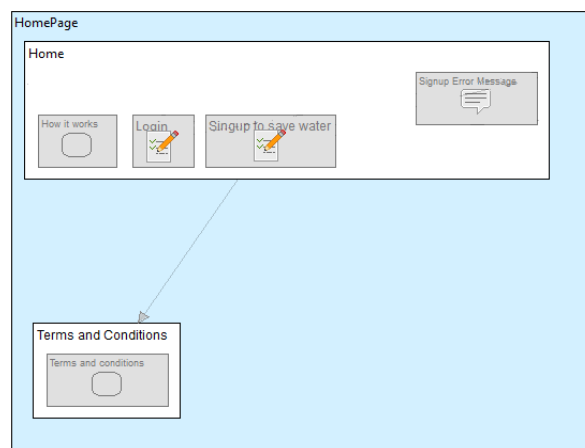


Figure 53 - HomePage Site View overview.

USE CASE 11.1 / 11.2: Customer Portal Sign-up / Gamification Engine Sign-up

Context: Home [Page]

Components:

 [NoOpContentUnit] How it works

This is an introductory textual explanation of the portal mechanics.

 [EntryUnit] Singup to save water

This is a form component to input all the information required by the signup procedure.

 [MultiMessageUnit] Signup Error Message

This is a message shown to the user when the signup procedure fails (e.g. the username is already assigned to

another user)

[EntryUnit] Login

This is a form component to perform the login.

Two different options are available for the customers: signup to the basic or to the advanced portal. Two different links represent these two options, as reported in Table 6 - Flows summary table.

Table 6 - Flows summary table.

→ Join Advanced Portal	from  [EntryUnit] Singup to save water	to  HomePage /  [ModuleInstanceUnit] Signup
→ Join Basic Portal	from  [EntryUnit] Singup to save water	to  HomePage /  [ModuleInstanceUnit] Signup
→ Sign In	from  [EntryUnit] Login	to  HomePage /  [LoginUnit] Login
→ Terms	from  [EntryUnit] Singup to save water	to  Terms and Conditions

8. Customer Portal Admin

8.1 Context

This set of use cases describes the interactions of the administrator users with the administration area of the portal, including content editing and gamification setting.

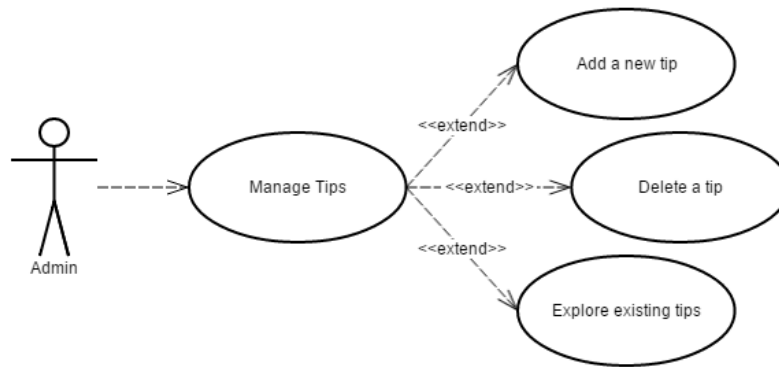


Figure 54 - USE CASE 12.1: Setting water consumption tips.

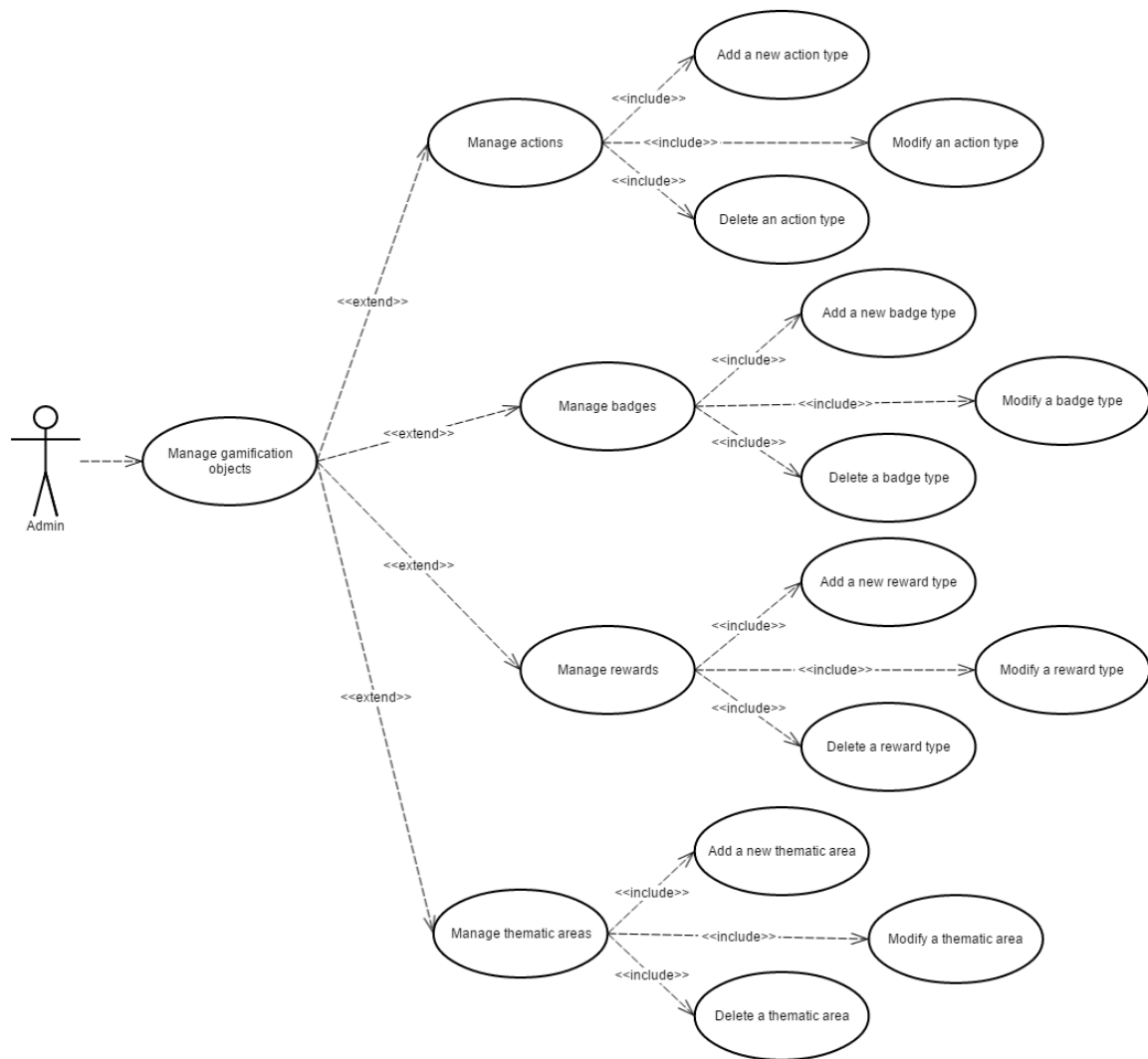


Figure 55 - USE CASE 12.2: Setting actions, badges and rewards.

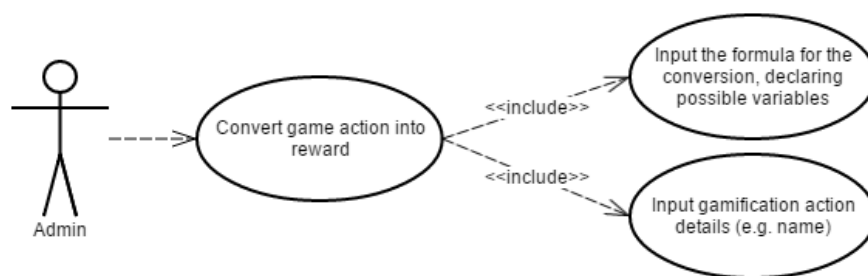


Figure 56 - USE CASE 12.3: Converting game actions into rewards.

8.2 Composition and structure

According to the IFML terminology, the Admin Customer Portal represents a Site View, accessible to all the customers belonging to the group “Admin”.

8.2.1 Summary Sections

In this section a description of the main sections of the site view is presented. Figure 57 represents a graphical overview of the Administration site view.

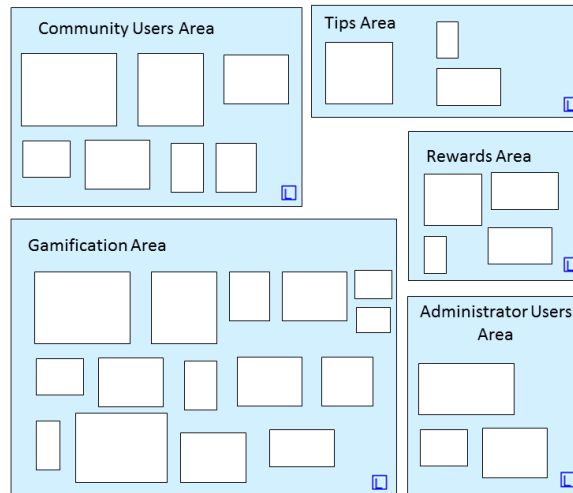


Figure 57 - Administration Site View – Overview.

Area Summary

Administrator Users Area

This area is related to the management of administrator users. In this area it is possible to explore admin users information and add new administrators.

Community Users Area

This area is related to the management of customers. In this area it is possible to explore customer users information, check their activities on the portal and monitor their consumption.

Gamification Area

In this area it is possible to manage gamification objects (thematic areas, actions, badges).

Rewards Area

This area is related to the management of rewards. It is possible to create/modify/delete rewards, monitor the status of acquired rewards and manage shipment details.

Rewards Area

This area is related to the management of rewards. It is possible to create/modify/delete rewards, monitor the status of acquired rewards and manage shipment details.

Tips Area

This area is related to the management of tips. Here it is possible to create/modify/delete tips.

Page Summary

UserProfile

The master page UserProfile contains all the elements which are constant in all the pages of the view. It consists of the top bar elements (username box, logout and localization dropdown).

Component Summary

[ModuleInstanceUnit] Change Language

This action contains the operations needed to change the current language of the portal.

[ModuleInstanceUnit] Logout

This action contains the operations needed to perform the logout.

Incoming Flow Summary

→ English	from  UserProfile /  [NoOpContentUnit] Change Language	to  [ModuleInstanceUnit] Change Language
→ Italian	from  UserProfile /  [NoOpContentUnit] Change Language	to  [ModuleInstanceUnit] Change Language
→ Spanish	from  UserProfile /  [NoOpContentUnit] Change Language	to  [ModuleInstanceUnit] Change Language
→ Logout	from  UserProfile /  [NoOpContentUnit] Logout&Profile	to  [ModuleInstanceUnit] Logout

8.2.2 Administrator Users Area [Area]

In this section a description of the Administrator Users Area is presented. Figure 58 represents the structure of the area, in terms of pages, links and components.

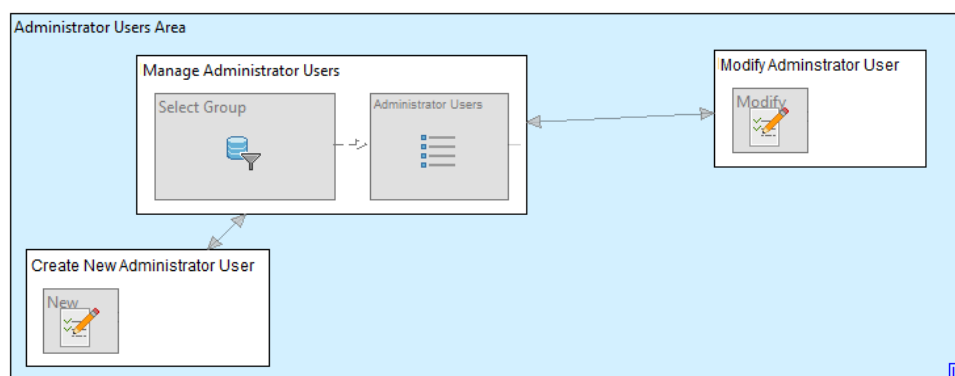


Figure 58 - Administrator Users Area – Overview.

Page Summary

 Create New Administrator User

In this page it is possible to add new administrator users.

Components:

 [EntryUnit] New

This is a form component to add a new admin user.

 Manage Administrator Users

In this page a list of admin users is provided and it is possible to explore some information about them (e.g. username, email).

Components:

 [PowerIndexUnit] Administrator Users

This component shows the list of administrator users.

 [SelectorUnit] Select Group

This component filters users selecting the ones belonging to the “Admin” group.

📄 Modify Administrator User

In this page it is possible to modify an existing user.

Components:

📄 [EntryUnit] Modify

This is a form component to modify an admin user.

8.2.3 Community Users Area [Area]

In this section a description of the Administrator Users Area is presented. Figure 59 represents the structure of the area, in terms of pages, links and components.

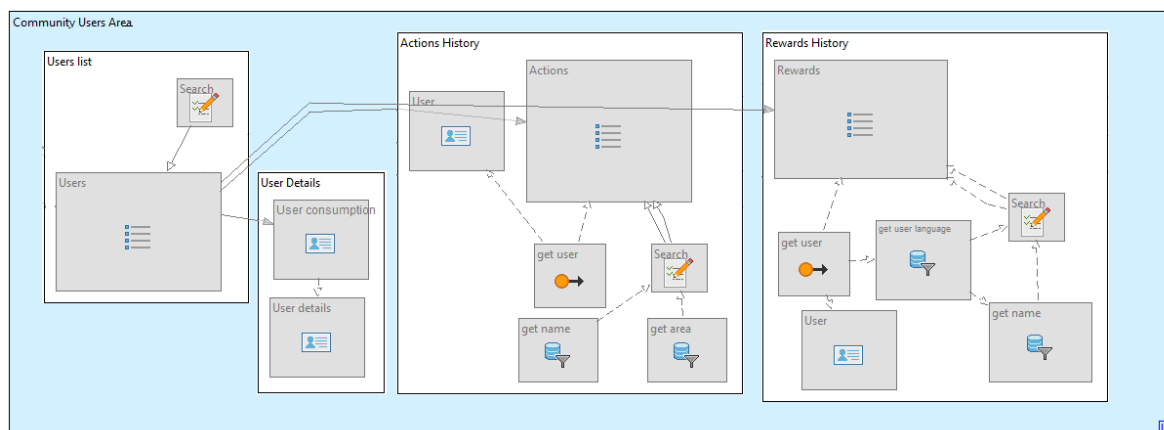


Figure 59 - Community Users Area – Overview.

Page Summary

📄 Actions History

In this page the list of the actions performed by the selected user is provided.

Components:

📄 [PowerIndexUnit] Actions

This component shows the list of past actions

📄 [SelectorUnit] get area

This component retrieves the thematic areas by which actions are grouped, to populate the selection filter in the form component.

📄 [SelectorUnit] get name

This component retrieves the actions name, to populate the selection filter in the form component.

➡ [GetUnit] get user

This component retrieves the identity of the user.

📄 [EntryUnit] Search

This is a form component to select the action name or area by which to filter the list of actions.

📄 [DataUnit] User

This component shows some details about the selected user.

Rewards History

In this page the list of the rewards obtained by the selected user is provided.

Components:

[SelectorUnit] get name

This component retrieves the rewards name, to populate the selection filter in the form component.

[GetUnit] get user

This component retrieves the identity of the user.

[SelectorUnit] get user language

This component retrieves language of the user to localize rewards name.

[PowerIndexUnit] Rewards

This component shows the list of rewards obtained by the selected user

[EntryUnit] Search

This is a form component to select the reward name by which to filter the list of rewards.

[DataUnit] User

This component shows some details about the selected user.

User Details

In this page some details about the selected user are provided.

Components:

[DataUnit] User consumption

This component allows the admin to monitor user consumption, plotting consumption data.

[DataUnit] User details

This component shows detailed information about the selected user (e.g. credits spent, total credits).

Users list

In this page a list of the community users is provided and it is possible to explore some information about them (e.g. username, email, credits collected).

Components:

[EntryUnit] Search

This is a form component to filter users by username.

[PowerIndexUnit] Users

This component shows the list of community users.

8.2.4 Gamification Area [Area]

In this section a description of the Gamification Area is presented. Figure 60 represents the structure

of the area, in terms of pages and links.

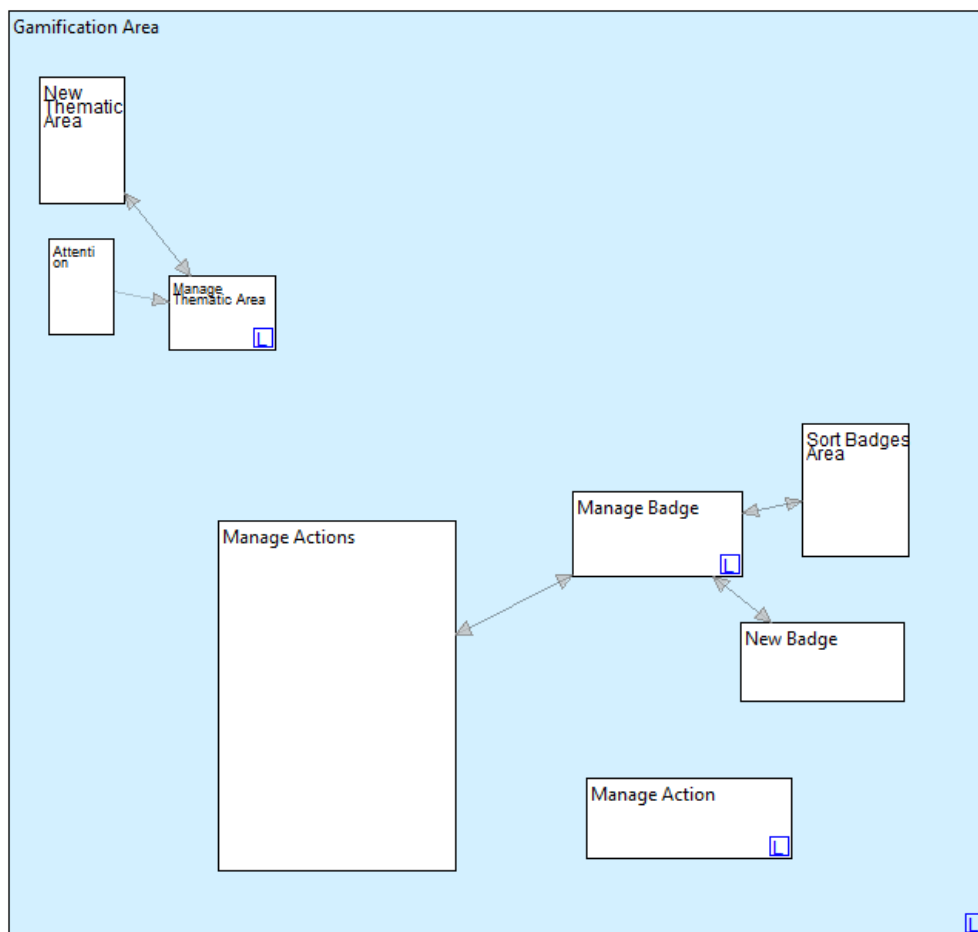


Figure 60 - Gamification Area – Overview.

Page Summary

Manage Action

In this page it is possible to modify an existing action.

Manage Actions

In this page it is possible to explore existing actions and select an action to be updated/deleted.

Manage Badge

In this page it is possible to explore existing badges and select a badge to update/delete it.

Manage Thematic Area

In this page it is possible to explore existing thematic areas and select an area to be updated/deleted.

New Badge

In this page it is possible to add new badge types.

New Thematic Area

In this page it is possible to add new thematic areas.

Sort Badges Area

In this page it is possible organize badges by areas.

8.2.5 Rewards Area [Area]

In this section a description of the Rewards Area is presented. Figure 61 represents the structure of the area, in terms of pages, components and links.

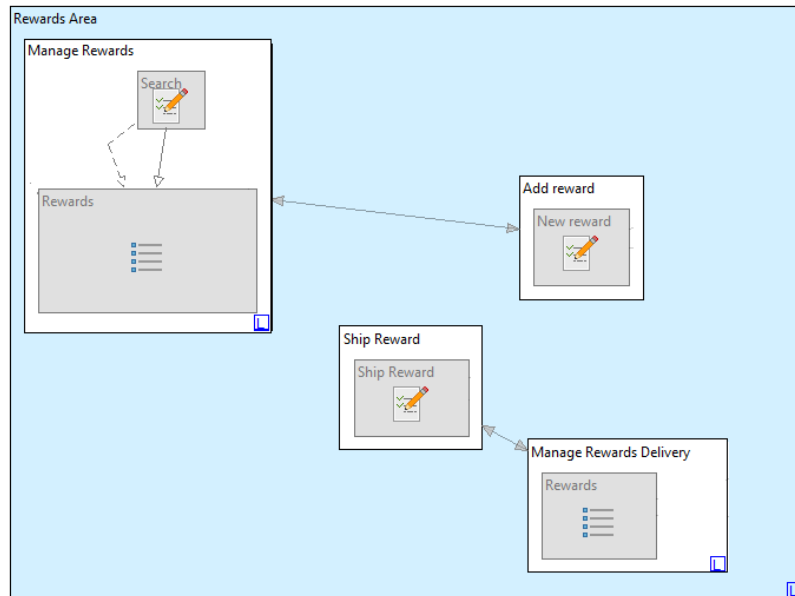


Figure 61 - Rewards Area – Overview.

Page Summary

📄 Add reward

In this page it is possible to add new reward types.

📄 Manage Rewards

In this page it is possible to explore existing rewards and select a reward to be updated/deleted.

📄 Manage Rewards Delivery

In this page it is possible to monitor the status of acquired rewards.

📄 Ship Reward

In this page it is possible to confirm the shipment of a reward.

8.3 Interaction, State dynamics

8.3.1 Use cases

USE CASE 12.1: Setting water consumption tips

Water saving tips can be managed by the administrators in the Tip Area. Figure 62 represents the structure of the area, in terms of pages, components and links.

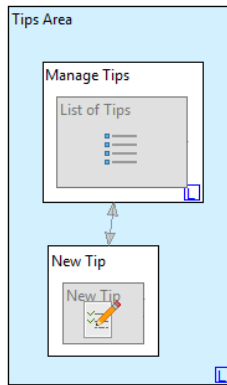


Figure 62 - Tips Area – Overview.

Context: Tips [Area]

Pages:

 New Tip

In this page it is possible to add a new tip.

 Manage Tips

In this page it is possible to explore existing tips and select a tip to be updated/deleted.

Components:

 [EntryUnit] New Tip

This is a form component to input title and body of a new tip.

 [PowerIndexUnit] List of Tips

This component shows the list of available tips.

USE CASE 12.2: Setting actions, badges and rewards

Context: New Action [Page]

Components:

 [EntryUnit] New Action

This is a form component to input a new action type.

 [SelectorUnit] Select Area

This component retrieves all the thematic areas by which actions can be grouped.

Context: New Badge [Page]

Components:

 [EntryUnit] New Badge

This is a form component to input a new badge type.

 [SelectorUnit] Select Area

This component retrieves all the thematic areas by which badges can be grouped.

Context: New Thematic Area [Page]

Components:

 [EntryUnit] Thematic Area

This is a form component to input a new thematic area.

Context: Add Reward [Page]

Components:

 [EntryUnit] New reward

This is a form component to input a new reward type.

USE CASE 12.3: *Converting game actions into rewards*

An admin user can convert games platform achievements into gamification engine actions. Figure 63 represents the structure of the page.

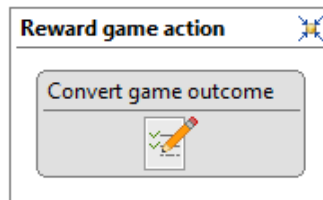


Figure 63 - Reward game action Page.

Context: Reward game action [Page]

Components:

 [EntryUnit] Convert game action

This is a form component to allow the user to manually input formulas to convert inputs (game achievements and points) into outputs (gamification actions).

9. Social Connector

9.1 Context

This set of use cases describes the interactions of the customer with external social networks, in the Customer Portal context.

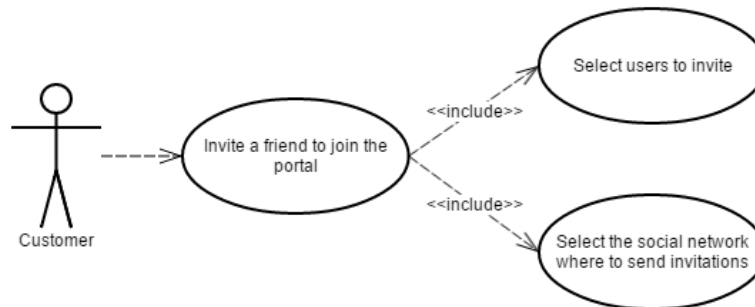


Figure 64 - USE CASE 10.4: Inviting friends on social networks.

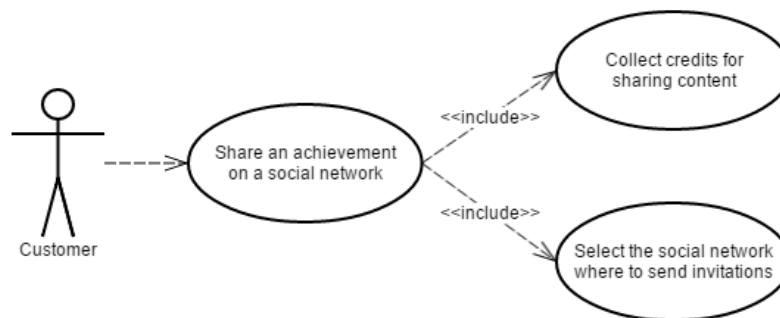


Figure 65 - USE CASE 10.5: Sharing achievements on social networks.

9.2 Composition and structure

According to the IFML terminology, the Social Connector is composed of two modules, in charge of inviting friends and posting content on social networks.

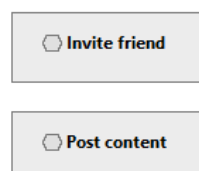


Figure 66 - Social Connector Modules.

9.2.1 Invite a friend [Module]

In this section a description of the “Invite a friend” module is provided. Figure 67 represents the structure of the module, in terms of components and execution flows.

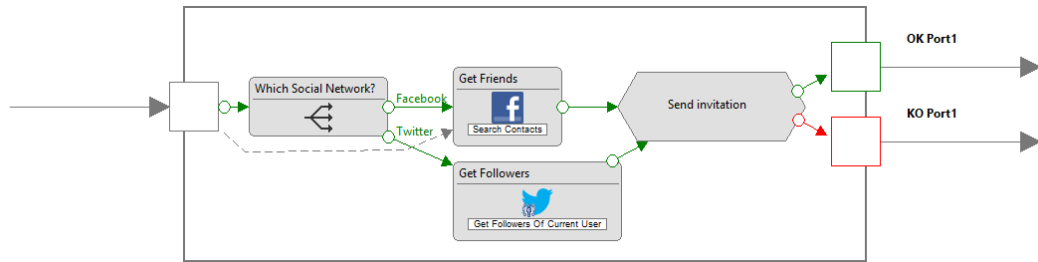


Figure 67 - Invite a friend module – Overview.

Module Components:

 [FacebookUnit] Get Friends

This component is in charge of selecting Facebook contacts.

 [TwitterUnit] Get Followers

This component is in charge of retrieving Twitter followers.

 [ModuleInstanceUnit] Send invitation

This component is in charge of sending invitation on the selected social network.

Figure 68 represents an example of invocation of the module, in the Customer Portal context.

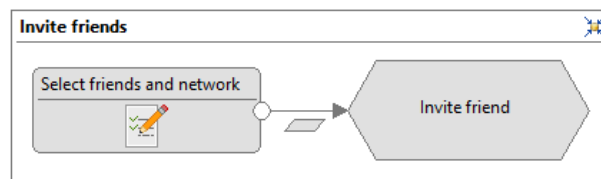


Figure 68 - Invite a friends on social network feature.

