

H2020 – EE – 11 – 2015

Research and Innovation Action

CleAnweb Gamified Energy Disaggregation



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 696170

D3.3 System Components (Short)

Report Identifier:	D3.3		
Work-package, Task:	WP3	Status – Version:	2.5
Distribution Security:	CO	Deliverable Type:	DEM
Editor:	WATTICS		
Contributors:	ALL		
Reviewers:	ED, PLEGMA		
Quality Reviewer:	ED		
Keywords:	Demonstrator, prototype, development, interfaces		
Project website: http://www.charged-project.eu/			

Disclaimer

Use of any knowledge, information or data contained in this document shall be at the user's sole risk. Neither the ChArGED Consortium nor any of its members, their officers, employees or agents accept shall be liable or responsible, in negligence or otherwise, for any loss, damage or expense whatever sustained by any person as a result of the use, in any manner or form, of any knowledge, information or data contained in this document, or due to any inaccuracy, omission or error therein contained.

The European Commission shall not in any way be liable or responsible for the use of any such knowledge, information or data, or of the consequences thereof.

This document does not represent the opinion of the European Union and the European Union is not responsible for any use that might be made of it.

Copyright notice

© Copyright 2016-2019 by the ChArGED Consortium

This document contains information that is protected by copyright. All Rights Reserved. No part of this work covered by copyright hereon may be reproduced or used in any form or by any means without the permission of the copyright holders.

Table of Contents

Abbreviations	7
Executive Summary	9
1 Introduction	10
1.1 Purpose and Scope	10
1.2 Intended Audience	10
1.4 Document Overview	10
2 ChArGED System Architecture	11
2.1 Logical Architecture	11
2.2 Process View	12
3 Data/Core Back-End System	13
3.1 SiteWhere Software Environment	13
3.1.1 Description and Operation	13
3.1.2 Implementation	14
3.1.2.1 Architecture	14
3.1.2.3 Installation and Configuration Steps	15
3.2 Databases (InfluxDB, MongoDB)	15
3.2.1 Description and Operation	15
3.2.2 Implementation	15
3.2.2.1 Architecture	15
3.3 User Management (WSO2 Identity Server)	15
3.3.1 Description and Operation	15
3.3.2 Implementation	16
3.3.2.1 Architecture	16
3.3.2.3 Installation and Configuration Steps	17
3.4 Asset management/User Interface	17
3.4.1 Description and Operation	17
3.4.2 Implementation	18
3.4.2.1 Architecture	18
3.4.2.2 Implementation Status	18
3.5 Raw Data Visualization (Grafana)	21
3.5.1 Description and Operation	21
3.5.2 Implementation	22

4	Sensor Gateway	22
4.1	Sensor Gateway System Architecture	22
4.2	HDM/SiteWhere Integration	24
4.2.1	Description and Operation	24
4.2.2	Implementation	24
4.2.2.1	Architecture	24
4.3	Modbus Device Integration to SiteWhere	25
4.3.1	Description and Operation	25
4.3.2	Implementation	25
4.3.2.1	Architecture	25
5	Analytics Backend	26
5.1	Wattics Back-End Architecture	26
5.2	Component for Automated Disaggregation and Allocation of Energy Savings	27
5.2.1	Description and Operation	27
5.3	Component for Automated Discovery of Energy Saving Opportunities	31
5.3.1	Description and Operation	31
5.3.2	Implementation	32
5.3.2.1	Architecture	32
5.4	Energy manager Front-End Dashboard	33
5.4.1	Description and Operation	33
5.5	Component for Microgeneration Energy Forecasting	34
5.5.1	Description and Operation	34
5.5.2	Implementation	34
6	Mobile App	38
1.1	Front-End	38
6.1.1	Description and Operation	38
6.2	Game Backend / Game Mechanics	43
6.2.1	Description and Operation	43
6.2.2	Implementation	44
6.2.2.1	Architecture	44
8	Conclusions	47

List of Figures

Figure 1: The ChArGED System Architecture	11
Figure 2: Organisation of SiteWhere system components	14
Figure 3: SiteWhere’s time series and noSQL databases	15
Figure 4: SiteWhere’s identity management through WSO2 Identity Server	16
Figure 5: The SiteWhere assets, device and metadata organisation	18
Figure 6: Illustration of the SiteWhere tenants created for each pilot site	19
Figure 7: Illustration of how the different pilots are decomposed into sub-sites within SiteWhere	20
Figure 8: Illustration of the assignment of devices to assets in SiteWhere for the pilot sites.....	21
Figure 9: Example of data collected by ZWave plugs in the DAEM pilot site	22
Figure 10: The Raspberry Pi (version 3 Model B used as hardware platform for the Sensor Gateway software	23
Figure 11: Illustration of the sensor gateway data collection, processing and communication to SiteWhere	24
Figure 12: Integration of the Sensor Gateway’s HDM with SiteWhere via MQTT	25
Figure 13: Modifications to the Sensor gateway for integration of Modbus meters	26
Figure 14: The Wattics cloud backend system architecture	26
Figure 15: Illustration of power drop measurement on a single electrical reading with NFC swipe events from users providing the name of the appliance causing the drop	28
Figure 16: Illustration that no kWh savings are achieved by a specific user control action, when that action was already conducted before ChArGED was introduced.	29
Figure 17: Illustration of the kWh savings achieved by a specific user control action	29
Figure 18: Illustration of the kWh savings achieved by a specific user control action delimited by a BLE alert	30
Figure 19: Illustration of the kWh savings calculation triggered at the start of the day covering the period from the NFC swipe event to when the user is expected to be returned at his desk.....	30
Figure 20: Illustration of the kWh savings calculation triggered at the start of the day covering the period from the first NFC swipe event to when the users are expected to be returned at his desk.....	31
Figure 21: Illustration of the kWh savings calculation delimited by BLE alerts when a user leaves and returns to a location	31
Figure 22: Model creation showing historical measurements (blue dots) and the model (green area)	33

Figure 23: Overview of theoretical principles of photovoltaic power production	35
Figure 24: PLEGMA’s Solar microgeneration forecast configuration screen	36
Figure 25: The PV prediction output from the system.....	37
Figure 26: The ChARGED mobile APP View of active challenges and who won challenges on the Dashboard	39
Figure 27: The ChARGED mobile APP Leader Board view with open details of a specific user	39
Figure 28: The ChARGED mobile APP review of challenges	40
Figure 29: an overview of The ChARGED mobile APP morning, lunch and evening challenges	40
Figure 30: The ChARGED mobile APP CHALLENGE SCREENS	41
Figure 31: NFC integration within the mobile app	42
Figure 32: BLE integration within the mobile app.....	42
Figure 33: The Game Backend INTERFACES	44
Figure 34: The Game Backend Internal architecture	45

ABBREVIATIONS

API	Application Programming Interface
BLE	Bluetooth Low Energy
CDC_ACM	Communications Device Class Abstract Control Model
D	Deliverable
eJRE	embedded Java Runtime Environment
ID	Identity
IoT	Internet of Things
IP	Intellectual Property or Internet Protocol
JSON	JavaScript Object Notation
kWh	Kilowatt hours
M	Month
MVP	Minimum Viable Product
mPRM	mPower Remote Manager
MQTT	MQ Telemetry Transport
NDEF	NFC Data Exchange Format
NFC	Near Field Communication
PC	Personal Computer
SSL	Secure Sockets Layer
SQS	Simple Queue Service
TCP	Transport Communication Protocol
TLS	Transport Layer Security
REST	Representational state transfer
UML	Unified Modeling Language
W	Watts

WP	Work Package
XSRF	Cross-Site Request Forgery

EXECUTIVE SUMMARY

This deliverable presents the final ChArGED platform architecture and the specifications of all the system components, functionalities and interfaces implemented.

The ChArGED system architecture consists of four main groups of functional blocks:

- The **Data/Core Back-end** group, responsible for providing an environment in which data, assets and users are stored and managed.
- The **Gateway** group, responsible for integration of energy use and environmental data to the Back-end system, to determine variations over the energy context within the building.
- The **Analytics Back-end** component, responsible for delivering insights that enable the ChArGED application to deliver custom and targeted feedback and incentives to the end-users.
- The **Gamification** group, responsible for processing field data and insights created from such data and make decisions as to the evolution of the game for each user, and for delivering the mobile app the end-users interact with.

The ChArGED platform implementation will be utilised in three pilot sites for a period of 12 months. Because of the unique challenges and specific environment conditions of each pilot, an Agile software development approach and modular architecture were used, to enable gradual implementation/development with new application requirements as they arise.

Functionality, architecture, and system interfaces are described for each component, and information about implementation, testing and configuration of the system components is also provided.

1 INTRODUCTION

1.1 PURPOSE AND SCOPE

The goal of this deliverable is to present the updated and final version of the ChArGED architecture and platform components implemented, providing a more detailed description of the components inner mechanisms and interfaces. This deliverable being of Demonstrator type, showcases the implementation of the system components via code samples, screenshots and run-time tests. Configuration and installation instructions are also given.

1.2 INTENDED AUDIENCE

The purpose of this deliverable is to document and demonstrate the software components and interfaces developed.

The intended audience is the project WP3 team especially developers. Sharing the architectural information with the other technical WPs constitutes an important knowledge foundation on which to reach an integrated framework. Members of the development team can use this report as guideline for future work.

This report also provides clarity on the deployment of the ChArGED system at a public building through various installation stages. This report represents the working modules that can be used for hands-on system demonstration and evaluation by intended public buildings.

1.4 DOCUMENT OVERVIEW

The key deliverable is the software development of the system components. This document provides an accompanying guide on the developed software components and is organised as follows:

- Chapter 2 presents a refined and final version of the ChArGED architectural design
- Chapter 3 describes architecture and software implementation details for the Data/Core Backend components
- Chapter 4 describes architecture and software implementation details for the Gateway component
- Chapter 5 describes architecture and software implementation details for the Analytics Backend components
- Chapter 6 describes architecture and software implementation details for the Gamification System components (mobile app and game engine)
- Chapter 7 concludes the document.

2 CHARGED SYSTEM ARCHITECTURE

This chapter presents the overall ChArGED platform together with the main functional blocks. The high-level dynamic aspects of the system are also illustrated with Unified Modeling Language (UML) Sequence Diagrams. A more detailed description of each component is provided in the next chapters, with details on their functionality, architecture, interfaces and implementation status.

