



## REPORT ON DELIVERABLE 3.2.1

# Sustainability Dashboard

PROJECT NUMBER: 619186  
START DATE OF PROJECT: 01/03/2014  
DURATION: 42 months



DAIAD is a research project funded by European Commission's 7th Framework Programme.

The information in this document reflects the author's views and the European Community is not liable for any use that may be made of the information contained therein. The information in this document is provided "as is" without guarantee or warranty of any kind, express or implied, including but not limited to the fitness of the information for a particular purpose. The user thereof uses the information at his/ her sole risk and liability.

Dissemination Level	Public
Due Date of Deliverable	Month 22, 31/12/2015
Actual Submission Date	08/07/2016
Work Package	WP3 Water Consumption Interfaces
Task	Task 3.2 Sustainability Dashboard
Type	Prototype
Approval Status	Submitted for approval
Version	1.1
Number of Pages	87
Filename	D3.2.1_DAIAD_Sustainability_Dashboard.pdf

## Abstract

This report provides an overview of the design, implementation, and function of the *Prototype Deliverable D3.2.1 "Sustainability Dashboard"*, a web-based dashboard for delivering information and stimuli to consumers for their water consumption. The Sustainability Dashboard comprises all *user interface (UI) elements* pertinent to the DAIAD front-end applications targeting consumers, and provides interventions, stimuli, exploratory, analysis, and knowledge extraction functions. The development of these individual UI elements is founded on our Year 1 evaluation and analysis regarding the effect of various interface artifacts for conveying water consumption knowledge and stimuli to consumers (*see Report Deliverable D3.1.1 'Design Studies'*). Finally, all UI elements have been instantiated, integrated, and adapted to serve the requirements of consumers in the DAIAD@home mobile and web applications.

## History

version	date	reason	revised by
0.1	10/12/2015	First draft	Anna Kupfer
0.2	10/01/2016	Revision and minor edits	Thorsten Staake
0.3	12/01/2016	Revision and minor edits	Samuel Schöb
0.4	15/01/2016	Revision	Thomas Stiefmeier
0.5	19/01/2016	Revision and addition of information	Nikolaos Georgomanolis, Yannis Kouvaras
1.0	21/01/2016	Final version	Spiros Athanasiou
1.1	08/07/2016	Updated all sections and introduced additional sub-sections following reviewers' feedback received during the Y2 review meeting; document re-authored as a Report deliverable	Yannis Kouvaras, Michalis Alexakis, Nikos Georgomanolis, Stelios Manousopoulos, Nikos Karagiannakis, Spiros Athanasiou

## Author list

organization	name	contact information
UNI BA	Anna Kupfer	<a href="mailto:anna.kupfer@uni-bamberg.de">anna.kupfer@uni-bamberg.de</a>
UNI BA	Thorsten Staake	<a href="mailto:thorsten.staake@uni-bamberg.de">thorsten.staake@uni-bamberg.de</a>
UNI BA	Samuel Schöb	<a href="mailto:samuel.schoeb@uni-bamberg.de">samuel.schoeb@uni-bamberg.de</a>
Amphiro	Thomas Stiefmeier	<a href="mailto:stiefmeier@amphiro.com">stiefmeier@amphiro.com</a>
ATHENA RC	Spiros Athanasiou	<a href="mailto:spathan@imis.athena-innovation.gr">spathan@imis.athena-innovation.gr</a>
ATHENA RC	Yannis Kouvaras	<a href="mailto:jkouvar@imis.athena-innovation.gr">jkouvar@imis.athena-innovation.gr</a>
ATHENA RC	Nikolaos Georgomanolis	<a href="mailto:ngeorgomanolis@imis.athena-innovation.gr">ngeorgomanolis@imis.athena-innovation.gr</a>
ATHENA RC	Michalis Alexakis	<a href="mailto:alexakis@imis.athena-innovation.gr">alexakis@imis.athena-innovation.gr</a>
ATHENA RC	Stelios Manousopoulos	<a href="mailto:smanousopoulos@imis.athena-innovation.gr">smanousopoulos@imis.athena-innovation.gr</a>
ATHENA RC	Nikos Karagiannakis	<a href="mailto:nkaragiannakis@imis.athena-innovation.gr">nkaragiannakis@imis.athena-innovation.gr</a>

# Executive Summary

This report provides an overview of the design, implementation, and function of the *Prototype Deliverable D3.2.1 “Sustainability Dashboard”*, a web-based dashboard for delivering information and stimuli to consumers for their water consumption. The Sustainability Dashboard comprises all *user interface (UI) elements* pertinent to the DAIAD front-end applications targeting consumers (*DAIAD@home mobile and web applications*), and provides interventions, stimuli, exploratory, analysis, and knowledge extraction functions.

The development of these individual UI elements is founded on our Year 1 evaluation and analysis regarding the effect of various interface artifacts for conveying water consumption knowledge and stimuli to consumers (*see Report Deliverable D3.1.1 ‘Design Studies’*). The elements of the Sustainability Dashboard have been instantiated, integrated, and adapted to serve the requirements of consumers & experts in DAIAD@home mobile and web applications.

In Section 1, we summarize the common underlying design principles of all UI elements and specific needs of our users.

In Section 2, we briefly discuss our approach towards rapid design, prototyping, and testing, which has enabled us to experiment with several concepts and iterations. We then present select prototypes of various iterations, portraying their evolution over time.

In Section 3 we present the various technologies applied for the development of Sustainability Dashboard. While all of its UI elements are web-based (i.e. apply Web technologies in their development and deployment) their application in different target platforms, i.e. mobile devices (mobile phone, tablet) and web (browser), has required the use of different frameworks and libraries.

In Section 4 we present in detail the UI elements that comprise the Sustainability Dashboard following the natural distinction of UI elements against their target platform. For each UI element we briefly describe its purpose, intended functionality, implementation details, and indicative examples for their invocation.

## Abbreviations and Acronyms

API	Application Programming Interface
AJAX	Asynchronous JavaScript and XML
AOP	Aspect Oriented Programming
APK	Android application package
BT	Bluetooth
CI	Continuous Integration
CORS	Cross-Origin Resource Sharing
CSRF	Cross-Site Request Forgery
CSS	Cascading Style Sheets
DOM	Document Object Model
DTO	Data Transfer Object
JSON	JavaScript Object Notation
MR	MapReduce
MVC	Model View Controller
MVP	Minimum Viable Product
OGC	Open Geospatial Consortium
ORM	Object Relational Mapper
RERO	Release Early, Release Often
REST	Representational State Transfer
RF	Radio Frequency
RPC	Remote Procedure Call
SPA	Single Page Application
SWM	Smart Water Meter
UI	User Interface

# Table of Contents

1. Principles of UI elements.....	8
1.1. Common UI Principles.....	8
1.2. Consumers.....	9
2. Development Process.....	11
2.1. Methodology.....	11
2.2. Prototype Iterations.....	13
3. Technology Stack.....	20
3.1. Common foundations.....	20
3.1.1. JavaScript.....	20
3.1.2. HTML5.....	20
3.1.3. CSS.....	20
3.1.4. Data API.....	20
3.2. Mobile UI elements.....	22
3.2.1. Application Patterns and Design.....	23
3.2.2. Libraries and Frameworks.....	23
3.3. Web UI elements.....	26
3.3.1. Application Patterns and Design.....	26
3.3.2. Libraries and Frameworks.....	27
4. UI elements.....	31
4.1. Web UI elements.....	31
4.1.1. Charts.....	31
4.1.2. Widgets.....	34



4.1.3. Events table .....	39
4.1.4. Events details modal.....	43
4.1.5. Messages.....	48
4.1.6. Message details.....	50
4.2. Mobile UI elements.....	53
4.2.1. Dashboard Gauge.....	53
4.2.2. Chart .....	56
4.2.3. Comparison .....	62
4.2.4. Gauge.....	64
4.2.5. Messages.....	65
4.2.6. Message details.....	67
4.2.7. Events table.....	70
4.2.8. Projections.....	72
4.2.9. Efficiency rating .....	74
5. Annex: Water Calculator .....	75
5.1. Water Calculators.....	76
5.2. DAIAD Water Calculator .....	77
6. Annex: Messages .....	80

# 1. Principles of UI elements

The Sustainability Dashboard comprises all UI elements of the DAIAD front-end applications for consumers and therefore must address all possible interaction, information, and knowledge exploration needs of our end-users. Further, the UI elements will be integrated in the DAIAD@home mobile & web applications and as such must be flexible enough to adapt and optimize to the respective devices (*mobile phones/tablets, pc/laptop*), adjusting for different viewport sizes, resolutions and control mechanisms (*touch, mouse, buttons*). In addition, they need to inherently support internationalization (*i.e. different languages*) and localization (*e.g. different units*) since DAIAD is targeting the entire EU population. Another important requirement concerns the need for facilitating *theme* changes (*e.g. colors, fonts*) in a streamlined manner. Finally, the UI elements should be as *lightweight* as possible in their implementation (*i.e. source code*) and *abstracted* from the underlying business logic to facilitate their reusability and performance.

In the following we briefly present the common design and development principles for all UI elements of the Sustainability Dashboard. With this foundation in place, we then discuss the specific needs of our target group (consumers) and our approach towards addressing them.

## 1.1. Common UI Principles

Web-based	UI elements are designed and implemented using exclusively Web standards ( <i>e.g. JS, HTML</i> ). This will streamline their porting and adaptation across mobile devices and desktop computers. The adaptation of existing elements from existing web frameworks should be preferred. If this is not possible, these frameworks should be extended, ensuring backwards compatibility and their coherent operation.
Responsive	UI elements should gracefully respond to different devices in a flexible manner [W3C]. They should scale the UI and any media assets accordingly, ensuring a fluid user experience. Adaptive design ( <i>i.e. gradually revealing additional functions and features depending on the screen size and resolution</i> ) should be avoided.
Visual Information-seeking Mantra	We will respect the visual information seeking Mantra [Shn96]: " <i>Overview first, zoom and filter, then details-on-demand</i> " both for individual UI elements, as well as during their assembly into screens and workflows.
Coherent visual identity	All UI elements will adhere to the common visual identity of DAIAD ( <i>color themes, fonts, framing</i> ). The only approved exception concerns the use of fonts for mobile devices in order to improve <i>readability</i> from users.

## Internationalization & localization

The UI elements should support their internationalization and localization in a streamlined and automated manner. Limits on character size for textual assets (*strings, labels*) should be established for all such assets.

## 1.2. Consumers

The common purpose of UI elements for consumer front-ends is to convey stimuli and information for inducing sustainable changes in consumption behavior. In Deliverable D3.1.1 ‘Design Studies’ we have identified, studied and evaluated against a user panel the relevant UI elements, selecting the leading design concepts in terms of effectiveness. Further, in Deliverable D1.2 ‘DAIAD Requirements and Architecture’ we documented and analyzed the requirements of consumers and consumer groups.

With these foundations in place, and before beginning our work in the individual UI elements, we established the basic operation of the consumer applications. This *top-down* approach was extremely useful, enabling us to establish the envisaged *high-level user experience*, ensure the *common understanding* of all involved Consortium members, and then *break it down* into individual components to be implemented. In the following we present this high level overview and discuss where relevant our design decisions.

When starting the DAIAD@home application (web & mobile), a **dashboard** provides an **overview** and basic information about ongoing consumption. This information includes metrics that are measured with the help of the DAIAD@feel sensor and/or smart water meter (*volume of water, energy, energy efficiency class, etc.*), in addition to social comparisons or goals, and saving tips/alerts. The overview is based on the Apple Watch paradigm of “Complications” [Ap15] that delivers quick access on metrics that are frequently used (*such as information about energy efficiency, shower timer, or goals*). For that reason, the consumer has the most wanted information at a *first glance* when opening the web-based or mobile application. The user can also *personalize* the displayed metrics accordingly to her interests, or replace them with *shortcuts* to functionalities she frequently uses.

Another element of the Sustainability Dashboard concerns the zoom-in function for detailed statistics. The Sustainability Dashboard provides more **detailed information** on water and energy consumption, projected savings and CO<sub>2</sub> emissions in the form of charts for different time horizons (*daily, weekly, monthly, yearly*). It is relevant that the user can navigate through this information for different DAIAD@feel sensors, as well as points in time. The user can *drill-down* into individual consumption events and gain access to more detailed information (time-series) and statistics, also enabling her to compare her consumption with to her past consumptions. When it comes to presenting information on consumption or savings, we chose bar charts as a main visualization for liters, kWh, Euros and CO<sub>2</sub> emissions. This decision is based on the experiences from the design studies, as most participants preferred *factual* information to metaphorical feedback information.

Moreover, **social comparisons** represent another element to nudge the user in sustainable water use. The consumer engages in a competitive behavior when learning about the consumption behavior of her neighbors, peers, or household members (*broader levels of comparison such as city- or regional-level are possible, too*). Furthermore, **goals** represent a vital UI element for supporting sustainable

water use. In the course of the self-tracking and '*quantified self*' trend and accordingly to the literature review of D.1.1 'State-of-the-Art Report' and D.3.1.1 'Design Studies', goals tremendously impact saving behavior. Users have the possibility to set goals for their water and energy consumption, monetary savings and CO<sub>2</sub> emissions. Users monitor their personal goal attainment in the goal section. Additionally, specific alerts and messages dynamically inform the consumer about her performance.

Finally, saving behavior will be nudged with **saving tips** and personalized messages/alerts. Personalized messages only display accordingly to a specific consumption or saving behavior (such as in order to reaffirm a certain behavior). Alerts, messages, and saving tips appear at the overview dashboard (with personalized information) and at the information section.

Concerning the navigation, several ways allow the user to navigate through the application menus. A navigation and tab bar help users to interact with the application according to their habits and preferences. Furthermore, we use icons for navigations (*such as settings, dashboard, statistics, goals, connected devices*) and for statistics (*such as water drop for water consumption, a lightning for energy consumption as well as a EURO-sign for monetary savings*).

## 2. Development Process

In this section, we briefly discuss our approach towards rapid design, prototyping, and testing of each specific element, which has enabled us to experiment with several concepts and iterations. Towards this, we introduce and present select prototypes of various iterations, portraying their evolution over time.

The development of the Sustainability Dashboard will continue throughout the duration of the project, integrating valuable feedback from the real-world trials. Our planning is to follow a rapid prototyping approach similar to the one presented in the following sub-section and frequently roll-out improved versions of all UI elements.

### 2.1. Methodology

We followed a rapid prototyping approach, consisting of small (1-3 weeks) iterative development cycles to improve the dashboard. In each cycle, we focused on a specific set of interventions and functionalities, producing a rough wireframe, data and user flow, and requirements. This was developed and tested internally by members of the consortium. The feedback was applied to further improve the prototype with another internal round of evaluation. After that, the prototype was either forwarded for another cycle, or graduated from the prototyping process.

In the following figures we provide a number of examples of how this process was followed in practice. In Figure 1 and Figure 2 we observe the original inception (wireframe) of the Settings screen and its current implementation. This very simple screen actually holds a number of important elements and design considerations that are used across the DAIAD applications. Without going into details, we would like to highlight (a) the use of a 'progress bar' on top of the profile as a means to nudge users towards completing it, (b) the common visual language for categories/labels, which can support vertically larger menus (e.g. multiple devices installed) and horizontally drill-down functions (e.g. device settings), (c) the simple common labels for buttons consisting of an icon and a text label to avoid ambiguity.

In Figure 3 we provide three snapshots in time that present the evolution of the starting screen for the mobile application. The differences at first may seem negligible, but actually correspond to significant effort, testing, and improvement. The first prototype integrates most of our original design elements and considerations, attempting to maximize the information provided to users. The top part of the screen is devoted to time and consumption controls, enabling users to select a time period and see their consumption in terms of water, energy and CO2. The middle part of the screen is dominated by one number (current consumption) and a circular progress bar visualizing water, energy, and CO2. The lower part of the screen is where the actual consumption is presented. Moving on to the second prototype of Figure 3, we see a number of important changes. First, the facilities for viewing and

filtering consumption have been moved to a completely separate screen to reduce information overload and simplify the operation. Second, the circular progress bar has been removed, as it is only relevant when a water budget (i.e. maximum water use) has been defined. Third, four complications around the edges of the screen have been added, which provide progress in a single glance. Fourth, we use the bottom third of the screen for presenting any important messages/tips to the user. Finally, the third prototype is much less drastic in its changes, focusing on ensuring a simpler and cleaner user experience, minimizing abstractions and guiding attention to the actual information.

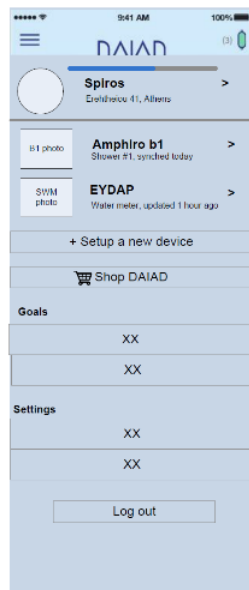


Figure 1: Wireframe of UI elements for Settings

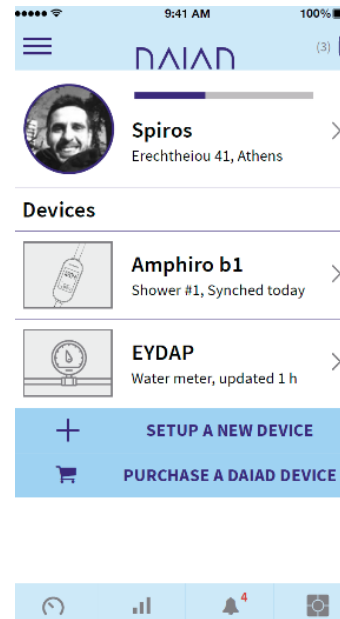


Figure 2: Implemented UI elements for Settings

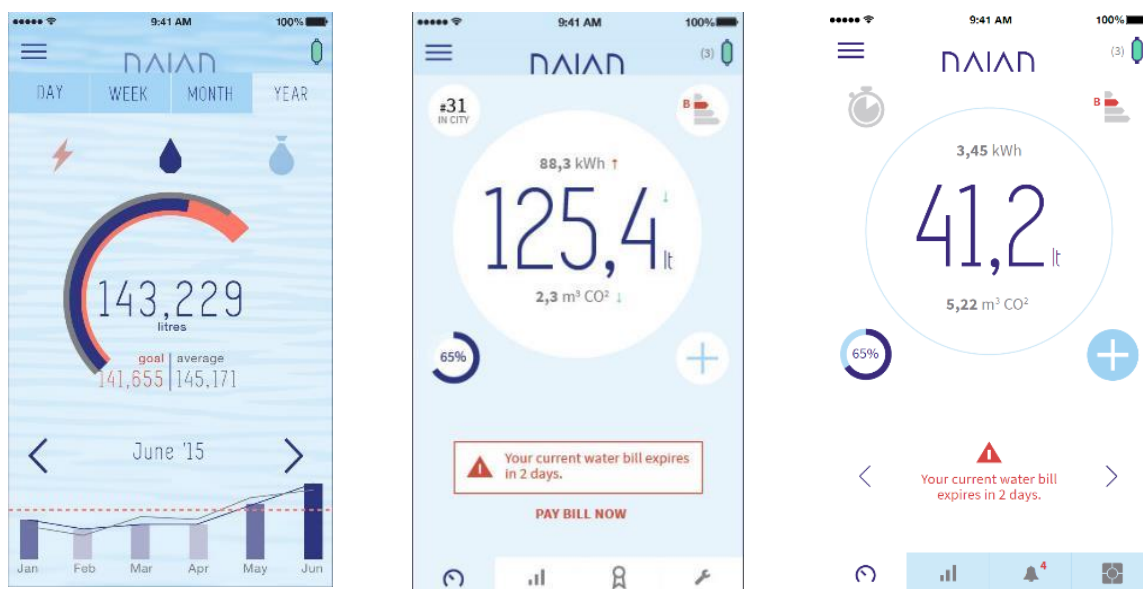


Figure 3: Evolution of main Dashboard

## 2.2. Prototype Iterations

The first development cycle consisted of design prototypes that represent an update of Deliverable D3.1. 'Design Studies'. Figure 4 to Figure 9 show the early prototypes displaying consumption (energy and water) information on several time horizons, combined with goals and consumption averages (descriptive feedback for consumed energy in kWh or water in liters), as well as social normative feedback (energy efficiency scale).



Figure 4: Weekly energy consumption in different metrics and visualizations



Figure 5: Yearly water consumption in different metrics and visualizations

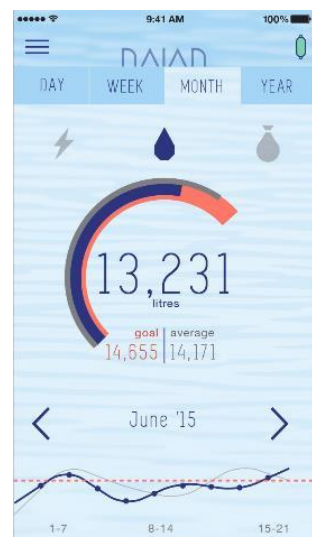


Figure 6: Monthly water consumption in different metrics and visualizations



Figure 7: Weekly water consumption in different metrics and visualizations



Figure 8: Navigation bar

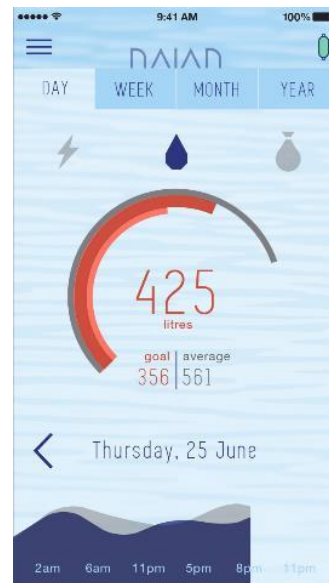


Figure 9: Daily water consumption in different metrics and visualizations



Figure 10: Very early prototype shows savings and relates them to a regional environmental impact they could induce

The second major iteration led to the development of a functional prototype with significantly revised and improved functions. First of all (see Figure 11), when starting the App, participants need to login or register/create a new account. They can also choose to login with their Facebook account. After successful login, the overview dashboard with real-time feedback and diverse overview information appears. The tap bar (on the bottom of all prototypes) or the navigation menu helps the user to find her way through the App. We chose basic icons to represent the dashboard, statistic, goals, and settings (see bottom of Figure 12). On the top right corner currently connected DAIAD@feel sensors can be seen and Figure 22 shows the specific menu for managing all water meters connected to the App. Figure 17 to Figure 20 present the implementation of the line and bar charts displaying the water and energy consumption, savings/spent money, and CO<sub>2</sub> emissions.