9.2.2 Post content [Module]

In this section a description of the “Post content” module is provided. Figure 69 represents the structure of the module, in terms of components and execution flows.

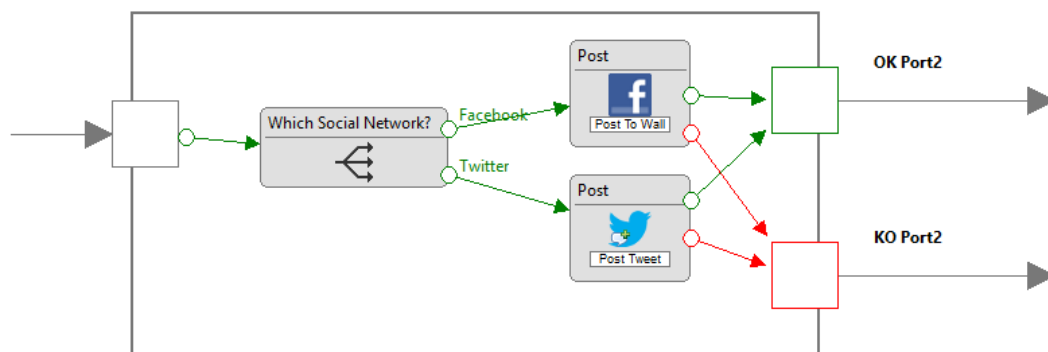


Figure 69 – Post content.

Module Components:

[FacebookUnit] Post

This component is in charge of sharing a multimedia content on Facebook.

[TwitterUnit] Post

This component is in charge of posting a multimedia content on Twitter.

Figure 70 represents an example of invocation of the module, in the Customer Portal context.

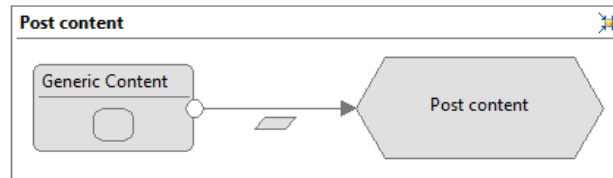


Figure 70 - Example of a posting event.

10. Gamification Engine

10.1 Composition and structure

The GE is composed by a list of components, web services and a Database.

- Configuration modules: set of modules to be used to configure the Gamification Engine by adding/deleting/editing Gamification objects (actions, badges, rewards)
- Gamification Logic modules: set of modules that implements the Gamification logic, i.e. how to assign a consumption goal etc. This component is described in section 10.4.
- Notification modules: set of modules used to notify (via mail) customers and admins.
- GE services: a list of REST services to manage Gamification objects form external application. The services description is specified in section 10.5
- GE external Authentication: REST services used to perform authentication from an external application
- GE DB: database for gamification data (users, actions performed etc), specified in section 10.2

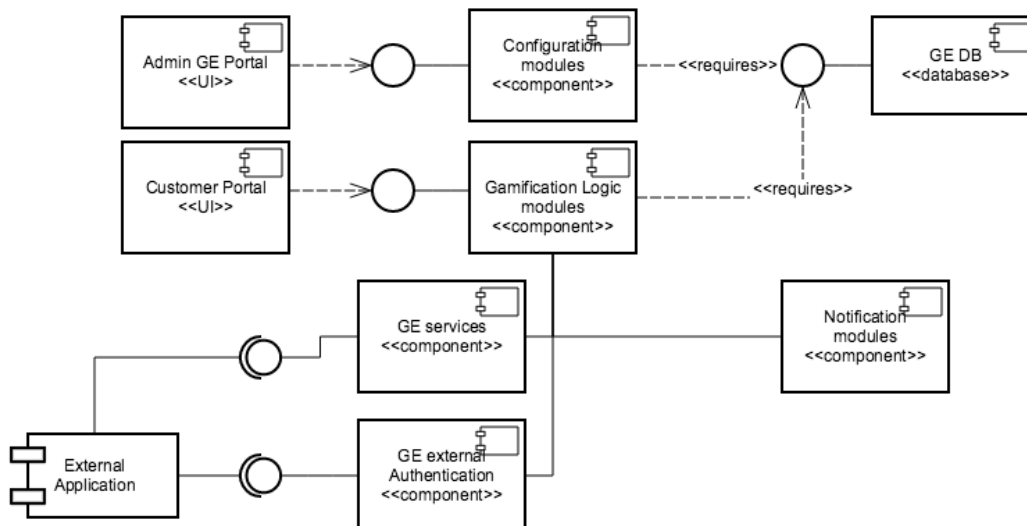


Figure 71 - Gamification engine structure.

10.2 Logical

In the following an overview of the gamification data is presented. The class diagram has been detailed in D3.2.

The **Gamification Engine Class Diagram** comprises the set of entities and relationships that express knowledge about user data produced and consumed by the Advanced Gamified Customer Portal. The following entities have been considered:

Community Users: the entity is a specialization of **User** and contains all the attributes that identify the user as a member of a community (like credits, bio information, ...).

Gamified Application: this table contains information about applications that call the gamification engine.

Action Type: the entity contains the dictionary of the actions of the gamification engine. The attribute values of an action are the specific features of the considered action.

Action Instance: the entity stores all the action instances performed by a user.

Badge Type: the entity contains the dictionary of the badges that a user can acquire.

Badge Instance: the entity contains all the badge instances acquired by the user.

Reward Type: the entity contains the dictionary of the rewards.

Reward Instance: the entity contains the instances of the rewards acquired by the users.

Text Mail: the entity contains information about the notification to send to users after a particular event in the gamification engine (e.g. a user gains a badge).

Notification: this entity contains the notification sent to users.

Thematic Area: this entity contains the thematic areas to organize actions and badges according to topics. Each thematic area is identified by a unique id and a name.

Game Result: this entity contains the possible outcomes of games, that need to be converted into credits. Each game result is identified by a unique identifier, a title (e.g. New level reached) and optionally by a score, a level and the current available lives. Each game result is mapped to an Action Type and, according to the game results attributes (score, level, lives) the game result is converted into credits.

Game Points Converter: each conversion is identified by a unique id, the game to which the conversion rule is applied, and the customizable formula which will take the attributes as inputs (score, level, lives) and will produce credits amount as output.

Alliance: this entity contains the coalitions created among competitor users. Each Alliance is identified by a unique id, a start date and an end date.

Goal: this entity contains the consumption goals assigned to users. Each goal is identified by a unique id, a title, a consumption value, and optionally the completion date. A goal can be assigned to a given user or to an alliance of users. Goal can be associated to a Badge Type, obtained by the user when the consumption goal is achieved.

10.3 Interaction & Dependencies

10.3.1 Registration

In Figure 73 the registration process is presented. The GE interacts with the ESB to save a new user and retrieve her baseline consumption.

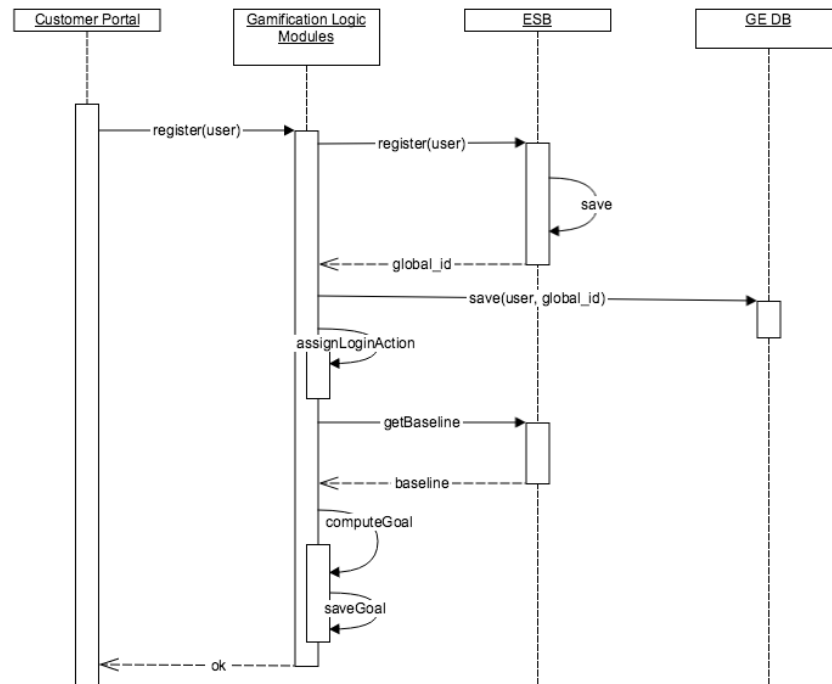


Figure 73 - User registration.

10.3.2 Visualize Consumption Overview

To build the consumption page the GE interacts with the ESB to retrieve consumption data associated with the customer and her neighbourhood.

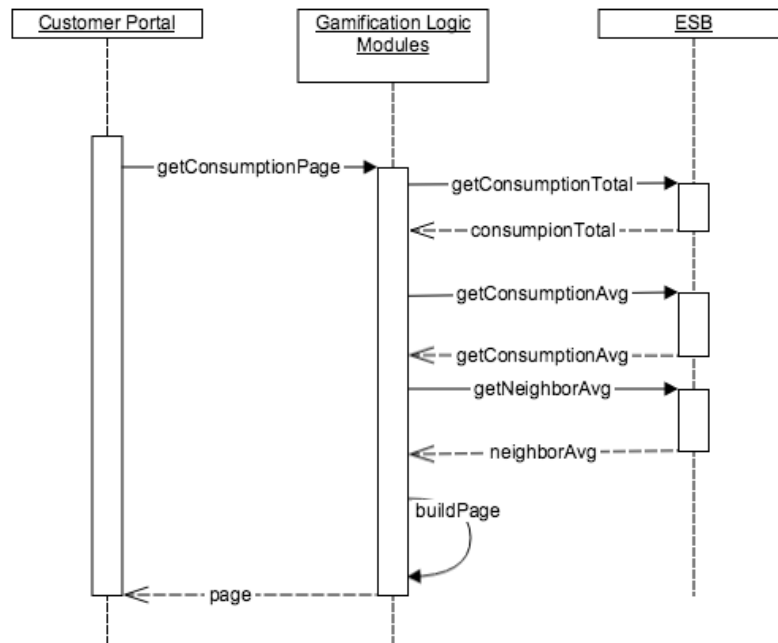


Figure 74 - Consumption visualization.

10.3.3 External Authentication

The following diagram specifies the interactions between the GE and an external application to perform the authentication of an external customer. The external application has to be authenticated using a ticket, only after the component authentication it can ask to authenticate a user. Once authenticate the customer will be redirect to the Customer Portal.

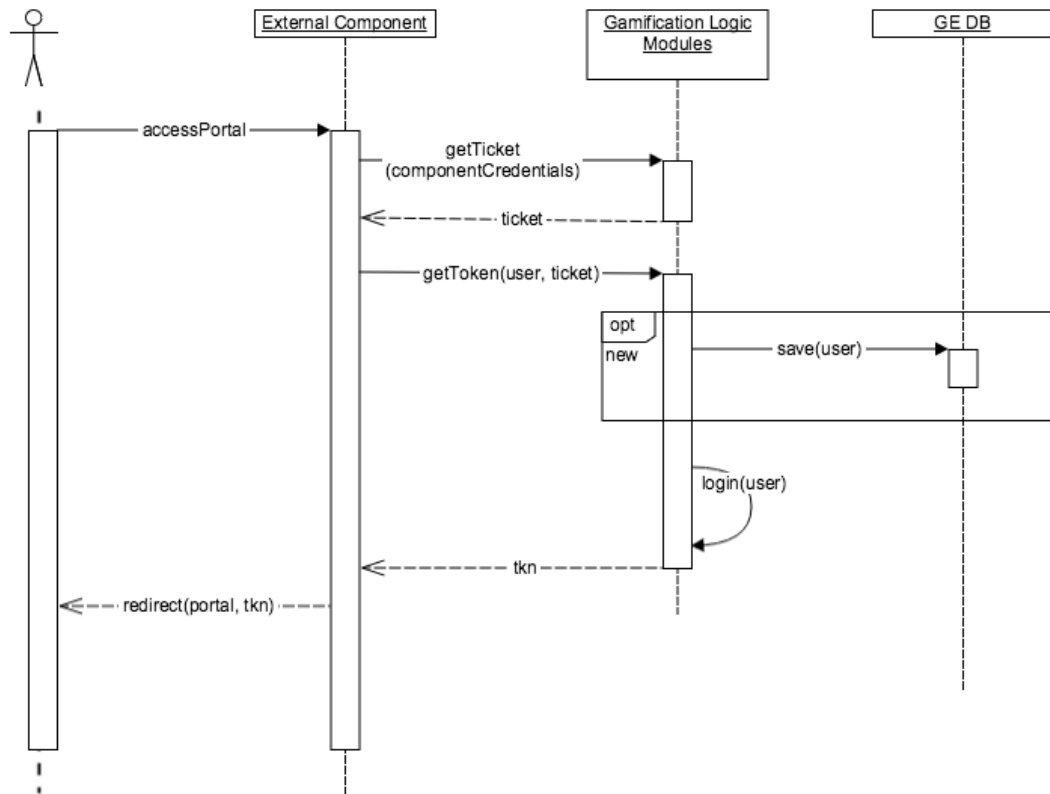


Figure 75 - External authentication.

10.4 State Dynamics

10.4.1 Assign an action to a user

The activity diagram of Figure 76 describes the steps performed by the GE when it has to assign an action to a user.

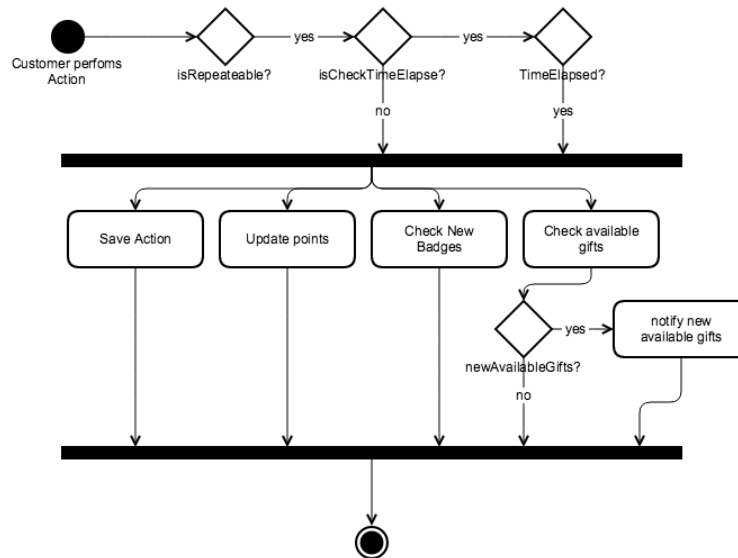


Figure 76 - Assign actions to an user.

10.4.2 Redeem reward

The activity diagram of Figure 77 describes the steps performed by the GE when a user asks to redeem a reward.

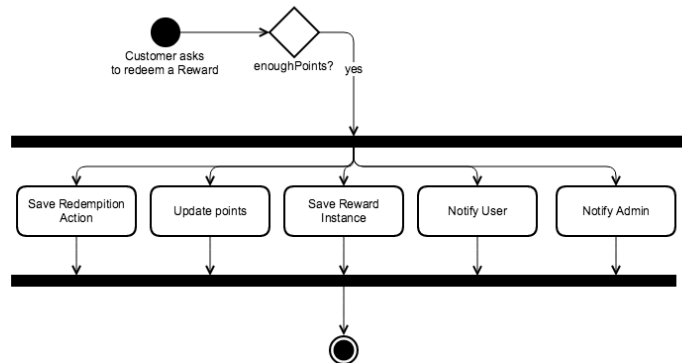


Figure 77 - Redeem reward.

10.4.3 Assign a new Badge

Every time a new action is assigned to a customer the GE checks if the customer achieved a new badge.

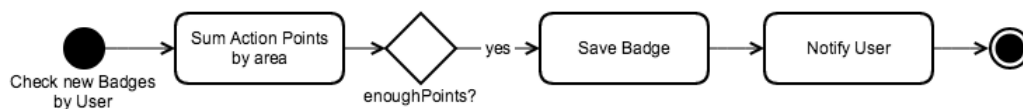


Figure 78 - Assign new badge.

10.4.4 Reset Score 7 days

The GE implements a Job that every day re-computes the weekly score for all the customers.

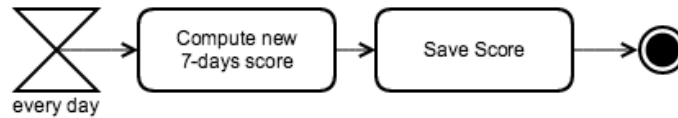


Figure 79 - Reset 7 days score.

10.4.5 Check consumption Goal

Once a week the GE checks the customer consumption to verify if she achieved the consumption goal.



Figure 80 - Check consumption goal,

10.4.6 Check podium action

The GE implements a job that every day compares the leaderboard podium with the one of the day before. If a new customer reaches the podium the GE assigns her an action.

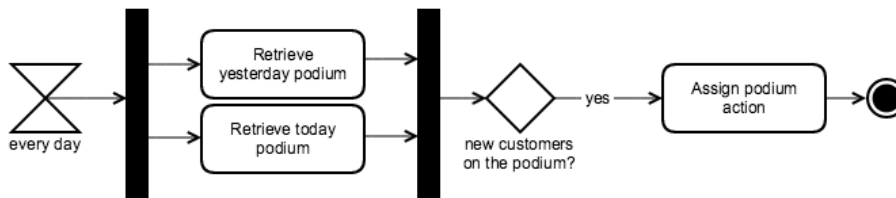


Figure 81 - Check podium action.

10.5 Interfaces

The GE exposes a set of REST services to be invoked from external applications:

- Authentication
 - Component authentication
 - User Authentication
- User:
 - Retrieve all users
 - Retrieve user by id
 - Retrieve credits by user id
- Thematic Area
 - Retrieve all areas
- Action
 - Retrieve all the actions
 - Retrieve an action by id

- Retrieve all the actions by user
 - Assign actions to users
- Badge
 - Retrieve badge areas
 - Retrieve all the badges
 - Retrieve badge by id
 - Retrieve badges by user
- Goals
 - Retrieve all the goals
 - Retrieve goal by id
 - Retrieve goal by user
- Leader board
 - Retrieve the weekly leader board
 - Retrieve the complete leader board
- Rewards
 - Retrieve all the rewards
 - Retrieve a reward by id
 - Retrieve all the rewards available for a user
 - Redeem reward

Error Handling

If there is an error during the process, due to a Bad Route or to an Internal Server Error an error message will be returned:

Code: 500 INTERNAL SERVER ERROR

If there is an error during the process due to a Missing parameter the error message will be:

Code: 400 BAD REQUEST

Content:

```
{
  "Cause": "400 BAD REQUEST",
  "Detail": "<parameter> is missing"
}
```

1.1.1 Authentication

Authenticate a Component

Description	Service to retrieve a valid ticket associated to the component
URL	/Authentication/GetTicket/getTicket
Method	POST
Headers	Content-Type: text/xml; charset=UTF-8 Accept: text/xml

URL Params	--
Data Params	Content: <pre><?xml version="1.0" encoding="UTF-8"?> <getToken> <credentials> <username> -string- </username> <password> -string- </password> </credentials> </getToken></pre> Example: <pre><?xml version="1.0" encoding="UTF-8"?> <getToken> <credentials> <username>Emivasa</username> <password>password</password> </credentials> </getToken></pre>
Success Response	Code: 200 OK Content-Type: text/xml;charset=UTF-8 Acceptpt: text/xml Content: <pre><?xml version="1.0" encoding="UTF-8"?> <Response> <authentication> <ticket> -string- </ticket> </authentication> </Response></pre> Example: <pre><?xml version="1.0" encoding="UTF-8"?> <Response> <authentication> <ticket> WR:RW1pdmFzYV8xNDM3ODM5MzkxNjU4:bU= </ticket></pre>

	<pre> </authentication> </Response> </pre>
Error Response	--
Notes	--

Authenticate a user

Description	Service to retrieve an authentication token for a user
URL	/Authentication/GetToken/getToken
Method	POST
Headers	Content-Type: text/xml;charset=UTF-8 Accept: text/xml Authorization: :ticket
URL Params	--
Data Params	Content: <pre> <?xml version="1.0" encoding="UTF-8"?> <getToken> <credentials> <user_id> -string- </user_id> </credentials> </getToken> </pre> Example: <pre> <?xml version="1.0" encoding="UTF-8"?> <getToken> <credentials> <user_id>15</user_id> </credentials> </getToken> </pre>

Success Response	Code: 200 OK Content-Type: application/json;charset=UTF-8 Content: <pre><?xml version="1.0" encoding="UTF-8"?> <Response> <authentication> <token> -string- </token> <result>success</result> <exception/> </authentication> </Response></pre> Example: <pre><?xml version="1.0" encoding="UTF-8"?> <Response> <authentication> <token>4c76b0a7-5908-44a4-a52e- e176638f551d</token> <result>success</result> <exception/> </authentication> </Response></pre>
Error Response	User not exists: <pre><Response> <authentication> <token/> <result>error</result> <exception>The user does not exist</exception> </authentication> </Response></pre> Ticket expired: <pre><Error> <Cause>Unable to execute the REST operation</Cause> <Detail>Expired authorization code</Detail> </Error></pre>

Notes	--

1.1.2 User Services

Retrieve all users

Description	Service to retrieve the list of all the users registered in the GE
URL	/UserActivityCreditWebServiceREST/GetAllUsers/getUsers
Method	GET
URL Params	--
Data Params	--
Success Response	Code: 200 OK Content-Type: application/json;charset=UTF-8 Content: <pre>{ "users": [<numeric>], "names": [<string>] }</pre> Example: <pre>{ "users": [14, 15, 16], "names": ["Fraternali", "Baroffio", "Pasini"] }</pre>

	<pre>] } </pre>
Error Response	--
Notes	--

Retrieve a user by id

Description	Service to retrieve the information about the specified users registered in the GE
URL	/UserActivityCreditWebServiceREST/GetUser/getUser?id=:id
Method	GET
URL Params	Required: id=[integer] example: id=49
Data Params	--
Success Response	Code: 200 OK Content-Type: application/json;charset=UTF-8 Content: <pre> { "image": <base64>, "id": <integer>, "email": <string>, "username": <string> } </pre> Example: <pre> { "image": "/9j/4AAQSk..", "id": "49", "email": "piero@test.com", "username": "Piero" } </pre>

Error Response	wrong id
Notes	--

Retrieve user credits

Description	Service to retrieve the summary of the credits of the specified user
URL	/UserActivityCreditWebServiceREST/ GetUserCredits/getUserCredits?user_id=:id
Method	GET
URL Params	Required: user_id=[integer] example: user_id=49
Data Params	--
Success Response	Code: 200 OK Content-Type: application/json;charset=UTF-8 Content: <pre>{ "userEmail": <string>, "totalCredit": <float>, "creditsSpent": <float>, "creditsAvailable": <float> }</pre> Example: <pre>{ "userEmail": "luca@test.com", "totalCredit": "18550.00", "creditsSpent": "0.00", "creditsAvailable": "18550" }</pre>
Error Response	

	--
Notes	--

1.1.3 Thematic Areas

Retrieve all Thematic Areas

Description	Service to retrieve all the available thematic areas
URL	/UserActivityCreditWebServiceREST/GetAreas/getAreas
Method	GET
URL Params	--
Data Params	--
Success Response	Code: 200 OK Content-Type: application/json;charset=UTF-8 Content: <pre>[{ "name": <string>, "oid": <integer> }]</pre> Example: <pre>[{ "name": "Water Saving Insights", "oid": "1" }, { "name": "Saving Water", "oid": "2" }]</pre>
Error	

Response	--
Notes	--

1.1.4 Actions

Retrieve all Actions

Description	Service to retrieve all the available actions
URL	/UserActivityCreditWebServiceREST/GetActions/getActions
Method	GET
URL Params	--
Data Params	--
Success Response	Code: 200 OK Content-Type: application/json;charset=UTF-8 Content: <pre>[{ "oid": <integer>, "name": <string>, "description": <string>, "area": <string>, name of the area associated to the action "are_oid": <integer>, id of the area associated to the action "score": <float> points earned performing the action "reputation": <boolean>, "participation": <boolean>, "check_time_elapsed": <boolean>, "time_elapsed": <float>, null if check_time_elapsed=false "repeatable": <boolean>, "active": <boolean> }]</pre> Example: <pre>[</pre>

	<pre> { "oid": 1, "name": "Create a team", "description": "Create a team with your friends and neighbors", "area": "Engagement", "area_oid": 3, "score": 200.0, "reputation": true, "participation": true, "check_time_elapsed": false, "time_elapsed": null, "repeatable": true, "active": true }, { "oid": 2, "name": "Achieve a common goal", "description": "Achieve a common goal with your team", "area": "Engagement", "area_oid": 3, "score": 500.0, "reputation": true, "participation": true, "check_time_elapsed": false, "time_elapsed": null, "repeatable": true, "active": true }] </pre>
Error Response	--
Notes	--

Retrieve an Action by id

Description	Service to retrieve an action given its id
URL	/UserActivityCreditWebServiceREST/GetAction/getAction?id=:id
Method	GET

URL Params	Required: id=[integer] example: id=1
Data Params	--
Success Response	Code: 200 OK Content-Type: application/json;charset=UTF-8 Content: <pre>{ "oid": <integer>, "name": <string>, "description": <string>, "area": <string>, "area_oid": <integer>, "score": <float> "reputation": <boolean>, "participation": <boolean>, "check_time_elapsed": <boolean>, "time_elapsed": <float>, null if check_time_elapsed=false "repeatable": <boolean>, "active": <boolean> }</pre> Example: <pre>{ "oid": 1, "name": "Create a team", "description": "Create a team with your friends and neighbors", "area": "Engagement", "area_oid": 3, "score": 200.0, "reputation": true, "participation": true, "check_time_elapsed": false, "time_elapsed": null, "repeatable": true, "active": true }</pre>
Error Response	Reason: wrong id

	Code: 400 BAD REQUEST Content: <pre>{ "Cause": "400 BAD REQUEST", "Detail": "The action does not exist" }</pre>
Notes	--

Retrieve all the Actions by User

Description	Service to retrieve all the actions performed by the specified user
URL	/UserActivityCreditWebServiceREST/GetUserActions/getActions?user_id=:id
Method	GET
URL Params	Required: id=[integer] example: id=15
Data Params	--
Success Response	Code: 200 OK Content-Type: application/json;charset=UTF-8 Content: <pre>[{ "oid": <integer>, id of the action instance "timestamp": <yyyy-mm-dd hh:mm:ss.c>, "action_oid": <integer> id of the action performed }]</pre> Example: <pre>[{ "oid": 77, "timestamp": "2015-06-29 13:28:11.0",</pre>

	<pre> "action_oid": 77 }, { "oid": 89, "timestamp": "2015-06-29 13:34:56.0", "action_oid": 89 }] </pre>
Error Response	Reason: wrong user id Code: 400 BAD REQUEST Content: <pre> { "Cause": "400 BAD REQUEST", "Detail": "The user ID does not exist" } </pre>
Notes	--

Assign actions to user

Description	Service to post the execution of an action to a user, the GE will save the action and add the points to the user total points
URL	/UserActivityCreditWebServiceREST/AssignActionsToUsersById/assignActionsToUsers
Method	POST
URL Params	--
Data Params	Accept: application/json Content-Type: application/json; charset=UTF-8 Accept-Encoding: gzip.deflate Content: <pre> [{ "user_id": <integer>, </pre>