2.1 LOGICAL ARCHITECTURE

This section presents the final logical architecture of the ChArGED platform, with the functional blocks and their interrelationships illustrated. The system architecture consists of four main groups of functional blocks:

- Data/Core Backend
- Gateway
- Analytics Backend
- Gamification System

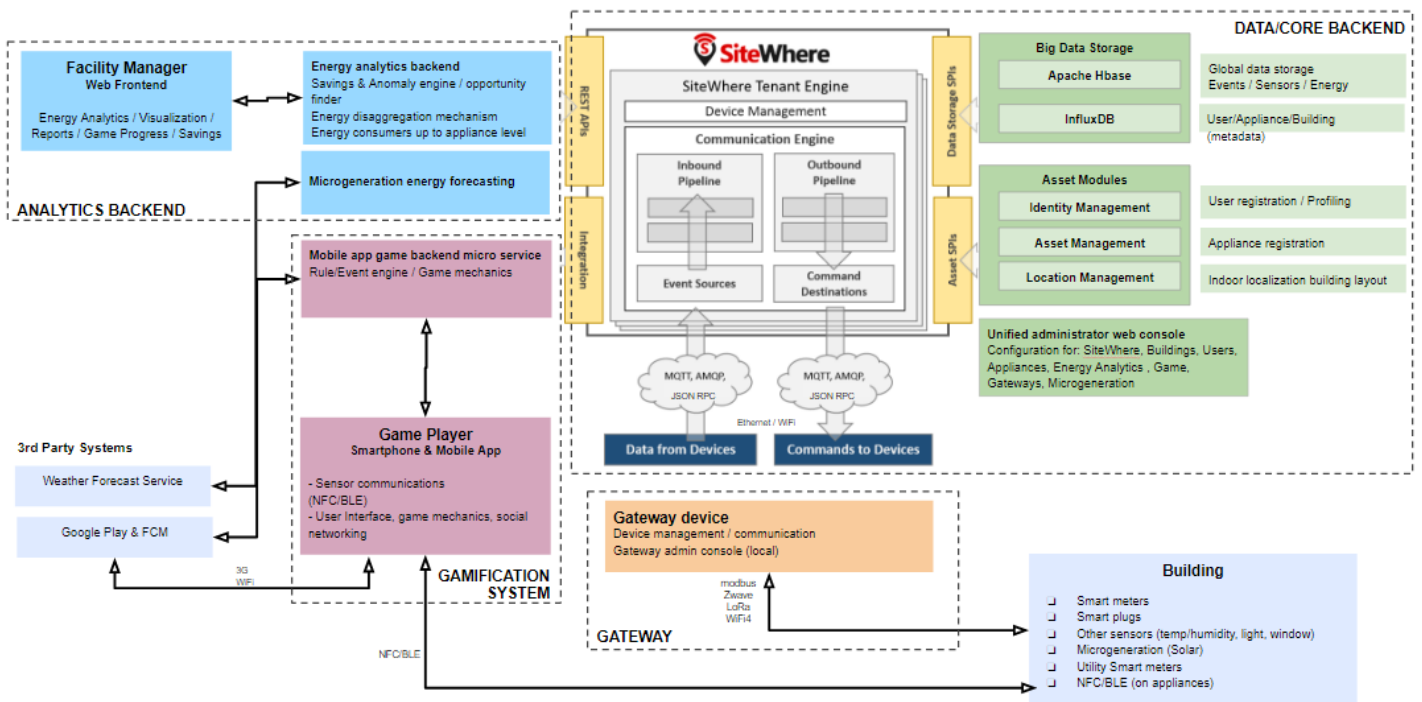


FIGURE 1: THE CHARGED SYSTEM ARCHITECTURE

The **Data/Core Backend** group is responsible for providing an environment in which data, assets and users are stored and managed. A collection of backend big data, IoT technologies based on Java Spring, SiteWhere, WSO2 Server, SQL and NO-SQL databases, etc provide the software infrastructure on which the ChArGED application is developed. That group of components is application agnostic, however it is tuned towards the needs of ChArGED project.

The **Gateway** group is responsible for integration of energy use and environmental data to the Data/Core Back-end system, to determine variations over the energy context within the building.

The **Analytics Backend** component is responsible for delivering insights that will enable the ChArGED application to deliver custom and targeted feedback and incentives to the end-users.

Finally, the **Gamification** group is responsible for processing field data and insights created from such data and make decisions as to the evolution of the game for each user, i.e. what the next step is towards more energy savings. This group also delivers the end user mobile app, which provides the user current game state and also information to the system about the users' behaviour towards the energy saving goals set.

2.2 PROCESS VIEW

This section deals with the dynamic aspects of the ChArGED platform. The system operation revolves around three main process flows, depicted in the following with UML Sequence Diagrams. Details about individual components and interfaces are described in the following chapters, and integration results will be presented in Deliverable D3.4.

1. **Collection of data from the building via meters and environmental sensors connected to the ChArGED gateway.** Metered data is communicated to the Data/Core Back-end system via MQTT, and then assigned to Devices for use by other system components. Sites, devices and assets are created beforehand via the Administrative User Interface.
2. **Claim of energy saving achievements from the end-users via the mobile app.** Energy measurements collected by the Gateway are sent to the Data/Core Back-end system and from there forwarded to the Analytics backend for modelling of energy use. When the mobile app reports user-triggered NFC and BLE events via MQTT or REST, such events are sent to the Data/Core Back-end system and from there are forwarded to the Analytics backend for modelling of the energy use and the Game backend for accessing their effects on the game state.
3. **Discovery of energy saving opportunities to assist the end users in saving energy.** Energy measurements collected by the Gateway are processed to create models of energy use. When new data are received, the Analytics backend compares them with the expected energy use at that time of the day and diagnoses deviations, which are in turn communicated to the Game backend. The Game backend will use that information to notify the end users if savings opportunities exist.

3 DATA/CORE BACK-END SYSTEM

This section presents the Data/Core Back-end system components and infrastructure that allows the overall platform to operate according to the game challenges requirements.

3.1 SITEWHERE SOFTWARE ENVIRONMENT

SiteWhere¹ was chosen as the Data/Core Back-end system for our application, providing an open-source platform with a number of rules and mechanisms for data exchange and operations.

3.1.1 DESCRIPTION AND OPERATION

SiteWhere is a system that facilitates the ingestion, storage, processing, and integration of device data. It provides the core backend functionality by fulfilling very specific roles.

IoT Server Platform

SiteWhere's main functionality is to supply a server based JAVA SPRING middleware between the sensing infrastructure and the different system components and acts as a controller for the processing of device data. It connects with noSQL & Timeseries databases in order to provide persistence of the logged data and scales with large number of devices so the whole history of the infrastructure is maintained and can be accessed at any point. It also provides the tools in order to structure the devices and categorize them according to their type, location and ownership and offers full control on a device's lifecycle (providing the functionalities of creating, deleting, updating, grouping, sending data). Moreover it provides an HTML5 administrative application that allows all of the system data to be viewed and manipulated in a structured way which makes their overview and administration easier and also more accessible to less experienced administrators.

Entity Structure/Device Management

As mentioned above, SiteWhere provides the tools to structure the entities according to their type, their location and their ownership, which greatly enhances the efficiency that the system can be administrated and the data processed. The available functionalities are:

- Each new asset or entity (i.e. a sensing device, an appliance, a specific location area, a person) is assigned a unique id and can be autonomously managed via external software. Specifically for devices a model for standard types of event data generated by them is provided (which includes measurements, alerts issued and location updated by the device) and the event that are logged are stored in massively scalable time series datastores (InfluxDB).
- The devices can be assigned to / associated with other entities. A device (a printer, an air conditioner, a pc etc) can be associated with a person, a location or another sensor device of our infrastructure thus giving us the ability to establish ownership room/location metadata and establish relationships with device.
- Devices can be grouped together according to a common role they fulfil, something that enhances efficiency by simplifying the way the devices can be retrieved by other backend processing services.

¹ <http://www.sitewhere.org/>

- Every top level entity is modelled as a tenant and can have a completely different configuration and structure without affecting other tenants. This can be used for modelling infrastructures that are unrelated to each other such as different locations, different buildings, pilot users etc on the same server.

Integration

SiteWhere provides an extensive list of third party frameworks and software tools that it can be connected with in order to extend its capabilities. The options include different databases, identity management frameworks, event streamers, event processors and others. Moreover being an open source software solution, new interfaces with other software tools and services can be created as needed. External communication with SiteWhere can be achieved via a built in extensive REST APIs. A communication interface utilizing the MQTT protocol is also implemented that can be used by devices and other embedded systems to send or get notified about new events and sensor data (e.g. NFC/BLE alerts and energy measurements).

3.1.2 IMPLEMENTATION

3.1.2.1 ARCHITECTURE

SiteWhere is the main data management solution, and provides APIs that the rest of the subsystems can use to send and request data from the database. Two public interfaces are used to establish this communication:

- The mosquitto MQTT broker, mainly used by the deployed IoT devices to provide efficient communication taking advantage of the lightweight characteristics of the MQTT protocol, and
- Rest APIs, mainly used by platform components and 3rd party software systems.

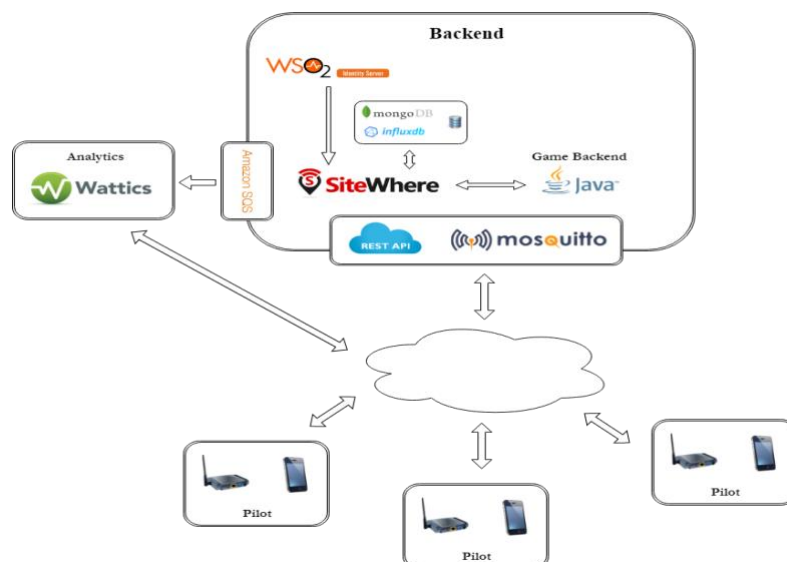


FIGURE 2: ORGANISATION OF SITEWHERE SYSTEM COMPONENTS

The server is exposed to the public internet, therefore is password protected so only authorized clients can login and use it. Credentials can be provided on request to the project coordinator.