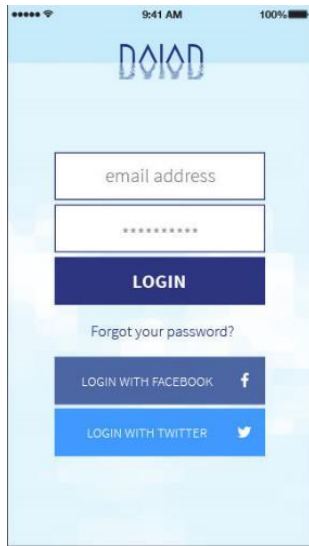


Figure 11: Log-in elements

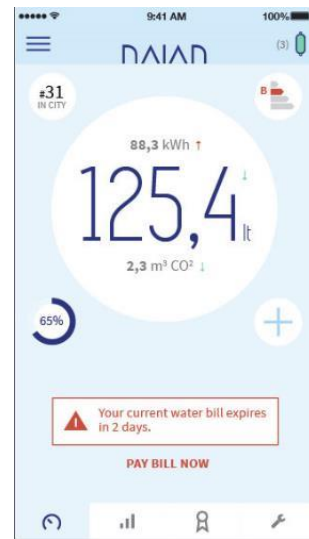


Figure 12: First overview

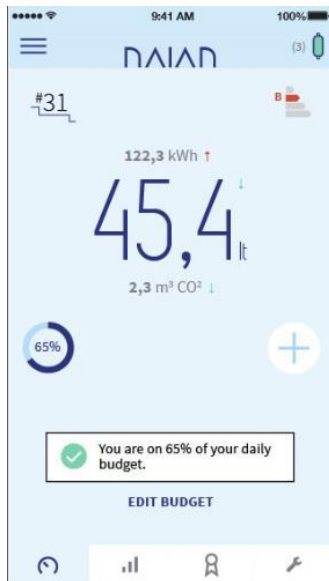


Figure 13: First overview II

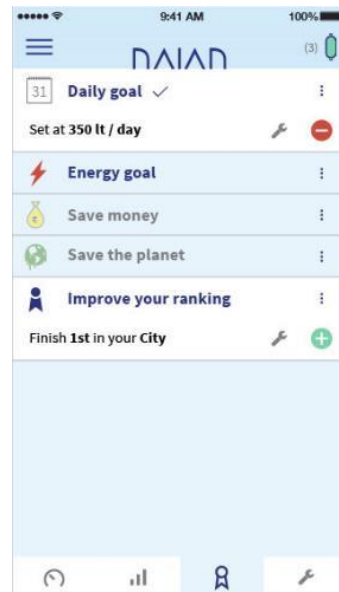


Figure 14: Goal setting overview and budget settings



Figure 15: First overview III

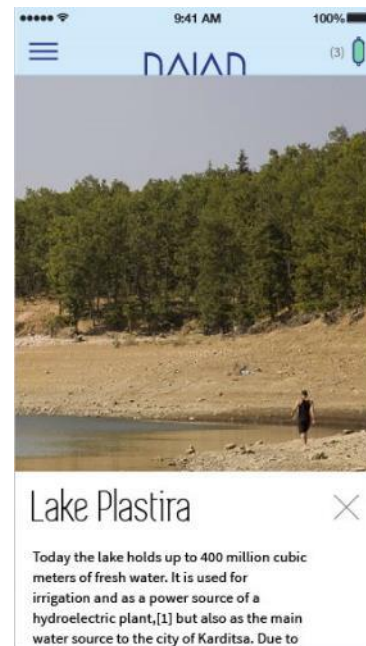


Figure 16: More information on the informative message of Figure 12

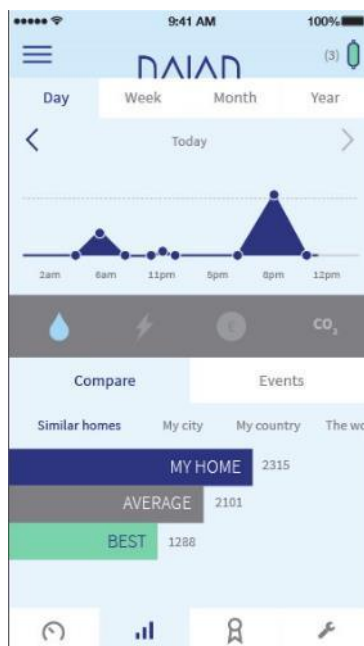


Figure 17: Historical feedback of daily water consumption and comparisons with similar homes



Figure 18: Historical feedback of daily water consumption and overview on individual showers in the event section



Figure 19: Historical feedback of weekly energy consumption and comparison on a city-level



Figure 20: Historical feedback of monthly CO2 emission and an overview different months in the event section

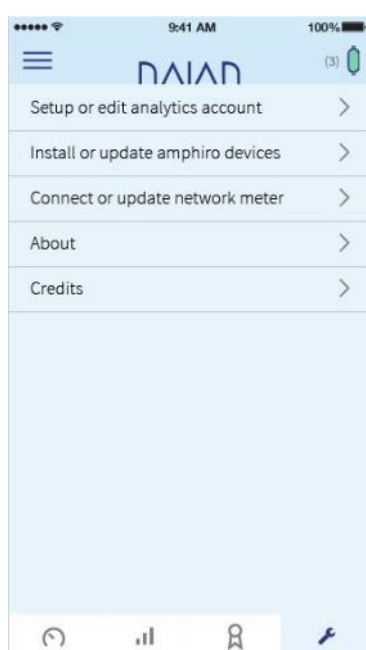


Figure 21: General settings

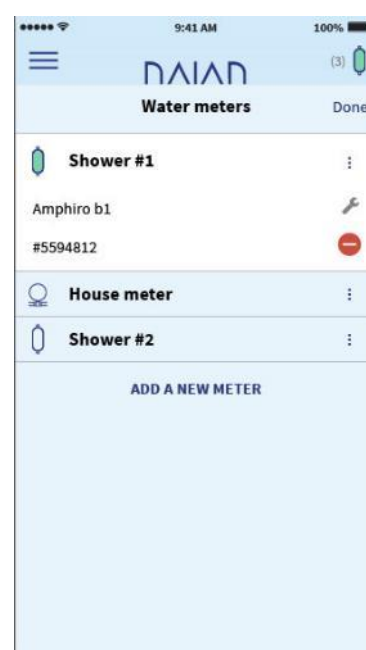


Figure 22: Overview on connected water meters and settings

The current iteration is briefly presented in the following. As the Prototype Deliverables D4.2.1 and D5.4.1 will present the DAIAD applications in detail, we will limit ourselves in presenting only major UI elements introduced in this version. First, we have developed an intelligent ‘Shower Timer’ which

enables users to set individual and family consumption goals per shower. The timer automatically starts as soon as water flow is detected and provides alarms to consumers when exceeding their targets.

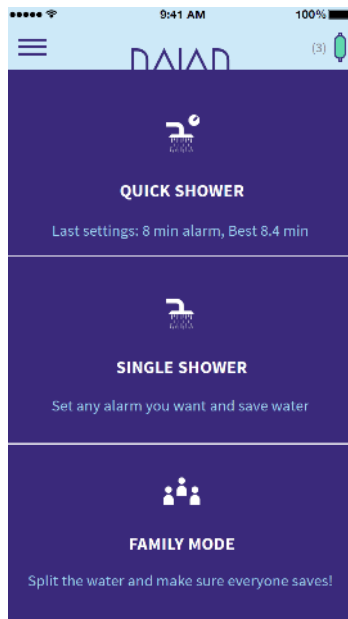


Figure 23: Shower selection

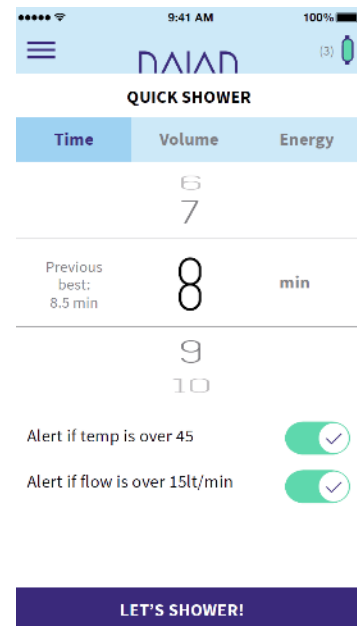


Figure 24: setting personalized goals

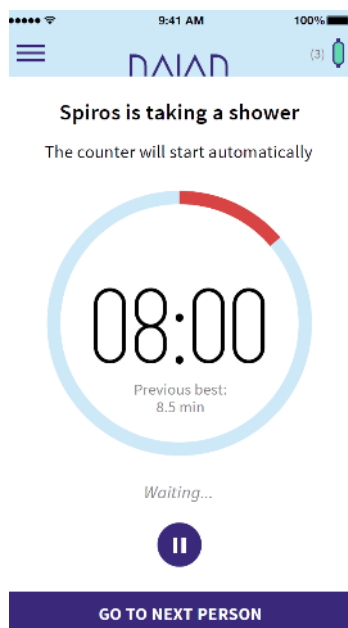


Figure 25: Automatic start/end

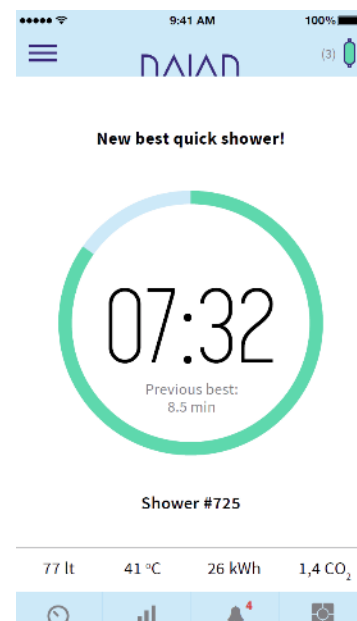


Figure 26: Shower statistics

Second, we have introduced a completely new Messages section, which integrates all messages pushed to a consumer. These include 'Alerts' (events requiring immediate attention), 'Tips' (generic suggestions for improving water efficiency), and 'Insights' (personalized recommendations for increasing efficiency). The whole Messages section resembles an email 'Inbox', with three separate sections depending on the urgency of the message.

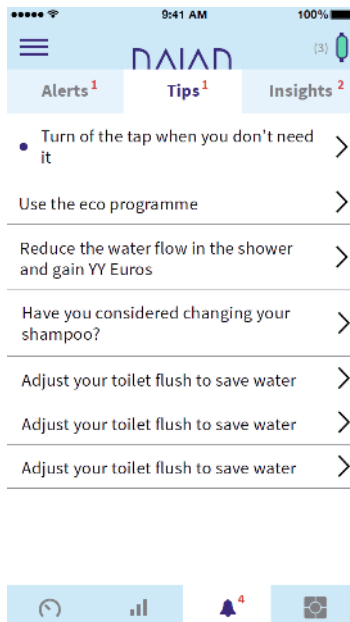


Figure 27: Tips for improving efficiency



### Turn of the tap when you don't need it



When you wash your hands, vegetables, or dishes, think about turning off the tap when you don't actually need the water. A running tap spends 2 liters every minute. It doesn't sound much, but over a year you could be saving more than 2,000

Figure 28: View of a specific tip

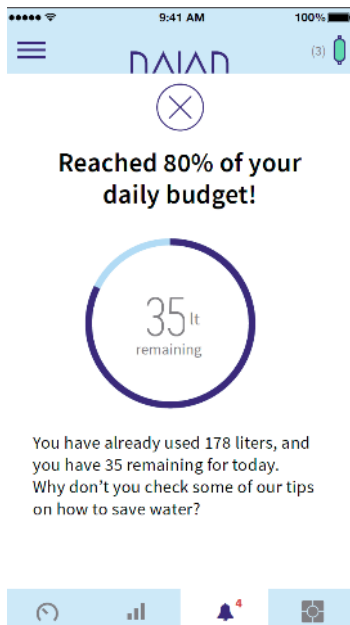


Figure 29: View of a specific alert

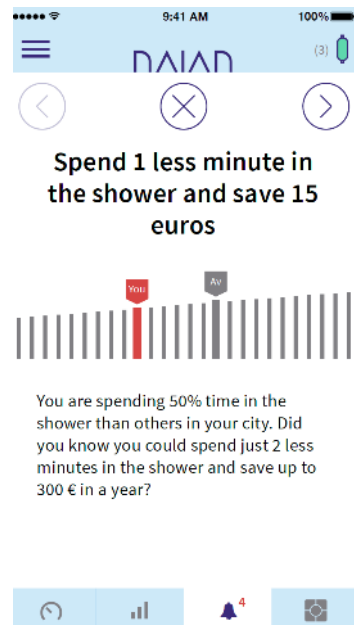


Figure 30: View of a personalized recommendation

## 3. Technology Stack

### 3.1. Common foundations

#### 3.1.1. JavaScript

The new version of JavaScript, codenamed [ES6 or ECMAScript 6 or ECMAScript 2015](#) brings many new features and standardizes several features previously found in third-party JS libraries. The multitude of tools and transformations necessitated automated handling for the course of development and the production deployment of both web applications.

#### 3.1.2. HTML5

HTML5 is the fifth and latest version of the popular markup language used for structuring and presenting content on the World Wide Web. It includes detailed processing models to encourage more interoperable implementations; it extends, improves and rationalizes the markup available for documents, and introduces markup and APIs for complex web applications. For the same reasons, HTML5 is also applied for cross-platform mobile applications, because it includes features designed with low-powered devices in mind.

#### 3.1.3. CSS

Cascading Style Sheets (CSS) is a style-sheet language used for describing the presentation of a document written in a markup language such as HTML. The latest version brings support for a lot of long-awaited novelties -that could only previously be found in browser-specific implementations- like rounded corners, shadows, gradients, transitions or animations, as well as new layouts like multi-columns, flexible-box or grid layouts.

#### 3.1.4. Data API

The Data Application Programming Interface (API) supports querying data persisted by the Big Water Data Management Engine developed in WP5 and presented in deliverable D5.1.1. It is exposed as a Hypertext Transfer Protocol (HTTP) Remote Procedure Call (RPC) API that exchanges JSON encoded messages and has two endpoints, namely, the Action API and HTTP API endpoints. The former is a stateful API that is consumed by the DAIAD web applications. The latter is a Cross-Origin Resource Sharing (CORS) enabled stateless API that can be used by 3<sup>rd</sup> party applications.

The API exposes data from three data sources, namely, smart water meter data, amphiro b1 data and forecasting data for smart water meters. The query syntax is common for all data sources. Moreover, smart water meter and amphiro b1 data can be queried simultaneously. However, a separate request must be executed for forecasting data.

The API accepts a set of filtering criteria as parameters and returns one or more data series consisting of data points which in turn have one or more aggregate metrics like sum, min or average values. More specifically the input parameters are:

- **Time:** Queries data for a specific time interval. An absolute time interval or a relative one (sliding window) can be defined. Optionally, the time granularity i.e. hour, day, week, month or year, can be declared that further partitions the time interval in multiple intervals. The Data API returns results for every of these time intervals.
- **Population:** Specifies one or more groups of users to query. For every user group a new data series of aggregated data is returned. A query may request data for all the users of a utility, the users of a cluster, the users of an existing group, a set of specific users or just a single user.

Clusters are expanded to segments before executing the query. A segment is equivalent to a group of users. As a result, declaring a cluster is equivalent to declaring all of its groups. Optionally, the users of a group may be ranked based on a metric.

- **Spatial:** A query may optionally declare one or more spatial constraints and filters. A spatial constraint aggregates data only for users whose location satisfies the spatial constraint e.g. it is inside a specific area. On the contrary, a spatial filter is similar to the population parameter and creates a group of users based on their location; hence a new data series is returned for every spatial filter.
- **Metric:** The metrics returned by the query. Data API supports min, max, sum, count and average aggregate operations. Not all data sources support all metrics.
- **Source:** Declares the data source to use. When forecasting data is requested, this parameter is ignored.

Detailed documentation on the Data API syntax and request examples can be found at:

- <https://app.dev.daiad.eu/docs/api/index.html>.

The Data API is implemented as part of the DAIAD Services presented in Deliverable 1.3. Figure 31 illustrates the Data API implementation in more detail.

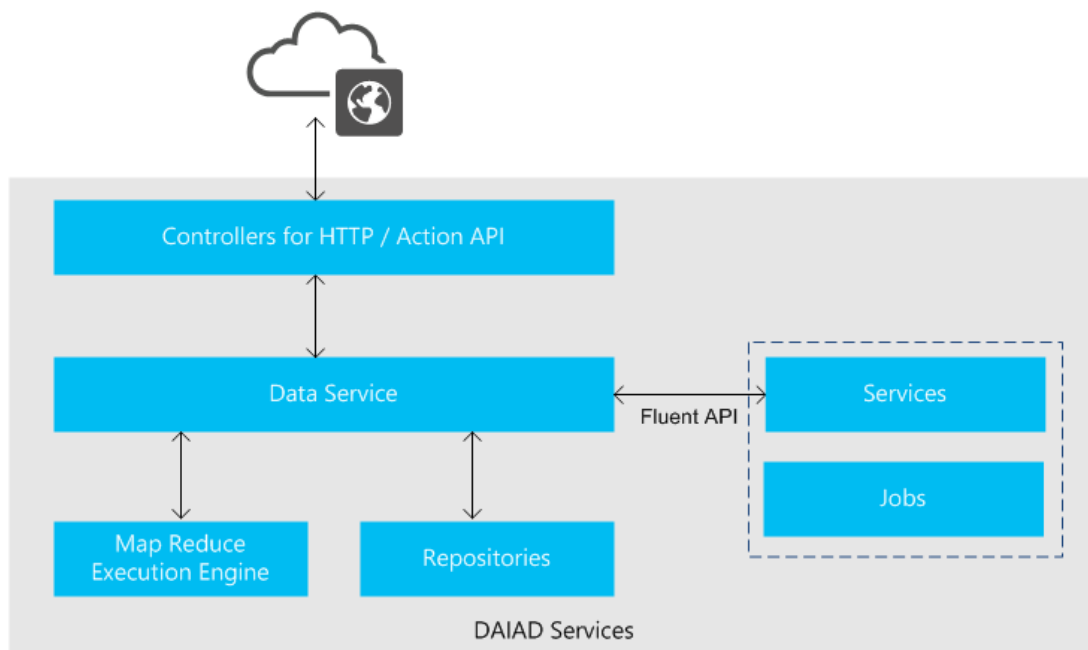


Figure 31: Data API implementation

Query requests are received by the DAIAD Services controller components and forwarded to the Data Service. The Data Service orchestrates data query execution. It accesses data from several repositories such as user profile information and smart water meter registration data and expands the query before execution. Query expansion refers to the process that selects all individual users and their corresponding devices for all groups to query. In addition, any spatial constraints are applied at this stage. The expanded query is submitted to the Map Reduce execution engine for computing the query results.

In addition to the HTTP endpoints, Data API also provides a fluent API for building queries at the server side. This feature is used by other services and jobs for querying water consumption data. Two distinctive examples are the Message Service<sup>1</sup> and the User Clustering Job<sup>2</sup> respectively. The former queries utility and individual user consumption data in order to generate alerts and recommendations. The latter clusters the users based on their total water consumption over a predefined time interval.

## 3.2. Mobile UI elements

The development of the mobile UI elements follows the architectural decisions and applies the technologies of the DAIAD@home mobile application. To facilitate the reader, in the following we briefly present the core technologies and frameworks used.

<sup>1</sup> <https://github.com/DAIAD/home-web/blob/master/src/main/java/eu/daiad/web/service/message/DefaultMessageService.java>

<sup>2</sup> <https://github.com/DAIAD/home-web/blob/master/src/main/java/eu/daiad/web/jobs/ConsumptionClusterJobBuilder.java>

### 3.2.1. Application Patterns and Design

In the DAIAD@home mobile application we apply the Single Page Application (SPA) web application design. This design is used extensively and strongly influences the structure of the source code. A short explanation; a broader coverage of these topics is outside the scope of this document:

- **Single Page Applications (SPAs)** offer increased UI usability that is in par with desktop applications. In contrast to traditional web applications, a SPA application is initialized by loading only a single web page. After initialization, any additional resources such data or JavaScript code files are loaded dynamically on demand using Asynchronous JavaScript and XML (AJAX) requests. Moreover, client side code is usually implemented using the MVC pattern or some variant of it.

### 3.2.2. Libraries and Frameworks

#### 3.2.2.1. Apache Cordova

Apache Cordova<sup>3</sup> is a framework for developing hybrid mobile applications. The application is implemented using HTML5, JavaScript and CSS. Cordova offers a set of JavaScript application programming interfaces that allow access to the most common native device APIs including geolocation, storage and network functionality among others. More significantly, these APIs are consistent across multiple platforms, hence, code has to be written only once. In the case that the developer requires access to a native API that is not supported e.g., Bluetooth connectivity, Cordova offers a plugin architecture for implementing JavaScript bindings in order to expose the required native functionality to the JavaScript code.

A generic overview of a Cordova application is shown in Figure 32.

---

<sup>3</sup> <https://cordova.apache.org/>.