	<pre> "action_id": <integer> }] </pre> Example: <pre> [{ "user_id": 15, "action_id": 2 }] </pre>
Success Response	Code: 200 OK Content-Type: application/json;charset=UTF-8 Content: <pre> { "action_ids": [<integer> id of the created action instance] } </pre> Example: <pre> { "action_ids": ["3299"] } </pre>
Error Response	Reason: missing/wrong input Code: 400 BAD REQUEST Content: <pre> { "Cause": "400 BAD REQUEST", "Detail": "..." } </pre>
Notes	--

1.1.5 Badges

Retrieve all the Badges

Description	Service to retrieve all the available badegs
URL	/UserActivityCreditWebServiceREST/GetBadges/getBadges
Method	GET
URL Params	--
Data Params	--
Success Response	Code: 200 OK Content-Type: application/json;charset=UTF-8 Content: <pre>{ "badges_list": [{ "area": <string> "area_id": <integer>, "badges": [{ "id": <integer>, "title": <string>, "score": <float>, score required to obtain the badge "icon": <base64> icon that represents the achieved badge }], "max": <float> score of the highest badge }] }</pre> Example: <pre>{ "badges_list": [{ "area": "Engagement", "area_id": 3, "badges": [{ "id": 5,</pre>

	<pre> "title": "Engager", "score": 500.0, "icon": "iVBORw0..." }], "max": 2500.0 }, { "area": "Profiling", "area_id": 4, "badges": [{ "id": 7, "title": "Beginner Profiler", "score": 100.0, "icon": "iVBORw0KGgoAAAANSUHE.." }, { "id": 8, "title": "Advanced Profiler", "score": 1500.0, "icon": "iVBORw0KGgoA.." }], "max": 2500.0 }] } </pre>
Error Response	--
Notes	--

Retrieve badge by id

Description	Service to retrieve the detailed description of a badge given its id
URL	/UserActivityCreditWebServiceREST/GetBadge/getBadge?id=:badgeId
Method	GET

URL Params	Required: id=[integer] example: id=8
Data Params	--
Success Response	Code: 200 Content-Type: application/json;charset=UTF-8 Content: <pre>{ "id": <integer>, "title": <string>, "score": <float>, score required to obtain the badge "icon": <base64> icon that represents the achieved badge }</pre> Example: <pre>{ "id": 8, "title": "Advanced Profiler", "score": 1500.0, "icon": "iVBORw0KGgoA.." }</pre>
Error Response	Reason: wrong id Code: 400 BAD REQUEST Content: <pre>{ "Cause": "400 BAD REQUEST", "Detail": "The badge does not exist" }</pre>
Notes	--

Retrieve badges by user

Description	Service to retrieve all the badges achieved by a user
URL	/UserActivityCreditWebServiceREST/GetUserBadges/getBadges?user_id=:id
Method	GET
URL Params	Required: user_id=[integer] example: user_id=8
Data Params	--
Success Response	Code: 200 Content-Type: application/json;charset=UTF-8 Content: <pre>{ "badges_list": [{ "area": <string> "area_id": <integer>, "badges": [{ "id": <integer>, "title": <string>, "score": <float>, score required to obtain the badge "icon": <base64> icon that represents the achieved badge }], "max": <float> score of the highest badge }, "data": [for each area the current badge status { "id": <integer>, area id "name": <string>,area name "score": <float>, current credits gained in the area "icon": <base64> last badge icon }] } }</pre> Example:

```

{
  "badges_list": [
    {
      "area": "Engagement",
      "area_id": 3,
      "badges": [
        {
          "id": 5,
          "title": "Engager",
          "score": 500.0,
          "icon": "iVBORw0..."
        }
      ],
      "max": 2500.0
    },
    {
      "area": "Profiling",
      "area_id": 4,
      "badges": [
        {
          "id": 7,
          "title": "Beginner Profiler",
          "score": 100.0,
          "icon": "iVBORw0KGgoAAAANSUhE.."
        },
        {
          "id": 8,
          "title": "Advanced Profiler",
          "score": 1500.0,
          "icon": "iVBORw0KGgoA.."
        }
      ],
      "max": 2500.0
    }
  ],
  "data": [
    {
      "id": 3,
      "name": "Engagement",
      "score": 500.0,
      "icon": "iVBORw0KGgoA.."
    },
    {
      "id": 4,

```

	<pre> "name": "Profiling", "score": 10000.0, "icon": "ivBORw0KG.." }] }</pre>
Error Response	Reason: wrong id Code: 400 BAD REQUEST Content: <pre> { "Cause": "400 BAD REQUEST", "Detail": "The user ID does not exist" }</pre>
Notes	--

1.1.6 Leaderboard

Retrieve the overall leaderboard

Description	Service to retrieve all the overall leaderboard
URL	/UserActivityCreditWebServiceREST/GetLeaderboard/getLeaderboard
Method	GET
URL Params	--
Data Params	--
Success Response	Code: 200 OK Content-Type: application/json;charset=UTF-8 Content:

	<pre>[{ "user_id": <integer>, "credits": <float> }]</pre> Example: <pre>[{ "user_id ": 48, "credits": "39150.00" }, { "user_id ": 46, "credits": "38850.00" }]</pre>
Error Response	--
Notes	--

Retrieve the weekly leaderboard

Description	Service to retrieve all the weekly leaderboard
URL	/UserActivityCreditWebServiceREST/ GetLeaderboardWeek/getLeaderboardWeek
Method	GET
URL Params	--
Data Params	--
Success Response	Code: 200 Content-Type: application/json;charset=UTF-8 Content:

	<pre>[{ "user_id": <integer>, "credits": <float> }]</pre> Example: <pre>[{ "user_id ": 48, "credits": "39150.00" }, { "user_id ": 46, "credits": "38850.00" }]</pre>
Error Response	--
Notes	--

1.1.7 Rewards

Retrieve all the Rewards

Description	Service to retrieve all the available rewards
URL	/UserActivityCreditWebServiceREST/GetRewards/getRewards
Method	GET
URL Params	--
Data Params	--
Success Response	Code: 200 OK Content-Type: application/json;charset=UTF-8 Content:

	<pre>[{ "needed_points": <float>, "available": <boolean>, "title": <string>, "oid": <integer>, "description": <string> }]</pre> Example: <pre>[{ "needed_points": "1200.00", "available": true, "title": "Home Water and Energy Saving Kit", "oid": 2, "description": "This award winning box of nine water saving devices (including educational..." }, { "needed_points": "2000.00", "available": true, "title": "Tankless Water Heater", "oid": 4, "description": "Mini Portable Gas Tankless Water" }]</pre>
Error Response	--
Notes	--

Retrieve reward by id

Description	Service to retrieve all the available rewards
URL	/UserActivityCreditWebServiceREST/GetReward/getReward?id=:id
Method	GET

URL Params	-- Required: id=[integer] example: id=1
Data Params	--
Success Response	Code: 200 OK Content-Type: application/json;charset=UTF-8 Content: <pre> { "needed_points": <float>, "available": <boolean>, "title": <string>, "oid": <string>, "description": <string> } </pre> Example: <pre> { "needed_points": "1200.00", "available": true, "oid": 2, "title": "Home Water and Energy Saving Kit", "description": "This award winning box of nine water saving devices (including educational..." } </pre>
Error Response	--
Notes	--

Retrieve redeemed reward by user

Description	Service to retrieve all the rewards redeemed by a user
URL	/UserActivityCreditWebServiceREST/ GetUserRewards/getRewards?user_id=:id

Method	GET
URL Params	Required: user_id=[integer] example: user_id=49
Data Params	--
Success Response	Code: 200 Content-Type: application/json;charset=UTF-8 Content: <pre>[{ "timestamp": <yyyy-mm-dd hh:mm:ss.c>, "user_id": <integer>, "reward_id":<integer>, "oid": <integer> }]</pre> Example: <pre>[{ "timestamp ": "2015-07-09T22:27:55+0200", "user_id": 49, "reward_id": 1, "oid": 1 }]</pre>
Error Response	Reason: wrong id Code: 400 BAD REQUEST Content: <pre>{ "Cause": "400 BAD REQUEST", "Detail": "The user ID does not exist" }</pre>

Notes	--

Redeem reward to user

Description	Service to redeem rewards for a user, the GE will send the notification to the admin and subtract the points to the user total points
URL	/UserActivityCreditWebServiceREST/RedeemUserRewardById/redeemUserReward
Method	POST
URL Params	--
Data Params	Accept: application/json Content-Type: application/json;charset=UTF-8 Accept-Encoding: gzip.deflat Content: <pre>[{ "user_id": <integer>, "reward_id": <integer> }]</pre> Example: <pre>[{ "user_id": 49, "action_id": 1 }]</pre>
Success Response	Code: 200 OK Content-Type: application/json;charset=UTF-8 Content:

	<pre>{ "user_id": <integer>, "availablePoints": <float> }</pre> Example: <pre>{ "user_id": 48, "availablePoints": "19150" }</pre>
Error Response	Reason: • The user doesn't have enough points to redeem the reward • The selected user doesn't exist • The user email must be provided to the Web Service • The reward id is not provided Code: 400 BAD REQUEST Content: <pre>{ "Cause": "Unable to redeem the reward", "Detail": "..." }</pre>
Notes	--

11. Games Platform

The Games Platform constitute the backend able to take care of the registration of the users, the tracking of all the actions performed by players within the games compatible with the Smarth2O platform and their profilation. The backed uses Nodejs enhanced by Express, to handle the REST APIs calls to and from the Games Platform framework and MongoDB to store the data related to the players and all the actions performed within the Smarth2O games. [Node.js®](#) is a JavaScript runtime built on [Chrome's V8 JavaScript engine](#). Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. [Express](#) is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications. [MongoDB](#) is an open source, document-oriented database designed with both scalability and developer agility in mind. Instead of storing your data in tables and rows as you would with a relational database, in MongoDB you store JSON-like documents with dynamic schemas.

In the following sections, UML Diagrams related to the interaction between the components and the Games Platform are described.

11.1 Structure

The data model related to the GamesPlatform is defined in the class diagram of Figure 82.

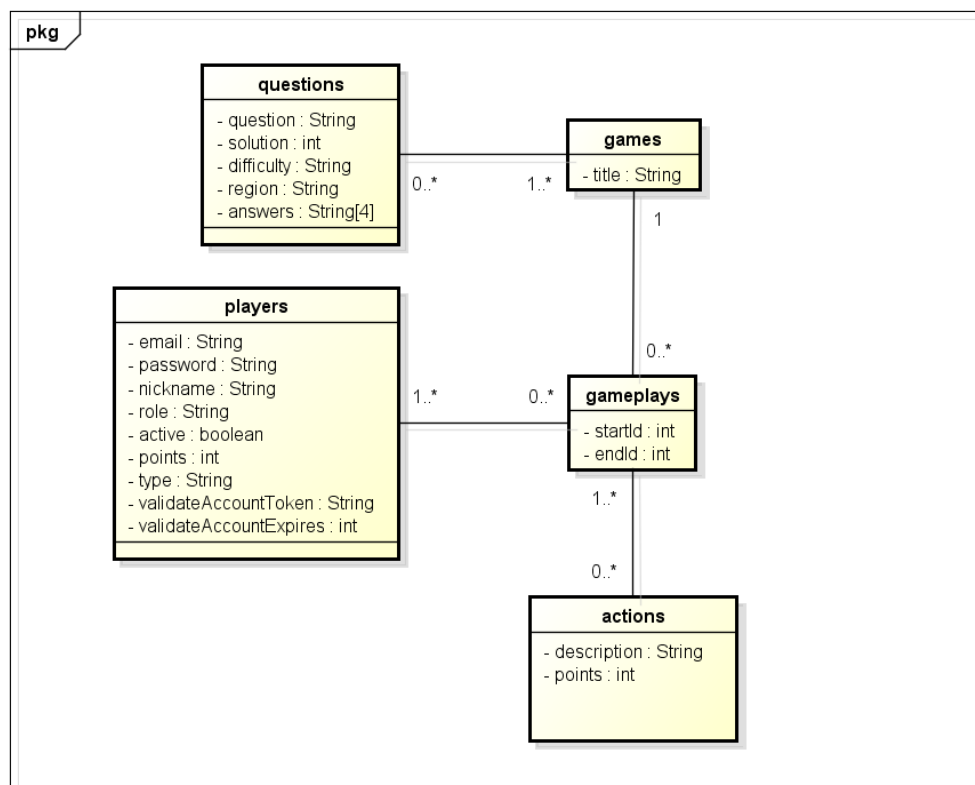


Figure 82 - Game Platform data model.

The Games Platform is able to support several games and store for each of those games the gameplay sessions composed by the actions performed by the registered users within the platform. Up to now since the only game that has been created was Drop! Question, the only feature needed to support the game in the platform was the extension of the game concept with questions. If further games will be developed, the game class could be extended in a similar way. The schema presented here has been implemented as NoSql documents in MongoDB.

Games is a table containing the list of the titles that have been activated within the platform, in order to distinguish the contributions on a “per game basis”.

Players is a table that contains the registered players within the platform. Along with the usual identification fields such as email, password and the activation status of the account, each user is associated with a role (player or admin are currently the only two meaningful roles), the type of the player (beginner, intermediate, expert) and the number of points he/she has accumulated within the platform.

Actions is a table containing all the actions that have been performed by a player within the platform. It has a description field in which the developer can store all the information related to the performed action and the list of points associated to the action that has been performed.

Gameplay is a table used to store gameplay sessions for the players within the platform. It collects all the actions performed by a player in a single gameplay instance with the starting and ending timestamp for the session.

11.2 Interaction

11.2.1 Sign-up

The Signup procedure interacts just with the Games Platform (GP), without trying to duplicate the users in the Advanced Customer Portal (ACP). If the ACM has a registered user with the same email used to register to the GP, the GP will send the appropriate calls to the ACP, otherwise the calls will just fail due to the lack of a corresponding registered user to the ACP. A user can register to the Games Platform only if a validated account with the same email is not present. If a fresh account is created, it has to be validated within the allotted time of three hours, otherwise the validation token will expire.

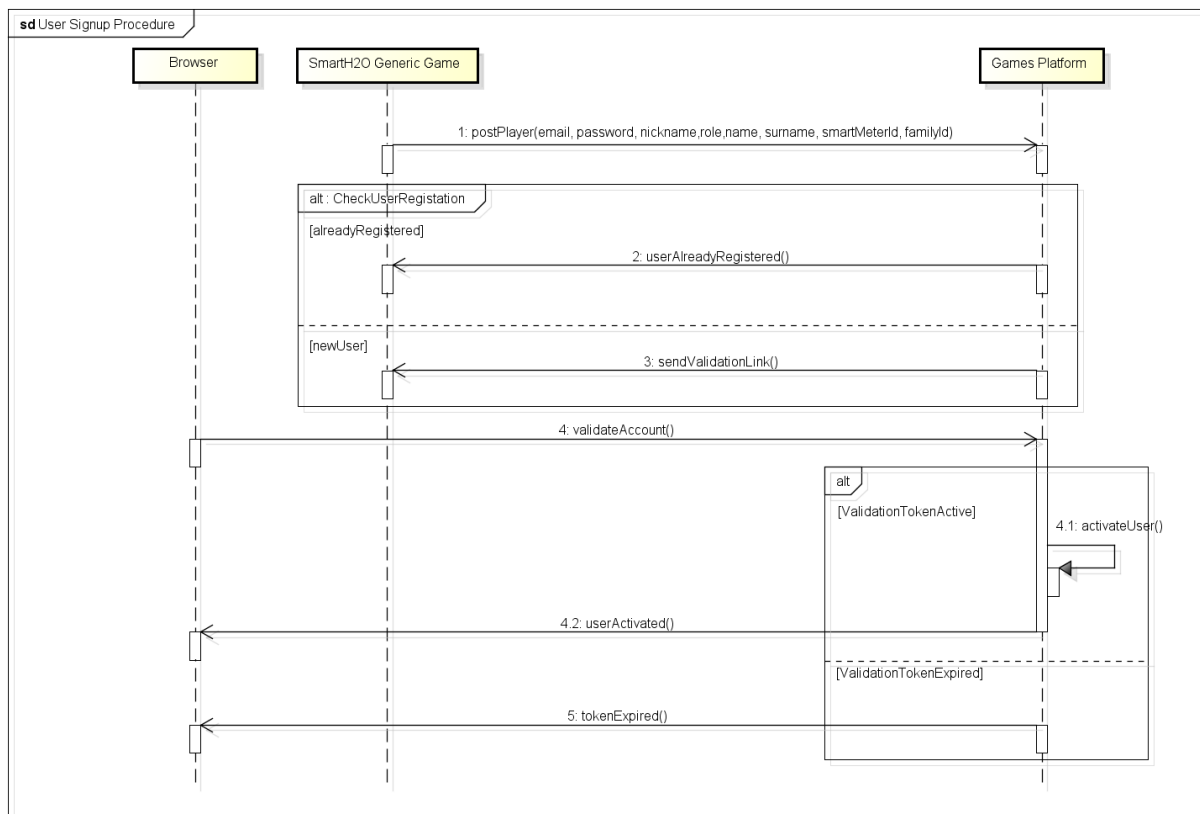


Figure 83 Game Platform signup flow.

11.2.2 Connecting Player Profile to the Gamification Engine

To connect the user of a SmarH2O game to the corresponding profile in the Gamification Engine, it is first necessary to authenticate correctly within the games platform. Once the operation has been performed, the game is allowed to connect the Games Platform account to the Gamification Engine as a standard action to be performed in the Games Platform. The connection fails if the authentication procedure has failed.

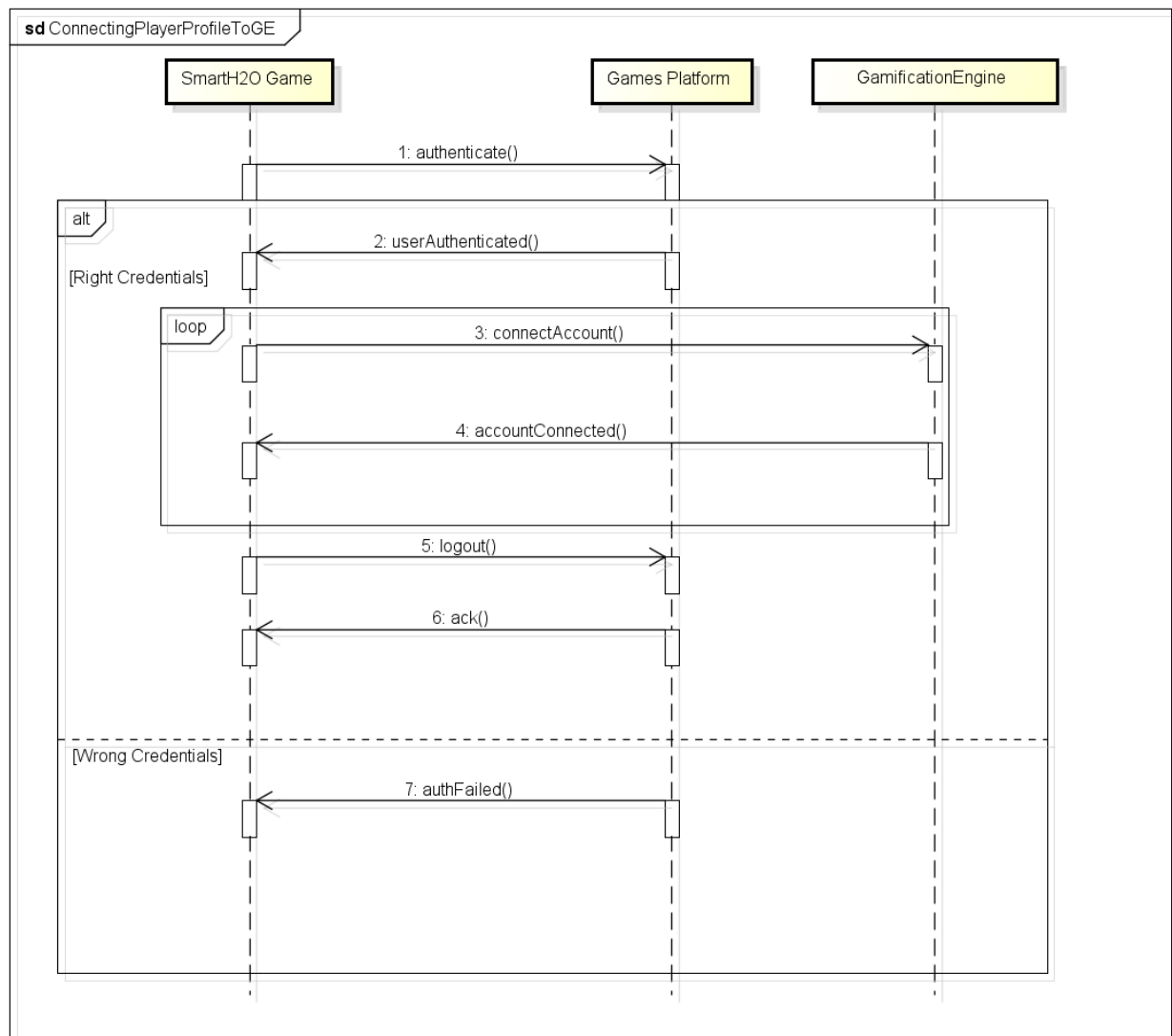


Figure 84 - Profile connection to the GE Flow.

11.2.3 Playing Standard Mobile Game (Drop! The Question)

A player willing to play a Game by making use of the functionalities offered by the Games Platform is first required to authenticate him/herself. Once it has authenticated the user, the platform must validate the game by providing the gameId to the game that has called the platform, prior to allow any other submission. Possible questions are then retrieved if supported by the game and a GameSession started until the game is closed.

Once a gamesession is opened, the user can submit answers to the questions that are shown to him and receive points, submitted to the Games Platform and Gamification Engine, for each correctly answered question. Once the game session has ended, it is closed and the user logged out.

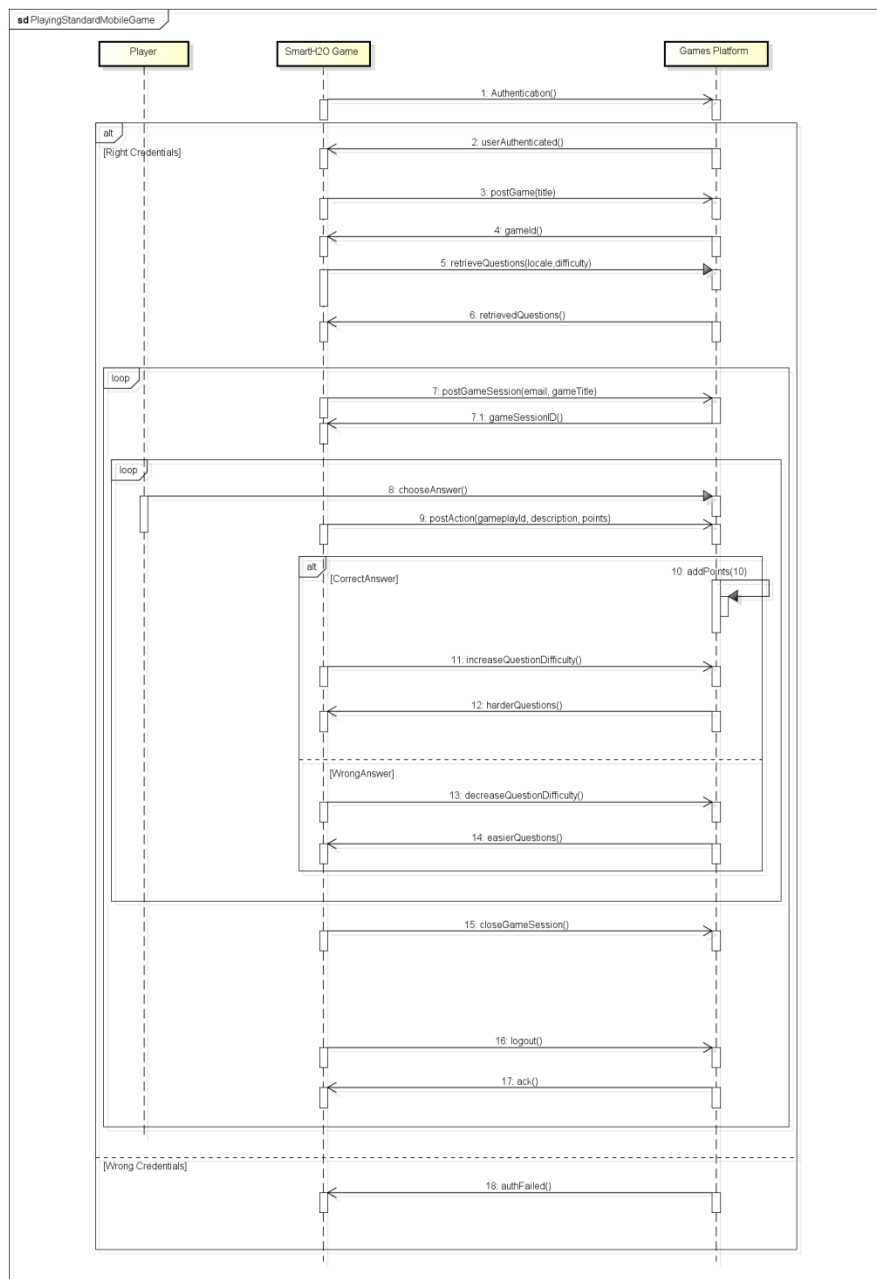


Figure 85 - Playing a game flow.

11.2.4 Play the card game and its digital Extension

Players can play the drop card game as specified within the rules of the boardgame. The boardgame is played out until the 7th monster card is picked from the deck or there are no more points chips

available. At each turn, a player can make a new bet or challenge the previous bet, if present. If she makes a new bet, she chooses the number of cards that she will be able to pick before picking a monster. If she challenge a bet, the previous players picks cards until the number of betted cards is revealed or a monster card is revealed. If no monster card is revealed, the challenged player scores all the points on the cards (no duplicates). If a monster card is revealed, the challenged player picks the monster card and all the positive cards are redistributed among the other players. Once the game has ended, the digital extension can start.

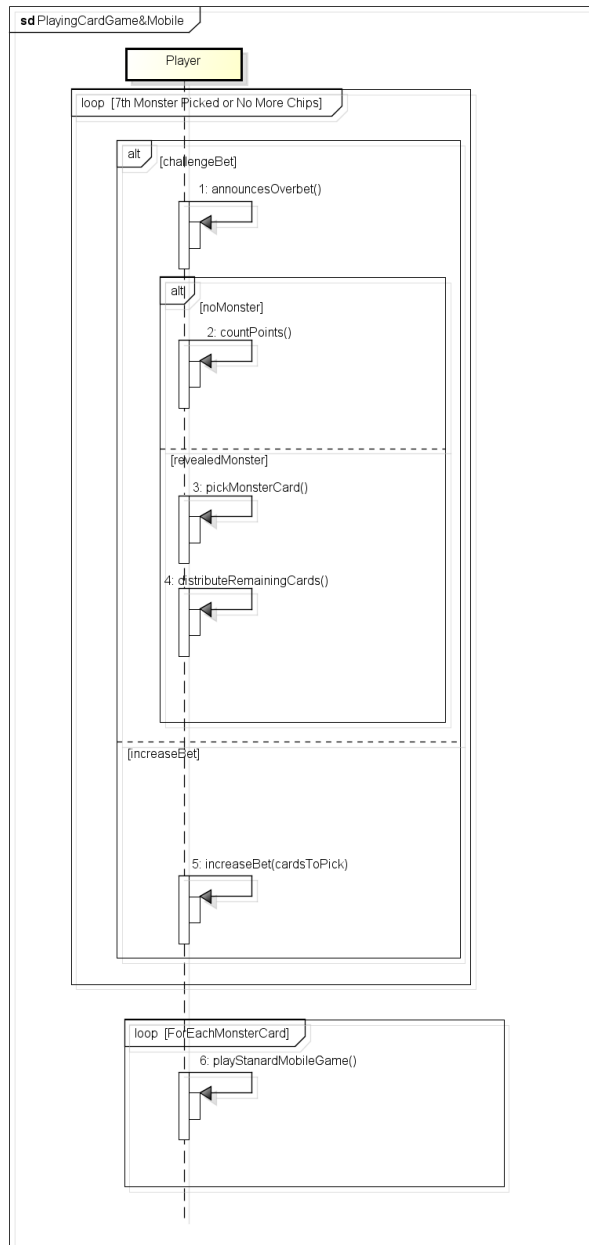


Figure 86 - Playing the card game and the digital extension flow.