3.1.2.3 INSTALLATION AND CONFIGURATION STEPS

In current setup the software is installed on a server running Linux CentOS 7. The software is extracted on the machine and the installation process is executed as described in detail in <http://documentation.sitewhere.io/userguide/installation.html>.

Upon successful installation, the user logs in with the default credentials (mentioned in the instructions above), changes the default passwords and creates new accounts as needed. Furthermore, communication with databases, MQTT brokers and the different backends needs to be configured. These configurations can be made from the 'Tenant Configuration' tab inside SiteWhere, as shown in the picture below.

3.2 DATABASES (INFLUXDB, MONGODB)

3.2.1 DESCRIPTION AND OPERATION

Time series (InfluxDB) and noSQL (MongoDB) types of database are used in the ChArGED platform. SiteWhere can use a hybrid approach where device management (meta) data is stored in MongoDB while event data (such as measurements and device alerts) is stored in InfluxDB which is optimized for fast, high-availability storage and retrieval of time series data. Both DBs are accessed through the SiteWhere's API. This choice provides a best of both worlds advantage, by enabling different features of each database for the different data stored in each one.

InfluxDB is a free and open-source cross-platform highly-scalable time series database that supports advanced clustering and performance.

MongoDB is a free and open-source cross-platform document-oriented database. Classified as a NoSQL database, MongoDB uses JSON-like documents with schemas. It is used to store the entities inside SiteWhere (i.e. sites, specifications, assets, devices, assignments and generally all data that are used to describe objects, users, locations and their relations). MongoDB is used because of its increased flexibility and performance.

3.2.2 IMPLEMENTATION

3.2.2.1 ARCHITECTURE

The databases are connected to and managed by SiteWhere which in turn provides a REST API and an MQTT interface that all the other system components can use in order to connect to the databases.

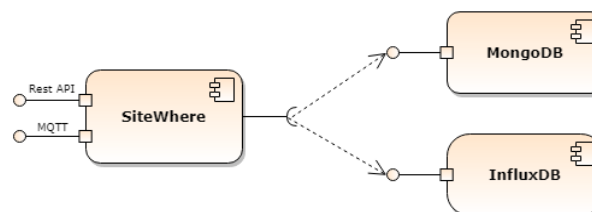


FIGURE 3: SITEWHERE'S TIME SERIES AND NOSQL DATABASES

3.3 USER MANAGEMENT (WSO2 IDENTITY SERVER)

3.3.1 DESCRIPTION AND OPERATION

WSO2 is an open source technology provider that offers an integrated enterprise platform for APIs, applications, and web services—locally and across the internet. Their WSO2 Identity Server² is natively integrated to SiteWhere to provide a secure identity management by managing identity and entitlements of the users securely and efficiently.

The Identity Server enables developers to reduce identity provisioning time, guarantee secure online interactions, and deliver a reduced single sign-on environment. The WSO2 Identity Server gives us the possibility to securely manage user accounts and enable persistence login for users across platforms, from the mobile app to other components. All user accounts and user information are created and managed inside this server. This provides us with a high level of security and protection of the sensitive user data. SiteWhere communicates with WSO2 Identity Server through a secure interface and obtains the information needed for importing a user in the system.

3.3.2 IMPLEMENTATION

3.3.2.1 ARCHITECTURE

The Identity Server provides a System for CrossDomain Identity Management (SCIM) interface which is a schema for representing users’ identities in conjunction with a REST API. This is used to manage users remotely in order to support user creation and user information management through the mobile app. The same interface is used by SiteWhere to automatically parse users and insert them into the system.

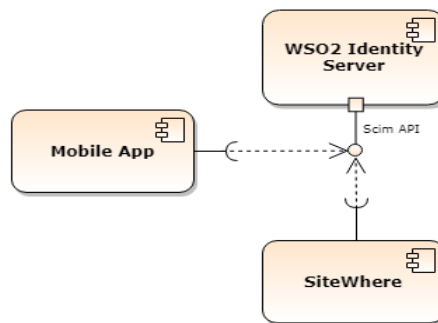


FIGURE 4: SITEWHERE’S IDENTITY MANAGEMENT THROUGH WSO2 IDENTITY SERVER

User Interface

The user interfaces in Sitewhere are oriented for the internal project developers of project components and the administrators of the pilots. It is not intended for end –users, who will access all related information only by the game user interface, specifically designed for this.

All the functionality is accessible through the interface’s side bar. Through the interface we can manage user provisioning (i.e. create, update and delete user accounts) as well as configure access rights and policies.

The identity server is configured to provide authentication and authorization services to the users already in the system. This is used by the mobile app in order to sign in/sign out the people that are playing the game. Currently the OAuth 2.0/OpenID Connect authorization/authentication scheme is used. This flow after the user is authenticated

² <http://wso2.com/identity-and-access-management>

provides the client (mobile app) with an expiring access token that in turn enables us to keep the user signed in for longer periods of time until the token expires or until the user signs out decreasing the frequency that the user needs to enter her credentials.

3.3.2.3 INSTALLATION AND CONFIGURATION STEPS

The WSO2 Identity Server is licensed under [Apache License Version 2.0](#) and available for download at <http://wso2.com/identity-and-access-management>. The software is downloaded and installed on the server. After the installation the default password is changed and new admin user accounts are added as needed.

3.4 ASSET MANAGEMENT/USER INTERFACE

3.4.1 DESCRIPTION AND OPERATION

SiteWhere provides a number of entities that can be used standalone or combined with other entities in order to represent our systems. The main categories are outlined below.

Tenants

SiteWhere provides an architecture where multiple IoT applications can run concurrently in separate containers. Each container is called a tenant and contains its own data and processing pipeline. Each tenant is associated with one or more system users which are allowed to configure and control it.

Device Specifications

Specifications are used to capture characteristics of a given hardware configuration. This is not necessarily a one-to-one mapping to a part number since some peripheral devices may have been added or certain characteristics upgraded. A device specification contains a reference to a hardware asset which provides the basic information about the hardware including name, description, image URL, etc

Devices

Devices are a SiteWhere representation of connected physical hardware that conforms to an assigned device specification. Each device is addressable by a unique hardware id that identifies it uniquely in the system. A new device can register itself in the system by providing a hardware id and device specification token. SiteWhere in turn creates a new device record via the APIs and (optionally) creates a placeholder unassociated device assignment (see below) to allow events to be collected for the device. Devices can be manually added via the REST services or via the admin UI.

Device Assignments

Events are not logged directly against devices, since a given device may serve in a number of contexts. For instance, a visitor badge may be assigned to a new person every day. Rather than intermingle event data from all the people a badge has been assigned to, the concept of a device assignment allows events to be associated with the asset they relate to at the time of the event. A device assignment is an association between a device, a site, and (optionally) a related asset. Some assignments do not specify an asset and are referred to as *unassociated*. A real-world example of this is a vending machine that has wireless connectivity to report inventory. The device is the asset, so there is no need to associate an external asset.

Assets

SiteWhere assets represent objects in the physical world – people, places, and (dummy) things. Device specification assets are used to describe the hardware information/configuration for a type of device. Device assignment assets are used to describe an entity associated with a device – a person associated with a badge, an appliance associated with a smart meter, or a room associated with a BLE sensor.

The above entities are used standalone or are combined in order to represent the objects, users and their relationships in our systems.

3.4.2 IMPLEMENTATION

3.4.2.1 ARCHITECTURE

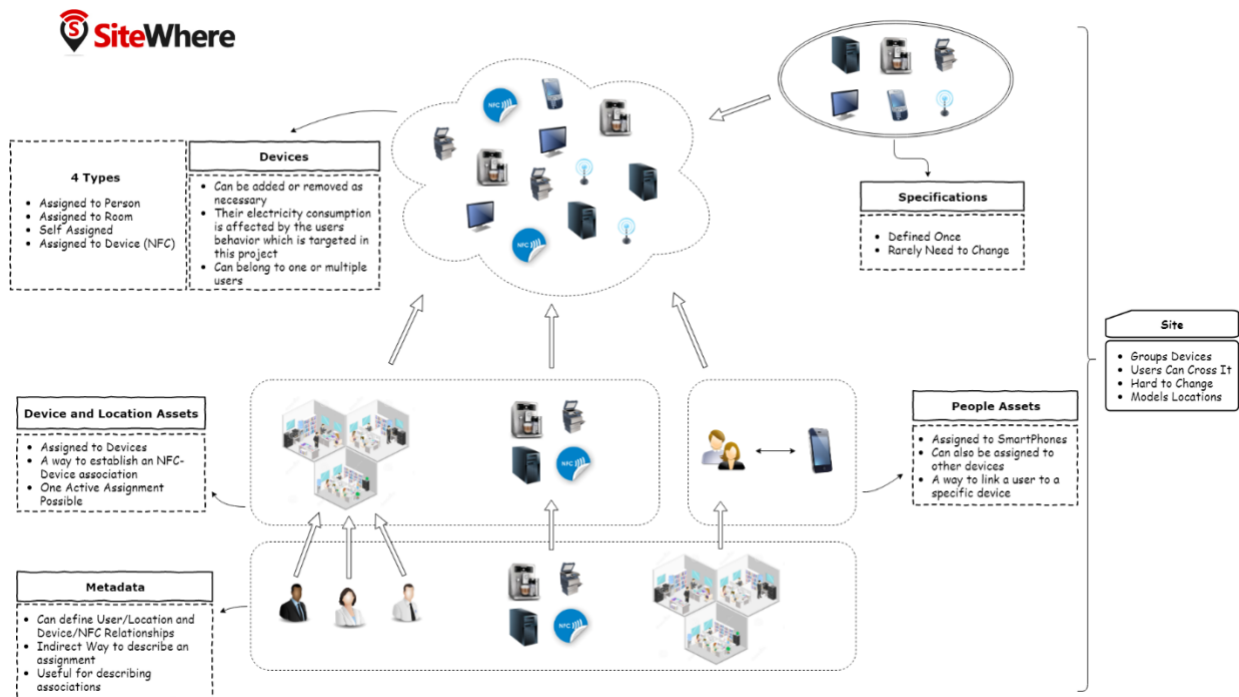


FIGURE 5: THE SITEWHERE ASSETS, DEVICE AND METADATA ORGANISATION

3.4.2.2 IMPLEMENTATION STATUS

In the CHARGED SiteWhere Backend, tenants have been used in order to model the different pilots. There is also a tenant dedicated to the development and testing.