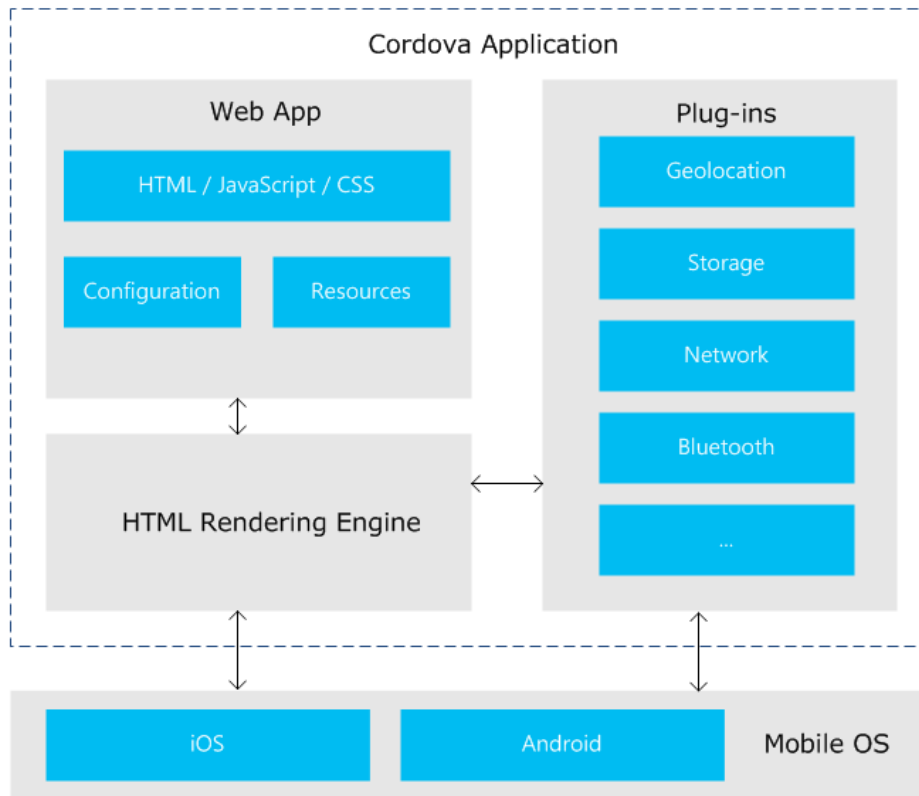


Figure 32: Cordova Application Architecture

The Web App block contains the majority of the application code and consists mostly of HTML, JavaScript and CSS files, like any traditional web application. Any required application assets such as image files are also packaged as resources. In addition to the source code and assets, the Web App contains configuration settings for building the application and managing Plug-ins.

Plug-ins, are special extensions of Cordova that provide access to device hardware features such as storage, network connectivity, native UI elements and Bluetooth. The most important plug-ins used by the DAIAD@home mobile application are enumerated below:

- cordova-plugin-ble-central 1.0.5: Access to the Bluetooth low energy (BLE) connectivity
- cordova-plugin-camera 2.1.1: Access to camera
- cordova-plugin-dialogs 1.2.0: Enables OS notifications
- cordova-plugin-file 3.0.0: Provides access to device storage
- cordova-plugin-geolocation 2.1.0: Allows the application to query the user's location
- cordova-plugin-globalization 1.0.3: Add globalization support to the application
- cordova-plugin-inappbrowser 1.2.1: Adds support for windows
- cordova-plugin-network-information 1.0.1: Provides information about device's cellular and Wi-Fi connection status and whether the device has internet access
- cordova-plugin-screen-orientation 1.4.2: Offers control over the device screen orientation
- cordova-sqlite-storage 0.8.2: Add access to SQLite relation database

All the plugins, except for the one of SQLite, are available at the Cordova Plugins repository at <https://cordova.apache.org/plugins/>.

Once the application is deployed, the HTML Rendering Engine, part of the Cordova runtime, executes the application and enables it to interact with the plugins and the supported mobile OS functionality. Currently, DAIAD@home mobile application supports iOS and Android.

#### 3.2.2.2. L20n

Internationalization is handled by [L20n](#), a localization framework developed by Mozilla for the Web. It allows localizers to put small bits of logic into localization resources to codify the grammar of the language. We have included the English language, as a default language and Spanish.

#### 3.2.2.3. Crypto

Crypto<sup>4</sup> is a growing collection of standard and secure cryptographic algorithms implemented in JavaScript using best practices and patterns. The mobile application has been implement using the AES (Advance Encryption Standard) and ECB (Electronic Codebook) components for encrypting and decrypting blocks of data.

#### 3.2.2.4. Framework7

Framework7<sup>5</sup> is a free and open source mobile HTML framework to develop hybrid mobile apps or web apps with iOS & Android native look and feel.

#### 3.2.2.5. jQuery mobile

jQuery mobile<sup>6</sup> is a HTML5-based user interface system designed to make responsive web sites and apps that are accessible on all smartphone, tablet and desktop devices. It allows creation of custom themes as well as configuration for laying out pages with minimal scripting. It is touch-optimized and platform-agnostic.

#### 3.2.2.6. jQuery validation

jQuery validation<sup>7</sup> makes simple client side form validation easy, whilst still offering plenty of customization options. It comes bundled with a useful set of validation methods, including URL and email validation, while providing an API to write your own methods.

#### 3.2.2.7. Flot

Flot<sup>8</sup> is a pure JavaScript plotting library for jQuery, with a focus on simple usage, attractive looks and interactive features. Supports all kinds of chart types such as line chart, bar chart, pie chart, and area chart. Furthermore, flot can be run on IE, Firefox, Chrome, Safari and Opera.

---

<sup>4</sup> <https://cdnjs.com/libraries/crypto-js>

<sup>5</sup> <http://framework7.io/>

<sup>6</sup> <https://jquerymobile.com/>

<sup>7</sup> <https://jqueryvalidation.org/>

<sup>8</sup> <http://www.flotcharts.org/>

#### 3.2.2.8. Mobile charts

Mobile charts<sup>9</sup> is a JavaScript library which provide convenience wrappers (around flot) for plotting consumption-related measurements on a mobile device. Supports line, bar and area charts for time-series events as well as charts for indexed based shower events.

#### 3.2.2.9. Moment

Moment<sup>10</sup> is a JavaScript library for validating, parsing, and manipulating dates and times.

### 3.3. Web UI elements

The development of the Web UI elements obviously follows the architectural decisions and applies the technologies of the DAIAD@home web application. To facilitate the reader, in the following we briefly present the core technologies and frameworks used.

#### 3.3.1. Application Patterns and Design

In the DAIAD@home web application we apply the Model View Controller pattern (MVC) and the Single Page Application (SPA) web application design. This pattern and design are used extensively and they strongly influence the structure of the source code. A short explanation for each follows; a broader coverage of these topics is outside the scope of this document:

- **Model View Controller (MVC).** The goal of the Model View Controller pattern is to separate code responsibilities into three parts. The Model, which represents application domain data and logic, the View, which is responsible for the data presentation and the Controller, who receives user interactions and updates the Model appropriately. This separation increases code testability and also improves a developer team's productivity. Nowadays, there are many variants of the MVC pattern and each MVC framework may implement the pattern in different ways. For the DAIAD implementation we are using the Spring Framework and its corresponding MVC module.
- **Single Page Applications (SPAs)** offer increased UI usability that is in par with desktop applications. In contrast to traditional web applications, a SPA application is initialized by loading only a single web page. After initialization, any additional resources such data or JavaScript code files are loaded dynamically on demand using Asynchronous JavaScript and XML (AJAX) requests. Moreover, client side code is usually implemented using the MVC pattern or some variant of it.

---

<sup>9</sup> <https://github.com/DAIAD/mobile-charts>

<sup>10</sup> <http://momentjs.com/>

### 3.3.2. Libraries and Frameworks

DAIAD@home web application executes on the client-side with authentication and data access achieved via communication with the backend APIs. It has been developed using the following technologies.

#### 3.3.2.1. React

React<sup>11</sup> is a JavaScript framework for building interactive User Interfaces. React can be thought as the View in the MVC pattern that allows users to build reusable UI components and promotes composition of existing ones. Each component maintains its internal state which controls the rendering process. Whenever state changes, only the parts of the Document Object Model (DOM) that are affected are updated. This is achieved by using a virtual representation of the DOM that efficiently detects changes to the actual DOM. The latter feature makes React interoperability with other UI libraries more challenging. Recommended templating in React is performed with the help of JSX<sup>12</sup>, an XML-like syntax with a smooth learning curve for HTML-familiar developers.

#### 3.3.2.2. Redux

Redux<sup>13</sup> is a predictable state container for JavaScript applications that is very popular for handling the increased application logic complexity that arises in Single Page Applications. In such applications the state management becomes increasingly harder since various user interactions –which quite often involve asynchronous requests - result in state changes. Redux attempts to manage state in a predictable way by imposing specific restrictions on how and when state updates can occur. Redux makes a perfect match to React by deferring component state management to Redux. It was based on the principle ideas of Flux<sup>14</sup> for making the flow of an application unidirectional. The main difference Redux introduced is the core idea of keeping the state in a single store (following a Single source of Truth principle), instead of multiple stores. The single application state maps at any moment to its view representation via React UI components. User actions such as clicks may dispatch actions that change the state in a predefined way with the help of reducers that dictate how a specific action modifies the application state. In that way a one-way flow is achieved, making the application easy to reason about, debug and scale. The abstract application flow is shown in Figure 33.

- Store: In redux the store holds the entire application state, which is the representation of the application at any given time. It is an object containing any number of valid JS data types, such as numbers, strings, booleans, arrays, or other objects. A key concept is that the state object cannot be mutated directly, but only by emitting actions.
- Actions: Actions can be dispatched by user interactions or other actions and cause the state to change. There are two types of actions, simple and complex actions or thunks that execute with the help of a special thunk middleware<sup>15</sup>. Simple actions are plain objects containing the unique action type and any data that needs to be passed to the store. Thunks are functions

---

<sup>11</sup> <https://facebook.github.io/react/>

<sup>12</sup> <https://facebook.github.io/jsx/>

<sup>13</sup> <http://redux.js.org/>

<sup>14</sup> <https://facebook.github.io/flux/docs/overview.html>

<sup>15</sup> <https://github.com/gaearon/redux-thunk>

that get access to the state and can perform asynchronous operations (such as fetching data from the API) and/or orchestrate multiple simple action dispatches.

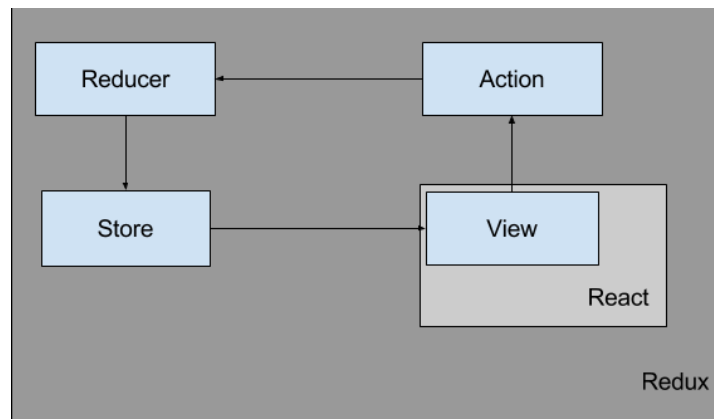


Figure 33: Redux Application flow

- **Reducers:** Reducers are pure functions that determine how an action modifies the state. Multiple reducers can be combined, each responsible for mutating a specific part of the state. Reducers do not mutate the state, but instead return a new state object, which allows easy recognition of any changes so that the view can be updated.

**Views:** Redux combines very well with React as its view layer with the help of react-redux library. In react-redux terminology, React components are divided into two types: smart components or containers and pure or presentational components. Containers are aware of redux and map parts of the state and action callback functions to react component properties, and are responsible for causing the components to re-render any time a mapped property has changed. On the other hand, presentational components are just pure functions of their input properties, completely ignorant of redux, allowing successful separation of logic and templating.

### 3.3.2.3. React-router

Application routing is handled by React-router<sup>16</sup>, a complete routing library for client-side routing for React applications. With react-router the user can enjoy a seamless experience when browsing through the different application pages even though they are actually rendered in a single page. React Router keeps the UI in sync with the URL. It has a simple API with powerful features like lazy code loading, dynamic route matching, and location transition handling built right in.

### 3.3.2.4. React-Router-Redux

React Router Redux is a JavaScript library that allows an application implemented using React and Redux to keep the application state in sync with routing information. This feature is achieved by automatically storing additional data about the current URL inside the state. This information is then propagated to React which can in turn suitably change the component tree rendering process. If there is no need for syncing routing information and application state, a simpler implementation can be

<sup>16</sup> <https://github.com/reactjs/react-router>

obtained by using the React Router<sup>17</sup> library. The latter provides support for keeping only the UI in sync with the URL.

### 3.3.2.5. React-Bootstrap

React-Bootstrap<sup>18</sup> is a library of reusable UI components for the React framework. It offers the look-and-feel of the Twitter Bootstrap<sup>19</sup> library using the React syntax, but has no dependencies on any 3<sup>rd</sup> party libraries like jQuery. React-Bootstrap offers a comprehensive list of UI components such as buttons, menus, form input controls, modal dialog, tooltips to name a few. All components can be used as provided or customized using CSS.

### 3.3.2.6. ECharts

ECharts<sup>20</sup> is a rich and versatile JavaScript charting library for building interactive charts, based on a standalone and lightweight rendering framework (ZRender). It manages (using an opaque handle) a given DOM node as a subtree, and directly draws to a canvas element lying inside this subtree. It supports a great variety of chart types including line, column, scatter, pie, radar, candlestick, chord, gauge, funnel, map and heatmap charts. It also supports data visualizations, not usually regarded as charts, including a tree map, a tree graph and a Venn diagram. Moreover, individual charts can be composed to create more complex data representations.

ECharts is highly optimized for handling hundreds of thousands (can easily cope with 200K) of data points, making it a seamless solution for big data analysis and visualization. Further, Echarts is *theme-able*, i.e. a great subset of appearance-related chart options can be supplied by referencing an external theme JSON object. This allows an entire applications to share (or override) a uniform look-and-feel for charts by keeping appearance-related options in a single place.

### 3.3.2.7. Reach-intl

Internationalization is based on react-intl<sup>21</sup>, a collection of JavaScript libraries developed by Yahoo for internationalizing React components. This library provides React components and an API to format dates, numbers, and strings, including pluralization and handling translations. The react-intl library can run both on server-side using Node.js, as well as the browser.

### 3.3.2.8. Grunt

Grunt<sup>22</sup> is a task runner used for development and deployment automation. The multitude of tools and transformations we introduced necessitated automated handling for the course of development and the production deployment for the web application. The most important tasks orchestrated by Grunt are listed below:

- NPM<sup>23</sup>: The popular package management library for Node.js for downloading and installing

---

<sup>17</sup> <https://github.com/reactjs/react-router>

<sup>18</sup> <https://react-bootstrap.github.io/>

<sup>19</sup> <http://getbootstrap.com/>

<sup>20</sup> <https://ecomfe.github.io/echarts/index-en.html>

<sup>21</sup> <http://formatjs.io/react/v1/>

<sup>22</sup> <http://gruntjs.com/>

<sup>23</sup> <https://www.npmjs.com/>

applications dependencies.

- Browserify<sup>24</sup>: Allows the modularization of javascript code in order to avoid global namespace pollution and several code transformations, such as babelify<sup>25</sup> for converting ES6 JS and JSX code to equivalent ES5 code that can execute in all browsers.
- Uglify & minify: Used for deployment in order to minimize the final bundle size that improves download time and thus improve user experience
- Sync: Performs the task of copying vendor files to necessary directories
- JSHint<sup>26</sup>: Used for early detection of errors or warnings in order to speed up development
- Watch: Used for development purposes for executing the entire task chain after any project file has been modified in order to reduce development time

### 3.3.2.9. Isomorphic fetch

Data fetching from the Data API backend is performed with the help of isomorphic-fetch<sup>27</sup> is a polyfill built on top of Github's window.fetch that implements the Fetch API specification<sup>28</sup> for making web requests from the browser. The global `fetch` function is an easier way to make web requests and handle responses than using a standard XMLHttpRequest.

### 3.3.2.10. Moment

Moment<sup>29</sup> is a JavaScript library for validating, parsing, and manipulating dates and times with ease in JS.

---

<sup>24</sup> <http://browserify.org/>.

<sup>25</sup> <https://github.com/babel/babelify>

<sup>26</sup> <http://jshint.com/>.

<sup>27</sup> <https://github.com/matthew-andrews/isomorphic-fetch>

<sup>28</sup> <https://fetch.spec.whatwg.org/>.

<sup>29</sup> <http://momentjs.com/>.

## 4. UI elements

In this section we present in detail the UI elements that comprise the Sustainability Dashboard. Our presentation follows the natural distinction of UI elements against their target platform, i.e. mobile devices (mobile phone, tablet) and web (browser). For each UI element we briefly describe its purpose, intended functionality, implementation details, and indicative examples for their invocation.

### 4.1. Web UI elements

#### 4.1.1. Charts

The Charts UI<sup>30</sup> element provides a flexible and direct way of presenting consumption data visualizations to users. It has been developed as an extensible and versatile component to cover all supported data sources and required visualization options.

The Charts UI element supports time-series, as well as distinct category values. Data can be visualized in multiple ways including line, bar and pie charts and is adjusted for quick setup, with the support for common customization features such as chart size, labels and series colors. User interaction is allowed by providing on-click callback functions that can perform custom actions. Further data exploration can be achieved by enabling an optional control that allows the user to limit the visible range in the chart if too many points or bars are on display and examine specific parts of the chart that interest him.

The Charts UI element has been implemented as follows:

- We have created a reusable Portal mixin class<sup>31</sup> from scratch in order to incorporate third-party components such as charts and maps into react components. The mixin class implements all functions necessary for handling the react lifecycle (e.g. mounting, unmounting). Also, we have included common functionality such as creating an enclosing element, adding class names or handling window events (e.g., resizing).
- We have developed a reusable presentational chart component as a wrapper to the [ECharts](#) JS library. Integration with React is achieved with the help of the previously mentioned Portal Mixin class we created. The Chart component provides a subset of commonly used options for easily creating Line, Bar and Pie charts from a wide range of available chart types and options available in the eCharts library.
- We have built a custom theme for the Chart UI element which complies with the general styling of DAIAD consumer applications. As the Chart element is a wrapper to the Echarts

---

<sup>30</sup> <https://github.com/DAIAD/home-web/blob/master/src/main/resources/public/assets/js/src/home/components/helpers/Chart.js>.

<sup>31</sup> <https://github.com/DAIAD/home-web/blob/master/src/main/resources/public/assets/js/src/home/components/helpers/PortalMixin.js>.

library, it is very easy to re-style according to needs.

The Charts UI element receives the parameters shown in the table below.

*Table 1: Charts UI element parameters*

Name	Type	Description	Example
<b>width*</b>	Number, String	The width of the container element in pixels or using css units.	100, 100%, 100vw
<b>height*</b>	Number, String	The height (pixels) of the container element in pixels or using css units.	200, 100vh
<b>type*</b>	String	The chart type. Must be one of 'line', 'bar', or 'pie'	Line, bar, pie
<b>xAxis</b>	String	The x axis type. Must be one of 'time', 'category'	time, category
<b>data*</b>	Array	An array of data series (see Table 2)	[{title: Athens, data: [5, 12]}]
<b>title</b>	String	The chart title	Average consumption
<b>subtitle</b>	String	The chart subtitle	per month
<b>mu</b>	String	The measurement unit for chart values display	lt
<b>colors</b>	Array	An array of colors for series that matches the order provided in data (acceptable css colors)	['red', '#666']
<b>invertAxis</b>	Boolean	Whether the x and y axis are inverted	False
<b>fontSize</b>	Number, String	The x, y labels font size	15
<b>clickable</b>	Boolean	Whether the points/bars/pie segments are clickable	True
<b>onPointClick</b>	Function	A callback that is called when a chart point/bar/pie segment is clicked with series, index integer parameters	function(series, index) { ... }
<b>dataZoom</b>	Boolean	Whether to show the data zoom for further zoom control on the chart	False
<b>dataZoomY</b>	String, Number	The position of the data zoom in pixels from the top or with a literal such as 'top', 'bottom'	bottom
<b>xMargin</b>	Number,	The left chart margin (number or css unit)	75

	String		
x2Margin	Number, String	The right chart margin (number or css unit)	10
yMargin	Number, String	The top chart margin (number or css unit)	20
y2Margin	Number, String	The bottom chart margin (number or css unit)	50

*Table 2: Data series parameters*

Name	Type	Description	Example
title*	String	The series title	Athens
data* (if xAxis is category)	Array	An array of distinct values in the same order as xAxis	[5, 12]
data* (if xAxis is time)	Array	An array of arrays containing a Date object and value	[[new Date("2016-05-30"), 5], [new Date("2016-06-15"), 12]]

In the following code snippet, we provide an example of how the Chart UI element can be invoked. In this particular example, we create an average rainfall line chart for 2016 in Athens and London.

```
<Chart {...{
  height: '350px',
  width: '100%',
  title: 'Average rainfall',
  type: 'line',
  subtitle: '',
  mu: 'mm',
  xAxis: "category",
  colors: ['#2d3480', 'black'],
  data: [{title: 'Athens', data:[55, 42, 38, 31, 28, 10, null, null,
null, null, null, null]}, {title: 'London', data: [52, 48, 45, 50, 60, 57,
null, null, null, null, null, null]}],
  xAxisData: ['January', 'February', 'March', 'April', 'May', 'June', 'July',
'August', 'September', 'October', 'November', 'December'],
  clickable: true,
  dataZoom: true,
  dataZoomY: 'bottom',
  onPointClick: function(series, index) { return null; },
  invertAxis: false,
  fontSize: 15,
  xMargin: 75,
  yMargin: 40,
  x2Margin: 20,
  y2Margin: 70
}} />
```

The output of rendering the Chart element with the above parameters can be seen in Figure 34.