11.2.5 Connecting user from Advanced Customer Portal

The Connection between users and the Advanced Customer Portal (ACP) is handled by the different Games supported by the Games Platform(GP). Once the player has established a connection with the GP, the game itself will be allowed to make any REST call to the ACP, by using its APIs.

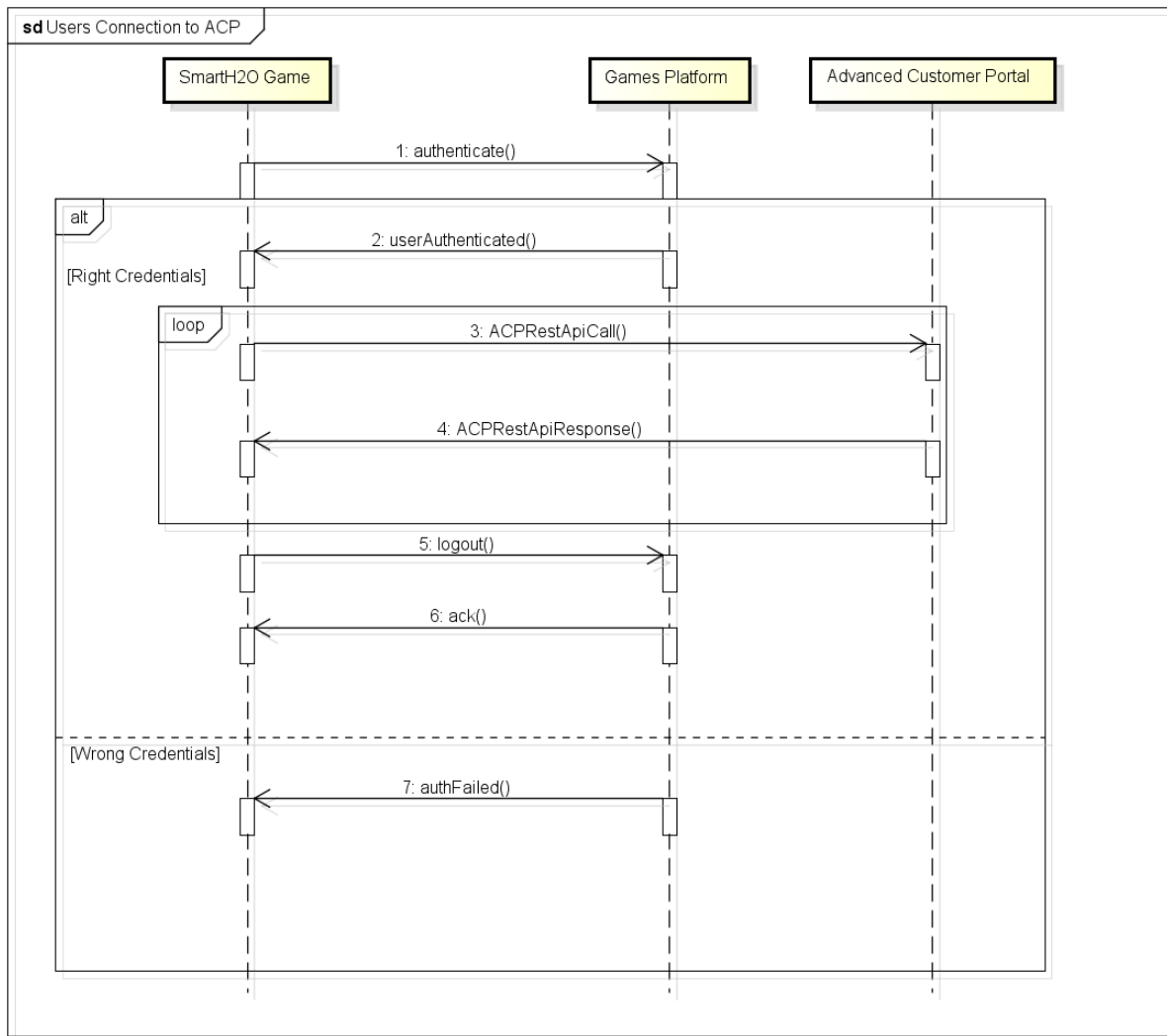


Figure 87 - Advanced Customer Portal connection flow.

11.2.6 Content administration & Players Profilation

The Games Platform (GP) allows the creation of customized content to support the online capabilities of the games that make use of the functionality it exposes. Up to now, just one game, the Drop! Digital Extension, makes use of the advanced storage capabilities offered by the GP with the retrieval of customized questions that could be added or removed from the platform to increase the challenge offered to the players, while also storing all the actions performed by the players in order to offer customized content. In order to submit new content to the platform, an Admin account must be used. The admin account can be configured in the configuration settings file of the GP. In order to submit the actions retrieved from the Players' gameplay, it is necessary first to create an entry for the game in the system, then open a gameplay session to associate to the actions performed by the players and then close the session once the player has ended his/her gameplay session. In the following, the sequence diagrams related to the submission of new content and the submission of players' actions within the platform are provided.

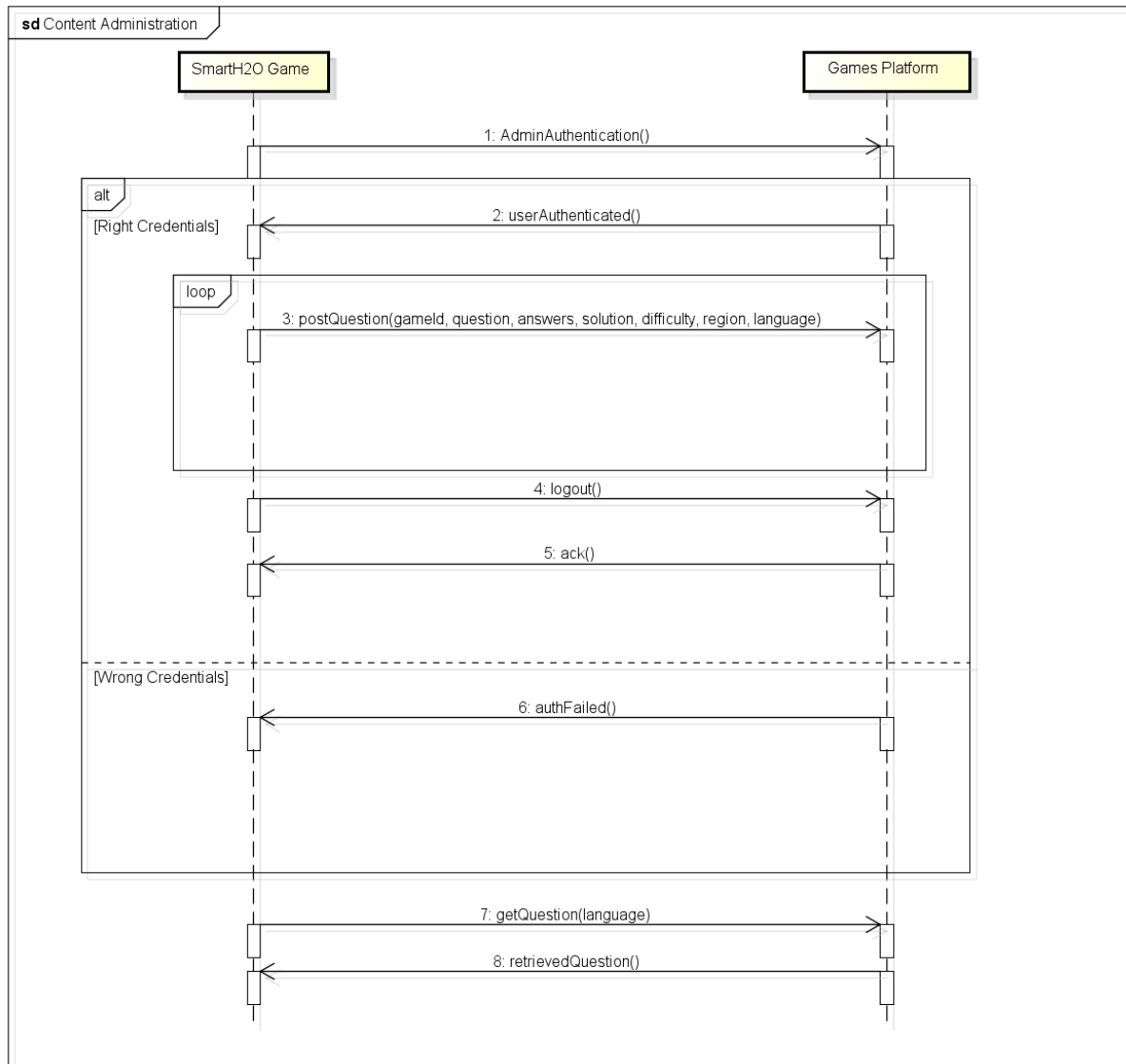


Figure 88 - Content Administration flow.

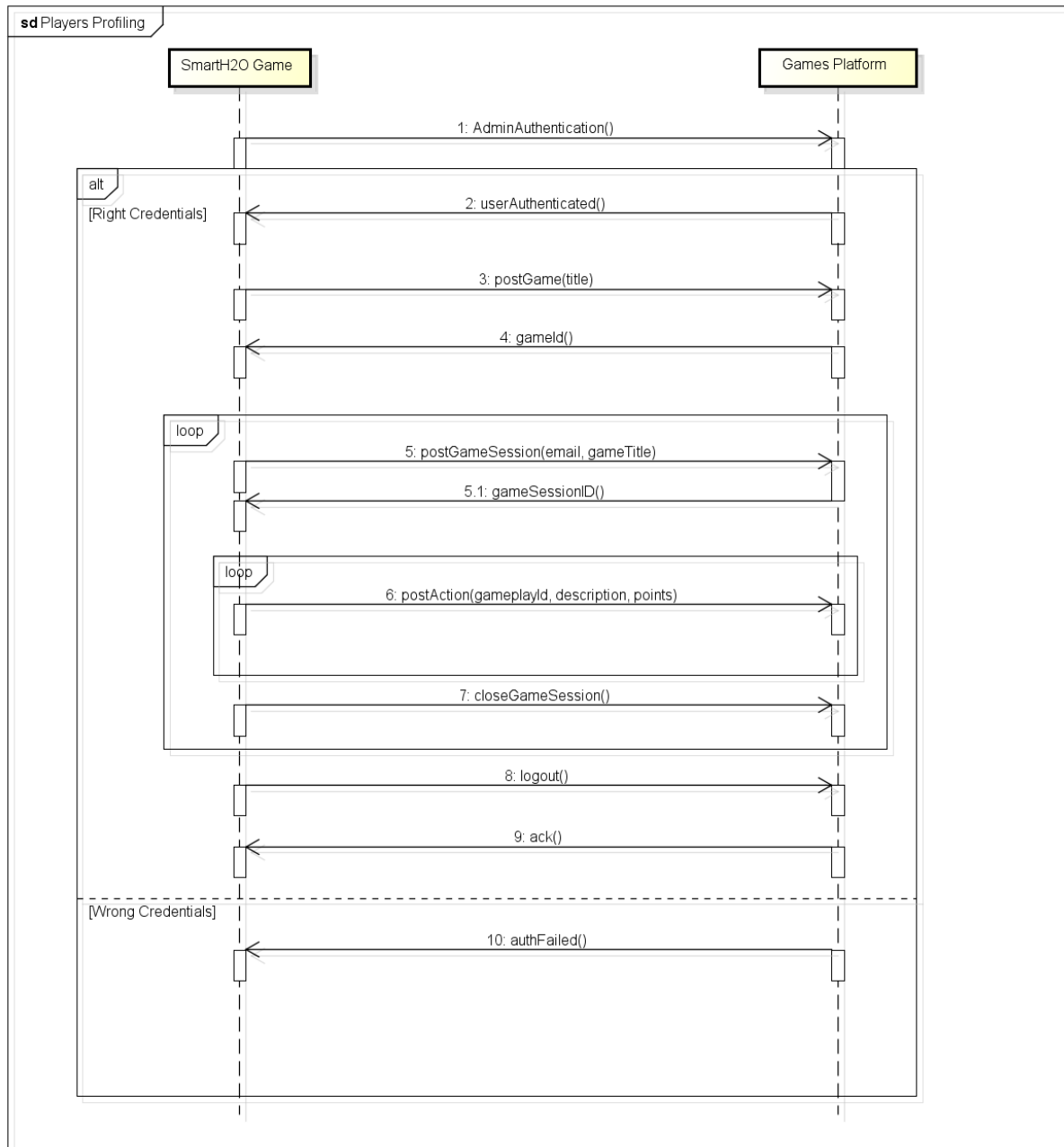


Figure 89 - Players Profiling Flow.

12. Social Network Crawler and Data Analyzer

12.1 Social network crawler

The social network crawler is a component capable of retrieving social data relevant for a specific topic according to a given set of querying criteria.

12.1.1 Logical

Class diagram overview

In Figure 90 an overview of the social network crawler class diagram is presented. As shown, the most important abstractions in the system are: the Generic Crawler, which models a social network crawler, and the Relevance Computer, which computes the relevance of content/users in the social network.

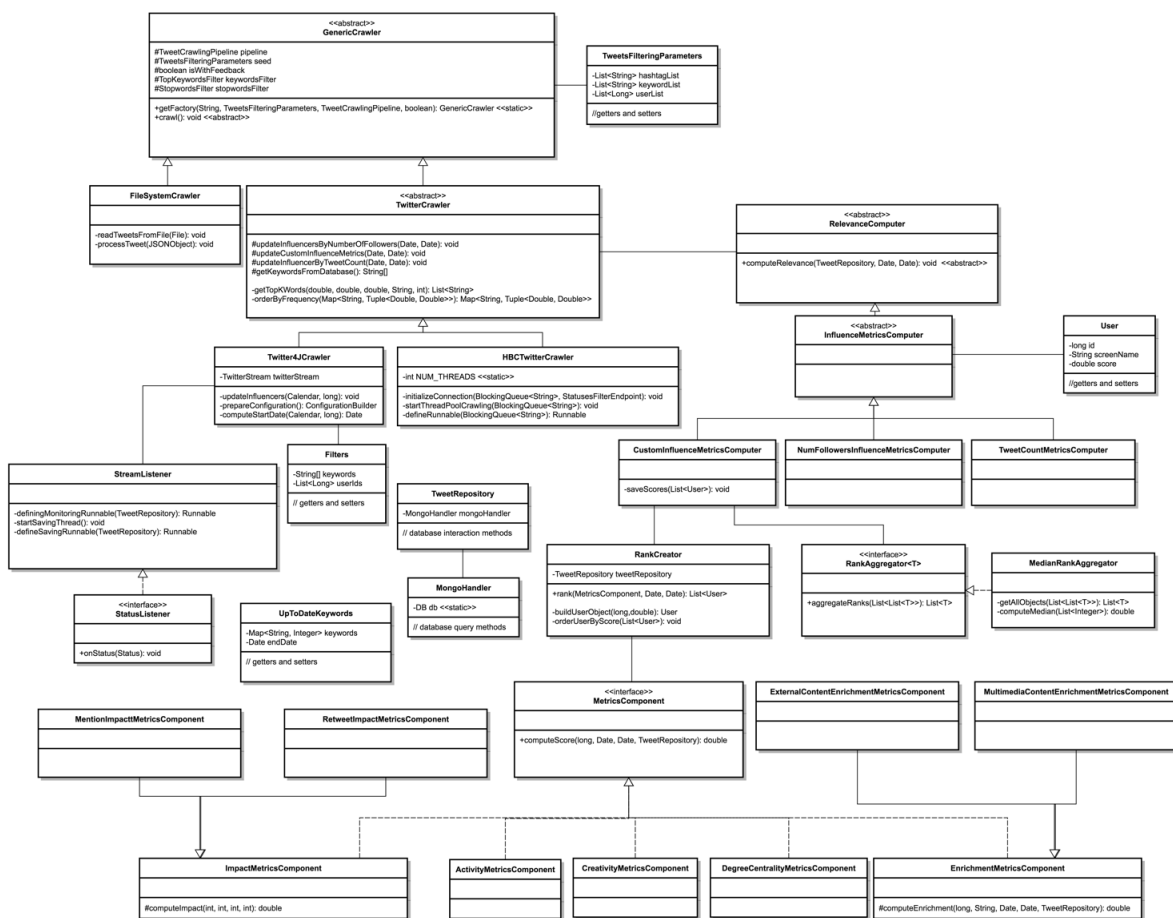


Figure 90 - Social Network Crawler class diagram.

The Generic Crawler hierarchy

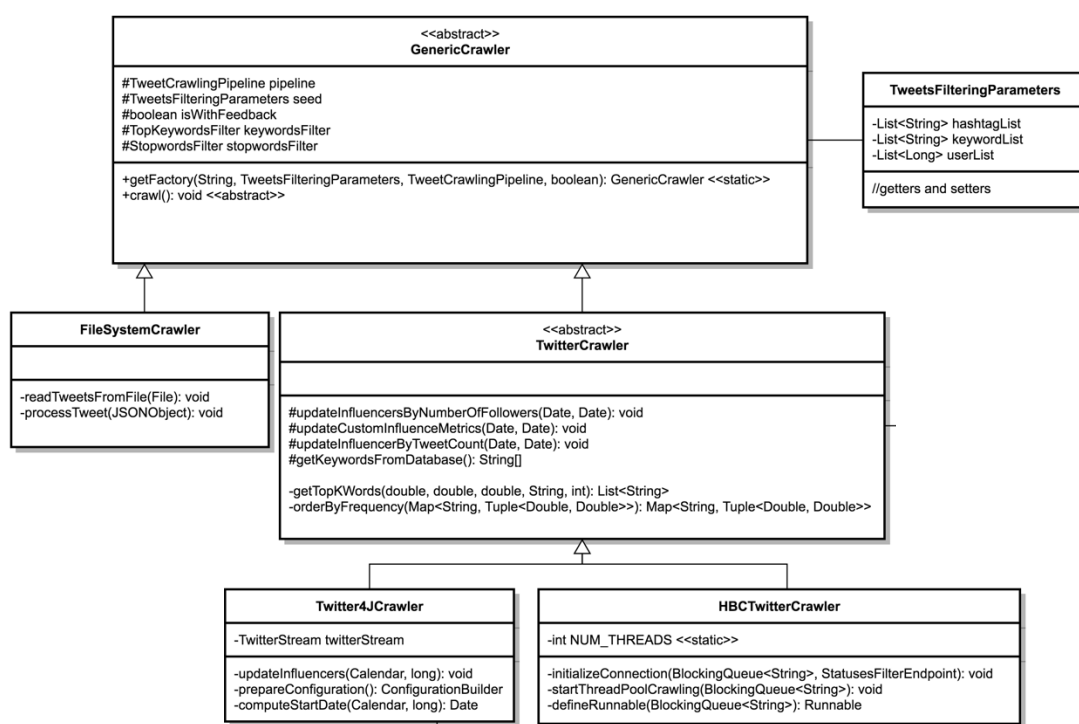


Figure 91 - Generic crawler hierarchy.

Currently, we support the collection of data from the Twitter microblogging platform. This is done by filtering the real-time feed, retaining only those posts containing one of the specified hashtags/keywords or produced by one of the specified users. From a more technical viewpoint, we provide two implementations of the Twitter crawler abstract class: the Twitter4J crawler (which internally uses the Twitter4J library²) and the HBC crawler (which internally uses the HBC library³). Moreover, a File System crawler which reads posts from JSON files is provided, to support offline storage.

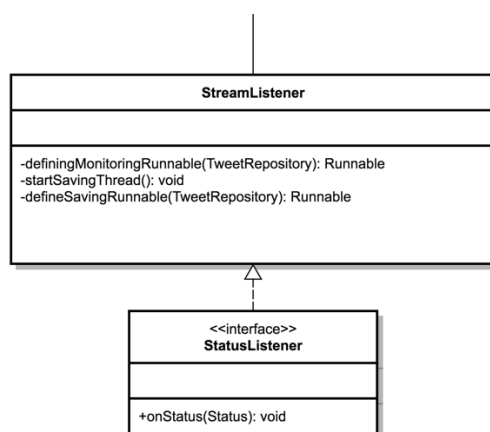


Figure 92 - Tweeter crawler.

A Twitter real-time crawler listens to the Twitter firehose and collects every status passing through it. When a tweet is captured by the crawler, it can be manipulated by the Stream Listener component, which stores it in the database, waiting for the analyzer to classify it as relevant/non-relevant for the selected topic of interest.

² <http://twitter4j.org>

³ <http://github.com/twitter/hbc>

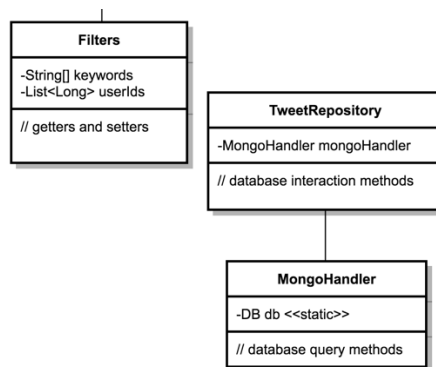


Figure 93 – Twitter repository.

The **TweetRepository** class is in charge of storing tweets in the database. In this case, a MongoDB database is used to store social data, since tweets are represented in JSON format and thus they can be naturally stored in such database.

Moreover, the **Filters** class contains the list of keywords, hashtags and users that will be used to filter the firehose.

The Relevance Computer hierarchy

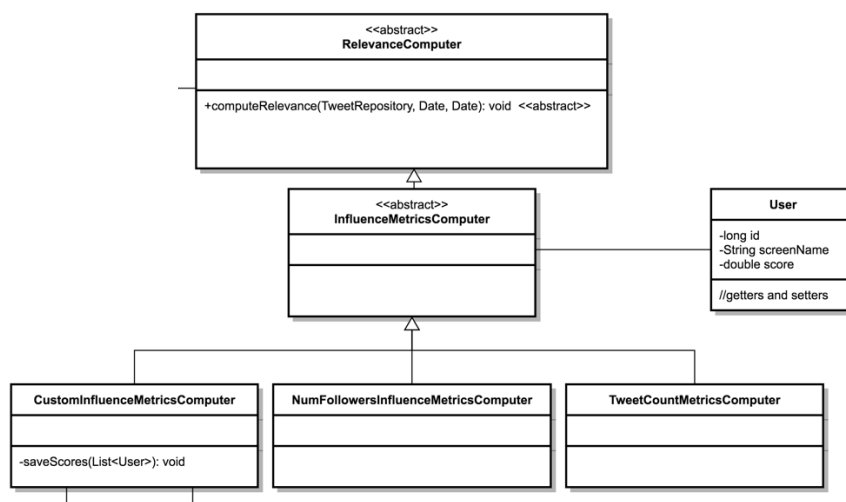


Figure 94 –RelevanceComputer hierarchy.

The **Influence Metrics Computer** computes the influence of a user whose tweets have been captured by the system.

Currently, we support three metrics:

- The **Custom Influence metrics** is composed by several signals collected from the user's profile and produced data (see below for further details)
- The **Number of Followers metrics** ranks users according to the number of followers they have on the Twitter platform
- The **Tweet Count metrics** ranks users according to the number of topically relevant tweets collected by the system

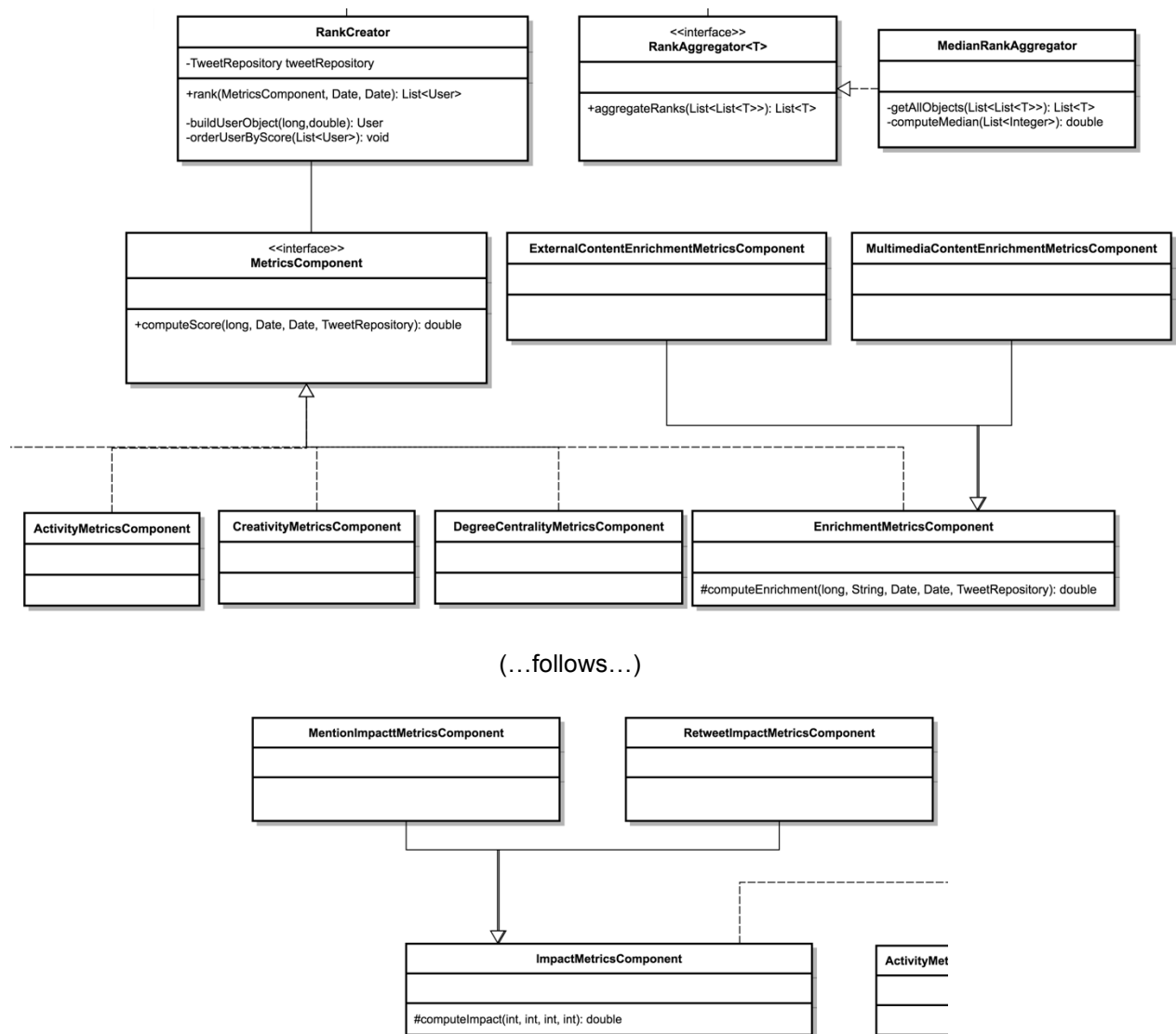


Figure 95 - Various metrics components.