Tenant		Development Tenant (development_tenant)	Select
Tenant		Pilot 1. EcoUrbanBuilding Barcelona (ecourbanbuilding_barcel...	Select
Tenant		Pilot 2. Musée National d'Histoire et d'Art (musee_national_lu...	Select
Tenant		Pilot 3. General Secretariat of the Municipality of Athens (athe...	Select

FIGURE 6: ILLUSTRATION OF THE SITEWHERE TENANTS CREATED FOR EACH PILOT SITE

Inside these tenants, we have modelled different sites for each pilot. These sites are modelled with regard to the building blueprints, as shown in the pictures below for the pilots in Luxembourg and Athens.

The screenshot shows a web interface for managing sites. At the top, there are navigation tabs: Sites (highlighted in red), Devices, Assets, Batch, and Schedules. A 'Charged User' dropdown is also visible. Below the navigation is a 'Manage Sites' section with a 'Filter Results' button and an '+ Add New Site' button. The main content area displays a list of four floor sites:

Floor	Created:	Updated:	Actions
Floor 0	2017-05-11 14:11:14	2017-05-11 14:13:28	[Edit] [Delete] [Next]
Floor 1	2017-05-11 14:11:25	2017-05-11 14:13:32	[Edit] [Delete] [Next]
Floor 2	2017-05-11 14:11:34	2017-05-11 14:13:35	[Edit] [Delete] [Next]
Floor 3	2017-05-11 14:11:46	2017-05-11 14:13:43	[Edit] [Delete] [Next]

Site Name	Created	Updated
Common Areas	2017-05-11 14:16:23	2017-05-11 14:18:50
EU Projects Department	2017-05-11 14:15:36	2017-05-11 14:18:53
General Secretariat of the Municipality of Athens	2017-04-27 12:17:20	2017-04-27 12:23:10
IT Department	2017-05-11 14:16:01	2017-05-11 14:18:56
Management and Administration	2017-05-11 14:16:55	2017-05-11 14:19:04

FIGURE 7: ILLUSTRATION OF HOW THE DIFFERENT PILOTS ARE DECOMPOSED INTO SUB-SITES WITHIN SITEWHERE

Assets are then created as described in the previous section. For example, device assets have been implemented to describe common devices found in the pilots, see **Error! Reference source not found.**

Assignments		Locations	Measurements	Alerts	Zones
<div style="text-align: right;"> Filter Results Refresh </div>					
Asset John Email: john@user.com Roles:	Device Samsung Galaxy Id: 06:0A:4B:C8:72:75 Info:	Assigned: 2017-08-08 13:31:19	Released: N/A	Status: Active	✖ ➤
Asset Ingrid Email: ingrid@user.com Roles:	Device Samsung Galaxy Id: 54:C7:26:D6:9A:C8 Info:	Assigned: 2017-08-08 13:30:56	Released: N/A	Status: Active	✖ ➤
Asset PC SKU: PC Info:	Device NFC Id: FC:D9:29:62:8A:BF Info:	Assigned: 2017-08-03 16:52:24	Released: N/A	Status: Active	✖ ➤
Asset NFC SKU: NFC Info:	Device PC Id: 70:C3:D0:22:D8:4C Info:	Assigned: 2017-08-03 16:45:49	Released: N/A	Status: Active	✖ ➤
Asset Printer SKU: Printer Info:	Device NFC Id: FC:D9:29:62:8A:BF Info:	Assigned: 2017-08-03 16:35:50	Released: 2017-08-03 16:40:02	Status: Released	✖ ➤
Asset NFC SKU: NFC Info:	Device Printer Id: 70:C0:E0:42:D8:4C Info:	Assigned: 2017-08-03 16:32:29	Released: 2017-08-03 16:40:00	Status: Released	✖ ➤

FIGURE 8: ILLUSTRATION OF THE ASSIGNMENT OF DEVICES TO ASSETS IN SITEWHERE FOR THE PILOT SITES

New devices and assignments will be created as needed to insert new assets and devices in our system (new employees, new devices etc).

3.5 RAW DATA VISUALIZATION (GRAFANA)

3.5.1 DESCRIPTION AND OPERATION

Grafana³ is a powerful open-source visualisation frontend optimized for time series databases such as InfluxDB. When working with sensor data a developer can be overwhelmed by the volume and the difficulty of exploring data with command line queries. Grafana allows us to quickly plot raw data from the SiteWhere database (InfluxDB) and create dashboards in a drag and drop manner. Within the scope of ChArGED project, it is mainly used as a development / debugging tool to quickly visualise raw sensor / user data, and assist integration between all subsystems, for any partner to use on his/her own. Furthermore, as the development progresses, Grafana embedded dashboards might be used to enrich specific user screens if needed. Grafana is part of the visualization of the Sitewhere, that facilitates the analytics. It is therefore also intended for the pilot site administrator or the energy analytics experts in the project.

³ <https://github.com/grafana/grafana>

3.5.2 IMPLEMENTATION

Grafana is a standalone server like software, developed in Go language, and runs along with the Data/Core backend server, in order to achieve direct/fast/easy access to the InfluxDB database.

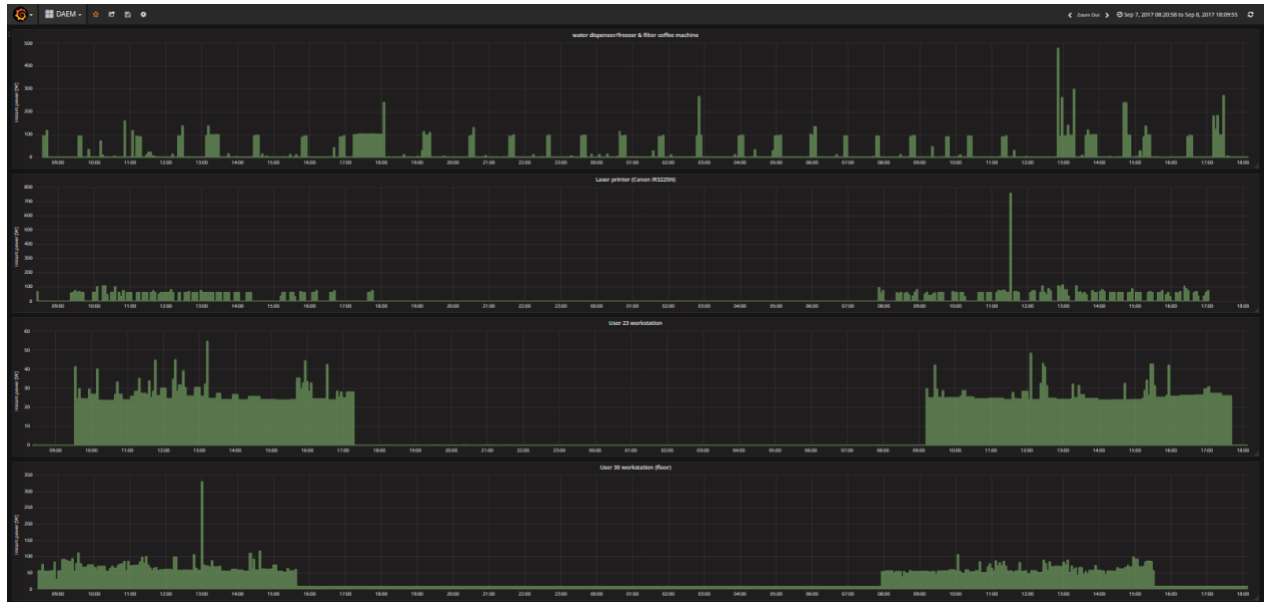


FIGURE 9: EXAMPLE OF DATA COLLECTED BY ZWAVE PLUGS IN THE DAEM PILOT SITE

4 SENSOR GATEWAY

This section focuses on the Gateway functionality developed for the purpose of ChArGED, which is about collecting and making data from the building / physical world environment available to the Core platform.

4.1 SENSOR GATEWAY SYSTEM ARCHITECTURE

To achieve the “Data Acquisition” process outlined in section 2.2, the Sensor Gateway has two connection interfaces within the global architecture:

- Building sensors (e.g. Smart plugs and Smart Meters)
- SiteWhere Data/Core backend

Various hardware and software requirements have to be fulfilled to support the needs of the platform. The *ProSyst mBS* gateway software⁴ by Bosch Software Innovations is used as basis of the Sensor Gateway used in the ChArGED project. For the remote software management and provisioning of the product, the *ProSyst mPower Remote Manager* (mPRM) by Bosch Software Innovations is also used (in a separate AWS instance, beside the ChArGED backend).

⁴ <https://www.bosch-si.com/iot-platform/iot-platform/gateway/software.html>

The Raspberry Pi⁵ (version 3 Model B) was chosen as hardware basis for the ChArGED Sensor Gateway. The Raspberry Pi is installed with a standard Raspian OS, including the Oracle embedded Java Runtime Environment (eJRE), the Communications Device Class Abstract Control Model (CDC_ACM) USB to serial driver and, as mentioned before, the ProSyst mBS Runtime for ChArGED.

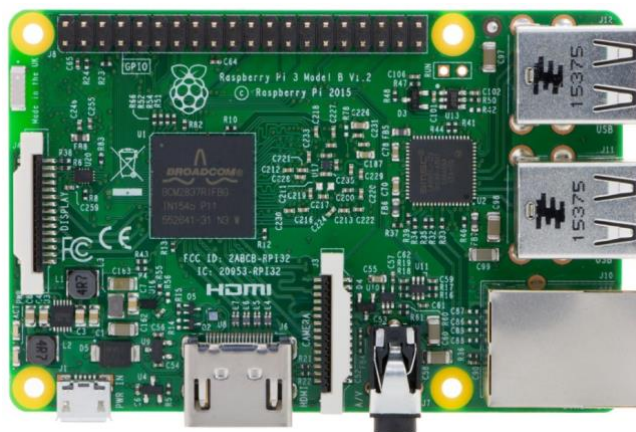


FIGURE 10: THE RASPBERRY PI (VERSION 3 MODEL B USED AS HARDWARE PLATFORM FOR THE SENSOR GATEWAY SOFTWARE

The data collection process required development of sensor drivers to retrieve data from third party sensors using industry leading communication protocols.