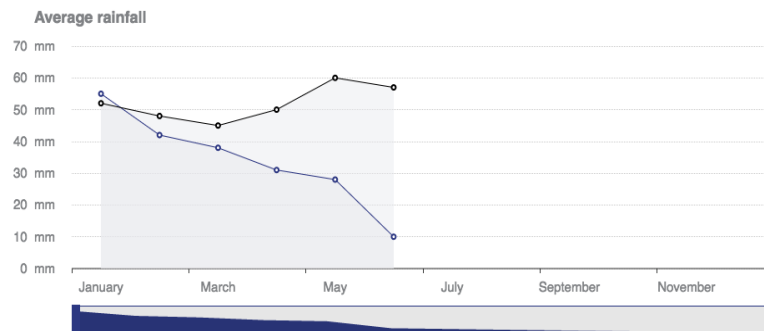


Figure 34: Example chart component invocation

Finally, in the following figures we present some indicative examples of how the Chart UI element is invoked and adapted in the context of the DAIAD@home web application.

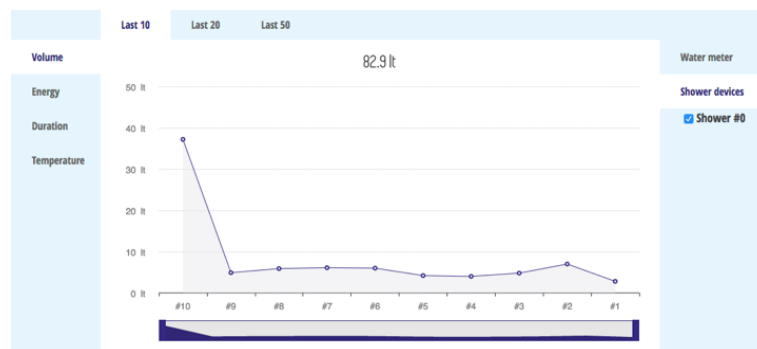


Figure 35: Index-based consumption data for an amphiro b1



Figure 36: Time-based consumption data for a SWM time-series

## 4.1.2. Widgets

Dashboard widgets<sup>32</sup> are UI elements that provide a quick overview of a user's consumption and can be fully customized by each user to fit her needs. Depending on the visualization type (chart, textual) and data source (SWM, amphiro b1), widgets can be one of the following:

- *Chart widgets* (line, bar or pie chart)
  - Total volume (SWM) is a line chart with aggregated data for the selected period
  - Water breakdown (SWM) is a bar chart presenting the breakdown of water per major

<sup>32</sup> <https://github.com/DAIAD/home-web/blob/master/src/main/resources/public/assets/js/src/home/components/sections/Dashboard.js#L43>.

- fixture category for the selected period
  - Forecast (SWM) is a bar chart presenting a forecast of water consumption (next month/year)
  - Comparison (SWM) is a bar chart displaying consumption comparisons against the user's neighborhood and similar homes for the selected period
  - Daily water budget (SWM) presents the current progress of the user's individual water budget
  - Last shower (b1) presents the water volume time-series for the last known shower
  - Last N showers volume/temperature/energy (b1) displays the water volume/temperature or energy for the last N showers
- *Textual widgets* (brief at-a-glance information)
  - Total volume (SWM) presents the total water consumption and its trend for the selected period
  - Last N showers volume/temperature/energy/energy efficiency (b1) displays the water volume/temperature or energy for the last N showers and a comparison with the previous N showers

Widgets of both types are split into three horizontal sections:

- Top section. Contains the title and any further customization options
- Main section. Differs depending on the widget type.
  - For *textual widgets* it is divided into two columns, one to the left taken up entirely by a highlight textual excerpt, and a column to the right with further information, such as comparisons.
  - *Chart widgets* have double the statistic widgets' height to provide more space for the chart that fills the entire main section. The chart can be either a line chart (e.g. for time-series), a bar chart with horizontal or vertical orientation, or a pie chart (e.g. for water budget).
- Bottom section. Provides a link to more detailed information.

The Dashboard widgets UI elements have been implemented as follows:

- We have created a Redux action<sup>33</sup> that handles data fetching from the API for all active widgets each time the user logs in or refreshes the page and -if successful- stores it in the application state.
- We have developed a smart React component<sup>34</sup> that maps the widget state to its properties and handles all necessary transformations on data fetch, in order to properly feed the presentational component.

<sup>33</sup> <https://github.com/DAIAD/home-web/blob/master/src/main/resources/public/assets/js/src/home/actions/DashboardActions.js#L185>.

<sup>34</sup> <https://github.com/DAIAD/home-web/blob/master/src/main/resources/public/assets/js/src/home/containers/DashboardData.js>.

- We have created a purely presentational React component<sup>35</sup> using JSX syntax and depending on the widget type (*chart* or *textual* widget) we render a specific React sub-component that handles the presentation for the main section in each case.
- The user can add or remove widgets at will. An 'X' symbol on the right of the top section removes the widget from display, while an 'Add widget' button triggers a modal which has been created to add widgets according to the user's preferences from a predefined selection (Figure 37).
- We have also created a responsive grid layout system<sup>36</sup> for dynamic manipulation and arrangement of the widgets using React-grid-layout<sup>37</sup>, built specifically for React. It allows resizing and dragging widgets and dynamically handles widget positioning based on the active window size. In all cases the user can save the active configuration and layout of widgets for future logins by clicking the Save button that appears when any change is detected.

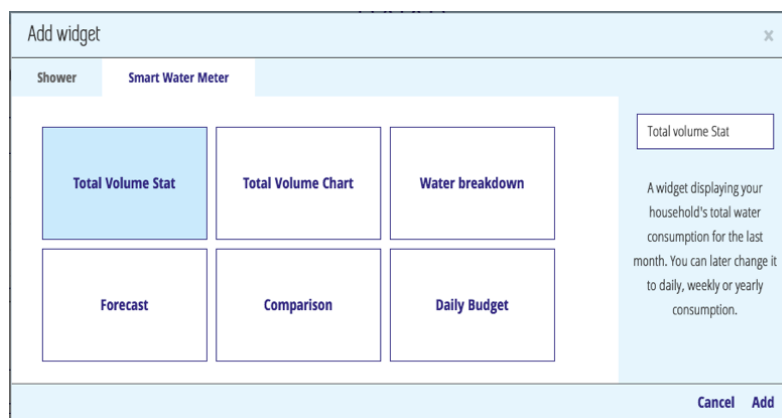


Figure 37: Add widget modal

The Widget UI element receives the parameters shown in the tables below.

Table 3: Widget UI element parameters

Name	Type	Description	Example
intl*	Object	The internationalization object provided by react-intl library	-
updateInfobox*	Function	Callback to handle links click	-
removeInfobox*	Function	Callback to handle delete widget click	-
infobox*	Object	The info box object (see Table 4)	

<sup>35</sup> <https://github.com/DAIAD/home-web/blob/master/src/main/resources/public/assets/js/src/home/components/sections/Dashboard.js#L43>.

<sup>36</sup> <https://github.com/DAIAD/home-web/blob/master/src/main/resources/public/assets/js/src/home/components/sections/Dashboard.js#L158>.

<sup>37</sup> <https://github.com/STRML/react-grid-layout>

Table 4: Infobox object parameters

Name	Type	Description	Example
display*	String	The widget display type. One of stat, chart	Stat, chart
title	String	The title to be displayed in the widget top-section	'Total consumption'
id*	String	A unique id that identifies the widget	'1'
periods	Array	An array of objects containing id and title (intl id) for extra options in the widget top-section	[[{id: 'year', title: 'periods.year'}, {id: 'month', title: 'periods.month'}]]
highlight (for textual widgets)	String	The highlight text to be displayed on the left column for textual widgets	25
mu	String	The measurement unit of the highlight text	lt
better (for textual widgets)	Boolean	A boolean whether the highlight metric was better than the comparison period	true
comparePercentage (for textual widgets)	Number	The percentage amount of improvement/degradation	20
chartData	Object	All options accepted by the Chart UI element (see Table 1)	

In the following we provide an example of how the Widget UI element can be invoked. In this particular example, we create a simple textual widget displaying the water consumption over the last 10 showers, which is 25% better than the previous 10 showers.

```
<InfoBox {...{
  intl: this.props.intl,
  infobox: {
    id: '1',
    periods: [{id: 'ten', title: 'periods.ten'}, {id: 'twenty',
title:'periods.twenty'}, {id:'fifty', title: 'periods.fifty'}]],
    highlight: 21,
    better: true,
    comparePercentage: 25,
    mu: 'lt',
    metric: 'volume',
    display: 'stat',
    title: 'Total volume',
    period: 'ten',
  }
}} />
```

The output of rendering the Widget component with the above parameters can be seen in Figure 38.

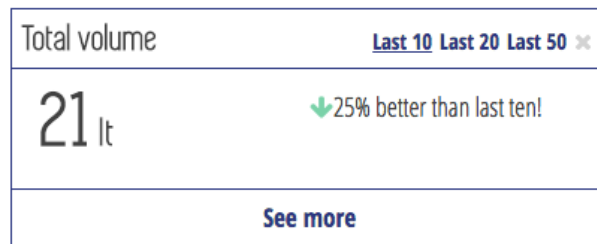


Figure 38: Example textual widget component invocation

Finally, in the following figures we present some indicative examples of how the Widget UI element is invoked and adapted in the context of the DAIAD@home web application in the Dashboard section.



Figure 39: Shower device last 20 showers' energy efficiency textual widget

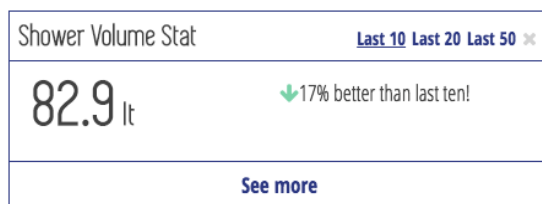


Figure 40: Shower device textual widget showing last 10 Showers' water consumption

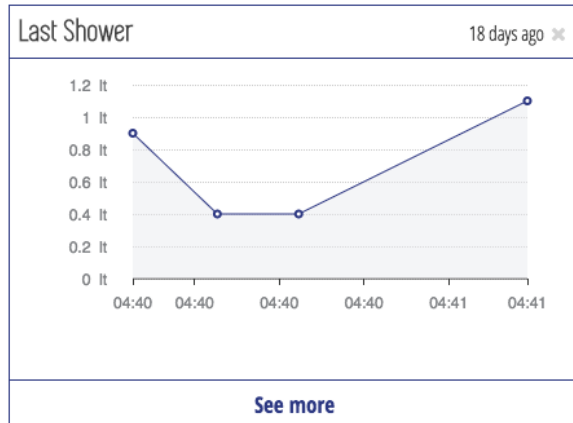


Figure 41: Last shower water flow chart widget

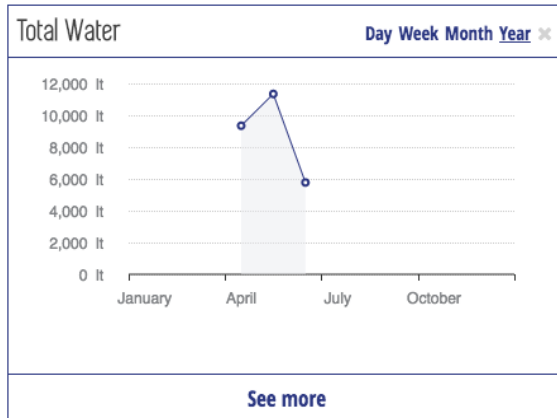


Figure 42: SWM total year water consumption chart widget

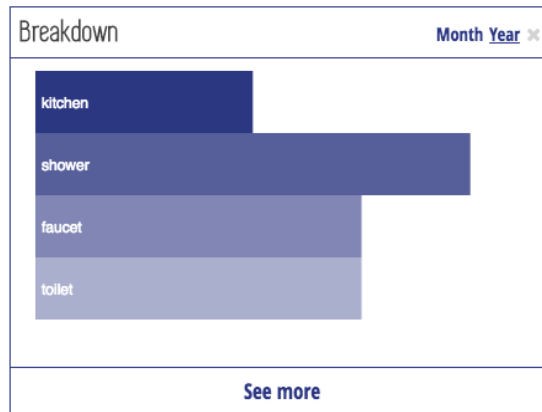


Figure 43: SWM Water breakdown chart widget

### 4.1.3. Events table

The Events table<sup>38</sup> UI element provides detailed information in tabular form for a selection of events and options to sort and extract the data. The Events table presents a selection of columns based on the device type (SWM, b1).

- For *SMMs*:
  - Water Volume
  - Reference time period
  - Simple comparison indicator (better, worse) comparing the current event with the immediate previous event
- For *b1* devices:
  - Water volume
  - Household member that had the shower
  - Shower date
  - Device name
  - Shower duration
  - Shower energy class
  - Average water temperature
  - Shower ID
  - Indication for real-time or historical shower event
  - Simple comparison indicator (better, worse) comparing the current event with the immediate previous event

<sup>38</sup> <https://github.com/DAIAD/home-web/blob/master/src/main/resources/public/assets/js/src/home/components/SessionsList.js#L99>.

The table of events can be sorted by different metrics (*time and volume for SWM and id, time, volume, energy, temperature and duration for b1*). Furthermore, the user can download the table data in CSV format by clicking the 'CSV' button. Note that the exported data maintains the format and sorting as selected by the element's controls. Each table item is clickable and the events details modal (link to events detail modal) is triggered on click with more detailed information.

The Events table UI element has been implemented as follows:

- We have created a Redux action<sup>39</sup> that handles data fetching from the API based on a series of user selections including device type (SWM, b1), period or range of showers, and -if successful- stores it in the application state
- We have developed a smart React component<sup>40</sup> that maps the data from the state to its properties and handles all necessary transformations on data fetch, in order to properly feed the presentational component
- We have created a presentational React component<sup>41</sup> using JSX syntax and HTML tables with predefined columns depending on whether SWM or Shower devices are active.

The Events table UI element receives the parameters shown in the tables below.

*Table 5: Events table UI element parameters*

Name	Type	Description	Example
sortOptions*	Array	Array of objects containing id and title with the available sort metrics	[{id: 'timestamp', title: 'sort.timestamp'}]
sortFilter*	String	The id of the active sort metric	timestamp
sortOrder*	String	Sort order. One of asc, desc	asc
activeDeviceType*	String	The active device type. One of 'METER', 'AMPHIRO'	METER
time*	Object	An object containing time information (see Table 6)	
sessions*	Array	An array of objects with the data to display in the table (see Table 7)	
csvData	String	A transformed string of the table sessions for download	"Device, Volume% total, Volume% difference,

<sup>39</sup> <https://github.com/DAIAD/home-web/blob/master/src/main/resources/public/assets/js/src/home/actions/HistoryActions.js#L174>.

<sup>40</sup> <https://github.com/DAIAD/home-web/blob/master/src/main/resources/public/assets/js/src/home/components/HistoryData.js>.

<sup>41</sup> <https://github.com/DAIAD/home-web/blob/master/src/main/resources/public/assets/js/src/home/components/SessionsList.js#L99>.

			Timestamp%0AC11FA586148% 2C81214%2C32%2C146724480 0000%0A"
--	--	--	--

*Table 6: Time object parameters*

Name	Type	Description	Example
startDate*	Number	The timestamp of the beginning of the period	1464739200000
endDate*	Number	The timestamp of the end of the period	1467631838603
granularity*	Number	The granularity for data aggregation. Accepted values: 0: second, 1: hour, 2: day, 3: week, 4: month	0

*Table 7: Sessions object parameters*


Name	Type	Description	Example
firstname*	String	The user's first name	John
date	String	The period range for aggregated data (only for SWM)	June 2016
timestamp*	Number	The timestamp of the exact shower time/aggregation time	1467631838603
devName	String	The name of the device to display (only for Shower device)	Shower #1
device	String	The device key used for CSV data extraction	fb430edf-179f-481a-b8c2-65a5493366f7
percentDiff	Number, null	If not null should contain the percentage of improvement/degradation against the immediate previous item in the table	-56, null
volume*	Number	The water volume (both for SWM and Shower devices)	21
friendlyDuration	String	The shower duration transformed for display (only for shower devices)	"01:25"
energyClass	String	The shower energy class (only for shower devices)	A+
temperature	String	The average shower temperature (only for shower devices)	36
history	Boolean	Whether it is a historical shower or not (only	false

		for shower devices)	
id	String	The shower id (only for shower devices)	789

In the following, we provide an example of how the Events table UI element can be invoked. In this particular example, we create a table displaying a single session for the month of June 2016.

```
<SessionsList {...{
  sortOptions: [{id: 'timestamp', title: 'sort.timestamp'}, {id:
'volume', title: 'sort.volume'}],
  sortFilter: 'volume',
  sortOrder: 'asc',
  activeDeviceType: 'METER',
  time: {
    startDate: 1464739200000,
    endDate: 1467244800000,
    granularity: 2
  },
  sessions: [{
    date: "June 2016",
    firstname: 'Stephen',
    volume: 81214,
    difference: 32,
    devName: "C11FA586148",
    device: "fb430edf-179f-481a-b8c2-65a5493366f7",
    percentDiff: null,
    timestamp: 1467244800000,
  }]
  csvData: "Device, Volume%0total, Volume%0 difference,
Timestamp%0AC11FA586148%2C81214%2C32%2C1467244800000%0A"
}} />
```

The output of rendering the Table UI element with the above parameters can be seen in Figure 44.

In detail 
Sort by:  

Volume	User	Date	
32lt	- Stephen	June 2016	>

*Figure 44: Example table element invocation for SWM*

Finally, in the following figures we present some indicative examples of how the Table UI element is invoked and adapted in the context of the DAIAD@home web application.

In detail

CSV

Sort by: ID

Volume

User

Date

Device

Dur

En

Temp

Real

Id

2.8 lt

↓

Stephen

18 days ago

Shower #0

00:40

A+

23 °C

✓

#33

>

7 lt

↑

Stephen

20 days ago

Shower #0

01:01

A+

34 °C

✓

#32

>

4.8 lt

↑

Stephen

20 days ago

Shower #0

01:01

A+

35 °C

✓

#31

>

4 lt

↓

Stephen

20 days ago

Shower #0

01:01

A+

30 °C

✓

#30

>

Figure 45: The events table when Shower device is selected

In detail	CSV	Sort by:	Time	↑
Volume		User	Date	
5771 lt	↓	Stephen	June 2016	>
11341 lt	↑	Stephen	May 2016	>
9342 lt	-	Stephen	April 2016	>

Figure 46: The events table when SWM is selected

#### 4.1.4. Events details modal

The Events details modal<sup>42</sup> provides detailed information about a specific event including tabular information and time-series charts (if available).

The Events details modal is divided into three sections:

- Top section. Contains the title and the close button
- Main section. Presents information about an event depending on the device type (SWM, b1) and can be one of the following:
  - SWM aggregated water consumption, which presents the aggregated data for a specific time period
  - Historical shower. Displays shower statistics, user, timestamp, device name, and

<sup>42</sup> <https://github.com/DAIAD/home-web/blob/master/src/main/resources/public/assets/js/src/home/components/Session.js#L141>.

comparison with the immediately previous shower.

- Real-time shower. Displays all available time-series for the particular shower (*volume, energy, temperature*) as well as all other information presented for a historical shower.
- Bottom section. Contains the Next and Previous events buttons, to visit other events in the list.

The Events details modal UI element has been implemented as follows:

- We have created a Redux action<sup>43</sup> that handles data fetching from the API in the case of Shower devices when the action that opens the modal is triggered and -if successful- stores it in the application state
- We have developed a smart React component<sup>44</sup> that maps the data from the state to its properties and handles all necessary transformations on data fetch, in order to properly feed the presentational component
- We have created a presentational React component<sup>45</sup> using JSX syntax. The React component rendering depends on whether the event is an SWM measurement or a shower. In the case of a real-time shower the Chart UI element (link to Chart UI element) is used to display a time-series line chart.

The Events details modal UI element receives the parameters shown in the tables below.

*Table 8: Events details modal UI element parameters*

Name	Type	Description	Example
showModal*	Boolean	A boolean which determines whether to show the session modal or not	true
intl*	Object	The internationalization object provided by react-intl library	-
activeDeviceType*	String	The active device type. One of 'METER', 'AMPHIRO'	METER
setActiveSession*	Callback	Function callback that is called when Next or Previous buttons are clicked and takes the following parameters: device key, shower id (for shower devices, otherwise null), timestamp (for SWMs,	function (dev, id, timestamp) { console.log(dev, id, timestamp); }

<sup>43</sup> <https://github.com/DAIAD/home-web/blob/master/src/main/resources/public/assets/js/src/home/actions/QueryActions.js#L79>.

<sup>44</sup> <https://github.com/DAIAD/home-web/blob/master/src/main/resources/public/assets/js/src/home/containers/SessionData.js>.