The Custom influence metrics ranks users according to the following signals:

- Their **creativity**, i.e., their ability of generating original content
- Their **activity**, i.e., the frequency with which they generate topic-related content
- Their **centrality** in the network, measured via the Degree Centrality metrics
- Their ability of **enrich** the content they produce via **multimedia content** and **external content** (e.g., blog articles)
- Their **impact** in the network, measured via the **retweet impact** (i.e., how much they are retweeted vs. how much they retweet) and their **mention impact** (i.e., how much they are mentioned vs. how much they mention)

Several rankings are generated for each metrics signal (via the Rank Creator component), and then the final influence ranking is computed by aggregating the set of produced ranks (via the Rank Aggregator component).

12.2 Data analyzer

The Data analyzer is the component used to assess the relevance of social data to the selected topic. Every collected post is classified via a multimodal classifier (i.e., a classifier which evaluates both textual and multimedia content) to assess its relevance. Every time a non-relevant post is found, it is deleted from the database. Thus, only the relevant posts constitute the final dataset retained by the

system.

12.2.1 Logical

Data analyzer overview

In the following, an overview of the data analyzer structure is presented.

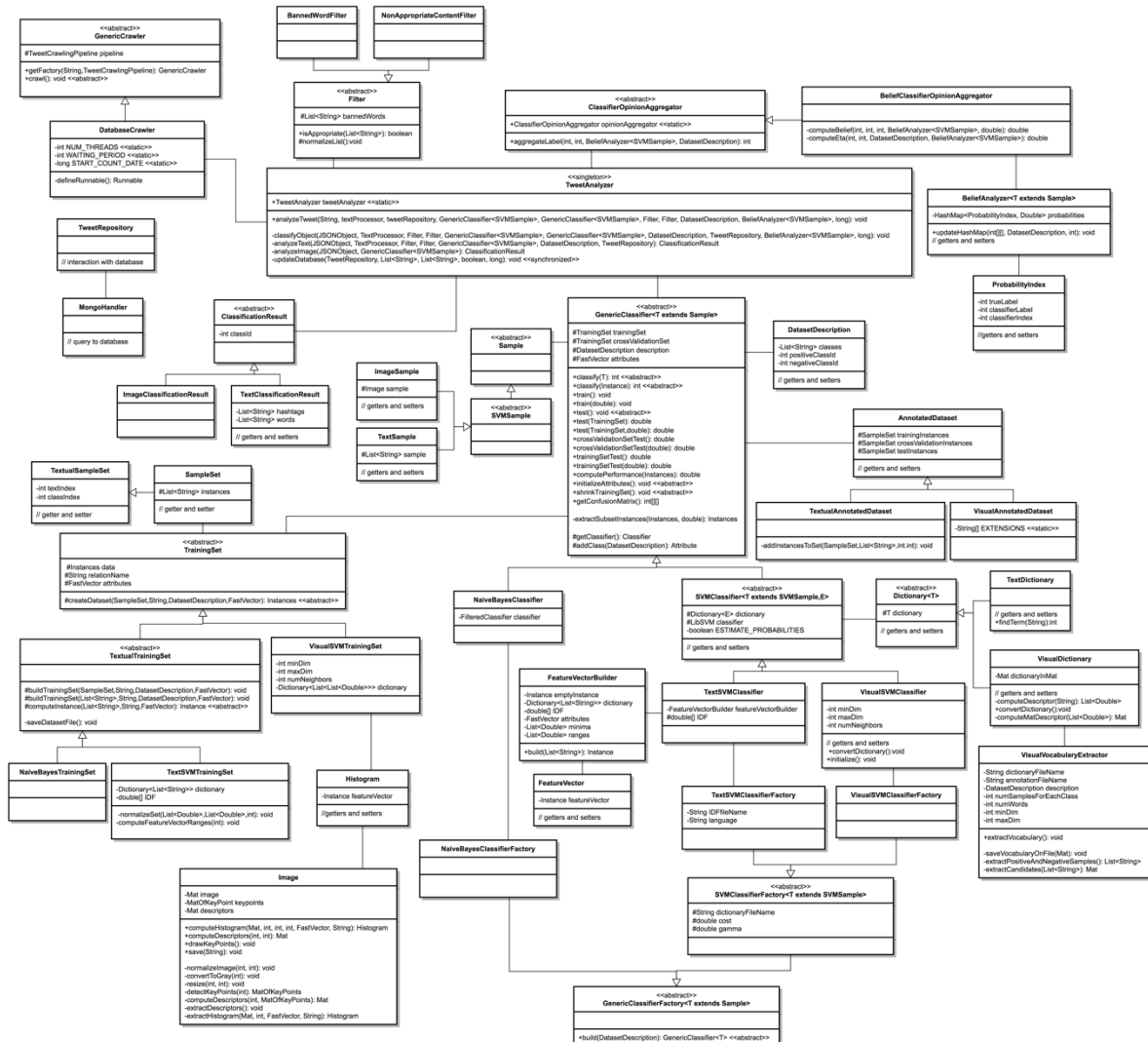


Figure 96 - Data analyser components.

The system can be subdivided in the following sub-areas:

- The **Core component**, which retrieves the posts the crawler collected, classifies them as relevant/non-relevant to the selected topic and discards them if non-relevant
- The **Classifier hierarchy**, which lists all the available classifiers (both textual and multimedia)
- The **Classifier aggregator component**, which aggregates the classifiers' opinions into a single label (i.e., relevant/non-relevant) for the processed post
- The **Classification results**, which represents the opinion of the applied set of classifiers
- The **Data corpus**, which represents a set of posts from a social network that were annotated as relevant/non-relevant to the selected topic
- The **Training sets**, which contain feature vectors with associated labels and are used to train the classifiers

The core component

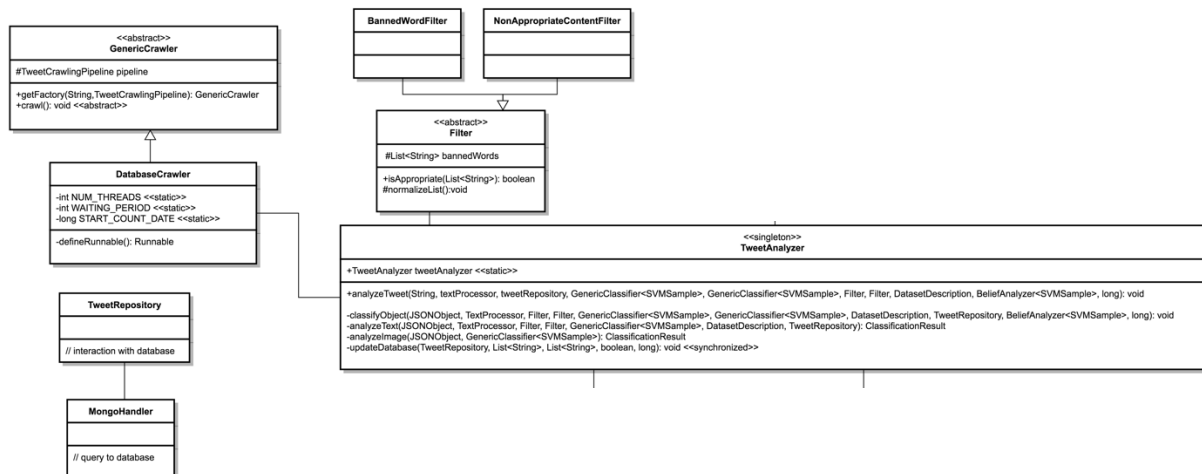


Figure 97 - Core component.

The core component:

1. Retrieves the social network posts from the database
2. Requires their classification as topically relevant/non-relevant
3. In case, discards them when they are annotated as non-relevant to the selected topic

As with the crawler, the Tweet Repository component is in charge of interacting with the database, which is MongoDB, since tweets are stored in JSON format.

The classifier hierarchy

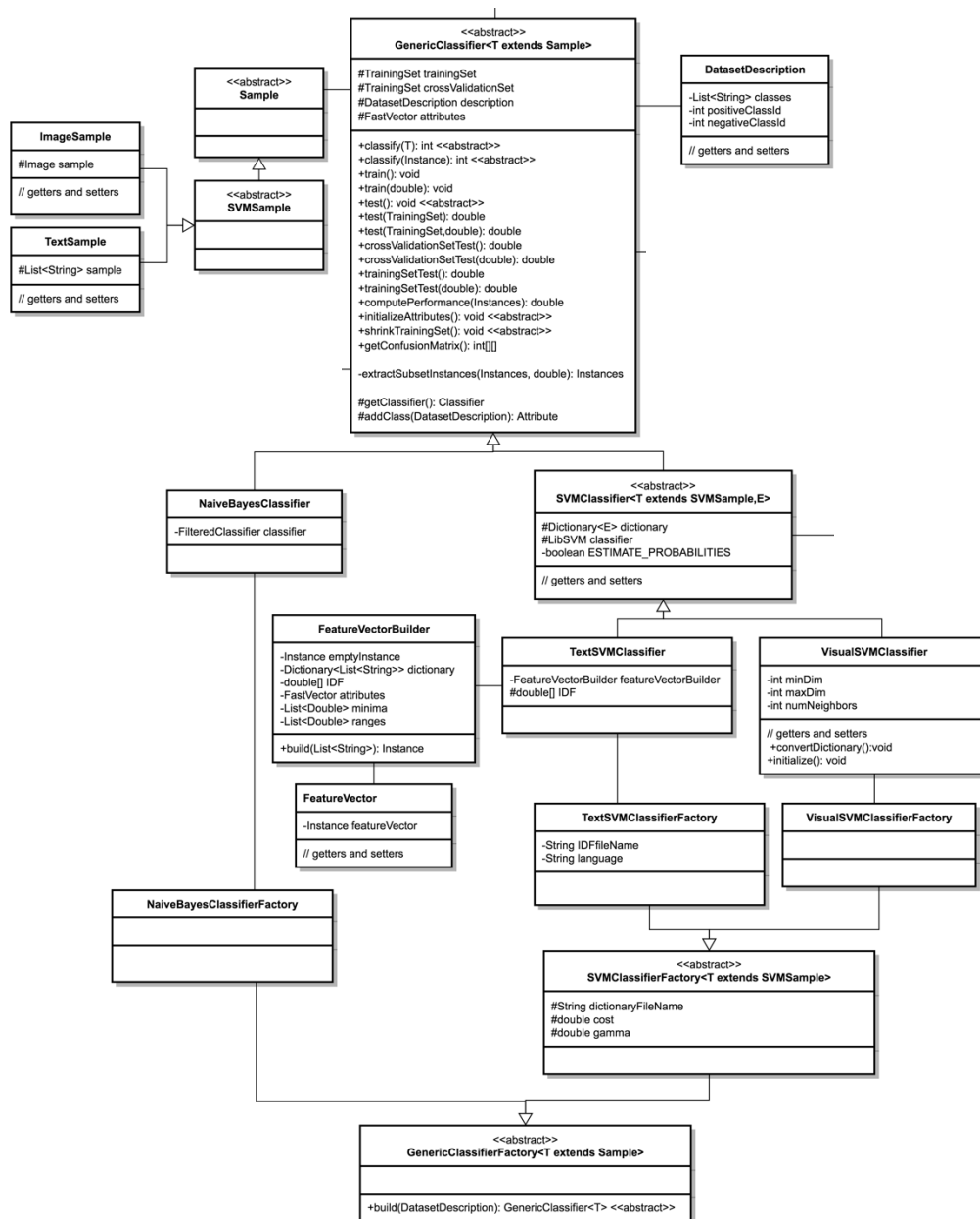


Figure 98 - Classifier hierarchy.

The system offers several classifiers to assess the topical relevance of posts:

- **Textual classifiers**
 - *Naïve Bayes classifier*
 - *Textual SVM classifier*
- **Visual classifier**
 - *Image SVM classifier*

The classifier aggregator component

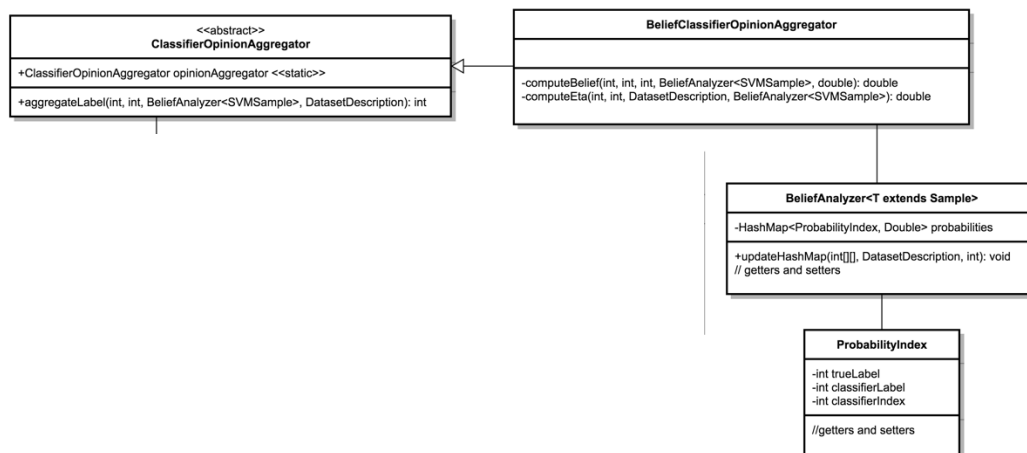


Figure 99 - Classifier aggregator.

The opinions of the applied classifiers (e.g., textual SVM classifier and visual SVM classifier to assess the relevance of, respectively, the textual and multimedia component of a post) are aggregated via the Classifier Opinion Aggregator component.

Currently, we have just one concrete aggregator implementation, which uses Bayes' belief functions to aggregate the hard labels produced by the classifiers.

The classification results

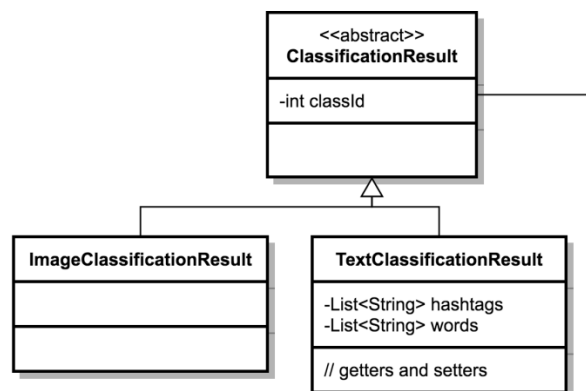


Figure 100 - Classification results.

The Classification Result component represents the label a classifier gives to a post.

The data corpus

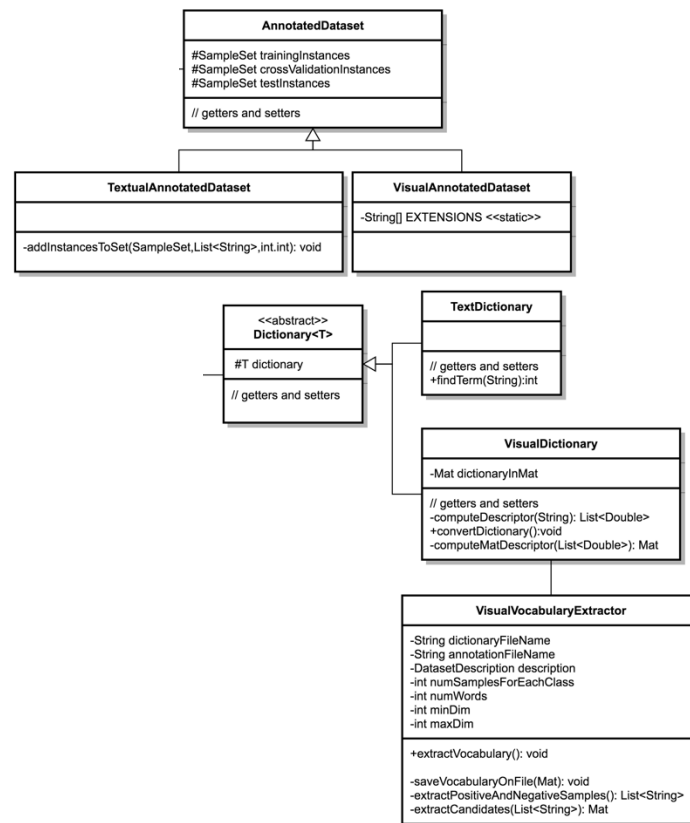


Figure 101 – Annotated dataset.

The Annotated Dataset component represents the corpus of manually annotated posts, which will be transformed into feature vectors and used to train the classifier(s).

The Dictionary component represents the dictionary (either textual in the form of a set of words, or visual in the form of a set of visual words) on which feature vectors will be computed.

The training sets

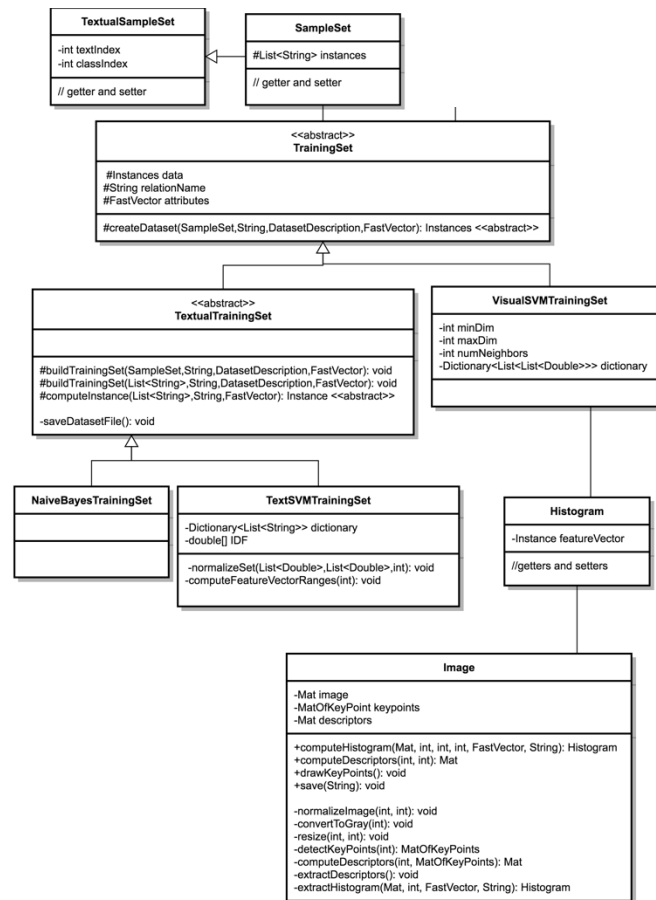


Figure 102 - Training dataset.

The Training Set represents a set of labeled feature vectors used to train the selected classifier(s). It can be either textual or visual, to train, respectively, textual and visual classifiers.

12.2.2 Information with data distribution overlay and physical volumetric overlay

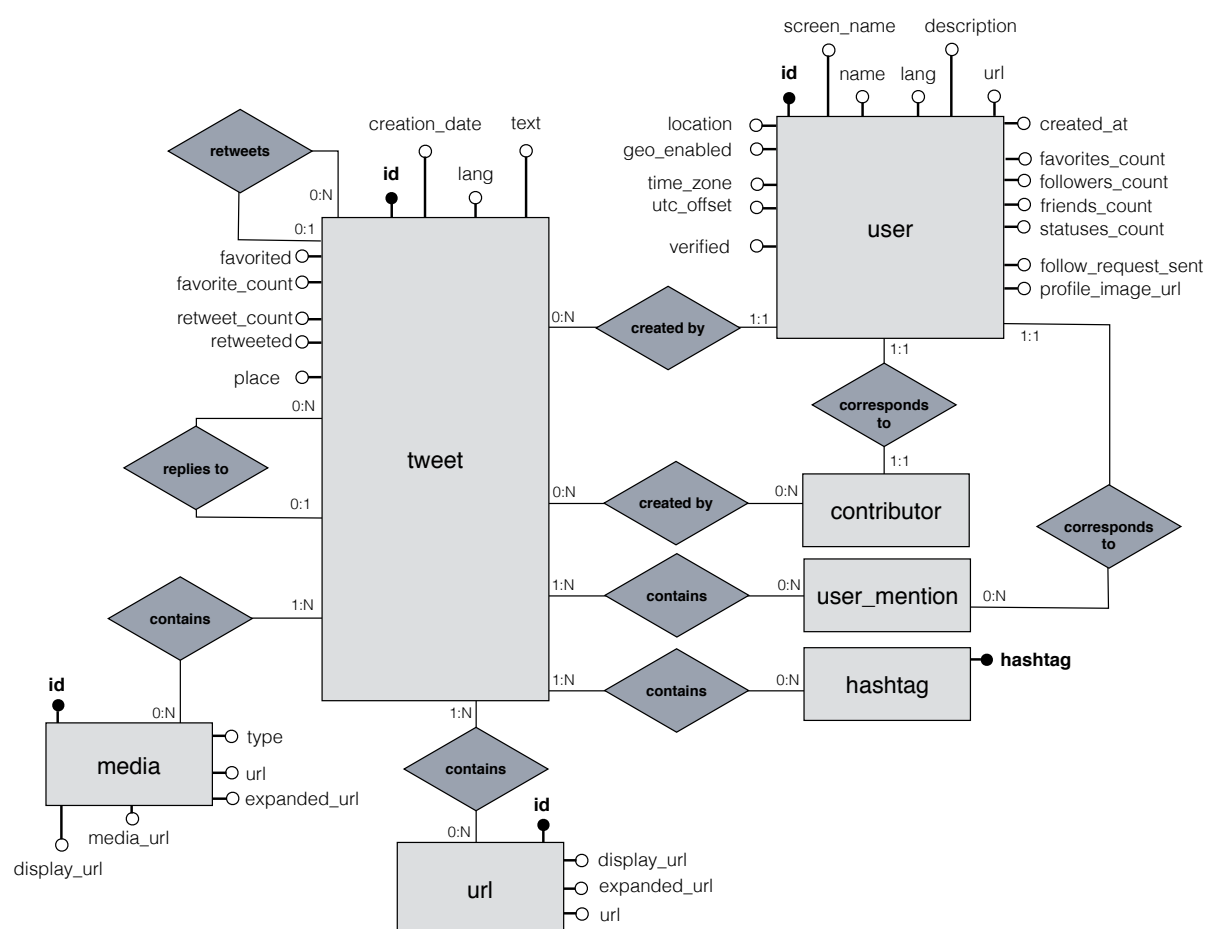


Figure 103 – ER data model of the media analyser.

The data model used by the system is similar to the one proposed by the Twitter platform to represent tweets, users and extra content (e.g., URLs, media, hashtags). All tweets are represented in JSON format.

13. Business Dashboard

13.1 Context

This set of use cases describes the interaction of utility staff with the customer consumption monitor, including customers monitoring and user groups management.

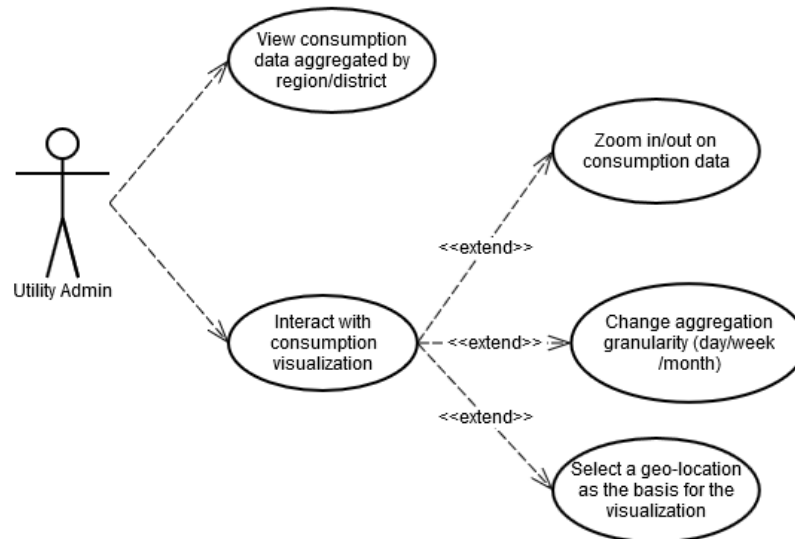


Figure 104 - USE CASE 14.1: Visualizing aggregate household consumption information by geo-location.

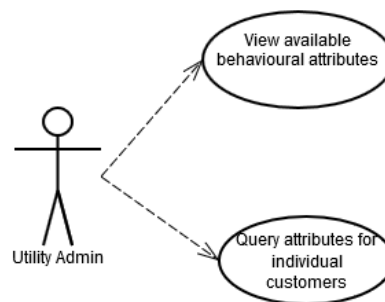


Figure 105 - USE CASE 14.2: Querying and displaying customer attributes.

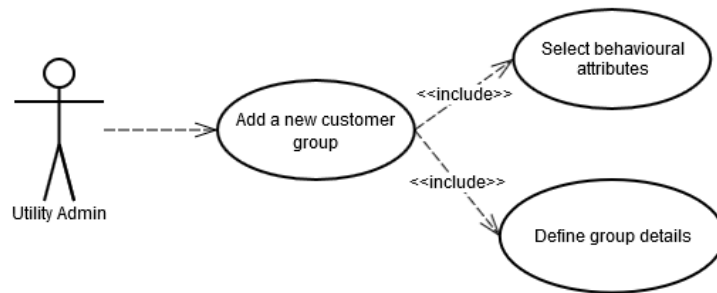


Figure 106 - USE CASE 14.3: Identifying customer groups.

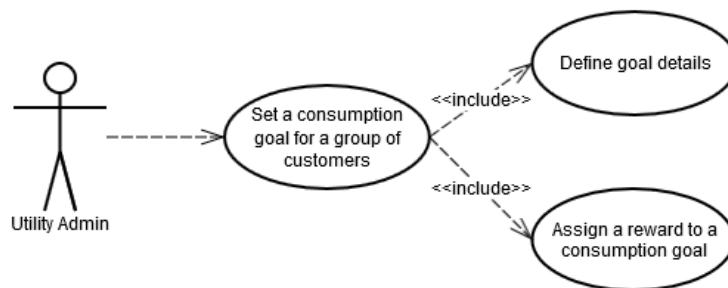


Figure 107 - USE CASE 14.4: Setting consumption goals and rewards for specific customer groups.

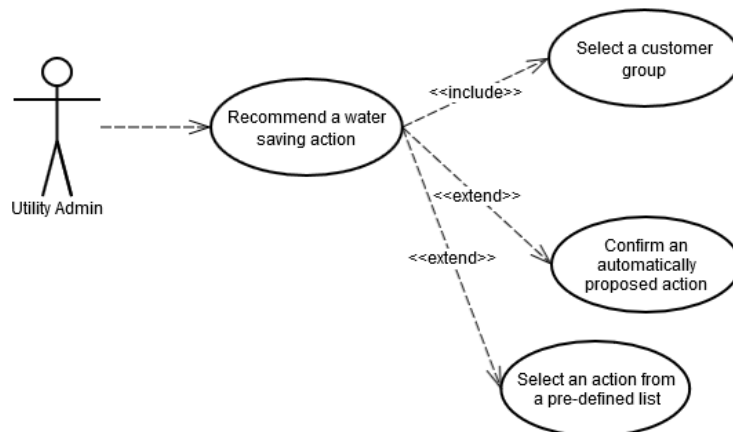


Figure 108 - USE CASE 14.5: Setting recommended water saving actions for specific customer groups.

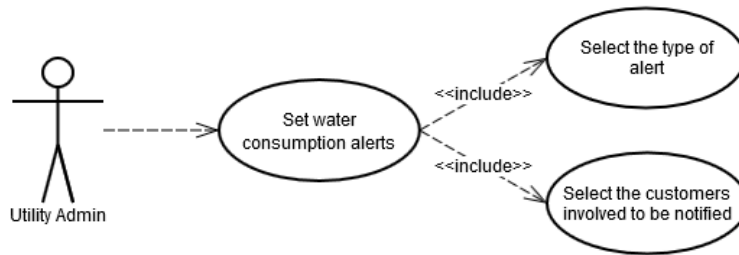


Figure 109 - USE CASE 14.6: Setting water consumption alerts.