For the connection of Z-Wave devices, many controller options have been investigated, and two units have been selected:

- “Razberry” GPIO Module for Raspberry Pi (for both testing and pilot site deployment)
- USB Z-Wave Controller (for testing)

Two Z-Wave devices were connected to the Sensor Gateway using those controllers:

- Fibaro Smart Plugs
- Fibaro 4in1 Sensor (Temperature, Humidity, Luminosity, Motion/Presence)

These devices are already managed by the mBS Runtime and included into the product portfolio, which allowed data to be immediately collected from the devices.

The *AcuRev 2000* multichannel Modbus meter by Accuenergy was also connected to the Sensor Gateway via the Modbus protocol, to collect energy measurements at the three pilot sites.

All connected devices communicate their data to the Sensors Gateway, which preprocesses and forwards it to the SiteWhere backend via MQTT, as depicted in Figure 11.

⁵ <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>

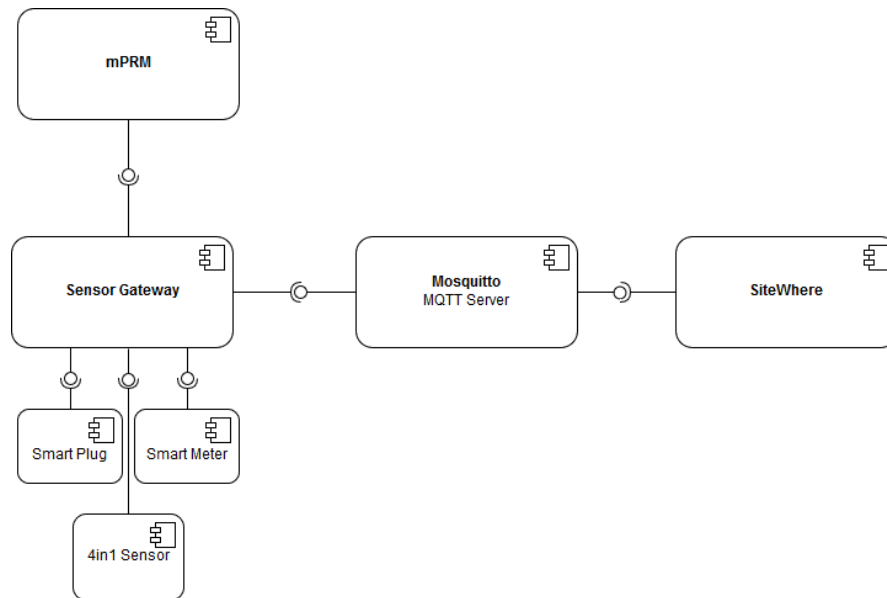


FIGURE 11: ILLUSTRATION OF THE SENSOR GATEWAY DATA COLLECTION, PROCESSING AND COMMUNICATION TO SITEWHERE

4.2 HDM/SITewhere INTEGRATION

4.2.1 DESCRIPTION AND OPERATION

The Bosch's *ProSyst mBS* Sensor Gateway component is not natively compatible with SiteWhere's asset management backend, thus a mapping was required and implemented.

On the *mBS* Sensor Gateway side, devices connected to the gateway (e.g. Z-Wave smart plugs) are assembled through the Home Device Manager (HDM) abstraction layer HDM, in order to have a protocol independent layer for developers, who want to develop against the same API for every connected device.

SiteWhere provides a SSL-TLS secured MQTT interface to add, delete, update and control assets connected to it. Therefore, integration between the HDM on client side and SiteWhere on the backend side has been developed.

4.2.2 IMPLEMENTATION

4.2.2.1 ARCHITECTURE

The diagram in Figure 12 shows the main blocks involved into the Raspberry Gateway software. On top of the standard OSGi modules new dedicated ZWave Connector and SiteWhere Service module have been developed, marked with blue colour.

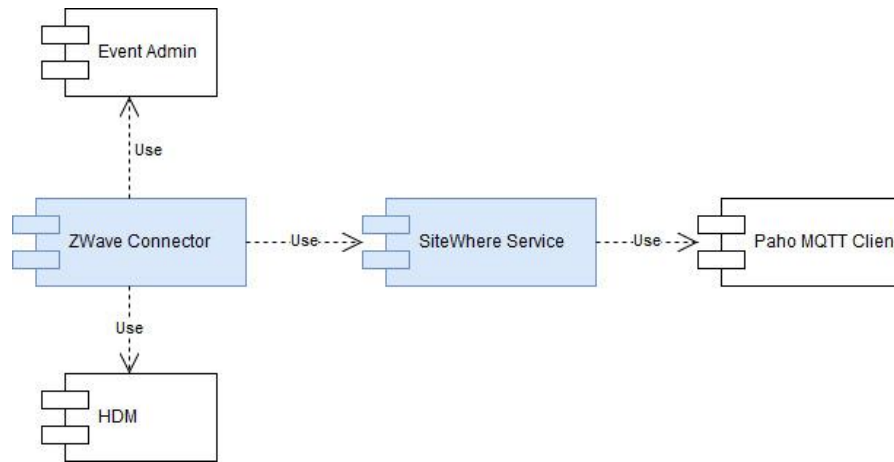


FIGURE 12: INTEGRATION OF THE SENSOR GATEWAY’S HDM WITH SITEWHERE VIA MQTT

4.3 MODBUS DEVICE INTEGRATION TO SITEWHERE

4.3.1 DESCRIPTION AND OPERATION

The Sensor Gateway was enhanced to collect energy readings in near real-time resolution (10 - 30 seconds) from the multi-channel power and kWh *AcuRev 2000⁶ meter* from Accuenergy installed at the pilot sites (reference device for the implementation) and the *Kaco Solar Inverter* (deployed at DAEM site) as additional integrated energy meter device.

The *AcuRev 2000* provides a Modbus interface which can be accessed in real-time. This interface was used to integrate the meter to the Gateway Software. Gathered data is sent directly to the SiteWhere backend.

The *Kaco Solar Inverter* provides its generated power and energy values also over a Modbus interface. It is integrated like the *AcuRev 2000* into the Gateway software and the SiteWhere backend.

4.3.2 IMPLEMENTATION

4.3.2.1 ARCHITECTURE

Figure 13 shows the main blocks involved into the Raspberry Gateway software. On the top of the standard OSGi modules are developed dedicated Modbus Connector and SiteWhere Service modules, marked with blue colour.

⁶ <https://www.accuenergy.com/product/acurev-multi-circuit-power-metering-system>

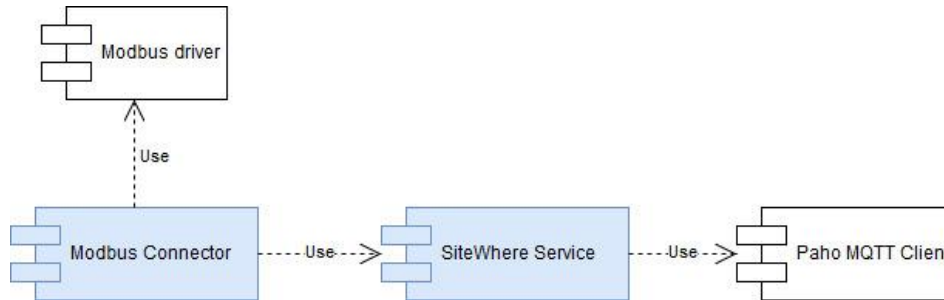


FIGURE 13: MODIFICATIONS TO THE SENSOR GATEWAY FOR INTEGRATION OF MODBUS METERS

5 ANALYTICS BACKEND

This section focuses on the Energy Analytics functionality of the ChArGED platform, based on the proprietary technology provided by Wattics. Wattics backend analytics engine already exists as a commercial product and we first outline its main features to provide context and system architecture in which the new components have been developed. In a second step, we document the new system components developed for ChArGED, and the components that have been modified for ChArGED, as well as showcase the means of integration within the ChArGED core backend.

5.1 WATTICS BACK-END ARCHITECTURE

The Wattics Energy Analytics backend, depicted in Figure 14, is connected to the ChArGED’s Sitewhere Core Back-End to enable the creation of actionable insights from raw energy measurements taken at the pilot sites, allowing the game engine to validate game challenges and measure the impact of various user actions over the overall energy expenditure.

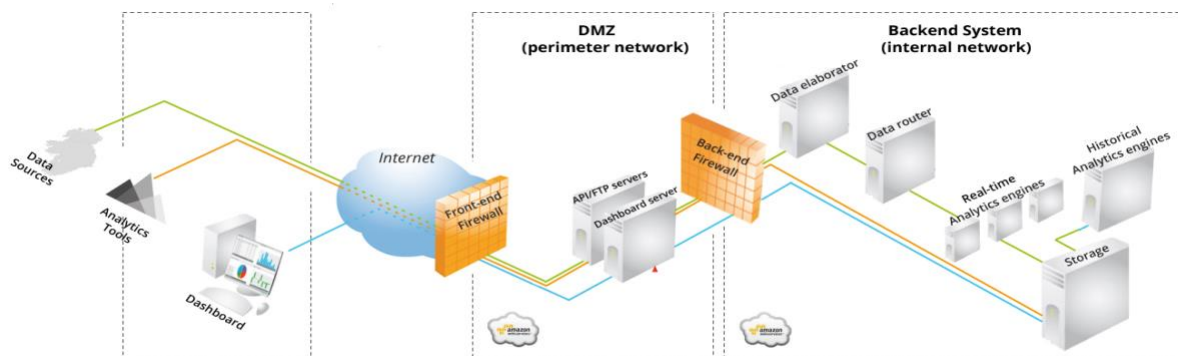


FIGURE 14: THE WATTICS CLOUD BACKEND SYSTEM ARCHITECTURE

The Wattics backend is interfaced with the Sitewhere backend via RESTful web services, which allow energy measurements, NFC swipe alerts and BLE location events to be received as input, and measurements of load demand reduction, energy savings and energy saving opportunities to be returned. Authenticated data streams are processed in real-time through parallel analytics engines to produce valuable insights for the application.

The Wattics backend infrastructure is brought to the project as background IP, and has been adapted to the needs of the project as follows:

- New API endpoints to process NFC and BLE data packets;
- New analytics engine to validate control actions have been taken by users (e.g. device switched off when going home or when away for more than N minutes), and to estimate the energy savings achieved by such actions;
- New analytics engine to diagnose inefficient operation of electrical devices based on concurrent power activity (e.g. A/C left on when window is open), and to estimate the energy wasted due to such actions;
- New notification mechanism to export insights generated to Sitewhere.