<sup>45</sup> <https://github.com/DAIAD/home-web/blob/master/src/main/resources/public/assets/js/src/home/components/Session.js#L141>.

		otherwise null)	}
<b>data*</b>	Object	An object containing the session data.	
<b>chartData (Shower devices with real-time data)</b>	Array	Array representing the time-series data to chart. It is an array of arrays each containing two elements: the point date as a JS Date Object and the value	[[new Date("2016-07-04T14:31:00"), 5], [new Date("2016-07-04T14:32:00"), 10]]

*Table 9: Events data object parameters*

Name	Type	Description	Example
<b>next*</b>	Array, null	Array of 3 values to link to the next event in the list. Its values are in turn: device key, shower id (for shower devices, otherwise null), timestamp (for SWMs, otherwise null)	["fb430edf-179f-481a-b8c2-65a5493366f7", 790, null], null
<b>prev*</b>	Array, null	Array of 3 values to link to the previous event in the list. Its values are in turn: device key, shower id (for shower devices, otherwise null), timestamp (for SWMs, otherwise null)	["fb430edf-179f-481a-b8c2-65a5493366f7", 788, null], null
<b>date (SWM)</b>	String	The period which the aggregated data concerns	June 2016
<b>firstname</b>	String	The logged-in user's first name for SWMs, or the family member first name that had the shower for Shower devices	John
<b>Id (Shower device)</b>	Number	The shower id	789
<b>History (Shower device)</b>	Boolean	Whether the shower is historical or not	true
<b>devName (Shower device)</b>	String	The device name to display	Shower #1
<b>Volume (Shower device)</b>	Number	The volume to display for both Shower devices and (in lt)	25
<b>difference (SWM)</b>	Number	The volume difference between two SWM measurements (in lt)	25

energy (Shower device)	Number	The total energy consumed during a shower (in W)	50
energyClass (Shower device)	String	The shower energy class	A+
temperature (Shower device)	Number	The average shower temperature (in C)	38
friendlyDuration (Shower device)	String	The duration to display	1 min
Timestamp (Shower device)	Number	The shower presumed shower time	1467644760681

In the following, we provide an example of how the Events details modal UI element can be invoked. In this particular example, we create a real-time Amphiro b1 shower event modal. The shower belongs to a list which has no previous elements and its next shower has an id of 790.

```
<SessionModal {...{
  showModal:true,
  setActiveSession: (dev, id, timestamp) => console.log(dev, id,
timestamp),
  intl:this.props.intl,
  activeDeviceType: 'AMPHIRO',
  data: {
    next: ["fb430edf-179f-481a-b8c2-65a5493366f7", 790, null],
    prev: null,
    firstname: 'Stephen',
    date: 'June 2016',
    id: 789,
    history: false,
    devName: 'Shower 1',
    percentDiff: -56,
    difference: 32,
    temperature: 38,
    friendlyDuration: '1 min',
    volume: 123,
    energyClass: 'A+',
    energy: 200,
    timestamp: 1467642771001
  },
  chartData: [[new Date("2016-07-04T14:31:00"), 5], [new
Date("2016-07-04T14:32:00"), 10]]
}} />
```

The output of rendering the Events details model with the above parameters can be seen in Figure 47.

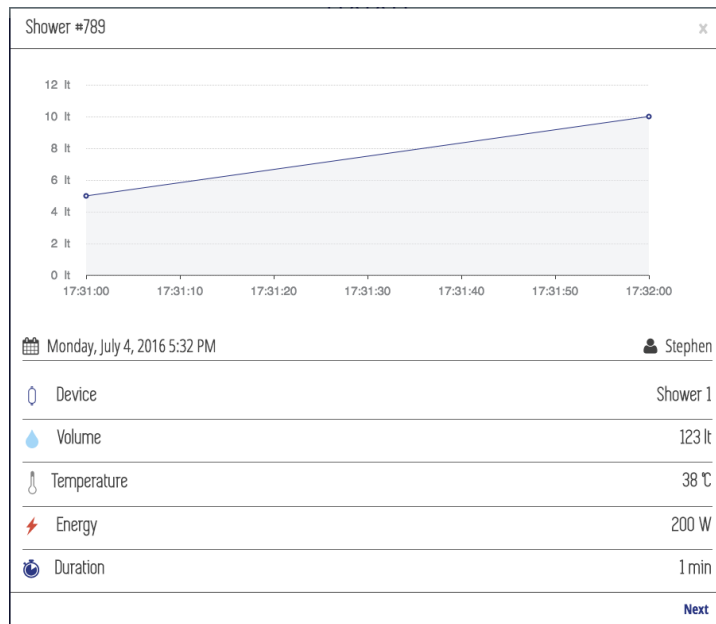


Figure 47: Example events detail modal UI element invocation for real-time shower

Finally, in the following figures we present some indicative examples of how the events details modal UI element is invoked and adapted in the context of the DAIAD@home web application in the Statistics section.

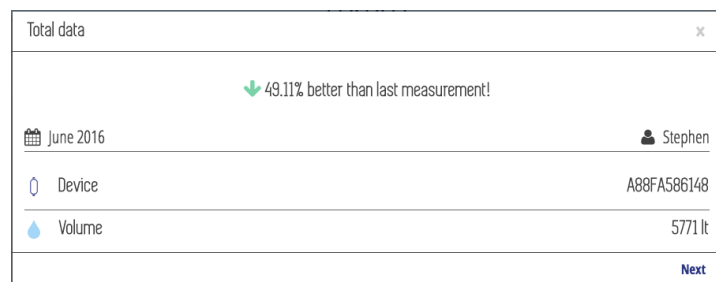


Figure 48: The event details modal for a total data aggregation for the month of June of 2016

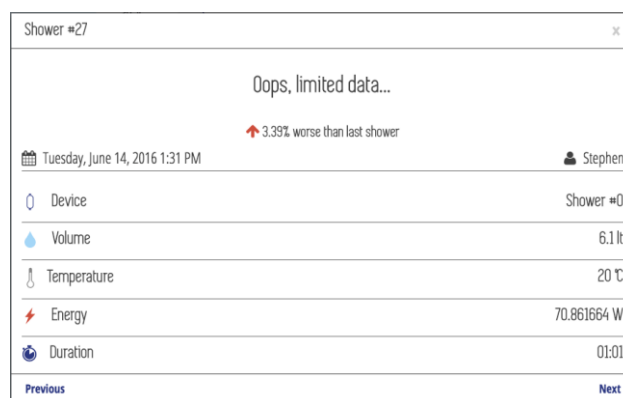


Figure 49: The event details modal for a historical shower event

## 4.1.5. Messages

The Messages UI element<sup>46</sup> lists the available messages for each category, and allows the user to view message details.

The Messages UI element is comprised of the following parts:

- A top bar listing the categories and allowing the user to switch between them
- The messages list for the selected category allowing the user to see more details for the message on click. Each message in the list includes:
  - The message title
  - A visible dot in case the message is still unread

The Messages UI element has been implemented as follows:

- We have created a Redux action<sup>47</sup> that handles data fetching from the API when the user logs in or refreshes the page and -if successful- stores it in the application state
- We have developed a smart React component<sup>48</sup> that maps the fetched message data from the state to its properties and handles all necessary transformations on data fetch, in order to properly feed the presentational component
- We have created a presentational React component<sup>49</sup> using JSX syntax. The React component renders the message list for the selected category using Bootstrap tabs and an HTML list element

The Messages UI element receives the parameters shown in the table below.

*Table 10: Messages UI element parameters*

Name	Type	Description	Example
intl*	Object	The internationalization object provided by react-intl library	-
categories*	Array	An array of objects that represent the tabbed categories. Each object must contain an id and title	[{id: 'alerts', title: 'Alerts'}]
messages*	Array	An array of objects containings all	

<sup>46</sup> <https://github.com/DAIAD/home-web/blob/master/src/main/resources/public/assets/js/src/home/components/sections/Notifications.js#L128>.

<sup>47</sup> <https://github.com/DAIAD/home-web/blob/master/src/main/resources/public/assets/js/src/home/actions/MessageActions.js#L115>.

<sup>48</sup> <https://github.com/DAIAD/home-web/blob/master/src/main/resources/public/assets/js/src/home/containers/MessageData.js>.

<sup>49</sup> <https://github.com/DAIAD/home-web/blob/master/src/main/resources/public/assets/js/src/home/components/sections/Notifications.js#L128>.

		available messages. See messages table for parameters	
<b>activeTab*</b>	String	The id of the active category	alerts
<b>setActiveTab*</b>	Function	Callback function for handling categories tab click. Takes a single parameter which is the category id	function(catId) { console.log(catId); }
<b>activeMessageId*</b>	Number, null	The active message id if active, or null otherwise	11, null
<b>setActiveMessageId*</b>	Function	Callback function for handling message click. Takes a single parameter which is the message id	function(msgId) { console.log(msgId); }

In the following, we provide an example of how the Messages UI element can be invoked. In this particular example, we create a sample messages element with two categories, and two alert messages one of which is unread.

```
<Notifications {...{
  intl: this.props.intl,
  categories: [{id: 'alerts', title: 'notifications.alerts'}, {id:
'tips', title: 'notifications.tips'}],
  messages: [{
    id: 12,
    title: 'Hello world!',
    Type: 'ALERT',
    description: 'This is a test message',
    acknowledgedOn: 1467648083879
  },
  {
    id: 13,
    title: 'Watch out, you have been using too much water!',
    Type: 'ALERT',
    description: 'We suspect there might be a leak, since your
water consumption has been increased dramatically over the last few hours.
Please check for leaks and take all necessary actions!',
    acknowledgedOn: null
  }
],
  activeMessageId: null,
  previousMessageId: null,
  nextMessageId: null,
  setActiveMessageId: (msgId) => console.log(msgId),
  activeMessage: {},
  activeTab: 'alerts',
  setActiveTab: (catId) => console.log(catId)
}} />
```

The output of rendering the Message element with the above parameters can be seen in Figure 50.



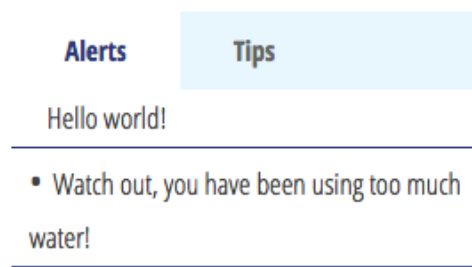


Figure 50: Example messages UI element invocation

Finally, in the following figure we present an indicative example of the Messages UI element in the context of the DAIAD@home web application.

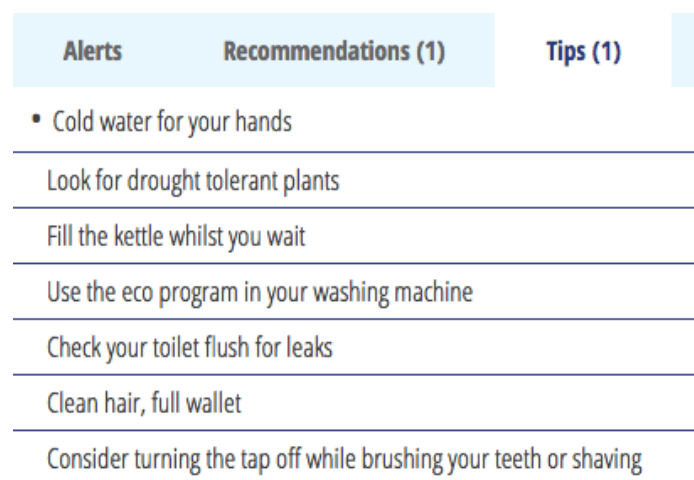


Figure 51: Messages UI element with tips category selected

#### 4.1.6. Message details

The Message details UI element<sup>50</sup> displays statically or dynamically produced information to the user in order to notify her of situations requiring attention or motivate her with towards improving water efficiency.

Message types are divided in the following categories (a list of all Messages is available in Annex: Messages):

- Alerts, which require immediate attention
- Tips, which are generic suggestions for improving water efficiency
- Insights, which are personalized recommendations for increasing efficiency

The Message UI element includes the following parts:

- A title, which is a short summarized text of the message
- The message description, which provides more textual details

<sup>50</sup> <https://github.com/DAIAD/home-web/blob/master/src/main/resources/public/assets/js/src/home/components/sections/Notifications.js#L15>

- Further visualizations which help emphasize and clarify the message information. Visualizations include:
  - Images, which accompany tip messages
  - Charts, which show personalized user consumption data in order to clarify a point being made, such as showing the last 10 showers' temperature for all shower devices.
- A short text showing whether the message has been read
- A next and previous button at the bottom allows the user to browse her available messages in the same category with ease

The Message details UI element has been implemented as follows:

- The message details UI element is fed data from the container we created for the Messages UI element<sup>51</sup>, by selecting the active message from the application state given the message category and id on user click
- When a message of type alert is clicked that requires displaying user-data charts, we call a reusable Redux action<sup>52</sup> that fetches data based on a predefined JSON for the specific alert message type, and stores the data in the messages state
- We have created a presentational React component<sup>53</sup> using JSX syntax. The React component renders the message type, description, image in base-64 if present and a chart if applicable using the Charts UI element.

The Message details UI element receives the parameters shown in the table below.

*Table 11: Message details UI element parameters*

Name	Type	Description	Example
id*	Number	A unique message id	12
title*	String	The message short title	Message title
type*	String	The message type. One of 'ALERT', 'ANNOUNCEMENT', 'RECOMMENDATION_STATIC', 'RECOMMENDATION_DYNAMIC'	ALERT
description*	String	The detailed message description	Message description

<sup>51</sup> <https://github.com/DAIAD/home-web/blob/master/src/main/resources/public/assets/js/src/home/containers/SessionData.js>.

<sup>52</sup> <https://github.com/DAIAD/home-web/blob/master/src/main/resources/public/assets/js/src/home/actions/QueryActions.js#L215>

<sup>53</sup> <https://github.com/DAIAD/home-web/blob/master/src/main/resources/public/assets/js/src/home/components/sections/Notifications.js#L15>

<b>previousMessageId*</b>	Number, null	The previous message id (if existing) in the message category list	11, null
<b>nextMessageId*</b>	Number, null	The next message id (if existing) in the message category list	13, null
<b>infobox</b>	Object	The infobox object containing necessary data for creating Chart	

In the following, we provide an example of how the Message details UI element can be invoked. In this particular example, we create an alert message with a JSON to create a chart from real-user data.

```
<NotificationMessage {...{
  notification: {
    id: 12,
    title: 'Watch out, you have been using too much water!',
    type: 'ALERT',
    description: 'We suspect there might be a leak, since your
water consumption has been increased dramatically over the last few hours.
Please check for leaks and take all necessary actions!',
    acknowledgedOn: 1467648083879
  },
  previousMessageId: null,
  nextMessageId: 13,
  infobox: {
    type: "total",
    display: "chart",
    period: "day",

    deviceType: "METER",
    metric: "difference",
    chartType: 'line',
    chartCategories: ['03:00', '04:00', '05:00', '06:00', '07:00',
'8:00', '9:00', '10:00', '11:00'],
    mu: 'lt',
    chartData: [{
      title: "C11FA586148",
      data: [22, 19, 26, 6, 80, 100, 120, 142, 167]
    }],
  }
}} />
```

The output of rendering the Message details UI element with the above parameters can be seen in Figure 52.



Figure 52: Invocation of example message details UI element with chart

Finally, in the following figure we present an indicative example of how the Events details modal UI element is invoked and adapted in the context of the DAIAD@home web application in the Messages section.

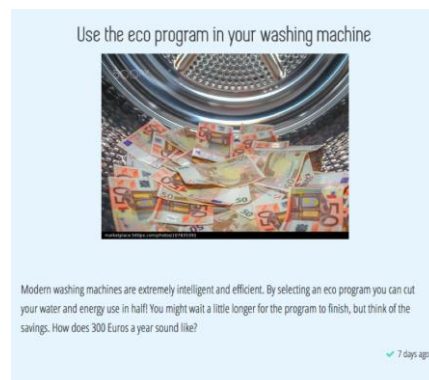


Figure 53: Tip message with image

## 4.2. Mobile UI elements

### 4.2.1. Dashboard Gauge

The Dashboard Gauge UI element comprises a central gauge and four complications focused on presenting a customizable overview of water use, as well as shortcuts to key statistics.

The central gauge displays information for a specific water consumption type (e.g. Total consumption) and the three out of four complications provide further water use information about last water consumptions as well as shortcuts to key statistics and other app functionalities. The last one is a static complication where the user can customize the entire gauge by setting personal preferences. In particular:

- Metrics. We have built the water flow, temperature, duration and money spent metrics

complications that can provide information about the average consumption over the last ten showers. Each complication is an html element `<a href="#" id="#"></a>` with a unique id and a link as attributes. Each of the aforementioned metrics complications has a title and an associated value e.g. Temperature 37C, Duration 37min

- Goals. We have developed the budget complication using the JQuery-plugin-circliful (SVG-based), contained and initialized in an html element.
- Trends. We have developed an efficiency rating complication that can provide information about the user's energy class (e.g. A, A+). We have used html `<img>` elements and media assets to visualize the efficiency rating.
- Ranking. We have developed the ranking complication that provides information about the current user's ranking in city and other similar homes.

The Dashboard Gauge UI element receives the parameters shown in the tables below.

*Table 12: Central Gauge parameters*

Name	Type	Description	Example
ConsumptionType	Number	A number from 0 to n, where n is the number of consumption types. Each consumption type is a list element.	4
callback	Function	A function that returns the water consumption	{ 'volume': 100, 'energy': 100 }

*Table 13: Metrics parameters*

Name	Type	Description	Example
title	String	Localized html element	Water flow, Temperature
value	Number	Metric value. This value is associated with the title	120

*Table 14: Budget parameters*

Name	Type	Description	Example
text	String	The text to be displayed inside the	'75%' or '75%

		circle	remaining'
percentage	Number	The percentage for circle's animation	75 out of 100

*Table 15: Efficiency rating parameters*

Name	Type	Description	Example
value	Number	A value used for computing the energy class.	1234
value	Number	A value used for computing the ranning	125

In the following, we provide details on the how the Dashboard Gauge UI element can be invoked. In this particular example, we create a new Dashboard Gauge overview for a household's Total consumption, with Water flow, Temperature, and Duration as active complications.

- The function `getConsumption` takes as arguments a chosen consumption type and a callback function. The first argument, number 4, belongs to the list element with title 'Daily Consumption' while the callback function holds the consumption data for the aforementioned type.

```
app.getConsumption (1,function({
  'volume' : 25,
  'energy' :0,
  'duration' : 120
})){
  var regg = new dashboardManager(data);
  regg.refreshDashboard();
}}
```

- The function `getLastTenShowers` takes as an argument a callback function and then calls the refresh function to set the active callbacks.

```
app.getLastTenShowers (function({
  "volume":23,
  "energy":0,
  "temp":26,
  "flow":5,
  "duration":120
})){
  var regg = new dashboardManager(data);
  regg.refreshComplications();
}}
```

The output of rendering the Dashboard Gauge UI element with the above parameters can be seen in Figure 54.

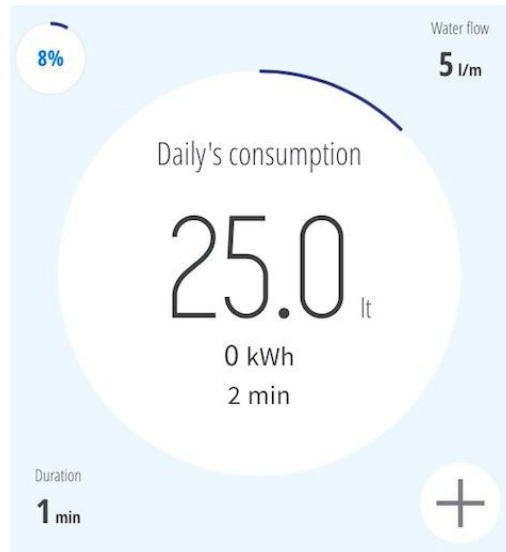


Figure 54: Dashboard Gauge UI element output

In the following figures we present some indicative examples of how the Dashboard Gauge UI element is invoked and adapted in the context of the DAIAD@home mobile application.



Figure 55: Dashboard Gauge overview



Figure 56: Dashboard Gauge overview

## 4.2.2. Chart

The Chart UI element provides a flexible and direct way of presenting consumption data visualizations to users. It has been developed as an extensible and versatile component to cover all supported data sources and required visualization options.

The Chart UI element has been developed to support the visualization of water consumption data SWMs (time-series) and amphiro b1 devices (shower sequences, time-series). Data can be visualized in multiple ways including bar, line and area charts setting customizable visualization options for

each type. Data exploration can be achieved by using the data limit controls that allow the user to change the range of the drawn chart and examine specific parts of the plotted data. Further, the Chart UI element can plot time-series data per day, week, month and year as well as single events based on the shower index. Finally, the Chart UI element has full-screen support (landscape mode) and swipe events (touch-based).

The Chart UI element has been implemented as follows:

- We have developed a CommonJS-compatible module that provides functions to assist plotting for all available data sources (SWM, amphiro b1) and needed time scales (e.g. week, month). All plotting functions wrap jQuery.Flott function by providing axis data and shaped (or simply mapped) series data. Apart from plotting functions, this module also hosts some utility functions and a large amount of plotting and theming defaults. The core module-level functions are:
  - generateTicks: generate ticks for a given numeric range and a given minimum resolution. This is a utility function that estimates the order of magnitude (of numbers in that range), chooses a meaningful tick step and precision, and then generates ticks on that computed step.
  - B1.plotForEvent: plot shower events from a b1 device based on their index (events are not directly associated to time). This is a plotting function that expects series of successive index-based events and produces a corresponding line/bar chart. Generates ticks using `generateTicks`.
  - Meter.plotForDay: plot consumption data from SWM measurements of a given day. This is a plotting function that expects series of time-based events and produces a bar chart. Uses a resolution of 1 hour and generates ticks using `generateTicks`.
  - Meter.plotForWeek: plot consumption data from SWM measurements of a given week. This is a plotting function that expects series of time-based events and produces a bar chart. Uses a resolution of 1 day and generates ticks for all week days.
  - Meter.plotForMonth: plot consumption data from SWM measurements of a given month. This is a plotting function wrapping jQuery.Flott that expects series of time-based events and produces a line chart. Uses a resolution of 1 day and generates ticks for the beginning of weeks.
  - Meter.plotForYear: plot consumption data from SWM measurements of a given year. This is a plotting function that expects series of time-based events and produces a line chart. Uses a resolution of 1 month, generates ticks using `generateTicks`.

The Chart UI element receives the parameters shown in the tables below.