13.2 Composition and structure

According to the IFML terminology, the Business Dashboard represents a Site View, accessible to all the customers belonging to the group “UtilityAdmin”.

13.2.1 Summary Sections

In this section a description of the main sections of the site view is presented. Figure 110 represents a graphical overview of the Administration site view.

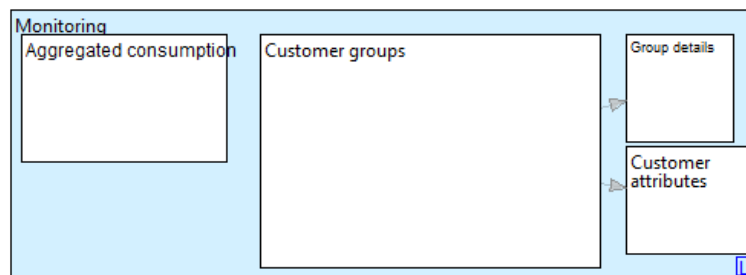


Figure 110 - Business Dashboard Site View – Overview.

Area Summary

Monitoring

This area is related to customers monitoring. In this area it is possible to monitor consumption information, identify and computer users attributes, segment customers based on these attributes and set targeted incentives to the different customer groups/segments.

Page Summary

Aggregated consumption

In this page it is possible to monitor consumption data aggregated by geographical region/district or household.

Customer groups

In this page it is possible to group and display information about customers according to their activity on the customer portal, on their consumption and on psychographic variables.

Group details

In this page the details for the selected customers group are provided.

Group details

In this page the attributes associated to the selected customer are provided.

13.3 Interaction, State dynamics

13.3.1 Use cases

USE CASE 14.1: Visualizing aggregate household consumption information by geo-location

Water utility staff is able to monitor aggregated consumption in the Aggregated consumption page.

Figure 111 represents the structure of the page in terms of components.

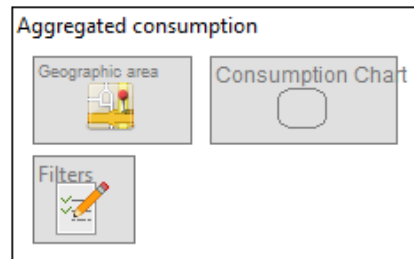


Figure 111 - Aggregated consumption page – Overview.

Context: Aggregated consumption [Page]

Components:

 [GoogleMapUnit] Geographic area

This component shows geographic regions/districts.

 [NoOpContentUnit] Consumption Chart

This component visualizes the aggregated metered water consumption data of a specific geographic region / district.

 [EntryUnit] Filters

This component allows the user to filter consumption data by choosing different zoom levels and to compare average to a set of available other aggregate averages.

USE CASE 14.2: Querying and displaying customer attributes

Staff users can query individual attributes/variables of customers in the Customer attributes page. Attributes can be automatically collected based on customer interaction with the platform or on psychographic variables from customer portal household profiles. Figure 112 represents the structure of the page in terms of links and components.

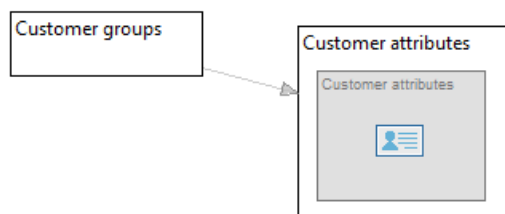


Figure 112 - Customer attributes page – Overview.

Context: Customer attributes [Page]

Components:

 [DataUnit] Customer attributes

This component shows the attributes for the selected user.

USE CASE 14.3: Identifying customer groups

The Utility Admin user can identify and save a specific customer group in the Customer groups page. Figure 113 represents the structure of the page in terms of links and components.

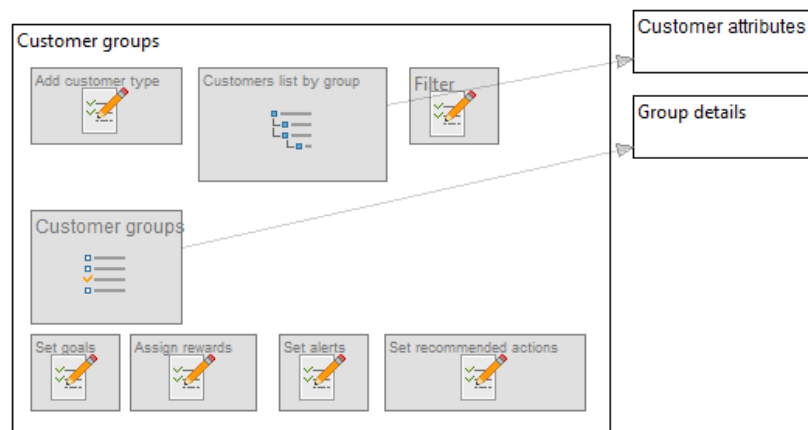


Figure 113 - Customer groups page – Overview.

Context: Customer groups [Page]

Components:

 [EntryUnit] Add customer type

This component allows the user to add a new customer group by selecting a subset of attributes.

 [HierarchicalIndexUnit] Customers list by group

This component shows the existing customer groups. It lists users organized by groups.

 [EntryUnit] Filter

This component allows to filter groups (e.g. by name).

USE CASE 14.4: Setting consumption goals and rewards for specific customer groups

Context: Customer groups [Page]

Components:

 [EntryUnit] Set goals

This is a form component set consumption goals for customer groups.

 [EntryUnit] Assign rewards

This is a form component to link a specific reward to a consumption goal.

USE CASE 14.5: Setting recommended water saving actions for specific customer groups

Context: Customer groups [Page]

Components:

 [EntryUnit] Set recommended actions

This is a form component to assign recommended water saving actions for a specific customer group.

 [MultiChoiceIndexUnit] Customer groups

This is a multichoice list showing all the available customer groups that can be selected.

USE CASE 14.6: Setting water consumption alerts

Context: Customer groups [Page]

Components:

 [EntryUnit] Set alerts

This is a form component to select a set of customers involved in a critical condition to be notified.

14. Pricing Engine

This component will be implemented as an service interface that provides PriceInfo based on different implementation of price schemes: block prices, dynamics pricing.

Actual pricing scheme implementation will be done on a project basis outside the scope of the current project, at Water Utility request.

The Platform will implement only the interface of this component as a generic provider of price information.

14.1 Structure

The structure of this component is presented in Figure 114.

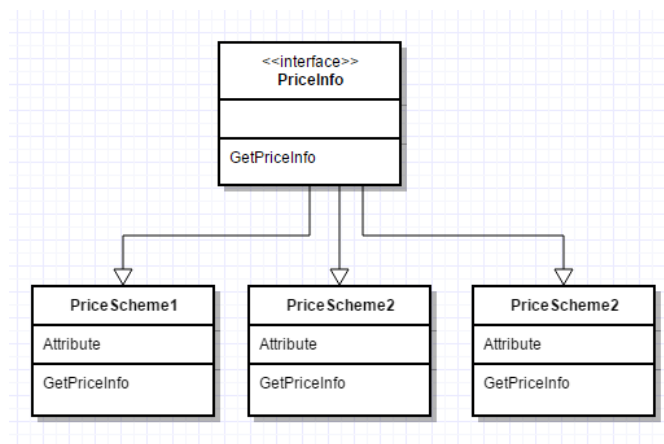


Figure 114 - Pricing Engine Class Structure.

The price interface will be implemented by various possible price schemes at request of Water Utility.

15. Agent-Based Consumption Simulator

15.1 Consumption simulation based on past information

15.1.1 Context

The ABM simulation based on past information allows us to understand how the model behaves while the household agents are set up with historical data statistics. Every household produces a water demand that is dependent on its past consumption and its meteorological sensibility.

The simulation is run by proprietary software, Anylogic, which provides agent components. Every agent is driven by the Anylogic framework, and is capable to exchange messages with other agents, to read data from external sources like a database. Every message exchanged can transport every kind of information like a number, a message string, a reference to another agent, a generic framework object.

The UML chart below describes the use case of the model: the researcher reads the smart meters readings produced by the SmartH2O DB, then he uses data mining techniques to produce the households attributes to be used by the simulator. The attributes are read by the Main model environment actor in order to populate the model with households instances. It's responsible to manage the internal model time and shows the model statistics of water consumption. The water supplier actor sends to all households a message with the Consumption request actor. Each household actor computes its water demand and publishes it, to put the Main model actor able to read it and to compute the consumption statistics.

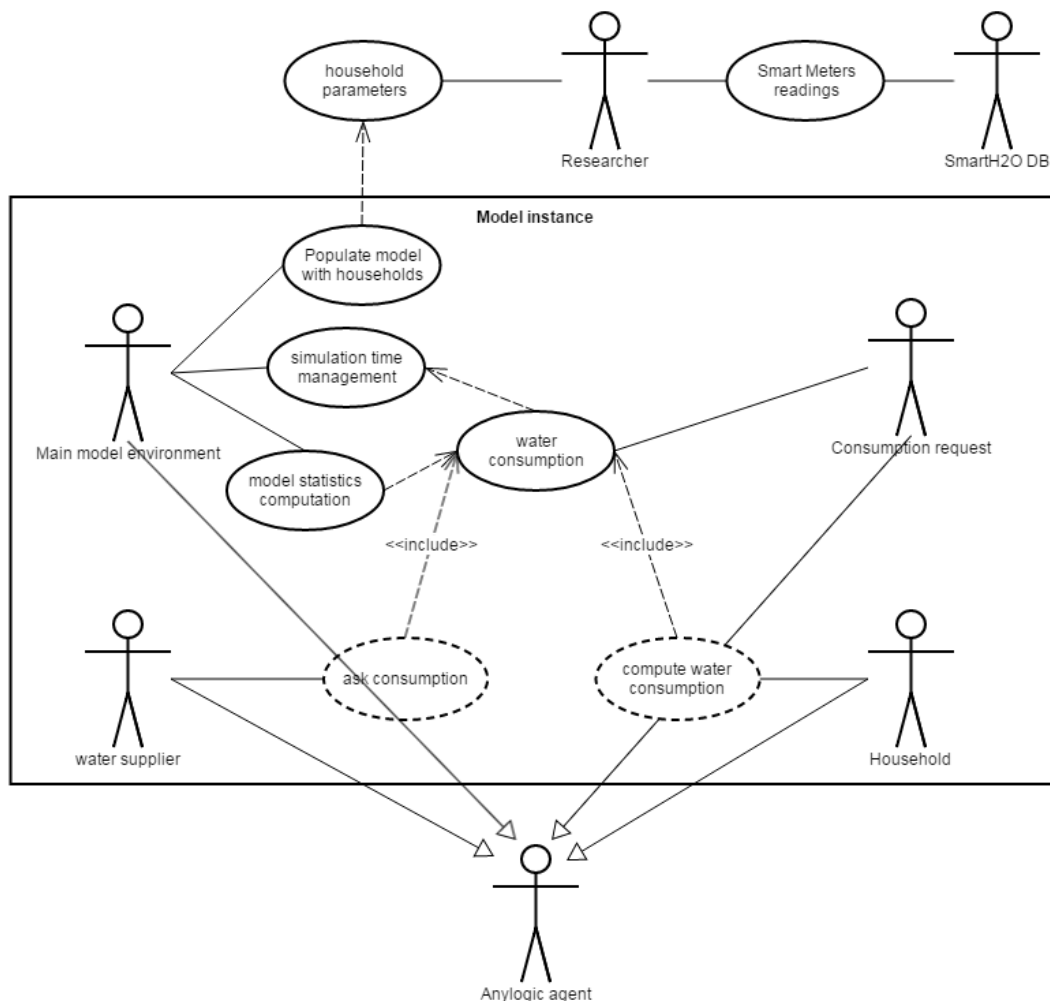


Figure 115 - ABM Use Cases.

15.1.2 Composition

In order to run the ABM simulation model, the operating system must provide the Java Runtime Edition and a graphical 3D chipset, required to render the 3D simulation view through OpenGL API. A file system access to the ground map pictures and the statistics flat file (CSV) are also needed. The Anylogic framework is embedded into the SmarH2O model.

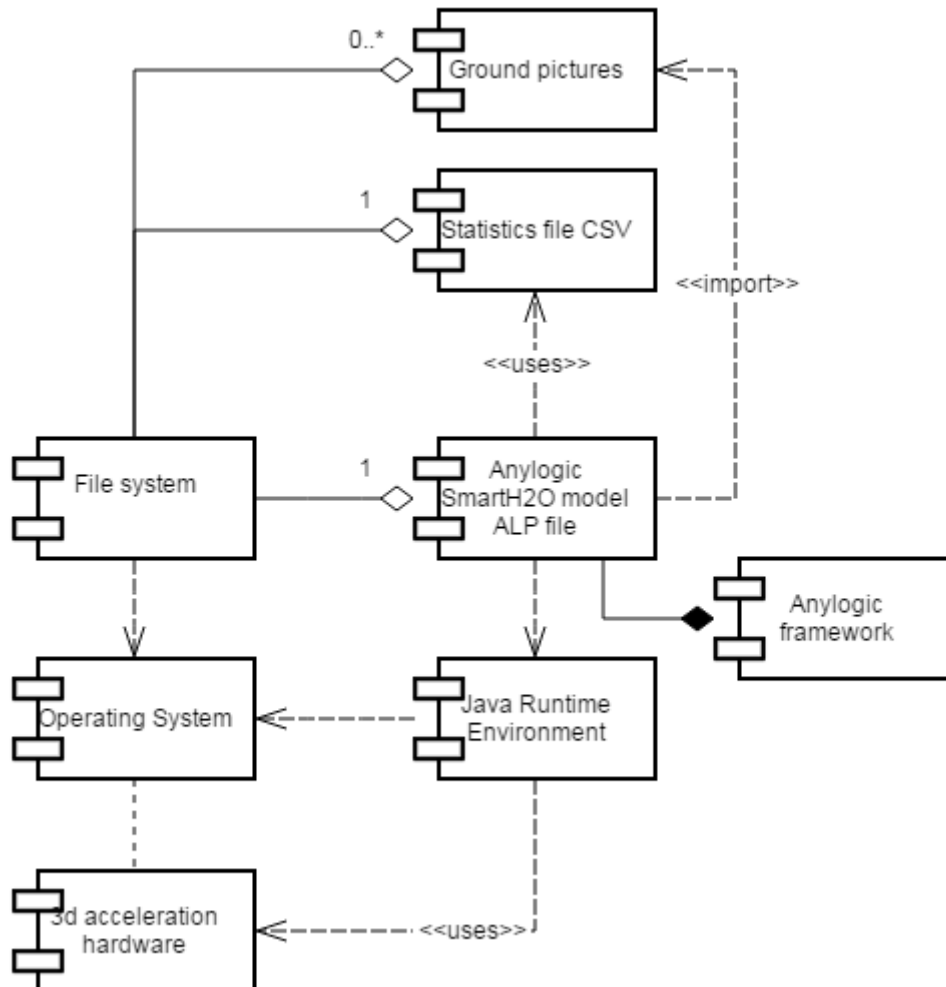


Figure 116 - ABM Component Diagram.

15.1.3 Logical

The UML class diagram shows that the Model simulation object, in its life owns all the others agents instances. The Model simulation will instance one single Main agent. The Main agent will instance all the Households described by the historical data, and one instance only, respectively, of Supplier and Consumption Request agent.

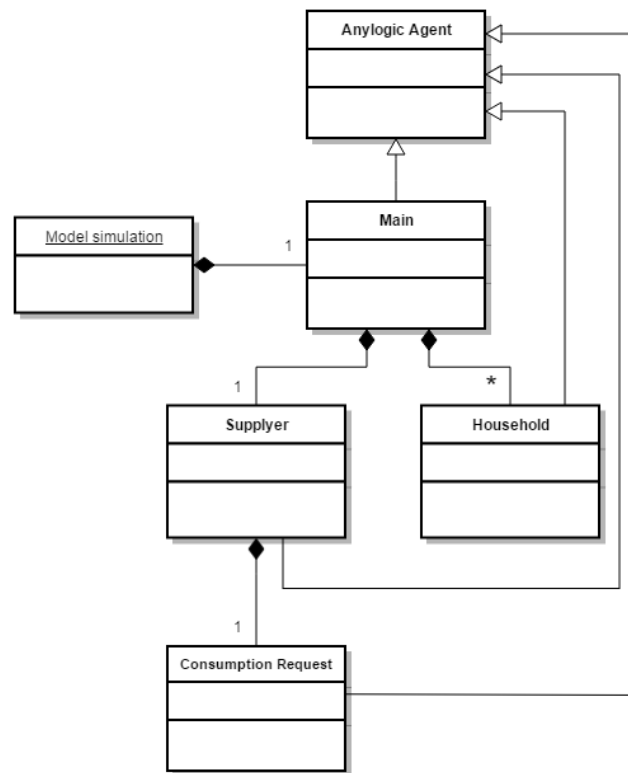


Figure 117 - ABM Class Diagram.

15.1.4 Dependency

There is no need to describe this viewpoint, the most important dependencies has been described in the Composition and Logical views

15.1.5 Information

The simulation will not produce any kind of data persistence that need to be described in this viewpoint.

15.1.6 Interface

The UML chart below shows up how the interfaces of the simulation entities are connected. The Main actor's interface exposes the sums of consumption that are accessed by the chart summary component, responsible to provide graphical statistics of the whole population. When The Household interface receives a Message object in the form of Consumption request object, it calculates its water consumption. The interface of Household is also accessed by the Main agent in order to update its interface for the charting component.

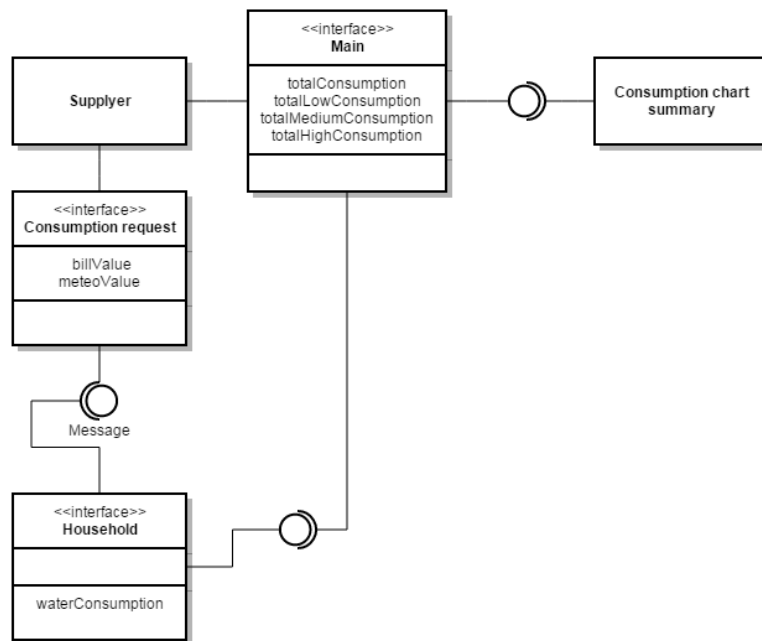


Figure 118 - ABM Interface Diagram.

15.1.7 Structure

There are not relevant structure constituents that needs to be described

15.1.8 Interaction

The UML chart below captures the essentials of interaction between the simulation entities, during a single tick (one day) of the time model simulation: firstly, the main agent send an asynchronous message carrying the time tick to all other entities of the scenario. The Supplier has an inner cycle that recomputes the water price and meteo statistics, and send them through the Consumption request instance to all the households asynchronously. The Household, when receive the Consumption request agent, recompute its statistics reading its attributes. Finally the main agent, in order to compute the model water consumption, synchronously reads the consumption statistics of every Household entity.

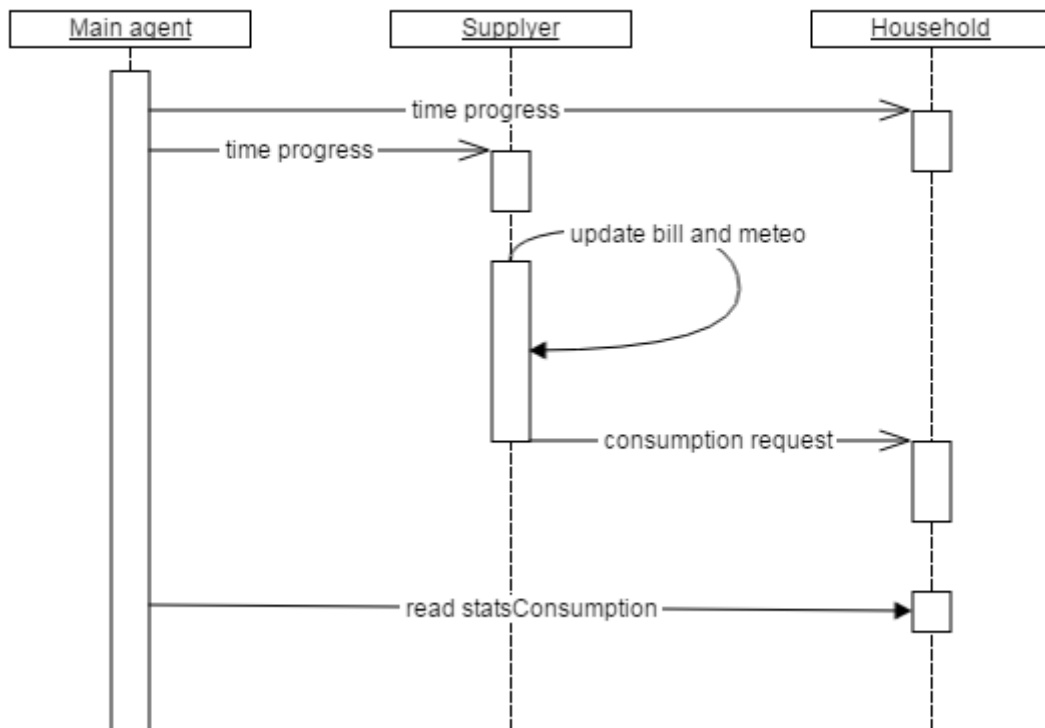


Figure 119 - ABM Sequence Diagram.

15.1.9 State Dynamics

The variation state of Household agent updates its consumption statistics. It occurs when the Supplier send the consumption request instance to it via the anylogic messaging system. Furthermore, after computed the consumption statistics, the Household refresh its graphical components shown by the User Interface of the model.

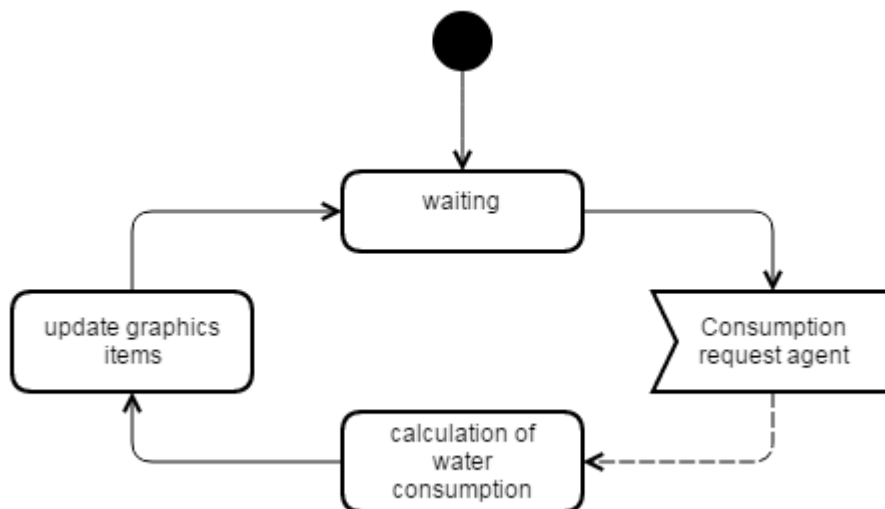


Figure 120 - ABM State Diagram.

15.1.10 Algorithms

The algorithm used by the Household to compute its water consumption is based on the classification of the day based on the temperature and precipitation. The day get classified in a cluster, for that

cluster every household has its own empiric probability distribution of water consumption. A random value of that distribution is returned by the algorithm.

Cluster = getClusterClassification(**day**, **temperature**, **precipitation**)

EmpiricDistributionConsumption = getEmpiricDistributionConsumptionForCluster(**Cluster**)

RETURN random value from the **EmpiricDistributionConsumption**

15.2 Consumption simulation based on incentive response

The current ABM prototype, submitted in the WP3 D3.3 deliverable, is an input-output model. It aims at determining future water consumption based solely on the specific values associated to the attributes of an agent, its current level of water consumption and the meteorological sensibility. In particular, this means that the model lacks a module to describe social interaction.

The influence on water consumption behaviour by the use of social awareness incentives is planned to be captured by the interaction module of the agent based model that will be developed for the WP3 D3.4 deliverable.

15.3 Consumption simulation based on pricing schemes

The pricing scheme mechanism is already implemented within the Water supplier agent but its influence on the consumption behaviour has yet to be implemented at the Household's agent level. As for the consumption simulation based on incentive response, this question is planned to be tackled with the WP3 D3.4 deliverable.

16. ESB

16.1 Context

The integration component ESB will act as a SOA integration layer for Platform components. A platform component that request data and/or processing from another component will address such a request against the integration component. This component will also provide the requester with the answer. The Use Case diagram of this component is depicted below:

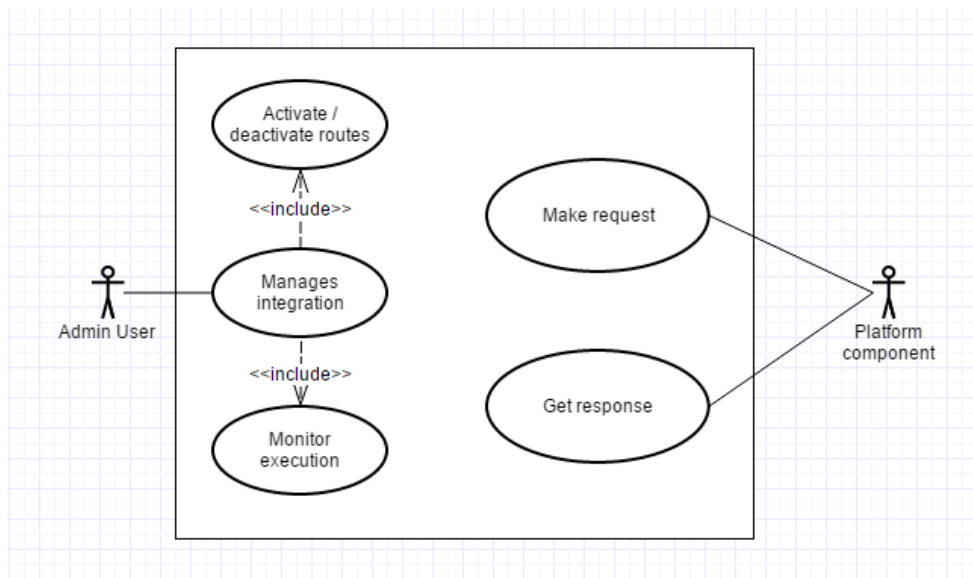


Figure 121 - ESB Use Case Diagram.

16.1.1 Make Request

When a Platform component require data and/or processing from other components it will make this request against the integration component (ESB).

16.1.2 Get Response

The Platform component will receive the response of its request from the ESB after the request was previously processed by ESB.

16.1.3 Administration

The Admin User will be able to manage integration routes in an Administration console.