In addition to these modifications to the Wattics backend, a micro service was developed to sit in between SiteWhere and the Wattics Analytics backend to enable seamless integration of both backend systems via Amazon SQS. The following provides more details about the new analytics engines developed specifically for the needs of the project, as well as the Wattics SQS Poller connecting the pieces together.

5.2 COMPONENT FOR AUTOMATED DISAGGREGATION AND ALLOCATION OF ENERGY SAVINGS

5.2.1 DESCRIPTION AND OPERATION

The game concept involves users saving energy within the floor plan through a range of energy conservation measures triggered by the game challenges, and claiming such savings by swiping their NFC enabled smartphones over NFC tags located nearby electrical appliances or their controllers. The game app therefore needs to validate that such actions are really taking place, and to assign impact / challenge scores accordingly. As such, the objective of this new software component is to disaggregate energy saving actions from the overall set of normal actions taken throughout the day by the various users, and assign the savings achieved to the users.

It is important that the platform back-end validates each action taken by a user, as the game relies on counting points based on the completion of challenges. Any action taken must lead to a reduction in energy use for a specific appliance (or group of appliances), which must be identified as the action occurs. As such the back-end analytics system must:

1. Monitor energy use for a given appliance (or group of appliances) affected by a user's control action. This means monitoring energy use within the premises at a granularity that allows for the energy use pattern of the specific appliances to be monitored individually.
2. Detect changes in energy use over that circuit that relates to the NFC swipe or BLE event from the user.

Submetering an entire electrical equipment inventory with submeters and smart plugs was not an option because it was too costly. Identifying an appliance's energy use pattern in software from an electrical measurement taken at the main incomer of a building was neither suitable, as the number and nature of devices fed from that single circuit does not allow for software load disaggregation based on power signatures. Indeed, in office buildings, all computers 'exhibit' a similar power footprint on the power line, preventing any software load disaggregation to work accurately.

The only commercially viable option, the one adopted within the project, is to submeter medium-level electrical circuits feeding groups of appliances (e.g. set of fancoil units in the same room), and run analytics on the data collected to monitor power variations of specific electrical equipment fed from such circuits. Concretely, this means for example monitoring a circuit feeding all the power plugs of a meeting room (as opposed to monitoring each wall socket of a meeting room), allowing localised analytics on a smaller subset of electrical units for better accuracy and

feedback for the game app. This approach of submetering a subset of key electrical circuits is reflected in Deliverable D4.2, where the monitoring setup for all pilot sites is documented.

A new analytics engine was developed to run on such medium-level circuit measurements in order to provide a validation and measure of savings achieved by a specific control on an electrical appliance fed from that circuit.

We have initially intended to convert the medium-level circuit measurements into appliance-level measurements with techniques such as software load disaggregation, as the number of electrical devices fed from the medium-level circuits would not be a major constraint anymore. Yet, while software load disaggregation is a methodology proven in domestic environments with many commercial products reviewed in Deliverable D2.1, it falls short in public buildings and generally larger-scale environments, for which no commercially viable solution exists. The main reason is that while households can be counted in millions with each site using the same set of home appliances, providing training datasets to enable self-learning and auto-configuration, public building environments all differ in size and type, with new equipment being added or replaced regularly. The latter calls for manual configuration and re-configuration for each deployment, and is not scalable for a commercial solution targeted to public buildings. For that reason, the cost of designing a software load disaggregation engine that should work in all public environments far exceeds the needs and scope of the project, and we have instead opted for an analytics engine that requires minimum to no configuration and that can scale to any environment.

Within ChArGED, the validation that a control action claimed by a user is really implemented is achieved with circuit-level measurements taken by physical meters, combined with metadata provided by the user swipe events and beacon location sensors. The principle of such analytics is energy use allocation, where energy reduction events are disaggregated from the power signal thanks to user-triggered notifications, and allocated to users associated to the equipment being operated.

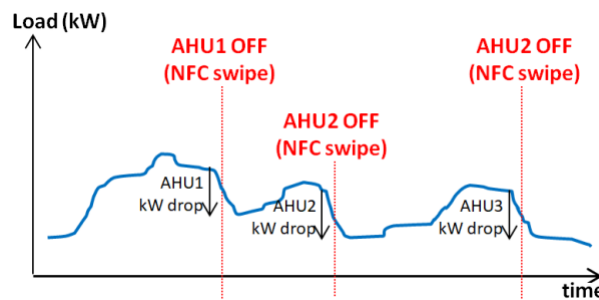


FIGURE 15: ILLUSTRATION OF POWER DROP MEASUREMENT ON A SINGLE ELECTRICAL READING WITH NFC SWIPE EVENTS FROM USERS PROVIDING THE NAME OF THE APPLIANCE CAUSING THE DROP

The disaggregation and energy allocation engine works as follows:

1. The Sitewhere's core/backend platform informs the Wattics Analytics engine that an appliance has been operated by the user after it received an NFC swipe alert from the user's mobile app.
2. The Wattics analytics software runs the NFC swipe alert against the power measurements of the circuit feeding the appliance operated by the user to detect significant power variations.
3. The Wattics analytics software analyses the power variations and informs the Sitewhere core/backend platform of the drop in energy use measured in relation to the user's operation of the appliance, as well as a quantification of the savings achieved by doing so.

- The game backend calculates the points to be given to the user and the savings are stored within the platform database.

In particular, the measure of power variations and associated drop in load demand (before and after NFC swipe event) as well as the energy savings generated are assessed by the analytics engine using kW variations, the expected period for which the equipment will have been left on, and other conditions which need to be taken account to adjust the real savings (e.g. atmospheric conditions).

Game challenges based on NFC swipes

It is expected that more than one device will be fed from any electrical circuit. For that reason, the power measurements collected by the platform correspond to the entire set of devices connected to a given circuit, and not only one computer or light bulb. The objective is to measure how much impact a user has had on the circuit’s energy use; this is done by comparing the circuit’s load pattern before and after ChArGED is introduced, using the NFC swipes as triggers and BLE/work hours information as time delimiter.

As illustrated in Figure 16, where the usual load demand pattern of a given electrical device is in green and the real load demand measured is in blue, the difference between the two load curves is minimal with no savings achieved if a user was already switching the device off before ChArGED was introduced.

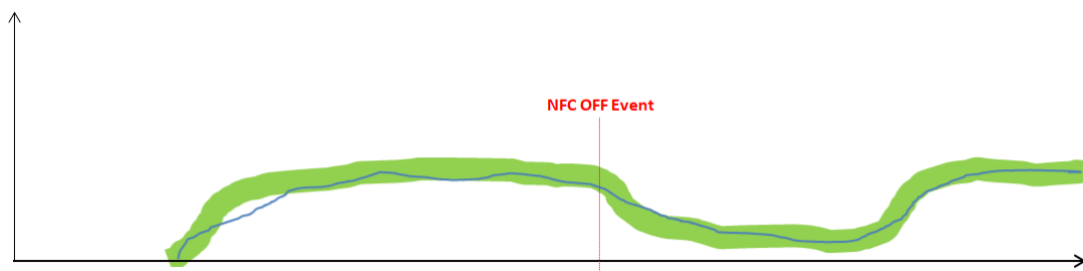


FIGURE 16: ILLUSTRATION THAT NO KWH SAVINGS ARE ACHIEVED BY A SPECIFIC USER CONTROL ACTION, WHEN THAT ACTION WAS ALREADY CONDUCTED BEFORE CHARGED WAS INTRODUCED.

On the other hand, a major reduction is measured if the user switching off the device is new, as illustrated in 38.

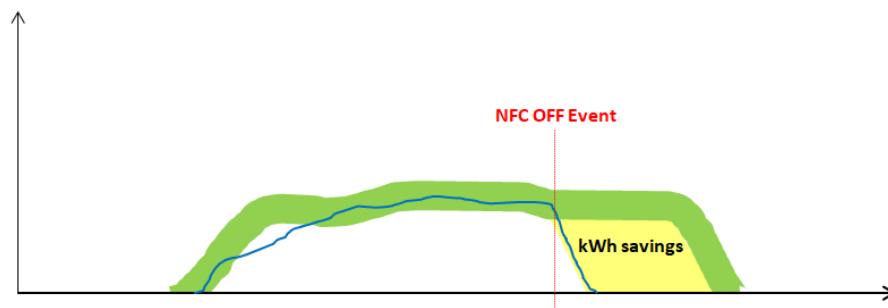


FIGURE 17: ILLUSTRATION OF THE KWH SAVINGS ACHIEVED BY A SPECIFIC USER CONTROL ACTION

The calculation of savings therefore requires:

1. The load demand curve for that circuit before ChArGED is introduced;
2. The load demand curve measured for the circuit;
3. Period for consideration (when the device would have potentially been left powered on)

PCs are operated by their owners, and are seldom switched off by other staff. For that reason, we can assume that the period for which savings can be assessed after a PC has been switched off is directly linked to when the user returns to his desk, either later in the day or the day after. This information can either be provided by BLE information, or inferred from office work hours.

Lights and A/C units are operated by more than one person, and the period for which savings can be assessed after a light or A/C unit has been switched off is directly linked to when any user enters the room. This information can either be provided by BLE information.

BLE alerts are generated when a specific user returns to its original location, or when any user returns to a location, which allows the Analytics backend to trigger the savings calculation, using the period between the NFC OFF event and the BLE alert.

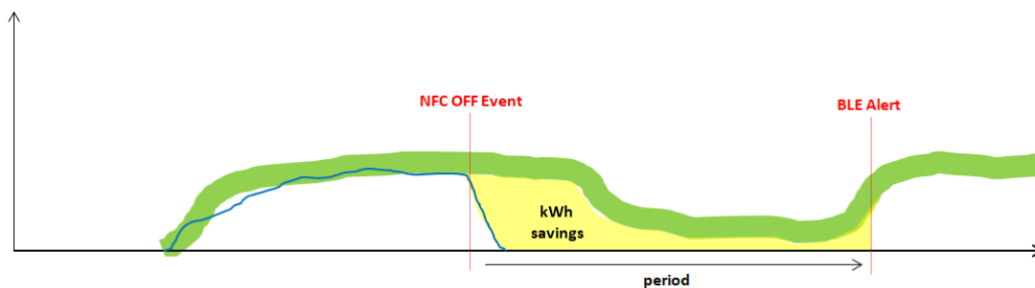


FIGURE 18: ILLUSTRATION OF THE KWH SAVINGS ACHIEVED BY A SPECIFIC USER CONTROL ACTION DELIMITED BY A BLE ALERT

When BLE alerts are not available, savings calculation is automatically triggered at the start of the day when most staff returns to work, e.g. 8AM, using the period between the NFC OFF event and the work hour.