*Table 16: Plot parameters*

Name	Type	Description	Example
selector	jQuery selector	Place to draw the graph	\$('#placeholder')
data	Array	Shower data to plot. The	(1, 123345567, 150)



		array contains the number of the shower, a UNIX timestamp and a value	
config	Object	Configuration object for graph initialization(see Day – Week config table below)	{resolution: 1 }

*Table 17: Plot day week config parameters*

Name	Type	Description	Example
resolution	Number	xAxis granularity e.g. 1 for day , 2 for week etc.	resolution = 1
levels	Array	Different bar chart colors based on the volume value	range: [0, 200], color: 'blue'

*Table 18: Plot month parameters*

Name	Type	Description	Example
selector	jQuery selector	Place to draw the graph	\$('#placeholder')
data	Array	Shower data to plot. The array contains the number of the shower, a UNIX timestamp and a value	(1, 123345567, 150)
config	Object	Configuration object for graph initialization (see Plot Month config table below)	{resolution: 1 }

*Table 19: Plot month parameters*

Name	Type	Description	Example
resolution	number	xAxis granularity e.g. 1 for day , 2 for week etc.	resolution = 1
weekLabel	String	Get the appropriate localized name of the week	English : Week Spanish : Semana

Table 20: Plot year parameters

Name	Type	Description	Example
selector	jQuery selector	Place to draw the graph	\$('#placeholder')
data	Array	Shower data to plot. The array contains the number of the shower, a UNIX timestamp and a value	(1, 123345567, 150)
config	Object	Configuration object for graph initialization(see Plot Year config table below)	{resolution: 1 }

Table 21: Plot year parameters

Name	Type	Description	Example
resolution	number	xAxis granularity e.g. 1 for day , 2 for week etc.	resolution = 1

Table 22: Plot for event parameters

Name	Type	Description	Example
selector	jQuery selector	Place to draw the graph	\$('#placeholder')
showers	Array	Shower data to plot. The array contains the number of the shower, a UNIX timestamp and a value	(1, 123345567, 150)
config	Object	Configuration object for graph initialization(see Plot for event config table below)	{colors: 'blue' }

Table 23: Plot for event config parameters

Name	Type	Description	Example
colors	Number	Color value	Colors= 'blue'

bars	Boolean	True for bars , false for line	bars = true
xaxis	Object	Object with graph configuration( see table below)	{bars : true}
ticks	Number	Number for yAxis and xAxis ticks generation	{ticks : 2}

In the following, we provide an example of how the Chart UI element can be invoked. In this particular example, the function plotForEvent will create a new bar chart for ten shower events. The showers array contains ten sub-arrays holding the shower indexes and the volumes, while the config json object holds customization options.

```
showers = [[1,
37.20], [2,4.90], [3,5.90], [4,6.10], [5,6.00], [6,4.20], [7,4.00], [8,4.80], [9,7.00], [10,
2.80]];
config = {
    color:'#fff',
    bars: true,
    xaxis: {
        ticks:5 ,
    },
};
charts.bl.plotForEvent($('#events'), showers, config );
```

The output of rendering the Chart UI element with the above parameters can be seen in Figure 57.

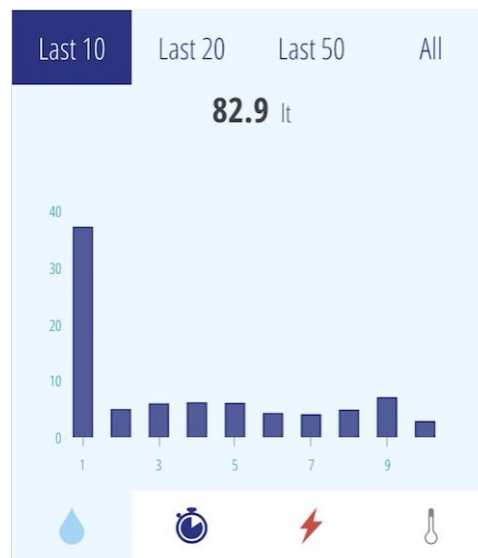


Figure 57: Chart UI element output

Finally, in the following figures we present some indicative examples of how the Chart UI element is invoked and adapted in the context of the DAIAD@home mobile application.



Figure 58: Plot for event bar chart



Figure 59: Plot for event bar chart

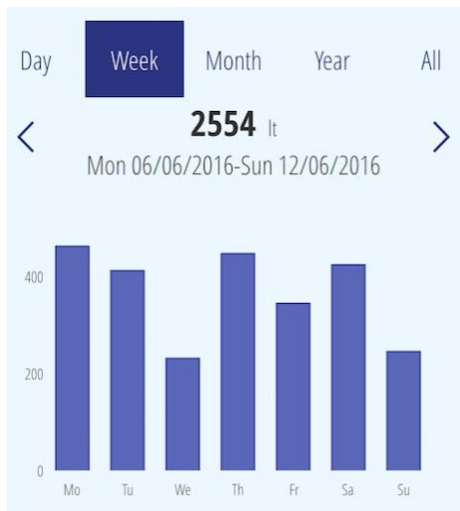


Figure 60: Plot for week bar chart



Figure 61: Plot for month area chart



Figure 62: Plot for week full screen bar chart



Figure 63: Plot for month full screen area chart

### 4.2.3. Comparison

The Comparison UI element provides a flexible and direct way of presenting social comparison data to users. It has been developed as an extensible and versatile customizable component to cover all supported groups comparisons.

The Comparison UI element consists of two different representations, a horizontal bar chart and a linear bar chart. The horizontal bar chart displays a household's water consumption for a particular time period compared to the average consumption of other consumer groups (e.g. similar households, city average). The linear bar displays the percentile (10) in which a households water consumption belongs to against the total population.

The Comparison UI element has been implemented as follows:

- We have developed a function that fetches consumption data for the required groups from the Data API and updates the local storage.
- Moreover, we feed the Charts UI element with the fetched data in order to display the two different representations: horizontal bar charts and linear bar charts.

The Comparison UI element receives the parameters shown below.

*Table 24: Comparison UI element parameters*

Name	Type	Description	Example
selector	Html element	jQuery selector	\$('#placeholder')
data	array	Arrays of arrays	[[ 'similar', 120 ]]
config	Json object	Configuration file for labels, colors and custom styles	{ paddingX: 8, margin: 8, align: 'right' }

In the following, we provide details on the how the Comparison UI element can be invoked.

- The function `plotBarsWithLabels` visualizes the consumption data as a horizontal bar chart. It takes as arguments a jQuery selector, data and a configuration file containing labels, colors and other custom styles.

```
charts.comparison.plotBarsWithLabels($('#bar-placeholder'), data, config);  
data = [[ 'similar', 120 ], [ 'me', 150 ], [ 'city', 220 ], ];  
config = {  
    // Provide labels for data points  
    points: new Map([  
        [ 'similar', {  
            label: 'Similar',  
            color: '#2D3580',  
            labelColor: '#FFF',  
        } ],  
        [ 'me', {  
            label: 'Me',  
        } ],  
    ]),  
};
```

```

        color: '#7BD3AB',
        labelColor: '#FFF',
    }],
    ['city', {
        label: 'City',
        color: '#A4D5F5',
        labelColor: '#445C92'
    }],
    ]),
    // Style labels
    labels: {
        paddingX: 8,
        marginX: 8,
        align: 'right',
    },
};

```

- The function `plotBarsWithMarkers` visualizes the consumption data as a linear bar chart. It takes as arguments a jQuery selector, data and a configuration file containing labels, colors and other custom styles.

```

charts.comparison.plotBarsWithMarkers($('#placeholder'), data, config)
data = [ ['best', 200], ['avg', 150], ['me', 100], ];
config = {
    // Define the expected range of values
    range: [50, 200],
    // How many steps in the linear
    //scale (keep it between 10 and 20)?
    numSteps: 10,
    // Provide labels for markers on data points
    points: new Map([
        ['me', {
            label: 'Me',
            color: '#3D97D8',
            labelColor: '#FFF',
        }],
        ['avg', {
            label: 'Avg',
            color: '#CE363B',
            labelColor: '#FFF',
        }],
        ['best', {
            label: 'Best',
            color: '#7BD3AB',
            labelColor: '#445C92'
        }],
    ]),
    // Style background bars (linear scale)
    bars: {
        color: '#BBB',
        widthRatio: 0.35,
    },
};

```

The output of rendering the Comparison UI element with the above parameters can be seen in Figure 64.

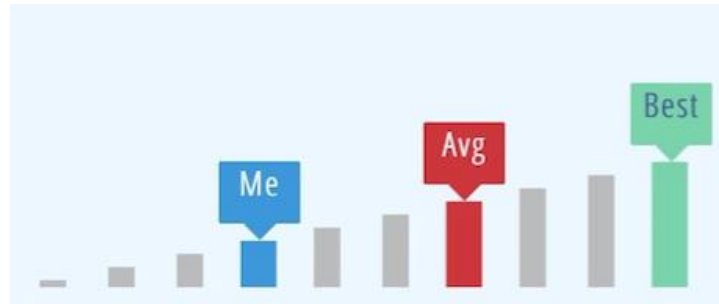


Figure 64: Comparison UI element output

In the following figures we present another example of how the Comparison UI element is adapted in the context of the DAIAD@home mobile application.

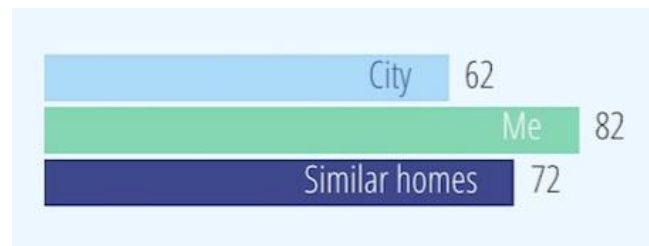


Figure 65: Horizontal comparison chart

#### 4.2.4. Gauge

The Gauge UI element consists of a central gauge focused on enabling users to set individual consumption goals. It has been developed to visualize any kind of consumption related events and to inform users when they exceed their targets.

The Gauge UI element has been implemented as follows:

- We used the jQuery-plugin-circlful plugin to create the circular HTML element and the presentation animations
- We modified the plugin to accept an extra text input parameter to display in the circle

The Gauge UI element receives the parameters shown in the table below.

Table 25 Gauge UI element parameters

Name	Type	Description	Example
data-percent	Number	Number from 0 to 100.	75
data-text	String	Main gauge text.	12 min 00 sec
data-info	String	Text. Optional	Temp 21C
data-fgcolor	String	Percentage color	#61a9dc
data-bgcolor	String	Border color	#eee

data-fill	String	Background color	#ddd
-----------	--------	------------------	------

In the following, we provide details on the how the Gauge UI element can be invoked.

First, we create a div container and pass the parameters to the chart by using the data-\* attributes:

```
<div id="myGauge" data-text="2 min 20 sec" data-info="Temp 22C" data-percent="24" data-fgcolor="#61a9dc" data-bgcolor="#eee" data-fill="#ddd"></div>
```

Then we initialize the Gauge UI element:

```
$('#myGauge').circliful();
```

Finally, we pass the data-\* options directly to the initialization process

```
$('#myGauge').circliful({
    data-percent: 24,
    data-text: "2 min 20 sec",
    data-info: "Temp 22C",
    data-fgcolor: "#61a9dc"
data-bgcolor: "#eee",
data-fill: "#ddd"
});
```

The output of rendering the Gauge UI element with the above parameters can be seen in Figure 66.



Figure 66: Gauge UI element output

## 4.2.5. Messages

Messages is a UI element comprising of a table with three specific message categories focused on presenting an overview of personalized messages to users.

The Messages UI element has the form of a mailbox and is separated into three major categories (Alerts, Tips, Insights). Each one includes a title that summarizes the point being made and a detailed description for the consumer to read. In the case of Tips we have included images to emphasize the point and motivate the consumer. Alerts may include charts that present specific consumption data the user needs to see in order to take action depending on the message. Finally, Insights are the personalized recommendations and include a title and a short description.



The Messages UI element has been implemented as follows:

- We have developed a function that fetches all the messages from the Data API during application initialization, and updates the messages in the local storage if necessary.
- For each message category we create an HTML tab with the category title and the category unread messages and an HTML list element containing the title, description and an HTML bullet if unread.

The Messages UI element receives the parameters shown in the table below.

*Table 26 Message UI element parameters*

Name	Type	Description	Example
id	Number	Number of message	15
title	String	Message short title	Reached 50% of your budget alert
txt	String	Message description	Using your dishwasher only when it is full can save water and energy ...
time	String	Human readable format	11:30 am
results	Array	Array containing budget values. The values are extracted directly from the message	[10,20,30]

In the following, we provide an example of the how the Message UI element can be invoked. In this particular example, we create a new Budget Alert. The function `rebuildMessages` internally reads all the available messages and create the list elements. It is called every time a new message is available.

```
{
  "id" : 31,
  "time":14588644782,
  "title":' Cold water for your hands',
  "txt" : ' Lather your hands before opening the tap and only rinse your hands
with cold water.',
  "type": 'RECOMMENDATION_STATIC',
}
rebuildMessages()
```

The output of rendering the Messages UI element with the above parameters can be seen in Figure 67.

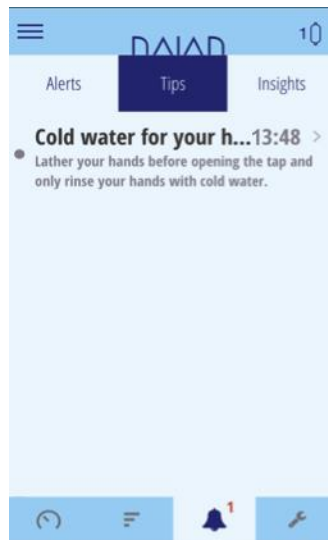


Figure 67: Messages UI element example

In Figure 68, we present an example of how the Messages UI element is adapted in the context of the DAIAD@home mobile application



Figure 68: Messages UI element

#### 4.2.6. Message details

The Message details UI element is focused on presenting a particular message to users, supporting three message categories: 'Alerts' (events requiring immediate attention), 'Tips' (generic suggestions for improving water efficiency), and 'Insights' (personalized recommendations for increasing efficiency). A complete list of available messages is provided in Annex: Messages.

The Message details UI element has been implemented as follows:

- Media assets. We have used html <img> elements as back and forward buttons for message navigation
- Title section. We have built an html <p> element with specific styles for the message title.

- Chart section. We have built a middle section to append charts, if exists, such as budget charts for budget alerts
- Description section. We have built an html <p> element with specific styles for the message description
- Alerts. We have developed an html template to handle all the Alerts except those with type “BUDGET”. This template consists of two parts. One for the message title and one for the message description.
- Budget Alerts. We have developed an html template to handle budget alerts. This template is separated into three parts. For the first and third parts, we have included a html <p> element with specific styles that presents the message title and message description. For the second part, we have used the Gauge UI element to present values and percentages as circle statistics
- Tips. We have developed an html template that handles the Tips. We have included two parts that provide information about the message title and the description and one more part as html <img> element to present images.
- Insights. We have developed an html template to handle the insights. This template consists of two parts. One for the message title and one for the message description. Furthermore we have added a specific html <h1> element to highlight important information such as numbers.

The Message details UI element receives the parameters shown in the table below.

Name	Type	Description	Example
type_id	Number	Number of message	15
type	String	Type of message. May be BUDGET, RECOMMENDATION_STATIC, RECOMMENDATION_DYNAMIC, ANNOUNCEMENT etc.	BUDGET
val	Number	Value for Budget alerts. Default is null	200
short	String	Message short description. Act like title	Reached 50% of your budget alert
long	String	Message long description	Using your dishawater only when it is full can save water and energy

*Table 27: Message details UI element parameters*

In the following, we provide details on the how the Message details UI element can be invoked. In this particular example, we create a new Budget Alert Message for the user.

Function createMessagePage take as arguments a unique message identifier, a short string (message title), the long message description, a value (e.g. 100), and the type of the message (e.g. BUDGET or RECOMMENDATION\_STATIC).



```
app.createMessagePage(15,"RECOMMENDATION_STATIC",200,"make sure your dishwasher is full",
"Using the dishwasher only when it is full can save you water and energy. Over a year, you could be saving up to 1,000 liters liters of water and 150 Euros")
```

The output of rendering the Message details UI element with the above parameters can be seen in Figure 69.

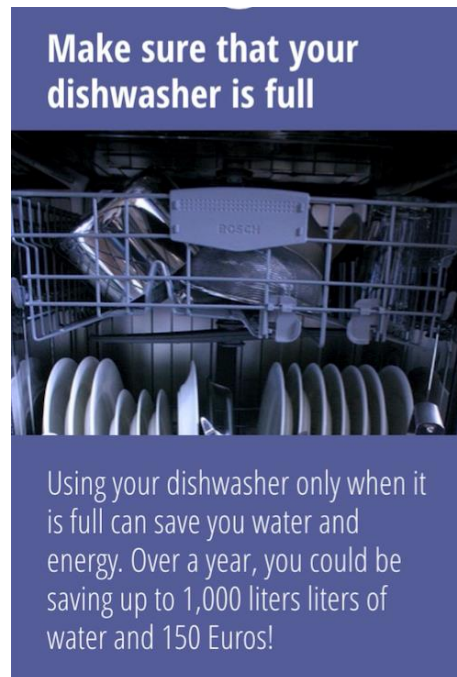


Figure 69: Message details output

In the following figures we present few examples of how the Messages Details UI element is adapted in the context of the DAIAD@home mobile application



Figure 70: Tip message



Figure 71: Budget alert example

## 4.2.7. Events table

The Events table UI element provides a general way to present consumption related events to users. It has been developed to cover events from multiple data sources.

The Events table UI element presents information for a variety of events such as Shower events, SWM events, budget and time alerts. For each event it displays information about the source of the event (e.g. shower) and the overall water consumption in a specific period. Further, depending the data source (SWM, b1) additional information is available.

The Events table UI element has been implemented as follows:

- Shower events:
  - We have added media assets as html `<img>` elements to define the source of the event (e.g. shower event) and up/down html `<img>` arrows to visualize the water consumption compared to the average consumption over the last ten showers (worse or better)
  - We have included multiple html `<span>` elements to provide information about the consumption's metrics such as the overall water consumption, duration and the time of the shower event. Further, we have added another `<span>` element providing information about the member that took the shower and one more `<span>` element to show either the efficiency rating of the shower or the shower device.
- Other events:
  - We have added media assets as html `<img>` elements to define the kind of the event such as budget alert, minimum or maximum consumption
  - We have added html `<span>` elements to provide information about the time which the event occurs, the kind of the event and the overall water consumption

The Events table UI element receives the parameters shown in the tables below.

*Table 28: Table UI element parameters*

Name	Type	Description	Example
selector	Html element	jQuery selector	<code>\$('#events')</code>
data	Json object	A json object that holds the consumption data of single event	See Table 29

*Table 29: Table events data object parameters*

Name	Type	Description	Example
volume	Number	Shower's volume in liters	120 liters

index	Number	Shower's index	21
energy	Number	Shower's energy spent in watt	1255 watt
temp	Number	Shower's temperature in Celsius	29celcius
duration	Number	Shower's duration in seconds	200 seconds
date	UNIX timestamp	Shower's datetime	125658999556
member	Number	Member's number. Default is 0	0 for the app owner
id	Number	Device Key	12535645-15555
category	Number	Shower's category. 17 for real time data and 18 for historical data	17

In the following, we provide details on the how the Events table UI element can be invoked. In this particular example, we create a shower event with shower values to an already created table. Thus, the function `appendEventsToList` will use the consumption data (second function argument) to create a shower event in the events selector (first function argument).

```
app.appendEventsToList($('#events'), {
  "volume":7,
  "index":21,
  "energy":1255,
  "temp":29,
  "duration":60,
  "date":1458986658,
  "member":0,
  "id":124855avfe-af-12fg,
  "category":17
})
```

The output of rendering the Events table UI element with the above parameters can be seen in Figure 72.

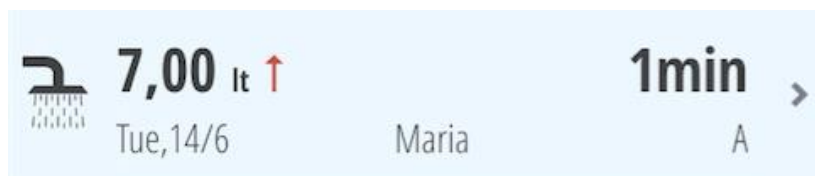


Figure 72: Events table UI element output

In the following figure we present an example of how the Events table UI element is adapted in the context of the DAIAD@home mobile application.

	<b>37,20</b> lt ↑	<b>8min</b> >
	Mon,13/6	Stephen
		A
	<b>19,80</b> lt ↑	<b>4min</b> >
	Mon,13/6	Stephen
		A
	<b>6,60</b> lt ↓	<b>1min</b> >
	Sun,12/6	Stephen
		A

Figure 73: Events table example

### 4.2.8. Projections

The Projections UI element provides a general way to present statistics for a specific shower event assuming a greater temporal and population scale.

The Projections UI element presents the statistics of a particular shower event, as well as estimations about the associated metrics (e.g. water, energy, CO2 emissions) on two scenarios (projections): at the city level (*what if everyone took similar showers?*), and at the year level (*what if all your showers within a year were the same?*).

The Projections UI element has been implemented as follows:

- We have developed a function that retrieves the necessary data for a specific shower event from the local database.
- We have created a function that computes the estimations of the water consumption data on a yearly basis and in the city based on the chosen shower's event data.
- We present the data as an HTML list of elements

The Projections UI element receives the parameters shown in the table below.

Name	Type	Description	Example
volume	Number	Total water used	100 lt
temperature	Number	Average water temperature	29 C
efficiency rating	String	Efficiency of the shower	A
energy	Number	Estimate of energy used for hot water	100 watt
co2	Number	Estimate of CO2 emissions	1kg

cost	Number	Estimate of how much the shower costs	1.20E
------	--------	---------------------------------------	-------

*Table 30 Projections UI element parameters*

In the following, we provide details on the how the Projections UI element can be invoked. In this example we create a projection of a single shower event.

```
$('#waterData').text(37.2);
$('#energyData').text(1);
$('#tempData').text(42);
$('#efficiencyData').text('A');
$('#costData').text(0.2);
$('#co2Data').text(0);
```

The output of rendering the Projections UI element with the above parameters can be seen in Figure 74.



*Figure 74: Projections UI element output*

In the following figures we present an example of how the Projections UI element is adapted in the context of the DAIAD@home mobile application



Figure 75: Projection for a shower event



Figure 76: Projection for a year

#### 4.2.9. Efficiency rating

The Efficiency rating UI element visualizes the efficiency of a given shower (or collection of showers) using efficiency classes (A-G, A being the most efficient, G the least efficient).