The Administration console of ESB will allow the Admin User to:

- activate and deactivate integration routes on the server
- deactivate or delete integration routes
- monitor execution

16.2 Composition

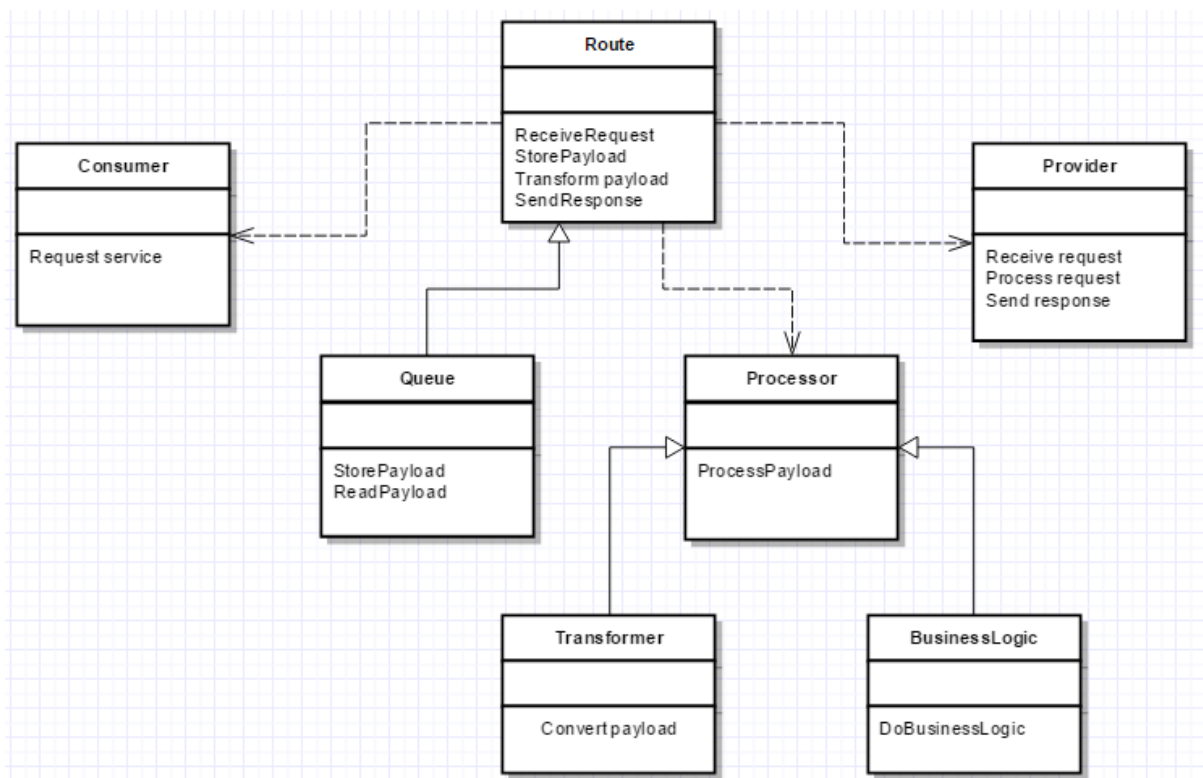


Figure 122 - ESB Class Diagram.

16.2.1 Consumers

Consumers are service requests of a Platform component that need data and/or processing from another component of the Platform.

16.2.2 Providers

Providers are services exposed by Platform components that will respond to Consumer requests. These services will be available to be called by ESB component and their response will be used by ESB to build the response for the Consumer.

16.2.3 Routes

The element that link components will be defined in ESB as a Route. A Route connects one Consumer to one or more Providers and processes the request payload to provide the response.

To accomplish the request of the Consumer, the ESB will perform one or more of the following steps:

- Receives the request and dispatches the request through one if its configured routes
- Store the request in a durable storage until it is processed [Optional]
- Check authentication of the requester and verifies the authorization of the request
- Transform the request payload [Optional]
- Perform certain business logic on the request payload [Optional]
- Consume one or more services published by other components to fulfill the initial request [Optional]
- Send the result to the initial requester

Note: The Platform will implement a SSO (Single Sign-On) mechanism that will allow that a user authenticated in any of the Platform's component to be automatically authenticated in any other

component. A central authentication authority will be implemented for the entire Platform.

Along the route, data can be subject of various operations like

- Queue storage
- Conditional filtering
- Data conversion
- Business logic operations
- Data split for different destinations
- Synchronization with another Publisher

Following integration routes will be configured in ESB component:

Component	Use Case (Consumer)	Publisher	Route description
Basic Customer Portal	Household consumption data	GetConsumptionData GetConsumptionDataHourly GetConsumptionSummary GetThisMonthAverageConsumption GetThisMonthTotalConsumption GetThisWeekAverageConsumption	Consumer request is authenticated and if authentication succeeds request is forwarded to Publisher. Publisher's response is forwarded to Consumer
	Comparison of individual data	GetConsumptionData GetNeighbourhoodConsumption	Consumer request is authenticated and if authentication succeeds request is forwarded to Publisher. Publisher's response is forwarded to Consumer
	Tip of the day	GetTips	Consumer request is authenticated and if authentication succeeds request is forwarded to Publisher. Publisher's response is forwarded to Consumer
	System notifications	GetSystemNotifications	
Business Dashboard	Aggregated consumption data visualization	GetNeighbourhoodConsumption	
	Customer segments		
	Alerts: leakage, shortage, quality		

17. Authentication Gateway

This component will handle single-sign-on authentication between Platform components. When a user authenticates against a Platform component he will be considered authenticated to other components.

The mechanics of authentication will be based on a Central Authentication Authority (CAS) that will handle initial user authentication and subsequent authentication requests across different components. CAS implementation will be done with the help of Apereo Central Authentication Service <https://www.apereo.org/projects/cas>

An authenticated user have a Ticket that can be used in subsequent requests against the component of that initiated the authentication process or other components of the platform.

Each component will embed a Client library that will access Central Authentication Authority to request a new TGT or to validate an existing TGT for authenticating users.

17.1 Context

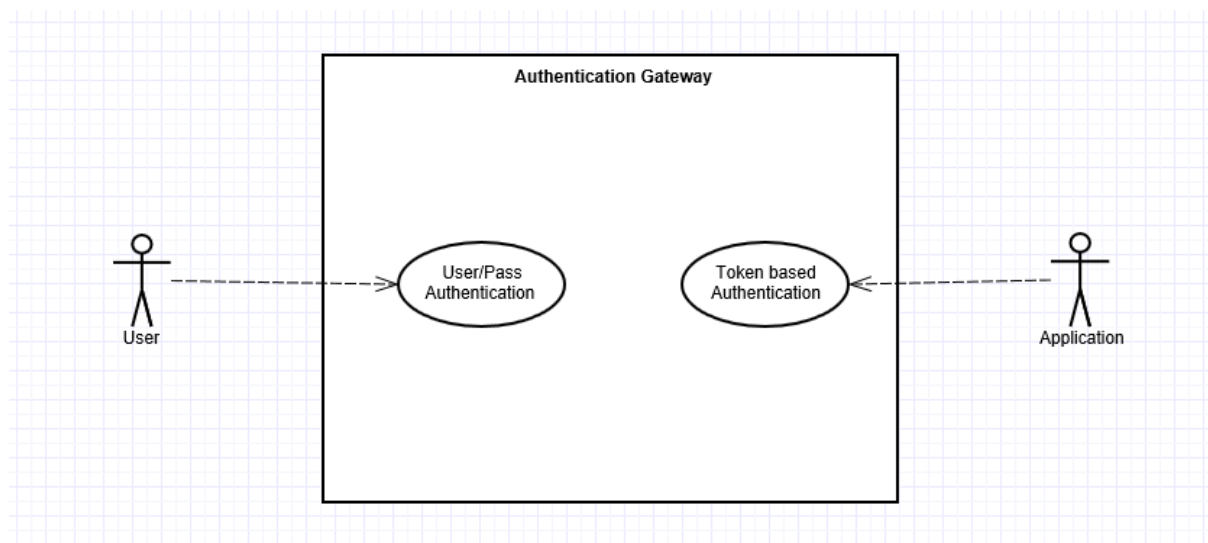


Figure 123 - Authentication Gateway Use Cases.

17.1.1 User/Pass Authentication

When a non-authenticated user (have no TGT or no valid TGT) tries to authenticate against a Platform component, it is prompted the user/pass form of the component. If the component successfully validates user/password against internal repository it then forwards a request to Authentication Gateway for a TGT that will be handed the user.

17.1.2 Token based Authentication

When a user makes request that have a TGT, the component forwards TGT authentication to Authentication Gateway. If TGT is valid then user is authenticated, if not the component will prompt the user the user/password form.

17.2 Interaction

The operation sequence when authenticating with user/pass. This scenario occurs for the first authentication or for authentication where user's TGT is not valid.

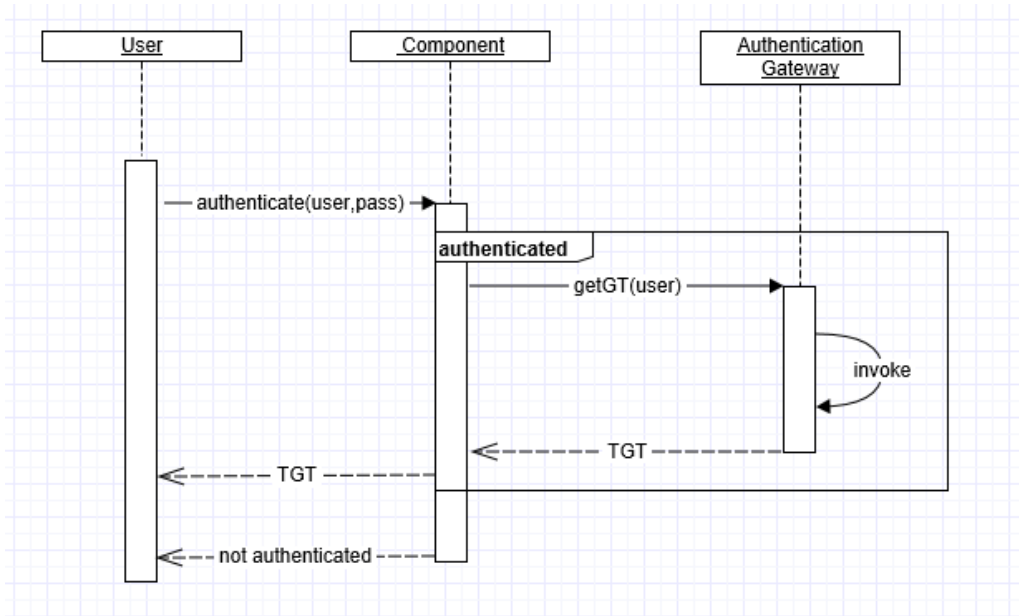


Figure 124 - Authentication Gateway Initial Authentication.

The operation sequence for TGT based authentication is presented bellow:

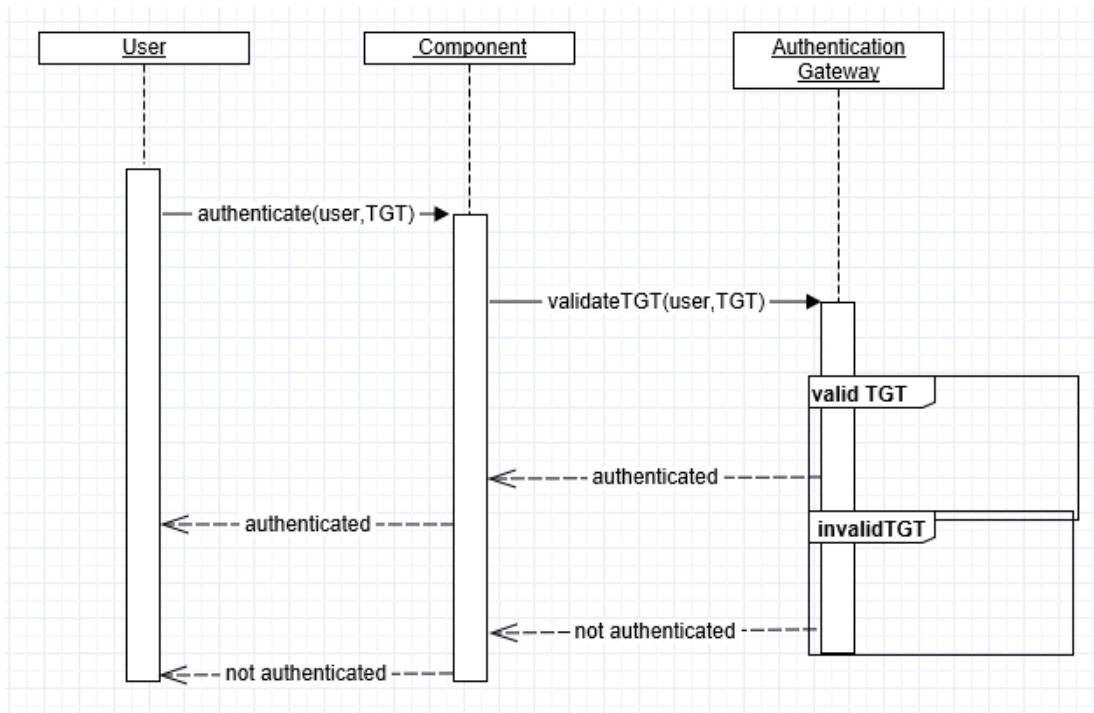


Figure 125 - Authentication Gateway TGT based authentication.

18. Design Rationale

Many of the design decisions were taken based on best industry practices, open source requirements of the project, latest technologies and also based on economical and viability criteria.

The Platform design was decided to be Service Oriented, even from the initial statement of project scope and objectives, with its components acting as Service Consumers and/or Service Providers.

Integration among various components will be done through the ESB component that is detailed at functional level in this document. Another general use component was the Authentication Gateway that provides Platform wide authentication services.

So the SOA design pattern respect the “loose couple” principle but also the components are glued together using specially designed integration components.

The distributed processing architecture using Apache Hadoop [HADOOP] environment for raw metered consumption data was decided to have a scalable and high-availability processing power able to handle large amounts data.

The user-interaction based components like Basic Portal, Advanced Portal, Business Dashboard will be implemented using a model-driven development platform that ensures high-fidelity tracking of User Requirements, rapid and consistent change process. The platform used will be WebRatio [WEBRATIO] backed by the theoretical sound foundation of IFML [IFML] standard.

This design decision will simplify maintenance of the portals, which are expected to be heavily customized in each site deployment.

Also, thanks to code generation, it will support an easy path to the implementation of the mobile version of the portal, because it will be possible to modify the IFML specification and then use an alternative code generator to produce the mobile code.

The ESB component will be implemented using JBoss Fuse [FUSE] integration framework that provides OpenSource tools and frameworks specialized for implementation and maintenance of integration scenarios.

The Authentication Gateway will be implemented using Apereo Central Authentication Service [APCAS] a Single Sign On service initially developed by Yale University. This was the most suited OpenSource solution for SSO requirements of SmarH2O Platform in respect to viability, usability and required features.

19. Conclusions

The document reveals the heterogeneous nature of the Platform and as a consequence the need to have a strong integration architecture and design. The Platform respect a SOA architecture ensuring both “loose couple” principles but also offering strong integration at the interface of components.

The Functional Specifications of the Platform stay close to User Requirements detailed in *D2.2 – Final Requirements* and a verification mapping is provided in Appendix B of current document. The user requirements are mapped more explicitly at components that have user interaction and less intuitive at components that act like services. For example ESB and Authentication Gateway components were not explicitly stated in user requirements but they derived rather from the integration requirements of the Platform.

Some Platform components like ABM and Social Network Crawler run as standalone systems but they serve to project’s objectives stated in the DoW.

The detail level of the document allows Consortium members to have a full image of the Platform functional specification but also implementation details for each component.

Functional specifications this document and screen mockups from *D2.2 – Final Requirements* provide enough information to implement most of Platform components that interact with Platform users. For components that are rather services, UI design will remain at the decision of implementing team as long they were not specified in user requirements.

20. References

- [IEEE 1016-2009] 1016-2009 IEEE Standard for Information Technology--Systems Design--Software Design Descriptions, July 20 2009, E-ISBN 978-0-7381-6900-2, <http://ieeexplore.ieee.org/servlet/opac?punumber=5981337>
- [UML] Unified Modeling Language, <http://www.omg.org/spec/UML/>
- [IFML] Interaction Flow Modeling Language, <http://www.omg.org/spec/IFML/>
- [HADOOP] Apache Hadoop distributed computing, <https://hadoop.apache.org/>
- [FUSE] JBoss Fuse integration platform, <http://www.jboss.org/products/fuse/overview/>
- [APCAS] Apereo Central Authentication Service <https://www.apereo.org/projects/cas>

21. Appendix A: Valencia EMIVASA Case Study

21.1 Valencia case study requirements elicitation

21.1.1 Outline of the approach

In this chapter, we provide an update of the requirements of the SmartH2O portal for the new case study in Valencia, as they could not yet be foreseen and considered in the previous deliverables.

The requirements were elicited in a workshop and other interactions with EMIVASA and UPV, to understand both the customers' and the utility's perspectives. As a consequence, we have refined the personas and main user stories to fit the new case study context. The requirements and use cases were adapted accordingly (for use case updates, see the following chapters), but the main new requirements regard the integration of the SmartH2O portal and the existing system from EMIVASA.

A separate workshop with customers was not undertaken as EMIVASA already has a thorough understanding of their customers and their needs, especially since they have had a virtual office in place for some time. We also don't expect the general requirements towards the system from the customer side to vary this largely from those in Switzerland and the UK. Instead, the main challenge was the integration of the SmartH2O portal or its elements into the existing system of EMIVASA. Therefore, our requirements analysis focused on the interaction with EMIVASA and their requirements towards the system, as well as insights they already know about their customers' consumption habits.

21.1.2 Requirements workshop with EMIVASA / Aguás de Valencia

Most of the key requirements for the new case study in Valencia have been elicited in a 2-day requirements workshop and meeting at the EMIVASA premises in the beginning of May. To ensure a comparable methodology with the previous requirements workshops and interviews with TWUL and SES, the same basic questions were used to guide the semi-structured interview in a similar way.

The company EMIVASA belongs to the larger utility group Aguás de Valencia and is responsible for managing the water supply for the city of Valencia (not outskirts). They are partially owned by the city council of Valencia (20% ownership) and because of this, they have to comply with certain city regulations. E.g., they are obliged to offer services in both local languages, Valencian (a variety of Catalan) and Spanish.

EMIVASA currently pursues three main goals: 1. Leak detection, 2. Provide information to consumers, 3. Improve service. They have been metering households with smart meters for the past 9 years, incrementally increasing their metered customer base. By the end of 2015, they plan to have 650.000 smart meters installed.

For the past 7 years, they have had an online customer portal in place, the "Virtual Office". It provides customers access to their bills online, but also shows their consumption over time, and includes multiple other services. Section 2.2.2 outlines the main features of the Virtual Office.

So far, no study on the impact of the smart metering technology or the virtual office has been conducted, which is one of the reasons EMIVASA is interested in the SmartH2O project and the accompanying validation planned.

There are two main challenges in Valencia regarding the water supply: the water is very hard with high mineral content, and it is scarce especially during the summer months. It rains only occasionally, and there are only two water sources, which is insufficient: The Tuka river, which has a small flow, and a reservoir, which suffers from large quality fluctuation. Especially at the end of the summer, the reservoir quality can be low, as the water level needs to be kept low to anticipate heavy rain in the fall.

And, while the water in Valencia is considered hard, the daily water quality in terms of hazardous residues, which could affect the drinking water quality, should always be good and stable, according to EMIVASA. Therefore, a water quality alert is not needed.

The customer and household types supplied by EMIVASA vary, especially the old town of Valencia vs. the surrounding neighbourhoods. The old town mostly consists of blocks of houses, but there are also neighbourhoods where e.g. single beach houses dominate the cityscape. The sample that will be

selected for the Smarth2O case study will purposely consist of a mix of different households, considering different areas and scenarios (e.g. single vs. family household).

EMIVASA customers are currently billed every two months based on one reading of the 2-month period. Customers can choose between the option of a paper-based or an online bill that they can access through the Virtual Office web application. The bill is based on the metered consumption, not a fixed price as is the case e.g. for most customers of TWUL. The bill consists of a fixed rate for network connection plus a variable amount based on the customers' actual consumption.

Smart Metering

Aguas de Valencia started to install smart meters 9 years ago, and by the end of 2015, 650'000 smart meters are supposed to be in place. Claiming to be the "most advanced company in Europe in terms of smart metering", their main objectives for applying smart metering are more accurate water balances and faster leak detection. They are using seven different smart metering technologies from different vendors, which are integrated using a common data format and platform.

Not all smart meters used provide hourly readings, but for the project they will ensure hourly samples are available for all participating households. Hourly samples are the only feasible smallest unit as battery life is affected by the reading frequency. In addition, the data samples are transmitted with 1 or 2 days delay. The data is already integrated in network management, enabling almost automatic leakage detection.

EMIVASA Virtual office

The Virtual Office is the online customer service portal of the Aguas de Valencia's group. It currently counts around 60 000 users. Customers can register as users with any official document.

One customer can also manage multiple household contracts and water suppliers in the Aguas de Valencia group under one user account, e.g. if they own multiple houses or apartments.

In the virtual office, customers can manage their online invoices, and view their consumption of the past three years if available (see Figure 126). The look and feel of the virtual office will be updated in the next few months to be more along the lines of their mobile app.

In the Virtual office access, consumption info is available to customers for the last 4 years, showing e.g. the total billed bi-monthly consumption and daily averages. Currently, they are implementing a means to also show hourly consumption in the virtual office.

Manual and semi-automatic input of meter readings in the case that the user has no smart meter yet are also possible. For the semi-automatic input, users can take a photo of their meter with the mobile virtual office app to read the current number automatically. In case the detected number is abnormal (e.g. not the same as official reading), a customer service loop is activated, where the customer is contacted by the utility to clarify the abnormality.

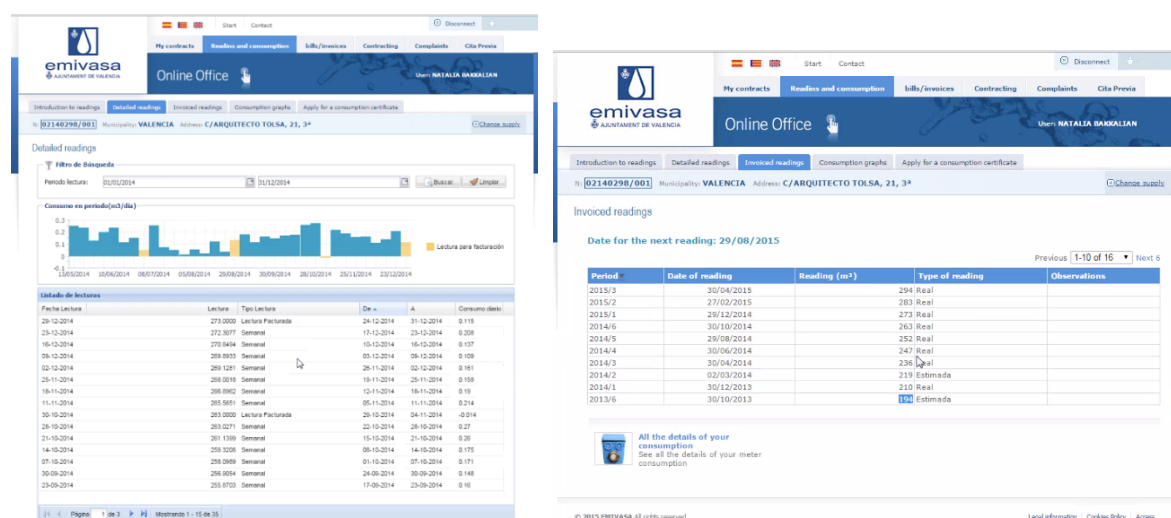


Figure 126 - Screenshots of the Virtual Office: Detailed consumption readings with bar chart.

visualization (l) and invoiced readings (r)

The virtual office is also accessible through a mobile app available on Google Play: “Aguás movil clientes”. At the time of the meeting in May, it had only been available for 1 week, and had 100 downloads (update July: play store downloads “100 – 500”). The numbers are expected to go up in the next months, with a total potential user base of around half a million contracts. The app is available in multiple languages: Spanish, English, Valencian Catalán, and general Catalán.

It features the following main functionalities:

- Consumption tips (18 general consumption tips available, of which 6 are displayed per login)
- Notification services
 - Abnormal consumption patterns
 - Invoice available (downloadable)
 - Cut offs in system (no access to service)
 - Programmed actions / accidents
- Access to the Virtual office (includes past bills, consumption info)
- Manual input of meter readings
- Leakage report to utility
- Opt-out of paper bill (all customers have access to the online bill)
- News and information about the utility

SmarrH2O customer portal

With their virtual office, EMIVASA has essentially developed an application similar to the basic customer portal, in web and mobile version. Therefore, the SH2O basic customer portal version is not needed by EMIVASA, since it duplicates the functions of the already existing portal. To integrate SmarrH2O and existing EMIVASA services based on this situation, only the gamification engine (GE) will be added to the current EMIVASA web and mobile virtual office.

For this, an extra tab that shows the gamified extension will be added to the EMIVASA virtual office and a call to the backend with the Gamification Engine services.

EMIVASA will locally install the GE backend and the two portals will run on the same domain. They will need an application to edit rules (actions, badges and rewards) to manage the GE on their own.

For the integration, a phased approach will be taken, starting with simple rewards for basic actions (e.g., logging in, downloading the mobile app) and then incrementing the monitored and rewarded actions progressively). The alpha test trial can start from the end of September, and selected customers will be notified using e.g. the ADV call centre.

The integration of Drop! is programmed for phase two of the trial. But because SH2O needs to start the production of Drop! for the Suiss case study, ADV has to provide the instructions translation soon in Spanish and Valencià.

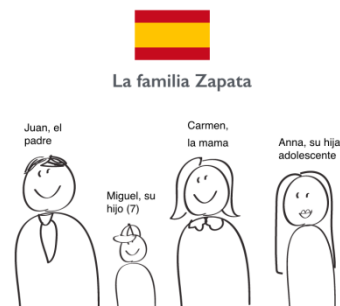
EMIVASA also voiced their concern that they are worried about customer feeling of “intrusion”. So, to counteract their anxiety, the SmarrH2O project is planning to test all the features and get feedback from users to tune the current application, starting in Switzerland. There, we already SmarrH2O has already recruited a group of core users with a “green attitude” who are really interested in participating in the first experiments with the SH2O platform.

SmarrH2O business dashboard

EMIVASA states that one access would be enough with which everyone in the company who is interested in using it could sign in. It would probably be most relevant for billing & customer service, especially since the network monitoring features are already in place in their existing applications. – Thus, the part of the SH2O Business dashboard for water consumption monitoring is not necessary for EMIVASA. They also already have some parts of the user modelling and consumption analysis algorithms at work (e.g., pattern detection).

21.1.3 Adapted personas and user stories

Consumer personas - Valencia



The Zapatas are a family of four living in a 4- bedroom flat in the centre of Valencia. Mr and Mrs Zapata both teach at a local school. They have a 15-year-old daughter, Anna. Anna likes to meet her friends at the beach and is always occupied with her smart phone. Miguel has a lot of friends who he plays with frequently after school.

During the summer months, the family often escapes the heat and city life and stays at their second home 1 ½ h north of Valencia.

They have had a smart meter installed for some years now in both their flat and summer home. They appreciate that their local utility has provided them with online access to their water bills, which helps them to remotely keep track of the cost and possible leakage of both properties. However, especially during the hot summer months, they have been having difficulties to convince their two children to save water.

Recently, they were called by a call centre about a new project of the utility: an extension of their existing online access, which would reward their water saving efforts in a playful way and engage all family members.

Water utility personas - Valencia

Sara is the customer service manager at a large water utility in Valencia. She is often planning large awareness campaigns to promote water saving but is looking for more effective solutions that can target different customer types with different kind of water saving incentives. She is open to experiment with different approaches, like providing vouchers for active savers. She is curious towards more innovative pricing schemes but would like to know more on customers' attitudes and likely behaviour towards such an approach.