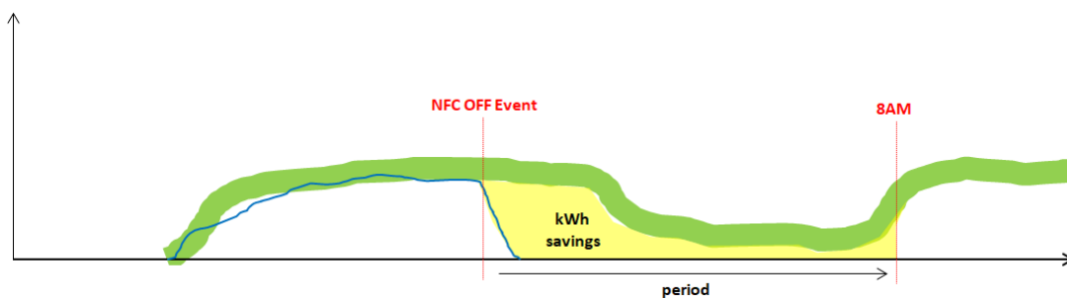


FIGURE 19: ILLUSTRATION OF THE KWH SAVINGS CALCULATION TRIGGERED AT THE START OF THE DAY COVERING THE PERIOD FROM THE NFC SWIPE EVENT TO WHEN THE USER IS EXPECTED TO BE RETURNED AT HIS DESK

In the case multiple users swipe their NFC stickers to indicate they switch their PCs off when going home, the period covered will not change but the savings measured on the circuit will increase according to all the control actions made by the users.

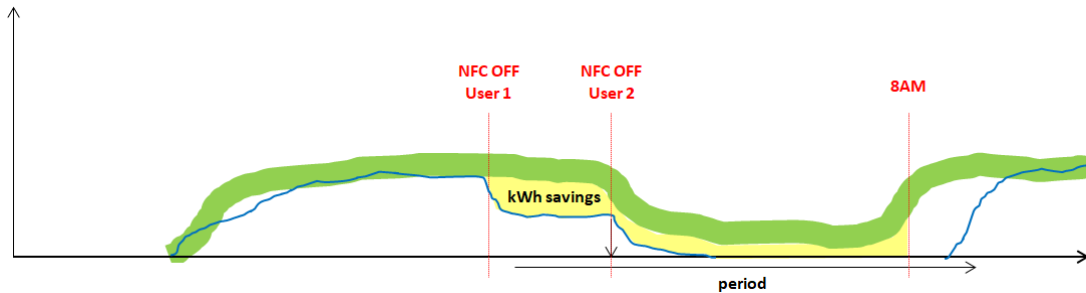


FIGURE 20: ILLUSTRATION OF THE KWH SAVINGS CALCULATION TRIGGERED AT THE START OF THE DAY COVERING THE PERIOD FROM THE FIRST NFC SWIPE EVENT TO WHEN THE USERS ARE EXPECTED TO BE RETURNED AT HIS DESK

A special case also exists when dealing with game challenges targeting elevators, when savings are calculated every time a user uses the stairs. For those game challenges NFC stickers are placed in the middle of the stairs and the users must swipe their phones to indicate they use the stairs. The savings calculation procedure depicted above cannot be implemented because the event in itself is too short for any power measurements to be made, unless the monitoring sampling granularity is at second-level. An elevator trip can indeed be a matter of seconds, so measuring the load demand is difficult. The approach taken here is to calibrate the energy used for an elevator trip, when the ChArGED application is commissioned, and add that figure to the savings achieved for every single staff trip via the stairs.

Game challenges based on BLE presence detection

BLE offers an advantage as opposed to NFC swipes, as it provides two alerts to the Core/Backend system, when a user leaves a location and when it returns to the location. For that reason, the period for consideration when measuring savings is delimited by those events.

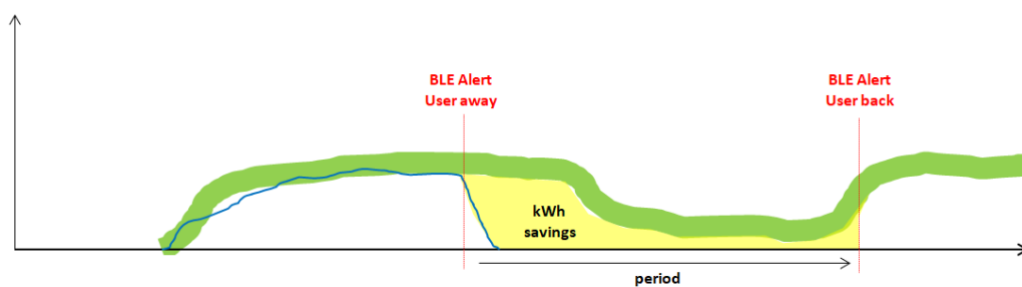


FIGURE 21: ILLUSTRATION OF THE KWH SAVINGS CALCULATION DELIMITED BY BLE ALERTS WHEN A USER LEAVES AND RETURNS TO A LOCATION

5.3 COMPONENT FOR AUTOMATED DISCOVERY OF ENERGY SAVING OPPORTUNITIES

5.3.1 DESCRIPTION AND OPERATION

The game app needs to provide suggestions in the game based on individual consumption patterns. This translates into the need for the platform to identify waste patterns and energy saving opportunities that can then be notified to the end users.

5.3.2 IMPLEMENTATION

Wattics' existing Sentinel analytics engine is used as infrastructure and was updated to serve the needs of the project. Wattics Sentinel Analytics is a software engine that models energy use over time and detects unexpected/abnormal energy patterns as they occur. For instance, the software can model what the energy use should be in a meeting room on a Monday at 3PM, and will be able to issue a notification should the actual energy use deviates from the expected pattern.

Within ChArGED, the engine is deployed to create the same energy model and detect anomalies, but the notifications and feedback to the SiteWhere backend are more valuable thanks to the additional context provided by presence detection beacons and NFC swipes. The analytics engine links notifications to specific users, and relates to specific control actions that can be taken within the area (room). Control actions taken in the past may also be used as references when new anomalies and energy saving opportunities are discovered.

5.3.2.1 ARCHITECTURE

The modelling engine requires time-series data, and takes as input:

- Energy measurements (electricity, water, gas, fuel, heat consumption etc)
- Real-time data (active/apparent/reactive power, flow, voltage, current, etc)
- Environmental data (temperature, humidity, occupancy etc)
- User data (NFC swipes, etc)

Environmental datasets are used for normalisation of the energy/real-time data before modelling, and other datasets such as user data are used to discard energy events linked to user triggered events (for validation that the deviation is not the result of a user action).

The model creation process is divided in three parts:

- Outlier detection to discard outliers periods,
- Cluster algorithm to group together periods with a similar pattern
- Creation of inner/outer band creation, which define the boundaries for the range of values that we expect to match with future readings.

1. Outlier detection

The outlier detection algorithm is configured to discard measurements based on specific rules dictated by the dataset constraints. For example, with a 4-week window period, all elements that do not fit within the 90% percentile can be discarded, with a maximum of one element discarded per period. In this implementation, we use the LOF outlier detection algorithm, looking at the Euclidian distance between elements. The output LOF coefficient is processed to discard elements outside the denser region.

2. Clustering mechanism

The clustering algorithm uses k-means to compare individual time periods and create clusters of similar periods. The metric used to compare the data is the squared Euclidean distance. As input to the clustering algorithm, we used one-day time periods defined by their 24 hourly mean, maximum and minimum load demand values. A metric based on Sum of Square Errors was used to decide the number of states that represents best the similarity between periods.

3. Inner/outer band

The measurements for the group of days identified within a same state are used to create a model, composed of two thresholds: an upper and a lower threshold. The area between the two thresholds indicates the area where we expect the values to fall inside.

We define the upper and lower thresholds as the 5th and 95th percentile, see Figure 22.

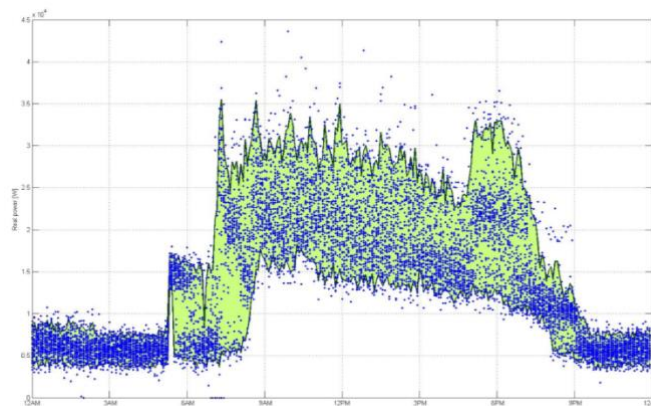


FIGURE 22: MODEL CREATION SHOWING HISTORICAL MEASUREMENTS (BLUE DOTS) AND THE MODEL (GREEN AREA)

When the models are created and enabled, an event recognition engine compares new measurements against the model to identify unexpected deviations. When detected, a message is sent to the user to point to the anomaly and the potential energy saving opportunity.

Various degrees of tolerance can be considered for issuing notifications, so as to avoid spam and false positives. For instance padding is added to the model and rules on the continuity of the deviation can be considered (e.g. duration, kWh cost, repetition, related user-triggered event etc).

Based on the user context, such as the time of the day, the location and the user's association to a specific device, the software will create a specific event to the gamification engine indicating the deviation specific and the device/circuit affected. This will allow the game mechanics to interact with the end-user and get them to address that saving opportunity.

5.4 ENERGY MANAGER FRONT-END DASHBOARD

5.4.1 DESCRIPTION AND OPERATION

The Energy Manager Front-End dashboard allows the facility manager, and possibly the manager of the game within the building, to monitor the overall savings achieved within the building. While the mobile app tracks individual achievements, the dashboard aims to review the overall impact of the savings strategy, with no interest on the users' scores and ranking.

The facility manager wants to quantify the overall savings achieved by the introduction of the ChArGED application. For this purpose, the facility manager will connect to the dashboard on a daily or weekly basis, to get update on the savings achieved and the energy use trend compared to same periods in the past. Because facility managers are very

busy, they will only spend a few minutes each time they connect, but may spend a few hours each month to produce monthly reports to management.