The Efficiency rating UI element has been implemented as follows:

- We have used html <img> elements and media assets to visualize the efficiency rating.
- We have used JavaScript code to compute the energy class based on a specific value

The Efficiency rating UI element receives the parameters shown in the table below.

Table 31: Efficiency rating UI element parameters

Name	Type	Description	Example
value	Number	A value to compute the energy class.	See table below for Energy class boundaries

In In this particular example we pass as parameter a value lower than 700, which will return the class A.

```
computeEfficiencyRatingFromEnergy(500)
```



Figure 77: Efficiency rating

## 5. Annex: Water Calculator

Water Calculators are simple forms (electronic or paper-based) provided by water utilities to consumers. They comprise an arbitrary set of simple questions and deliver an estimate of the total water demand for a household and its breakdown into major water uses (e.g. toilet flush, irrigation). Water calculators are a powerful instrument for informing and educating consumers regarding water use, enabling them to get insights about:

- What their water use *should be*, i.e. compare the results of the water calculator with their actual water consumption and thus understand where they can become more efficient
- How their *choices affect* their water use, i.e. how simple changes in their behaviors (e.g. *one less minute in the bathroom*) and/or equipment (e.g. *installing a dual flush toilet*) can have a great aggregate effect in their total water demand.

In addition, the responses to the actual water calculators (e.g. *number of household members, type of flushes*) provide water utilities with critical data regarding their customers they typically do not have access to. This data source can in turn be applied to facilitate demand management, improve the underlying water consumption models to anticipate water demand, and assist in the design and implementation of targeted retrofit/rebate campaigns.

Understandably, to harness these benefits water calculators are created and provided by water utilities and NGOs worldwide. However, a challenge in their implementation lies within the deeply *local nature* of water use, and thus the underlying assumptions and methodologies for building water calculators. For example, the typical values (target flows) for water fixtures are produced from water audits, i.e. extensive in situ studies of a sample of actual households for a specific location or utility. Similarly, on a methodological level the greatest difference is whether inelastic water demand per consumer (i.e. the absolute minimum amount of water a person needs) is used or implied from questions targeting water fixtures. At all cases, water calculators (mix of questions, typical values, assumptions) greatly differ around the world and evolve over time, due to weather, cultural, educational, technology, and local differences. As a result, water calculators developed for one water utility/location for a specific time-period cannot be applied *as is* to another population.

The lack of water audits and water calculators from EU-based water utilities (except the UK) compared to other parts of the world (especially Australia, USA) makes the task of developing a water calculator for DAIAD quite challenging. Our approach was to study a number of water calculators used in the literature, document the various assumptions and typical values, develop a set of super-set questions covering all methodological approaches, and deliver two different formulas for calculating the results. Our water calculator has been integrated both in the DAIAD apps and the surveys completed from all trial participants before the start of the pilots. During the course of the trials we will validate the assumptions of our water calculator against the actual water demand of our trial participants and use this information to update the water calculator accordingly. This process will be automated and integrated in the DAIAD system as ‘virtual water audits’ enabling water utilities to

automatically localize the generic water calculator based only on smart water meter data, i.e. without a need for expensive water audits.

## 5.1. Water Calculators

Name	Questions and Values		
CC Water <sup>54</sup> Calculator	<b>Type</b>	<b>Water used</b>	<b>Notes</b>
	People per household	11,000 liters per year and per person	This takes into account all personal activities (e.g. washing up, brushing teeth, etc)
	Baths in the household per week	80 liters per bath	
	Showers taken per week	46 liters per shower	
	Toilet flushes/day	7.5 liters per flush	
	Washing machine use/week	56 liters per wash	
	Dishwasher use/week	25 liters per wash	
	Garden hose (hours per year)	800 liters per hour	
City of Bellingham Washington <sup>55</sup>	<b>Type</b>	<b>Water used</b>	<b>Notes</b>
	People per household	105 gallons per person and per week	
	Showers per week (number <i>per week</i> , average length in min, showerhead flow rate)		5 for standard, 2.5 for low flow (gallons per minute)
	Baths in the household per week	6 gallons per bath	
	Toilet flushes (number <i>per week</i> , gallons per flush)		5 for standard, 1.6 for low flow (gallons)
	Dishes by hand (number <i>per week</i> , minutes each time)		2 gallons per minute
	Dishwasher use/week	15 gallons per wash	
	Washing machine/week	41 gallons per wash	
SW Florida Water Management District <sup>56</sup>	<b>Type</b>	<b>Water used</b>	<b>Notes</b>
	Showers per day	4 gallons per minute (old), 2 gallons per minute (low-flow)	
	Baths per week	35 gallons per bath	
	Toilet flushes per day	4 gallons per flush (old), 2 gallons per flush (low-flow)	
	Running water (minutes per day)	4 gallons per minute (old), 2 gallons per minute (new)	
	Dishes by hand (number <i>per week</i> , minutes each time)	same as running water	
	Dishwasher use/week	12 gallons per wash (old), 4 gallons per wash (new)	
	Washing machine/week	43 gallons per wash (old), 27 gallons per load (new)	
Hunter Water <sup>57</sup>		Annual Typical Household Usage (Kilo - Liters)	Breakdown of total water use
	Bathroom	88	50.6%
	Kitchen	12	6.9%
	Laundry	30	17.2%
	Lawn/Garden	32	18.4%
	Pool	0	0.0%
	Car/Boat	12	6.9%

<sup>54</sup> <http://www.ccwater.org.uk/watermetercalculator/>.

<sup>55</sup> <https://www.cob.org/services/utilities/water-calculator.aspx>.

<sup>56</sup> <http://www.swfwmd.state.fl.us/conservation/thepowerof10/>.

<sup>57</sup> <http://www.hunterwater.com.au/Save-Water/Water-Usage-Calculator.aspx>

	174		
Home Water Works <sup>58</sup>	Pre-EPA <sup>ct</sup> *	Post-EPA <sup>ct</sup> **	Volume (gal/year)
	25.5%	19.7%	Toilet use
	22.1%	20.7%	Clothes washer use
	17.4%	21.3%	Shower use
	15.1%	18.0%	Faucet use
	12.4%	14.0%	Leak use
	4.2%	2.4%	Other use
	1.8%	2.5%	Bathtub use
	1.5%	1.4%	Dishwasher use
	Annual Indoor water use is estimated using the following formula ( <i>result is in gallons</i> ) for Pre-EPA <sup>ct</sup> (i.e. pre 1992) and Post-EPA <sup>ct</sup> (i.e. after 1992), where y is the number of residents.		
<ul style="list-style-type: none"><li>PreEPA<sup>ct</sup> (ignore, too old) = <math>87.41 y^{0.69} \times 365</math> (gallons)</li><li>PostEPA<sup>ct</sup> = <math>63.30 y^{0.63} \times 365</math> (gallons)</li></ul>			
Melbourne <sup>59</sup>	Type	Water used	Notes
Toilet	Average 35 flushes/person per week <ul style="list-style-type: none"><li>Dual flush 5 liters per fill</li><li>Singe flush 11 liters per fill</li></ul>	Leaking toilet adds 308 liters/week	
Shower	<ul style="list-style-type: none"><li>Low flow: 7.5 liters/minute (AAA)</li><li>High flow: 12 liters/minute (conventional)</li></ul>		
Baths	96 liters per bath		
Washing machine	<ul style="list-style-type: none"><li>AAAA front loading 40 liters per load</li><li>Top loading 130 liters per load</li></ul>		
Kitchen	<ul style="list-style-type: none"><li>Try to save water 473 liters per week</li><li>Do not 525 liters per week</li></ul>	Very minor differences, we can use 500 liters per week for kitchen use at all cases	
Garden	<ul style="list-style-type: none"><li>Try to save water 651 per week</li><li>Do not, 1116 liters per week</li></ul>	Highly gross estimate	
Wash teeth	<ul style="list-style-type: none"><li>5 liters per minute (running water)</li><li>1 liter if water is not running</li></ul>	Leaking tap 200 liters per day	

## 5.2. DAIAD Water Calculator

This is the set of questions used for the water calculator

Question	Notes
Number of members in a household	
Showers in household per week	Avoid asking per day, some people don't shower every day
Number of minutes per shower (water running)	
Baths in household per month	Avoid asking per week, for some taking long baths is an infrequent luxury
Toilet flushes per day in the household	Can be indirectly estimated based on the number of members in a household (35 per person per week)
Washing machine per week	
Dishwasher per week	
Dishes by hand per day (minutes)	

<sup>58</sup> <http://www.home-water-works.org/calculator>

<sup>59</sup> [https://www.melbourne.vic.gov.au/Sustainability/SavingWater/Documents/water\\_household\\_calculator.pdf](https://www.melbourne.vic.gov.au/Sustainability/SavingWater/Documents/water_household_calculator.pdf)

Garden hose per day (minutes)	
Other running water per day (minutes)	

This is the version of our water calculator taking into account the exact number of household members (monthly household water consumption)

Question	Value	Notes	Monthly consumption
Number of members in a household	11,000 per person, per year	<i>This includes water consumption for all personal uses not included in the other categories</i>	X1
Showers in household per week	50 per shower		X2
Number of minutes per shower (water running)	NOT USED		
Baths in household per month	80 per bath		X3
Toilet flushes per day in the household	8 per flush		X4
Washing machine per week	56 per laundry		X5
Dishwasher per week	25 per wash		X6
Dishes by hand per day (minutes)	NOT USED		
Garden hose per day (minutes)	NOT USED		
Other running water per day (minutes)	NOT USED		

This is the version of our water calculator that ignores the exact number of household members (monthly household water consumption)

Question	Value	Notes	Monthly consumption
Number of members in a household	NOT USED		
Showers in household per week			
Number of minutes per shower (water running)	9.5 liters per minute		X2
Baths in household per month	96 per bath		X3
Toilet flushes per day in	8 per flush		X4

the household			
Washing machine per week	60 per laundry		X5
Dishwasher per week	25 per wash		X6
Dishes by hand per day (minutes)	5 liters per minute		X7
Garden hose per day (minutes)	5 liters per minute		X8
Other running water per day (minutes)	5 liters per minute		X9

## 6. Annex: Messages

In the following we provide the current water saving tips, alerts, and prompts automatically provided to DAIAD@home users. For each one we provide its English and Spanish version (primer and long string) and category (see legend below).

Legend – categories of textual interventions

W: helps to conserve water

E: helps to save energy

+: Very high impact

F: suited for flats and houses

H: Habit

O: One time effort

I: requires financial investments

W: Ayuda a ahorrar agua

E: Ayuda a ahorrar energía

+: Alto impacto

F: Apropiado para pisos y casas

H: Hábitos y costumbres

O: Necesario realizar sólo una vez

I: Necesita Inversión

### Shower and Bath

ID	Primer	Primer (ES)	Long Text (ES)	Long Text	Category
S1	Consider buying an efficient showerhead	Plantéate comprar un teléfono de ducha más eficiente	Los nuevos modelos mezclan agua y aire para ahorrar agua sin afectar a la calidad de la ducha. Tendrás la misma presión gastando menos agua. ¡Podrás gastar hasta la mitad de agua!	Modern showerheads mix water with air to save water without affecting your shower comfort. You will have the same pressure but spend less water. Using one can cut your shower water use in half!	W&E + F O I
S2	Don't multi-task!	¡No hagas tres cosas a la vez!	Puede que hayas oído que puedes ahorrar agua lavándote los dientes en la ducha - ¡NO! Tres minutos más en la ducha pueden suponer más de 20 litros de agua gastada. Hazlo en el lavabo y cerrando el grifo.	Some tips advise you to brush your teeth in the shower - DON'T. If you brush for three minutes that's about 20 liters wasted. Brush in the sink with the tap turned off	W&E + F H
S3	Consider turning the tap off while brushing your teeth or shaving	Mientras te laves los dientes, cierra el grifo del lavabo	!Dejar el grifo abierto un par de minutos al día puede suponer más de 1000 litros al año! Utilizando un vaso de agua ahorrarás agua suficiente para ducharte 20 veces.	A running tap for a few minutes every day can amount to losses over 1000 liters in a year! Using a glass of water instead can save enough water for 20 showers.	W&E F H
S4	Knocking a minute off your shower will save about 4 liters each time	Reduciendo en un minuto tu ducha diaria, ahorrarás 10 litros cada vez.	Es una de las formas más fáciles de ahorrar agua y energía. Ahorrando en cada ducha 1 minuto por cada miembro de la casa, ahorrarás 6000 litros de agua / El 40% del agua caliente utilizada en la vivienda se realiza en la ducha.	This is one of the easiest ways to save water and energy. By spending just one minute less in the shower a family of four can save in a year up to 6,000 liters of water and 300 Euros. / In a household usually 40% of the hot water are used for showering. When reducing the shower	W&E + F H

				time you can reduce your hot water consumption.	
S5	Try showering with slightly less hot water	Intenta ducharte con agua menos caliente	Puedes reducir tu factura energética simplemente reduciendo la temperatura del agua mientras te duchas. 1 o 2 grados pueden marcar la diferencia. Date una oportunidad y encuentra tu temperatura de confort / Por ejemplo, en una vivienda de dos personas, reducir en un minuto la ducha y un grado de temperatura puede suponer un ahorro importante a lo largo del año.	You can reduce your energy bills by slightly reducing the temperature of water in the shower. Even 1-2 degrees can make a difference. Give it a try and find a temperature that is comfortable for you / For example for a two person household, showering one minute shorter and with a water temperature of one degree colder as usual, you can save about 100€ a year when using a boiler.	E F H
S6	Consider turning off the water while soaping during the shower	Considera apagar el agua mientras te enjabonas	Intenta apagar el agua cuando no lo necesitas. Te sorprendería lo sencillo que es, ya que el baño se mantendrá caliente. Inténtalo y verás qué sencillo es.	Try turning the water off when you do not actually need it. You might be surprised to find this quite comfortable, as the bathroom will already be quite warm. Try it once and see how easy it is!	W&E F H
S7	Consider taking a shower instead of a bath	Considera ducharte en vez de bañarte	Un baños largo y calentito es muy agradable, pero consumes tres veces más agua y energía que si te das una ducha. Si lo haces frecuentemente, intenta reemplazar los baños por las duchas y verás como reduces a un cuarto tu consumo de agua. Bañarte supone un consumo de 150-200 litros, mientras que una ducha, 60-80 litros.	Warm, long baths are certainly nice, but use almost 3 times more water and energy compared to a shower. If you take frequent baths, try replacing half of them with showers. / Showering only consumes one fourth of the amount of water than taking a bath. / When taking a bath you consume 150-200 liters of water. When showering you only need 60-80 liters.	W&E + F H
S8	Clean hair, full wallet	Pelo limpio, cartera llena	Apaga el grifo del agua mientras te enjabonas el pelo y ahorra dinero reduciendo tu consumo de agua, energía y champú, lo cual repercutirá en un ahorro económico.	Turn of the tap or shower while you lather your hair and save money on water, energy and shampoo	W&E + F H
S9	Try using a timer when having a shower	Intenta usar un cronómetro mientras te duchas	¡Un cronómetro puede ayudarte a conocer cuánto tiempo necesitas para ducharte y reducirlo! Nuestra app lo lleva integrado, ¡Pruébalo!	A shower timer can help you keep track of how much time you spend in the shower and help you reduce it! Our mobile app has an integrated timer you can try!	W&E + F H I
S10	Fill your bathtub the smart way	Llena tu bañera de forma eficiente	Pon el tapón de la bañera antes de encender el agua y ajusta con la temperatura mientras se va llenando.	Plug the bathtub before turning the water on, and then adjust the temperature as the tub fills up.	W&E F H
S11	Exchange old double lever mixers	Cambia los antiguos “dobles” mezcladores	Cambia los dobles mezcladores por simples. Necesitarás menos tiempo para alcanzar la temperatura deseada.	Exchange old double lever mixers with single lever mixers. Then, it does not take that long to reach the expected water temperature.	W&E F O I
S12	Avoid long pipes	Evita largas conducciones	Instala el calentador del agua lo más cerca del baño que puedas para evitar pérdidas de calor.	Install the boiler close to the bathroom in order to avoid heating losses.	E O I
S13	Save energy and do not scald	Ahorra energía, no hace falta poner el agua a 60° en el calentador	Reduce la temperatura de tu calentador a 50-60°C. Además de ahorrar energía, alargarás la vida de tu calentador.	Set the heating temperature of your hot water heating system up to 50-60°C. Thus, you can save energy and you won't risk to scald due to too hot water.	E +
S14	For most people, 4 minutes of shower is enough. Try it yourself once!	Alternativa a S5: ¡Intenta ducharte en 4 minutos! Para la mayoría de la gente, con 4 minutos es suficiente...	¡Reducir tu tiempo en la ducha no es tan difícil! La próxima vez que te duches piensa en cuando y por qué estás usando el agua. Te sorprenderás de cómo consumes menos agua con el mínimo esfuerzo. ¡Y todo mientras disfrutas de la ducha!	Spending less time, and thus water, in the shower is not that hard! Next time you take a shower try and think when and why you use water in the shower. You will find out that you spend less time with minimal effort, while still enjoying a nice shower!	W&E + F H
S15	Alternative to S1: Consider installing a low flow shower head	Alternativa a S1: Plantéate comprar un teléfono de ducha de bajo consumo	Comprando un teléfono de ducha eco-eficiente (EcoBooster) podrás ahorrar 6.000 litros de agua caliente (Vivienda de 2 personas).	By buying a water saving showerhead (EcoBooster) you can save up to 63'000 liters of hot water (of a 2 persons household).	W&E + F O I
S16	Be aware of	Para lavarte las manos y los dientes	Asegúrate de que cuando abras el grifo,	Be aware of use cold water when you	

	use cold water when you brush your teeth and wash your hands	asegúrate de tener el grifo con el agua fría	estés utilizando sólo agua fría cuando te lavas los dientes o las manos, ya que si no, estarás realizando un consumo de agua del calentador que no llegará a salir por el grifo en la mayoría de las ocasiones	brush your teeth and wash your hands because you will spend hot water that will not arrive to the tip	
--	--	--	--	---	--

## Toilet

ID	Primer	Primer (ES)	Long Text (ES)	Long Text	Category
T1	Remember to use the low flush in your toilet whenever you can	Recuerda utilizar el caudal corto del inodoro siempre que puedas	Si tu inodoro tiene doble cisterna, utiliza el caudal corto siempre que puedas, consumirás la mitad de agua. Más de 40.000 litros por año para una familia media, equivalente a 250 baños. / La cantidad de agua utilizada cada vez que tiras de la cisterna es equivalente a la que necesita un joven en un país en desarrollo para sobrevivir durante todo un día (beber, lavar y cocinar)	If your toilet has a low flush, use it whenever possible! You will be consuming half the water each time you flush. That's more than 40,000 liters per year for an average family, equivalent to 250 long baths.  / The amount of water used for one toilet flush is equivalent to the amount of water a kid from a developing country consumes during a day for drinking, washing, and cooking.	W + F H
T2	Put a cistern displacement device in your toilet cistern	Reduce el volumen de tu cisterna del inodoro	Puedes introducir una botella llena de agua en tu cisterna para reducir el agua que necesitas cada vez que tiras de la cadena entre 1 y 3 litros.	Most water companies will give you a free CDD, you can put these in your toilet cistern and reduce the flush by between 1 and 3 liters.	W + F O I
T3	Ultra low flush and high efficiency toilets consume less than half water than conventional ones	Las cisternas de baño eficientes consumen menos de la mitad que las cisternas convencionales	Si estás pensando en renovar tu inodoro, busca los que tengan cisterna eficiente. Pues llegar a ahorrar más de 40.000 litros al año en un hogar de cuatro personas, equivalente a 250 baños.	If you consider changing your toilet flush, prefer an ultra low flush and high efficiency toilet. You will be consuming half the water each time you flush. That's more than 40,000 liters per year for an average family, equivalent to 250 long baths.	W + O I
T4	Check your toilet flush for leaks	Asegúrate de que tu inodoro no tenga fugas	Una pequeña fuga en tu inodoro es algo que solemos ignorar. Sin embargo, ¡esto puede suponer hasta 5.000 litros al año! Equivalente a 80 duchas o 30 baños. ¿Para qué desperdiciar agua y dinero?	A small leak in your toilet flush is something you may easily ignore. However, this small leak can consume up to 5,000 liters a year! That's equivalent to taking 80 showers or 30 long baths. Why lose water and money?	W + O I
T5	Adjust your toilet flush to save water	Ajusta el volumen de agua de tu inodoro	¿Sabías que la mayoría de los inodoros pueden ser regulados para utilizar menos agua? Con un simple ajuste podrías reducir hasta 2 litros por descarga ahorrando unos 15.000 litros al año en una vivienda de 4 personas.	Did you know that almost all toilet flushes can be adjusted to use less water? A simple adjustment to save 2 liters per flush will have no effect in the effectiveness of flushing. However, it can save an average family over 15,000 liters of water in a year!	W + F O
T6	Don't use your toilet as dustbin	No utilices el inodoro como papelería	Cada descarga de inodoro son 5 litros, utiliza la papelería para lo que sirve en vez del inodoro.	Every time you flush you use about 5 liters, use the bin instead.	W F H

## Washing machines

ID	Primer	Primer (ES)	Long Text (ES)	Long Text	Category
W1	Make sure your washing machine is full	Asegúrate de llenar por completo la lavadora	Utilizar la lavadora sólo cuando está llena ahorra agua y energía. ¡A lo largo del año puedes ahorrar hasta 2.000 litros.	Using your washing machine only when it is full can save you water and energy. Over a year, you could be saving up to 2,000 liters of water and 300 Euros!	W&E F H
W2	Consider purchasing an eco-efficient washing machine	Considera comprar una lavadora eco-eficiente	Reemplazar tu antigua lavadora por una eco-eficiente, aunque inicialmente suponga una inversión un poco superior, a la larga ahorrarás en tu bolsillo.	Modern washing machines are intelligent enough to use the optimal amount of water and energy required for each load. Replacing your old washing machine with an eco-efficient one can save you money.	W&E F O I