User stories - Valencia

The following relevant user stories were adapted to the Valencia case study.

User story 1A: Visual exploration of water consumption information

Goal: Raising individual water consumption awareness by visualizing metered consumption data and providing water saving tips.

Using his desktop computer, Juan logs into the "Virtual office", the application from his local utility. On the virtual office, he sees how much water his family has consumed and is billed for during the past day, week, and month. Looking at the consumption from the last month, he is shocked to see a strong peak marked as a very bad performance. He remembers guiltily that it was a very hot week, which led them to take extra showers during the day and in the evenings. With the virtual office mobile app Juan also gets water saving tips and alerts in case of emergency, like abnormal consumption or cut-offs.

User story 2A: Earning rewards by water saving and information collection

Goal: Gamification of individual water consumption and household data provision through points, goals and rewards to raise individual awareness of water consumption and stimulate water saving

In addition to the existing utility application "Virtual office", customers in Valencia can now activate the gamified portal features from Smarth2O.

When activating the gamified portal, Juan sees his current point score and the badges he achieved on the platform. For key actions on the Virtual office, he can get more points and earn new badges. He can e.g. complete his customer profile (how many family members there are, what kind of appliances they have), and undertake specific water saving actions that are recommended to him by the system. He can also set his own water saving goals for which he earns extra points.

For now, Mr Zapata decides to download their latest bill, which earns him 10 points. When he checks his score again he sees that he only needs 150 more points to redeem a "10% off at the cinema" voucher as a reward and 1350 more points to receive a high quality water saving showerhead.

In addition, Juan and his family opt to share their score with other users. Now they can compare

themselves to the other users on a leaderboard and on the neighbourhood's map.

User story 2D: Saving water by playing augmented card games

Goal: Raising water consumption awareness of younger children to motivate the family as a whole to save water

One of the rewards in the list is the “push your luck” card game **Drop!** that can be played by the whole family. It's all about avoiding the water waster monster with his bad habits! **Drop!** can also be played in combination with a mobile app, where the users can teach the monster how to improve his bad habits and provide additional household information for the utility. When successful, they earn points. Family members can log in with their account to redeem the **points** earned in the mobile game.

As soon as they have earned enough points, the Zapata's redeem them in exchange for a copy of the game. The whole family enjoys playing it, especially during their summer stays at their second home where there is not so much to do during the evenings. Especially the Zapata's son loves the **Drop!** monsters and even shows the game to his friends at school. Soon, they all want a copy, and convince their parents to also signup so they can earn points for their own copies.

21.2 Requirements updates

The following table lists additional functional requirements, including those needed for the Valencia case study, and existing requirements that have been updated (marked with “*”; or “DEL” when obsolete). For the full list of requirements, please refer to D2.2.

	Visual water meter: Basic customer portal
DEL VF5	The application should provide the option to manually insert consumption data.
	Advanced customer portal: Gamified water meter
9.10*	The advanced customer portal should provide the functionalities available in the basic version extended with the gamification features. Alternatively, the gamification engine can be configured as an extension to an external application, e.g. EMIVASA virtual office.
	Customer portal user account management
11.2*	The application should provide the user the option to register to the basic or the advanced version of the Customer Portal according to their own preference, if both application versions are available to them.
11.7*	The application should allow users who are registered to the advanced gamified version of the Customer Portal to opt-out of the gamification features.
11.9	The gamification engine should enable an external application (e.g. EMIVASA virtual office) to authenticate users.
11.10	User actions should be logged to the extent they are needed for the KPI and success criteria validation.
11.11	Sign-up, log-in and gamification opt-in should provide the option to display links to external evaluation questionnaires.
	Customer portal admin
12.4	The application should allow the Admin to monitor user activities on the gamified portal (actions, badges, rewards).
12.5	The gamification engine should provide services to assign gamified actions from an external application (e.g. EMIVASA virtual office), i.e. to gamify and reward points to actions from an external application.
12.6	The gamification engine should be configurable independently for different case studies, i.e. features can be (de)activated e.g. when they are already implemented by the utility's existing application.
	Business dashboard: customer consumption monitor
14.4*	The application should enable the user to set specific consumption goals for individual customer groups.

22. Appendix B: Mapping Final Requirements to Functional Specifications

The following table ensures that all User Requirements (D2.2 + EMIVASA) will be implemented by one or more components of SH2O Platform. It also specifies which components will address each requirement. One can easily study in detail specification of each component involved in a Use Case.

Please note that EMIVASA requirements are detailed in Appendix A of the current document.

Table 7 of D2.2: List of use cases

#	Use Case	Components that implement Use Case
8	Use cases of basic customer portal: Visual water meter	
8.1	Use case: Collecting consumption data with smart meters	Portal Exchange Data Manager Basic Customer Portal User and Consumption Database ESB
8.2	Use case: Manually collecting consumption data	
8.3	Use case: Visual exploration of water consumption information	Portal Exchange Data Manager Basic Customer Portal User and Consumption Database ESB
8.4	Use case: Visual exploration of water consumption at fixture/appliance level	Portal Exchange Data Manager Basic Customer Portal User and Consumption Database ESB
8.5	Use case: Providing feedback on disaggregated consumption data	Portal Exchange Data Manager Basic Customer Portal User and Consumption Database ESB
8.6	Use case: Getting water consumption alerts	Business Dashboard Basic Customer Portal User and Consumption Database ESB
8.7	Use case: Getting water consumption tips	Business Dashboard Basic Customer Portal User and Consumption Database ESB
8.8	Use case: Getting system notifications	Business Dashboard Basic Customer Portal User and Consumption Database

		ESB
8.9	Use case: Learning interactively about innovative pricing schemes	Business Dashboard Basic Customer Portal User and Consumption Database ESB
9	Use cases of advanced customer portal: Gamified water meter	
9.1	Use case: Making gamification actions and exploring results	Advanced Customer Portal Gamification Engine
9.2	Use case: Self setting consumption goals	Advanced Customer Portal Gamification Engine
9.3	Use case: Fulfilling consumption goals	Advanced Customer Portal Gamification Engine
9.4	Use case: Recommending water saving actions	Customer Portal Admin Advanced Customer Portal Gamification Engine
9.5	Use case: Declaring water saving actions	Advanced Customer Portal Gamification Engine
9.6	Use case: Contributing household and user profiling information	Advanced Customer Portal Gamification Engine
9.7	Use case: Declaring water end-use events	Advanced Customer Portal Gamification Engine
9.8	Use case: Verifying manually inserted consumption	Advanced Customer Portal Gamification Engine
9.9	Use case: Making actions and earning digital credits with the Games Platform	Advanced Customer Portal Games Platform
9.10	Updated. The advanced customer portal should provide the functionalities available in the basic version extended with the gamification features. Alternatively, the gamification engine can be configured as an extension to an external application, e.g. EMIVASA virtual office.	Advanced Customer Portal Gamification Engine
10	Advanced Customer Portal: Social water meter	
10.1	Use case: Comparing achievements with other households	Advanced Customer Portal ESB User and Consumption Database
10.2	Use case: Inviting another user to join a team	Advanced Customer Portal ESB User and Consumption Database

10.3	Use case: Achieving goals collaboratively as a team	Advanced Customer Portal ESB Gamification Engine
10.4	Use case: Inviting friends on social networks	Advanced Customer Portal Social Connector ESB Gamification Engine
10.5	Use case: Sharing achievements on social networks	Advanced Customer Portal Social Connector ESB Gamification Engine
11	Customer portal user account management	
11.1	Use case: Customer Portal Sign-up	Basic Customer Portal
11.2	Updated. Use case: Gamification Engine Sign-up The application should provide the user the option to register to the basic or the advanced version of the Customer Portal according to their own preference, if both application versions are available to them.	Advanced Customer Portal
11.3	Use case: Modifying User Settings	Advanced Customer Portal
11.4	Use case: Upgrading to the advanced gamified version / downgrading to the basic version of Customer Portal	Basic Customer Portal Advanced Customer Portal
11.5	Use case: Leaderboard opt-in / opt-out	Advanced Customer Portal
11.6	Use case: Geolocation opt-in / opt-out	Advanced Customer Portal
11.7	Updated . Use case: Customer Portal Unsubscription The application should allow users who are registered to the advanced gamified version of the Customer Portal to opt-out of the gamification features.	Basic Customer Portal Advanced Customer Portal
11.8	New – EMIVASA. The gamification engine should enable an external application (e.g. EMIVASA virtual office) to authenticate users.	Advanced Customer Portal
11.9	New – EMIVASA. User actions should be logged to the extent they are needed for the KPI and success criteria validation.	Advanced Customer Portal
11.10	New – EMIVASA. Sign-up, log-in and gamification opt-in should provide the option to display links to external evaluation questionnaires.	Advanced Customer Portal
12	Customer Portal Admin	
12.1	Use case: Setting water consumption tips	Customer Portal Admin

12.2	Use case: Setting actions, badges and rewards	Customer Portal Admin
12.3	Use case: Converting game actions into rewards	Customer Portal Admin
12.4	New – EMIVASA. The application should allow the Admin to monitor user activities on the gamified portal (actions, badges, rewards).	Customer Portal Admin
12.5	New – EMIVASA. The gamification engine should provide services to assign gamified actions from an external application (e.g. EMIVASA virtual office), i.e. to gamify and reward points to actions from an external application.	Customer Portal Admin
12.6	New – EMIVASA. The gamification engine should be configurable independently for different case studies, i.e. features can be (de)activated e.g. when they are already implemented by the utility's existing application.	Customer Portal Admin
13	Use cases of Games Platform	
13.1	Use case: Games Platform Sign-up	Games Platform
13.2	Use case: Playing a standard mobile game	Games Platform
13.3	Use case: Playing the card game and its digital game extension	Games Platform
13.4	Use case: Gaining power-ups based on the Gamification Engine credits	Will not be implemented. There is a more solid business scenario to earn points in various systems like: Water Games, Portal, other utility applications and have them redeemed in a single location, in Advanced Customer Portal. The Admin can define a consistent redeeming set of rules and awards inventory and tracking functionality can be easily added in the future.
13.5	Use case: Connecting player profile to the Gamification Engine	Games Platform
13.6	Use case: Setting content of game questions	Games Platform
14	Use cases of business dashboard: Customer consumption monitor	
14.1	Use case: Visualizing aggregate household consumption information by geo-location	Business Dashboard ESB User and Consumption Database
14.2	Use case: Querying and displaying customer attributes	Business Dashboard ESB User and Consumption Database
14.3	Use case: Identifying customer groups	Business Dashboard ESB User and Consumption Database

14.4	Updated. Use case: Setting consumption goals and rewards for specific customer groups The application should enable the user to set specific consumption goals for individual customer groups.	Business Dashboard ESB User and Consumption Database
14.5	Use case: Setting recommended water saving actions for specific customer groups	Business Dashboard ESB User and Consumption Database
14.6	Use case: Setting water consumption alerts	Business Dashboard ESB User and Consumption Database
15	Use cases of agent-based customer consumption simulator	
15.1	Use case: Modelling behaviour based on consumption	Agent Based Consumption Simulator
15.2	Use case: Predicting customer segment consumption behaviour	Agent Based Consumption Simulator
15.3	Use case: Predicting behaviour based on incentive response	Agent Based Consumption Simulator
15.4	Use case: Predicting customer segment response to pricing schemes	Agent Based Consumption Simulator

23. Appendix C: IFML Concepts

For usability and easy understanding of IFML diagrams in this document, we reiterate here IFML Concepts from *D6.2 – Architecture and design* document. More details can be found in D6.2 document, chapter *ANNEX2 – User Interaction Flows*.

23.1 Overview of IFML main concepts

An IFML diagram consists of one or more top-level *ViewContainers*, i.e., interface elements that comprise components for displaying content and supporting interactions.

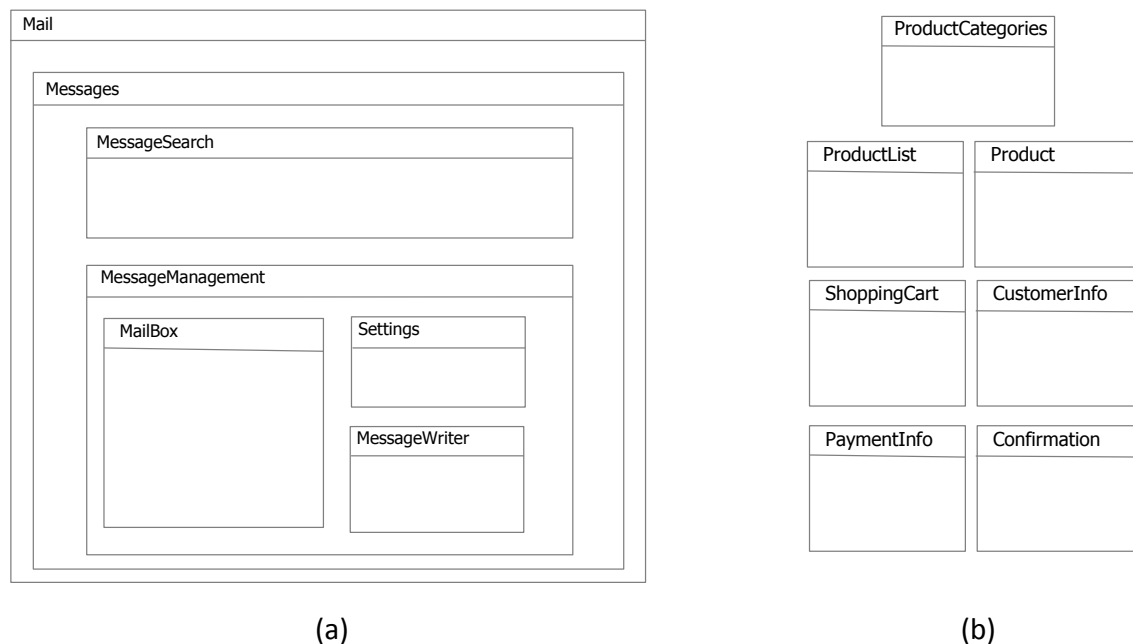


Figure 48: example of different top-level interface structures

Figure 48 contrasts two different organizations of the GUI: a mail desktop or rich internet application (a) consists of a top-level container with embedded sub-containers at different levels; an e-commerce web site (b) organizes the user interface into different independent view containers corresponding to page templates.

Each view container can be internally structured in a *hierarchy of sub-containers*. For example, in a desktop or rich internet application, the main window can contain multiple tabbed frames, which in turn may contain several nested panes. The child view containers nested within a parent view container can be displayed simultaneously (e.g., an object pane and a property pane) or in *mutual exclusion* (e.g., two alternative tabs). In case of mutually exclusive (XOR) containers one could be the *default container*, displayed by default when the parent container is accessed. The meaning of a container can be specified more precisely, by adding a stereotype to the general-purpose construct. For instance, a ViewContainer can be tagged as «window», as in the case of the “Mail” ViewContainer in Figure 49, to hint at the nature of its expected implementation.



Figure 49: example of mutually exclusive sub-containers

In Figure 49, the “Mail” top-level container comprises two sub-containers, displayed alternatively: one for messages and one for contacts. When the top level container is accessed, by default the interface displays the “Messages” ViewContainer.

A ViewContainer can contain *ViewComponents*, which denote the publication of content (e.g., a list of objects) or the input of data (e.g., entry forms).

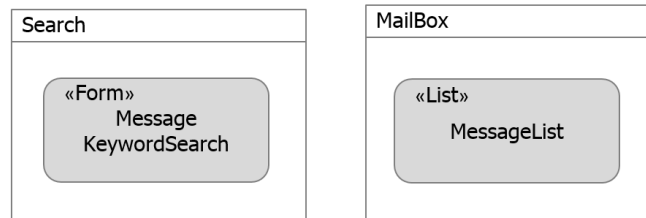


Figure 50 - example of ViewComponents within view containers

Figure 50 shows the notation for embedding ViewComponents within ViewContainers: the “Search” ViewContainer comprises a “MessageKeywordSearch” ViewComponent that represents a form for searching; the “MailBox” ViewContainer comprises a “MessageList” ViewComponent that denotes a list of objects.

A ViewComponent can have *input and output parameters*. For example, a ViewComponent that shows the details of an object has an input parameter corresponding to the identifier of the object to display; a data entry form exposes as output parameters the values submitted by the user; and a list of items exports as output parameter the item selected by the user.

A ViewContainer and a ViewComponent can be associated with *events*, to express that they support the user’s interaction. For example, a ViewComponent can represent: a list associated with an event for selecting one or more items, a form associated with an event for input submission, or an image gallery associated with an event for scrolling through the gallery. IFML events are mapped to *interactors*⁴ in the implemented application. The way in which such interactors are rendered depends on the specific platform for which the application is deployed and is not captured by IFML, but is rather delegated to transformation rules from Platform-Independent Model (PIM) to Platform-Specific Model (PSM). For example, the scrolling of an image gallery may be implemented as a link in an HTML application and as a swipe gesture handler in a mobile phone application.

The effect of an event is represented by an *interaction flow*, which connects the event to the ViewContainer or ViewComponent affected by the event. For example, in an HTML web application the event produced by the selection of one item from a list may cause the display of a new page with the details of the selected object. This effect is represented by an interaction flow connecting the event associated with the list component in a top-level ViewContainer (the web page) with the ViewComponent representing the object detail, positioned in a different ViewContainer (the target web

⁴ By interactor we mean any interface widget that supports the user’s interaction, such as a button, a link, a check box, and so on.

page). The interaction flow expresses a change of state of the user interface: the occurrence of the event causes a transition from a source to a target web page.

For example, in Figure 51 the “MailBoxList” ViewComponent shows the list of available mailboxes and is associated with the “MailBoxSelection” event, whereby the user can open the “MailBox” ViewContainer and access the messages of the mailbox selected in the “MessageList” ViewComponent .

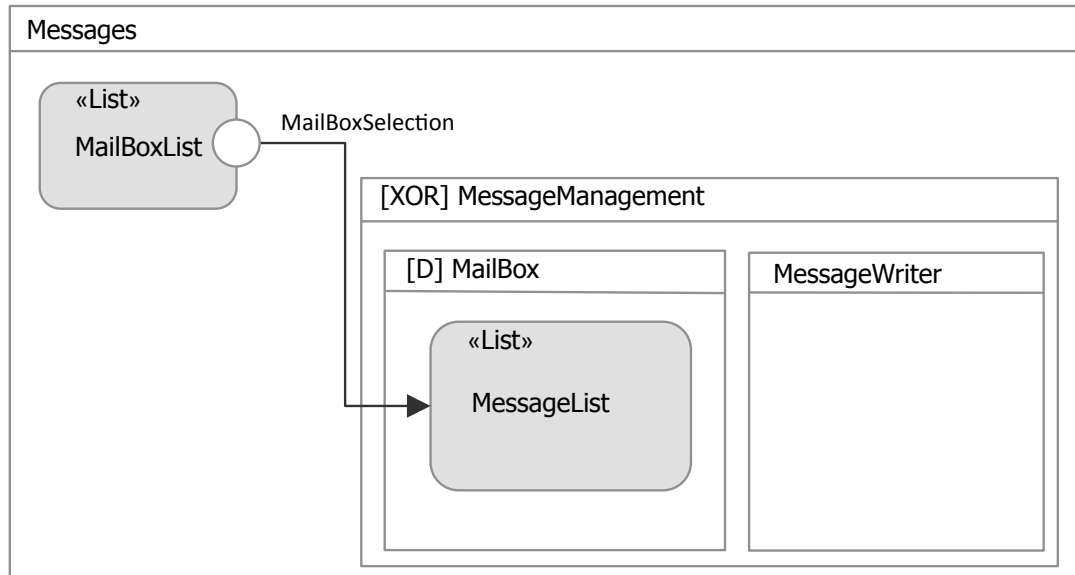


Figure 51 - Example of interaction flow between ViewComponents

An event can also cause the triggering of an action, which is executed prior to updating the state of the user interface; the effect of an event firing an action is represented by an interaction flow connecting the event to an action symbol, consisting of an hexagon. For example, in a mail management application the user can select several messages from a list to delete them; the selection event triggers a delete action, after which the ViewContainer with the updated list is displayed again. The result of action execution is represented by an interaction flow that connects the action to the ViewContainer or ViewComponent affected by it.

In Figure 52, The “Message toolbar” ViewContainer is associated with the events for deleting, archiving and reporting mail messages. Such events are connected by a flow to an action symbol (a labelled hexagonal icon), which represents the business operation. The outgoing flow of the action points to the ViewContainer displayed after the action is executed; if the outgoing flow of an action is omitted, this means that the same ViewContainer wherefrom the action has been activated remains in view (as illustrated for the “Archive” and “Report” actions in Figure 52).

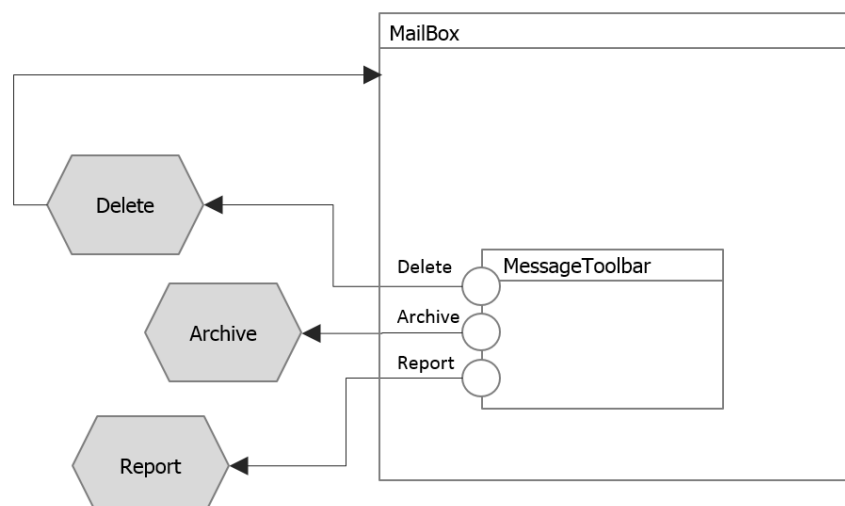


Figure 52: Example of events triggering business actions

The model of Figure 52 does not express the objects on which the business actions operate. Such an *input-output dependency* between view elements (ViewContainers and ViewComponents) or between view elements and actions requires the specification of *parameter bindings* associated with interaction flows. More specifically, two kinds of interaction flows can host parameter bindings: *navigation flows*, which represent navigation between view elements, and *data flows*, which express data transfer only, not produced by the user's interaction. Parameter binding rules are represented by annotations attached to navigation and data flows, as shown in Figure 53.

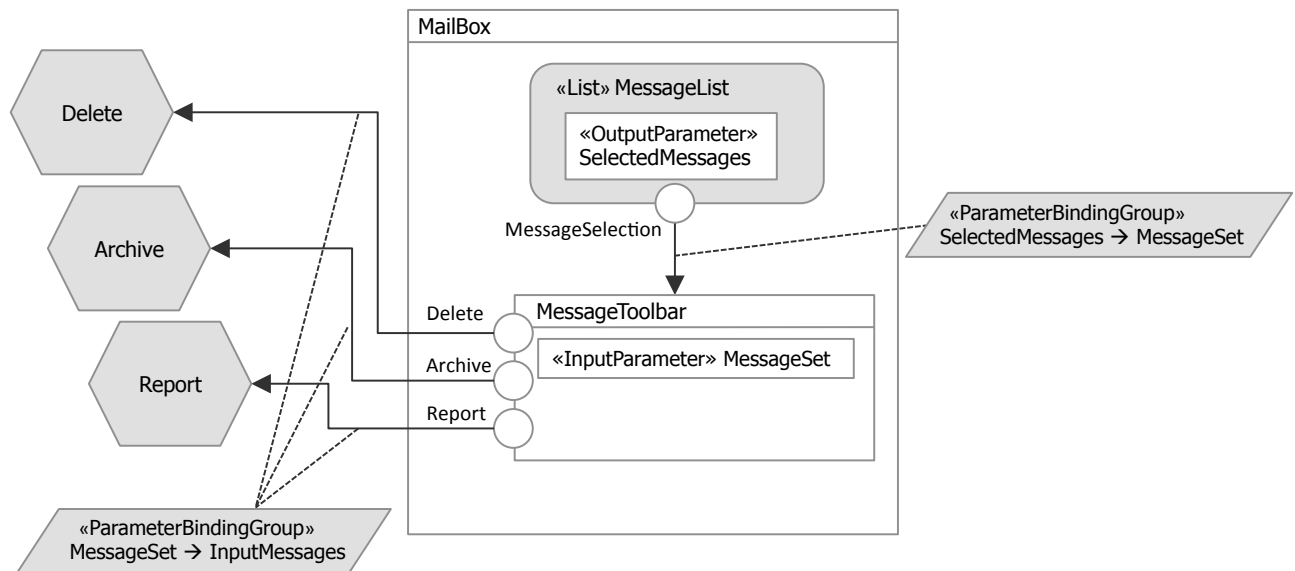


Figure 53: example of parameter bindings used for expressing input-output dependencies

In Figure 53, the “MessageToolbar” ViewContainer has an input parameter “MessageSet”; its value is set to the messages selected from the “MessageList” ViewComponent, when the user produces the “MessageSelection” event. Another parameter binding rule is associated with the Delete, Archive and Report events: the value of the “MessageSet” parameter is bound to the “InputMessages” parameter of the triggered action.

23.2 Role of IFML in the development process

The development of interactive applications is typically managed with agile approaches, which traverse several cycles of “problem discovery” / “design refinement” / “implementation”. Each iteration of the development process generates a prototype or a partial version of the system. Such an incremental lifecycle is particularly appropriate for modern Web and mobile applications, which must be deployed quickly and change frequently during their lifetime to adapt to the user's requirements. Figure 54 schematizes a possible development process and positions IFML within the flow of activities.

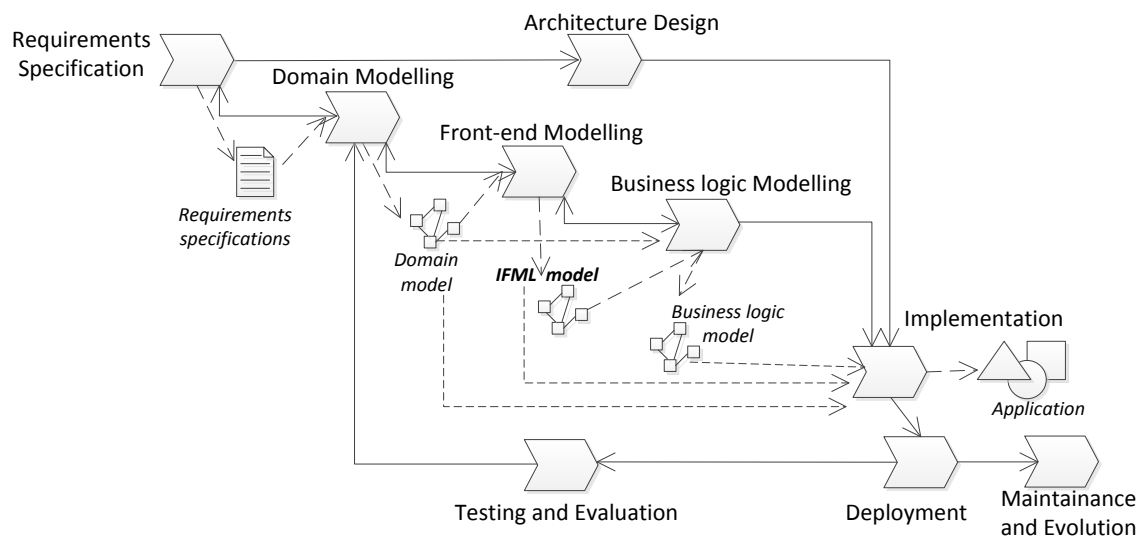


Figure 54 - role of IFML in the development process of an interactive application