W3	Use the eco program in your washing machine	Utiliza el programa de lavado ECO	Las lavadoras modernas permiten seleccionar programas ECO para reducir hasta la mitad de agua y energía. Puede que tengas que esperar un poco más para tender, pero piensa en los ahorros, Hasta 7.000 litros al año.	Modern washing machines are extremely intelligent and efficient. By selecting an eco program you can cut your water and energy use in half! You might wait a little longer for the program to finish, but think of the savings. How does 300 Euros a year sound like?	W&E F H
W4	Washing dark clothes in cold water	Lava las prendas oscuras con agua fría	Lavar las prendas oscuras con agua fría ahorra energía además de mantener mejor el color de las mismas.	Washing dark clothes in cold water saves water and energy, and helps your clothes retain their color	E + F H
W5	Do your laundry the smart way	¿Lavas de forma eficiente?	Considera lavar a 20-30°C en vez de a 90°C. Ahorrarás mucha energía conservando tus prendas mejor.	Consider washing your laundry with 30-20°C instead of 90°C. This saves a lot of energy and conserves your laundry.	E + F H
W6	Do not use pre-washing for usual laundry	No uses prelavado	No uses el programa de prelavado. Los nuevos detergentes permiten lavar sin problemas a 30-40°C.	Do not use the pre-washing program of your washing machine. Due to modern detergent slightly soiled laundry can be already cleaned at 30-40°C.	W&E F H
W7	Consider installing new appliances	Considera instalar nuevos dispositivos	Considera renovar tus dispositivos. Los nuevos son más eficientes tanto en caudales como energéticamente. Una nueva lavadora puede ahorrar hasta 75 litros por lavado.	Consider installing new appliances. They are more water and energy-efficient than older appliances. A new washing machine can save up to 75 liters per load.	W&E F O I

## Kitchen

ID	Primer	Primer (ES)	Long Text (ES)	Long Text	Category
K1	Wash dishes the smart way	Limpia los platos de forma inteligente	Si friegas a mano, no dejes el agua correr. Enjabona todo primero, y luego aclara.	When washing dishes by hand, don't let the water run. Fill one basin with wash water and the other with rinse water	W&E + F H
K2	Use a kettle for heating up water for tea, coffee, or cooking	Utiliza una tetera para calentar el agua.	Controla la cantidad de agua que necesitas cuando cocinas. ¡Puedes ahorrar energía!	Control the amount of water you actually need for your beverage or for cooking (use a measuring jug for example). Then you can save energy!	W&E F H
K3	Fill the kettle whilst you wait	No tires el agua que no utilizas cuando la calientas	A veces tienes que añadir o quitar agua del cazo para controlar la temperatura que buscas. Si tienes que quitar, ponla en un vaso para utilizarla luego para regar.	Sometimes you have to run the tap to get top or cold water (lagging might cut the wait time) use this water to fill your kettle or keep a watering can by the sink for houseplants	W&E F H
K4	Make sure that your dishwasher is full	Asegúrate de llenar el lavavajillas	Utilizar el lavavajillas sólo cuando está lleno ahorra agua y energía. Al año, puedes llegar a ahorrar hasta 1.000 litros	Using your dishwasher only when it is full can save you water and energy. Over a year, you could be saving up to 1,000 liters of water and 150 Euros!	W&E F H
K5	Don't use running water to thaw food	No utilices agua caliente del grifo para descongelar	No enciendas el grifo del agua caliente para descongelar comida, hazlo en el frigorífico o a temperatura ambiente.	Don't use running water to thaw food. Defrost food in the refrigerator.	W&E F H
K6	Select the proper pan size for cooking	Utiliza el tamaño adecuado de cazo cuando cocines	Elige bien el tamaño adecuado de cazo cuando cocines. Cuanto más grande, más agua necesitarás.	Select the proper pan size for cooking. Large pans may require more cooking water than necessary.	W&E F H
K7	Choose a good dishwashing detergent	Elige un buen lavavajillas de mano	Un buen lavavajillas de mano de ayuda a ahorrar agua, energía y tiempo.	A good dishwashing detergent helps you save water, energy, and time.	W&E F I

## Irrigation/Gardening

ID	Primer	Primer (ES)	Long Text (ES)	Long Text	Category
G1	Look for drought tolerant plants	Busca plantas resistentes a la sequía	Cuanto más resistente a la sequía sea la planta, menos tendrás que preocuparte de ella y más agua ahorrarás	The more drought tolerant your plants are, the less you have to care about them. This saves effort and water.	W + F O

					I
G2	Think about irrigation in your garden	Plantéate el modo de riego de tu jardín	Capta el agua de lluvia y utilízala para regar. Puedes recolectarla usando los canales y almacenando en un depósito.	Collect rainwater and use it to irrigate the plants in your garden. You can set up a rain barrel and even connect it to the rainwater gutter.	W O I
G3	Think about the right time for irrigating your garden	Piensa cuando es el mejor momento del día para regar	Recuerda que regar durante la noche evita evaporación de agua. Evita también que tus plantas se quemen por regar de más.	Remember that irrigating your plants during noon only supports evaporation of the water. Conserve your plants also from “burning” by irrigating in the early morning or at the evening.	W H
G4	Irrigate your lawn correctly	Riega el césped correctamente	El césped no suele necesitar mucha agua, sobre todo en los meses de invierno cuando hay riesgo de que se seque.	Lawns normally do not use much water – mainly during drought season there is a risk of dry out.	W F
G5	Aerate your soil	Airea el suelo	Cuando aireas el suelo, el agua llega más rápido a las plantas ahorrando agua mientras riegas.	When you aerate your soil, the faster the water reaches the plants and you can save water for irrigation.	W + H
G6	Use water reservoirs for balcony plants.	Utiliza depósitos de agua para las plantas del balcón.	Con depósitos de agua para las plantas no tendrás que regar diariamente. Existen dispositivos que indican la humedad que tiene la tierra y si el depósito se ha vaciado.	With water reservoirs for water plants you do not have to daily irrigate your plants. A display shows the humidity and indicates an empty reservoir.	W O I
G7	Use water from bathing for watering your flowers	Usa el agua del baño para regar las plantas.	La mayor parte del agua que usas cuando te duchas puede aprovecharse para regar. ¡Úsala!	Water that you have just used in your bath tube does not harm your plants. You can easily use it for irrigation.	W F H

## Leaks and inefficient taps

ID	Primer	Primer (ES)	Long Text (ES)	Long Text	Category
L1	Don't forget to fix your dripping taps	No olvides reparar tus grifos cuando gotean	Un simple goteo puede llegar a suponer cientos de litros anualmente. ¿Por qué no arreglarla y darte un buen baño en su lugar?	A single dripping tap can lead to annual water losses of hundreds of liters. Why not save this water and take a long bath instead?	W&E F H
L2	A new low-flow faucet can easily half your water use	Un grifo de bajo consumo puede reducir hasta un 50% tu consumo	Los nuevos modelos de grifos de bajo caudal puede reducir tu consumo de agua sin que se aprecie notablemente. Si instalas uno en casa, ¡ahorrarás agua cada vez que enciendas el grifo!	Modern low-flow faucets can reduce your water consumption without any discomfort. If you install one you will be saving water every time you turn on the faucet! And without even trying.	W&E + O I
L3	Aerators and flow restrictors have an excellent cost benefit ratio	Los dispositivos de aireación y los reguladores de caudal tiene un buen ratio coste-beneficio	Los dispositivos de aireación modernos reducen tu consumo de agua sin que apenas se aprecie. Si instalas uno en casa, ¡ahorrarás agua cada vez que enciendas el grifo!	Modern Aerators reduce your water consumption without any discomfort. If you install one you will be saving water every time you turn on the faucet! And without even trying. Most faucets can be easily retrofitted.	W&E + F O I
L4	Use your water meter to check for hidden water leaks	Utiliza tu contador de agua para encontrar las fugas ocultas	Aprovecha cuando estés solo/a en casa y toma lectura de tu contador antes y después de salir de casa. Si la lectura es la misma, ¡Perfecto! no tenemos fugas (Ten en cuenta si tienes algún sistema de riego automático o llenado de depósitos como el del calentador de agua).	Wait until everyone has left your apartment and read your water meter before you go and when you come back. The meter reading should be the same unless there's a leak (or if you have an automatic top-up on a fish tank or a trickle irrigation system).	W F O
L5	Check your monthly water bill for leaks	Comprueba tu factura para encontrar fugas	Revisa tu factura para comprobar que no hay un consumo excesivo. Comprueba tu contador por las noches o en los fines de semana para encontrar fugas. No debería correr la aguja del contador cuando todos los grifos están cerrados.	Review your water bill monthly to check for unusually high use. Check water meters at night or on the weekend to detect leaks. There should be no flow when all fixtures have been turned off.	W&E + F H
L6	Check your toilet flush for leaks	Comprueba la descarga de tu cisterna para detectar fugas	Una pequeña fuga en la cisterna del baño puede pasar fácilmente desapercibida. Sin embargo, ¡puede llegar a consumir	A small leak in your toilet flush is something you can easily ignore. However, this small leak can consume up to 5,000	W + F O

	(important, listed also under T)		5.000 litros al año! Esto equivale a 80 duchas o 30 baños. ¿Por qué desperdiciar el dinero así?	liters a year! That's equivalent to taking 80 showers or 30 long baths. Why lose water and money?	I
L7	Exchange old double lever mixers	Cambia tus antiguos mezcladores de agua	Cambia tu antiguo mezclador doble de agua por uno simple. No necesitarás tanto tiempo para encontrar tu temperatura de confort.	Exchange old double lever mixers with single lever mixers. Then, it does not take that long to reach the expected water temperature.	W&E + F O I

## General Tips

ID	Primer	Primer (ES)	Long Text (ES)	Long Text	Category
G1	Think about indirect consumption	Piensa en los consumos indirectos	Puedes reducir tu consumo de agua incluso comprando. Se necesita mucha agua para producir algunos alimentos u otros bienes. Por ejemplo, la producción de unos pantalones vaqueros necesita unos 8.000 litros de agua.	You can also reduce your water consumption while shopping. A lot of water is used for the production of food and other goods. For example, the production of a Jeans might need up to 8'000 liters of water. Less economic consumption means less water consumption.	W&E + F H
G2	Cold water for your hands	Agua fría para tus manos	Enjabona tus manos antes de encender el grifo y cuando lo hagas, hazlo con agua fría.	Lather your hands before opening the tap and only rinse your hands with cold water.	W&E F H
G3	Turn off the tap when you don't really need it	Cierra el grifo cuando realmente no lo necesitas	Cuando te lavas las manos, verduras, o platos sucios piensa en cerrar el grifo cuando realmente no necesitas agua. Cada minuto con el grifo abierto son 2 litros de agua. No parece mucho, pero podrías llegar a ahorrar 2.000 litros de agua / Puedes ahorrar un 50% de agua cuando cierras el grifo cuando realmente no lo necesitas / Usa un cuenco o el fregadero cuando laves verduras, frutas o ensalada.	When you wash your hands, vegetables, or dishes think about turning off the tap when you don't actually need the water. A running tap spends 2 liters every minute. It doesn't sound much, but over a year you could be saving more than 2,000 liters of water! / You can save up to 50% of water when turning off the tap, when you don't really need it. / Use a bowl or the sink when washing vegetables, fruits, or salad.	W&E F H
G4	Insulate your water pipes	Aísla tus tuberías de agua	Aíslando tus tuberías puedes evitar que éstas revienten en invierno. También reducirá tu consumo de agua caliente y regularás la temperatura del agua más rápidamente.	Insulating your pipes can stop bursts in the winter, will cut your hot water bill and means that you get hot and cold water from the tap much faster.	E O I
G5	The sun provides hot water	El sol calienta el agua	Un sistema de calentador de agua solar reduciría nuestro consumo de energía.	A solar heating system warms up my hot water.	E + F O I
G6	Use energy efficient hot water generation	Utiliza el agua caliente de forma eficiente	Puedes conectar directamente tu lavavajillas con la línea de agua caliente ya que el calentador de agua general es más eficiente que el del lavavajillas.	You can directly connect your dishwasher to the hot water piper because the hot water generation is more energy efficient than the heating of the dishwasher.	E F O
G7	Save energy when leaving for holidays	Ahorra agua cuando estás de vacaciones	Acuérdate de desconectar el calentador cuando te vas de vacaciones. No malgastes energía si no la vas a usar.	Remember to turn off your boiler when you are leaving your home for vacation. Thus, you do not waste energy for providing hot water when it is not used.	E F H
G8	Invest in new hot water generation technologies	Invierte en nuevas tecnologías para calentar agua	Considera cambiar tu viejo calentador eléctrico ya que consume mucha energía. Anualmente podrás reducir tu consumo hasta en un 60% si utilizas una bomba de agua caliente.	Consider removing your electrical boiler because it uses a big amount of energy. This form of hot water generation is not energy efficient. Yearly electricity consumption can be reduced by 60% with a hot water heat pump.	E + O I
G9	Save water by turning off small hot water reservoirs	Ahorra agua desconectando el calentador cuando no lo necesitas	Los pequeños calentadores de agua regulan la temperatura para mantenerla más o menos fija. Piensa en apagarlo cuando no lo vayas a usar durante un tiempo.	Small hot water reservoirs/boilers regulate the water inside the reservoir at a certain temperature. Consider turning off these reservoirs when they are not used.	E + F H

G10	Intelligent refreshing with water during the summer	Refréscate de forma inteligente durante el verano	Cuando usas juguetes acuáticos como piscinas, toboganes acuáticos... piensa que éstos no consumen tanta agua como parece.	When using water toys such as wading pools, water slides, etc. consider that wading pools do not consume far as much water as water slides and other toys.	W F H
-----	---	---	---	--	-------------

## Alerts

ID	Short string	Full string	Short string (ES)	Full string (ES)
A1	Check for water leaks!	We believe there could be a water leak in your house. Please check all fixtures for leaks and contact your water utility	¡Comprueba las fugas de agua!	Creemos que tienes una fuga de agua en casa. Por favor, comprueba tus dispositivos y contacta con tu compañía suministradora de agua.
A2	Shower still on!	Someone forgot to close the shower?	¡Te has dejado la ducha encendida!	¿Alguien ha olvidado cerrar el grifo de la ducha?
A3	Check the water fixtures!	Please check your water fixtures, there may be water running in one of them!	Comprueba tus dispositivos que funcionen con agua	Por favor, comprueba tus dispositivos que funcionen con agua, puede que te hayas dejado alguno funcionando.
A4	Unusual activity detected!	Water is being used in your household at an unusual time. Is everything OK?	¡Actividad inusual detectada!	Se está utilizando agua en una hora un poco inusual. ¿Está todo bien?
A5	Water quality not assured! (Date)	Remember to leave the water running for a few minutes when you return home after a long period of absence. Temperatures have been over 28 since you last used water. Better be safe from Legionella.	¡Calidad del agua no asegurada!	Recuerda dejar correr un par de minutos el agua en el grifo cuando vuelvas a casa después de estar mucho tiempo fuera. Si la temperatura ha superado los 28°C en algún momento podrías contraer legionela.
A6	Water too hot!	Be careful, the water in your shower is extremely hot!	¡Agua demasiado caliente!	¡Cuidado, el agua de tu ducha está demasiado caliente!
A7	Reached 80% of your daily water budget (Date)	You have already used XX liters, and you have YY remaining for today. Want some tips to save water?	Ya has consumido el 80% de lo que sueles consumir normalmente al día.	Ya has consumo XX litros, y normalmente gastas YY al día. ¿Quieres algunas recomendaciones para ahorrar agua?
A8	Reached 80% of your weekly water budget	You have already used XX liters, and you have YY remaining for this week. Want some tips to save water?	Ya has consumido el 80% de lo que sueles consumir normalmente a la semana	Ya has consumo XX litros, y normalmente gastas YY a la semana. ¿Quieres algunas recomendaciones para ahorrar agua?
A9	Reached 80% of your daily shower budget	You have already used XX liters, and you have YY remaining for today. Want some tips to save water?	Ya has consumido el 80% de lo que sueles consumir normalmente al día en la ducha.	Ya has consumo XX litros, y normalmente gastas YY al día duchándote. ¿Quieres algunas recomendaciones para ahorrar agua?
A10	Reached 80% of your weekly shower budget	You have already used XX liters and you have YY remaining for this week. Want some tips to save water?	Ya has consumido el 80% de lo que sueles consumir normalmente a la semana en la ducha	Ya has consumo XX litros, y normalmente gastas YY a la semana duchándote. ¿Quieres algunas recomendaciones para ahorrar agua?
A11	Reached daily/weekly Water Budget	You have used all XX liters of your water budget. Let's stick to our budget tomorrow! Want some tips to save water?	Alcanzado el consumo medio diario/semanal de agua	Has consumo los XX litros que consumes normalmente. You have used all XX liters of your water budget. ¡Intentemos mejorarlo mañana! ¿Quieres algunas recomendaciones para ahorrar agua?
A12	Reached daily/weekly Shower Budget	You have used all XX liters of your shower budget. Let's stick to our budget tomorrow! Want some tips to save water?	Alcanzado el consumo medio diario/semanal de agua en la ducha	Has consumo los XX litros que consumes normalmente en la ducha. You have used all XX liters of your water budget. ¡Intentemos mejorarlo mañana! ¿Quieres algunas recomendaciones para ahorrar agua?
A13	You are a real water champion!	Well done! You have managed to stay within your budget for 1 whole month! Feel you can do better? Try targeting a slightly lower daily budget.	¡Eres un genio del agua!	¡Bien hecho! ¡Has conseguido mejorar tu consumo diario durante 1 mes completo! ¿Podrías hacerlo mejor? Intenta optimizar un poco más tu consumo.
A14	You are a real shower champion!	Well done! You have managed to stay within your shower budget for 1 whole month! Feel you can do better? Try targeting a slightly lower daily shower budget.	¡Eres un genio de la ducha!	¡Bien hecho! ¡Has conseguido mejorar tu consumo diario en la ducha durante 1 mes completo! ¿Podrías hacerlo mejor? Intenta optimizar un poco más tu consumo
A15	You are using too much water	You are using twice the amount of water compared to [your anticipated water consumption] [similar households] [local households] [city average]	¡Estás gastando mucha agua!	Estás gastando el doble que [en tu anterior consumo] [un/a consumidor/a similar] [tu compañero/a de vivienda] [la media ciudadana]

		[country average] [your typical water use] You could save up to XX Euros. Want to learn how?		[la media del país] [tu consumo típico] Puedes ahorrar sobre XX €. ¿Quiéres aprender cómo?
A16	You are using too much water in the shower	You are using twice the amount of water compared to [other consumers] [your typical shower use] You could save up to XX Euros. Want to learn how?	¡Estás gastando mucha agua en la ducha!	Estás consumiendo el doble que [otr@s clientes] [tu consumo típico de agua en la ducha] Puedes ahorrar sobre XX €. ¿Quiéres aprender cómo?
A17	You are spending too much energy for showering	You are spending too much hot water in the shower. Reducing the water temperature by a few degrees could save you up to XX Euros. Want to learn how?	Estás gastando mucha energía en tu ducha	Estás gastando mucha agua caliente. Reduce la temperatura un par de grados, podrías ahorrar XX €. ¿Quiéres aprender cómo?
A18	Well done! You have greatly reduced your water use	You have reduced your water use by XX% since you joined DAIAD! Congratulations, you are a water champion!	¡Bien hecho! Has reducido tu consumo de agua	¡Has reducido tu consum de agua en un XX% desde que participas en el piloto DAIAD!
A19	Well done! You have greatly improved your shower efficiency	You have reduced your water use in the shower by XX% since you joined DAIAD! Congratulations, you are a water champion!	¡Bien hecho! Has mejorado la eficiencia de tus duchas	¡Has reducido tu consumo de agua en la ducha un XX% desde que participas en el piloto DAIAD! ¡Enhorabuena, eres un genio del agua!
A20	Congratulations! You are a water efficiency leader	You are a true leader for water efficiency! If everyone adopted your behavior your city could save XX liters of water	¡Enhorabuena! ¡Eres un máquina ahorrando agua!	¡Eres realmente un máquina ahorrando agua! Si todos se comportaran como tú la ciudad ahorraría XX litros de agua.

## Prompts

ID	Prompt	Prompt (ES)	Decision Rule	Maximum frequency	Remarks
P1	Keep up saving water!	¡Ahorremos agua!	Randomly if no other prompt is available	Once per month	Use sparsely as it might create reactance. Only use when no specific prompt is available
P2	You are doing a great job!	¡Buen trabajo!	(Savings above 6% over last month) & (consumption below average) or (savings above 25%)	Once per week or month	Display actual savings
P3	You have already saved XX liters of water!	¡Vamos, ya has ahorrado XX litros!	(Savings > 100l)	Once per week (alternating with P4-P5)	Can be show shown in combination with P2
P4	Congratulations! You are one of the top 25% savers in your region/household/city.	¡Enhorabuena! Estás en el top 25 del piloto!	(consumption within lowest 25%)	Once per week (alternating with P3, P5)	Can be show shown in combination with P2
P5	Congratulations! You are among the top group of savers in your region/household/city.	¡Enhorabuena! Estás entre los que más ahorran del piloto!	(consumption within lowest 10%)	Once per week (alternating with P3-P4)	Can be shown in combination with P2
P6	Did you know? + Link for more information	¿Sabías qué...? + Link for more information	Randomly, can be combined with P1-P5	One time only	Show saving tips concerning the reduced shower time/water temperature accordingly to the shower behavior (when the user has showered 3 times in a row with one degree lower than before, when the user has shower 3 times shorter than before, e.g.)
P7	Did you know? + Link for more information	¿Sabías qué...? + Link for more information	Randomly, can be combined with P1-P5	One time only	Show saving tip concerning new equipment when detecting a change in one flow rate for one point of consumption (e.g. for the shower head and faucets or a new washing machine)