Facility managers with energy management background are at ease with modern technology, and used to work with energy data. Facility managers converted to energy managers over time may not have the same expertise, but our experience tells us that they are comfortable with graphs and simple data analysis.

The dashboard was designed to be used as a tool to monitor and report on the savings achieved, and as such does not require the minimum granularity available at the meters or real-time display. The data is however normalised and unified to be presented with a common periodicity for analysis, and with a delay that does not exceed 15mn.

5.5 COMPONENT FOR MICROGENERATION ENERGY FORECASTING

5.5.1 DESCRIPTION AND OPERATION

Solar energy generation has been deployed in the DAEM pilot site, in order to maximize the building energy savings, increase end user awareness and also try to use the mobile app to maximize the solar-based electricity consumption during production, avoiding the need of energy storage.

The solar solution utilizes a solar inverter (Kaco) with rich data communication capabilities (modbus TCP protocol) in order for the generated electricity to be as analytically monitored as possible to assist production forecasting mechanisms. Solar inverter will be connected to the gateway and its gateway middleware IoT integration software mBS SH. Through the device abstraction of the mBS SH the data will be prepared for the ChArGED backend and sent via MQTT to the ChArGED core platform and distributed to all other system components.

The Component for Microgeneration Energy Forecasting calculates periodically and sends to the SiteWhere core backend the expected energy generation per hour for the following day(s).

An external server, part of Plegma's cloud infrastructure, is utilized to provide the solar power microgeneration forecast, for the system deployed at DAEM pilot site, based on weather forecast of the specific area (Athens, Greece). The server is connected to a 3rd party weather forecast provider APIs to pull the weather forecast data (OpenWeatherMap). The service collects data for each day and a 5 day ahead forecast and saves it into a database. For each day, there is hourly forecast that is used to predict the PV power output.

5.5.2 IMPLEMENTATION

Implementation is based on theoretical principle of photovoltaic power production (analysis follows) enriched with Plegma's optimizations for taking in to account the characteristics of the particular system deployed in one of the pilot sites (DAEM) in order to produce a more accurate forecast.

Photovoltaic Collectors

For solar photovoltaic collector the electric power produced is given by,

$$\dot{W}_{\text{elec}} = \eta_{\text{PV}} A_{\text{coll}} G_i$$

where

η_{PV} : conversion efficiency of the photovoltaic cell

A_{coll} : area of the collector

G_i : incident solar radiation

The efficiency of the solar cell is given by

$$\eta_{\text{PV}} = \eta_o [1 - \beta_{\text{PV}} |T_o - T_c|]$$

where

η_o : conversion efficiency of the photovoltaic cell at the reference temperature

T_o : reference temperature, 25°C

β_{PV} : temperature coefficient for the solar cell

T_c : average solar cell temperature

with

$$T_c = T_{\text{amb}} + C_f (218 + 823K_t) \frac{\text{NOCT} - 20}{800}$$

where

K_t : monthly clearness index (comes from weather data)

NOCT: Nominal Operating Cell Temperature

C_f : tilt correction factor

$$C_f = 1 - (1.17 \times 10^{-4})(s_M - s)^2$$

where s_M is the optimum tilt angle and s is the actual tilt angle, both expressed in degrees. The constants for the above equations are provided in the table below for several different types of solar cells.

Table 1: PV Module Characteristics for Standard Technologies

PV module type	η_o (%)	NOCT (°C)	β_{PV} (%/°C)
Mono-Si	13.0	45	0.40
Poly-Si	11.0	45	0.40
a-Si	5.0	50	0.11
CdTe	7.0	46	0.24
CIS	7.5	47	0.46

FIGURE 23: OVERVIEW OF THEORETICAL PRINCIPLES OF PHOTOVOLTAIC POWER PRODUCTION ⁷

The incident solar radiation has been taken from the PVGIS⁸, a European founded project supported by the European Commission Joint Research Centre.

⁷ <http://www.egr.msu.edu/classes/me417/somerton/SolarCalc.pdf>

⁸ <http://re.irc.ec.europa.eu/pvgis/>

In order to be able to calculate the PV production on an hour interval we used more advanced calculations. One such model, used in the basic solar energy system model, is the half-sine solar irradiance model. The only input required is the times of sunrise, sunset, and the peak noontime solar irradiance level

$$I = I_{noon} \sin \left[\frac{180 * (t - t_{sunrise})}{(t_{sunset} - t_{sunrise})} \right]$$

The input values for sunrise and sunset are taken from the a 3rd party API (OpenWeatherMap) and by using the Momentjs library and the country of the installation we can calculate the time *t* for the formula.

Using the above formula, we can modify our prediction from daily to hourly.

Plegma’s implementation has a form to define the values and the location that you want to predict the PV energy Forecast for.

Solar Energy Predictions

Effective Area (in square meters)

Panel Technology

Size of panel (in cm)
 Height
 Width

Solar Tracking

Panel angle from the horizontal (in degrees)
 Angle

Module Efficiency (%)

Performance of panel (in kWh)

Descending performance of panel per year

Year of installation

Choose Location




FIGURE 24: PLEGMA’S SOLAR MICROGENERATION FORECAST CONFIGURATION SCREEN

That form sends through a web-socket the information to the server. The server checks the location and in case it is a new location then a prediction site is created otherwise the prediction is sent to the client. For each prediction site, the server calls the weather API to get forecast data and starts to calculate the PV production.

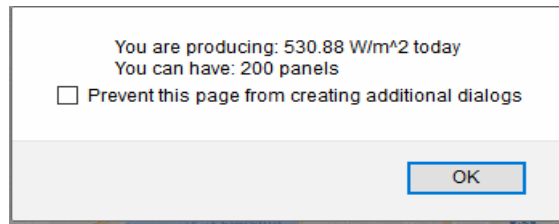


FIGURE 25: THE PV PREDICTION OUTPUT FROM THE SYSTEM

The server responds with the prediction and the number of panes you can install on the area you entered. These data are saved to a database and can be used again to just get the forecast.

6 MOBILE APP

1.1 FRONT-END

IMPORTANT NOTE: This task has been implemented so far by partner PEAK. There is a first version that has been implemented with the current plan. As the participation of PEAK in the project has been terminated, it is taken over to be implemented by European Dynamics SA. The implementation plan is changed and will be reported in the periodic report.

6.1.1 DESCRIPTION AND OPERATION

The Charged Mobile App provides the front-end towards the user, visualizing the whole backend mechanics and provided data in a user friendly, appealing, modern and motivating interface to ensure continuous usage of the app.

The app is intended to run on Android Mobile phones supporting API Layer 21 (Lollipop) and being equipped with NFC on the hardware side.

The Mobile App is not natively developed, but uses a technology called ReactJS. React follows current modern approaches using web technologies on mobile phones making it easy to implement apps in HTML, JavaScript and CSS plus the benefits of using a custom made interfaces and designs not following usual templates. Besides this, React offers real advantages on data changing over time, as no active updates are necessary like in usual JS applications, but React uses the “render” methods to automatically update the representation of the data.

The Front-End connects to a Mobile-App-Gateway ensuring the communication between other backends and the mobile App. The Mobile-Game-Backend ensures constant updates of running challenges and delivery of the deltas to the Front-End where these find their representation once the User opens the app. This ensures battery and performance advantages compared to have this functionality within the Mobile App directly. Additionally this comes with performance advantages due to the app connecting only to one Backend, and not multiple such as SiteWhere and the Rues Engine (Game Backend).

DASHBOARD

The dashboard will always be the first touchpoint for the user when opening the app. The dashboard contains most of the useful information to quickly get an overview on the last updates, current leader board position and the team’s performance.

From here the user can jump to various functionalities such as:

- Accept his / her first challenge
- Check the leader board and the current own status
- Check his / her currently active challenges and stats
- Check the Won / Lost challenges
- View details and accept new challenges
- Get a glimpse of the emotionalized item used to motivate the user using the app



FIGURE 26: THE CHARGED MOBILE APP VIEW OF ACTIVE CHALLENGES AND WHO WON CHALLENGES ON THE DASHBOARD

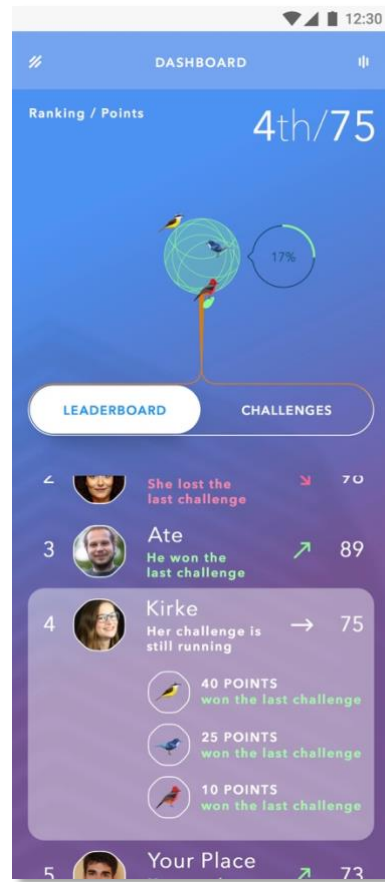


FIGURE 27: THE CHARGED MOBILE APP LEADER BOARD VIEW WITH OPEN DETAILS OF A SPECIFIC USER

Depending on the user role, the App will have different functionalities. Depending on the type of the account different functionalities will be offered to the users. A regular user will be able to only play the game but a local admin will also be able to add/remove devices, NFCs from the system etc. The role can be configured at the user account in the Backend and delivered within the user object.

DISPLAY A CHALLENGE / ACCEPT A CHALLENGE / COMPLETE A CHALLENGE

The challenges view supports daily, weekly and monthly challenges which can be played in parallel. Filter will easily make these browsable.

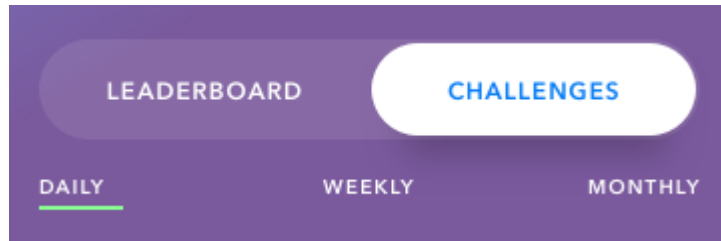


FIGURE 28: THE CHARGED MOBILE APP REVIEW OF CHALLENGES

Morning, Evening and lunch challenges' views:

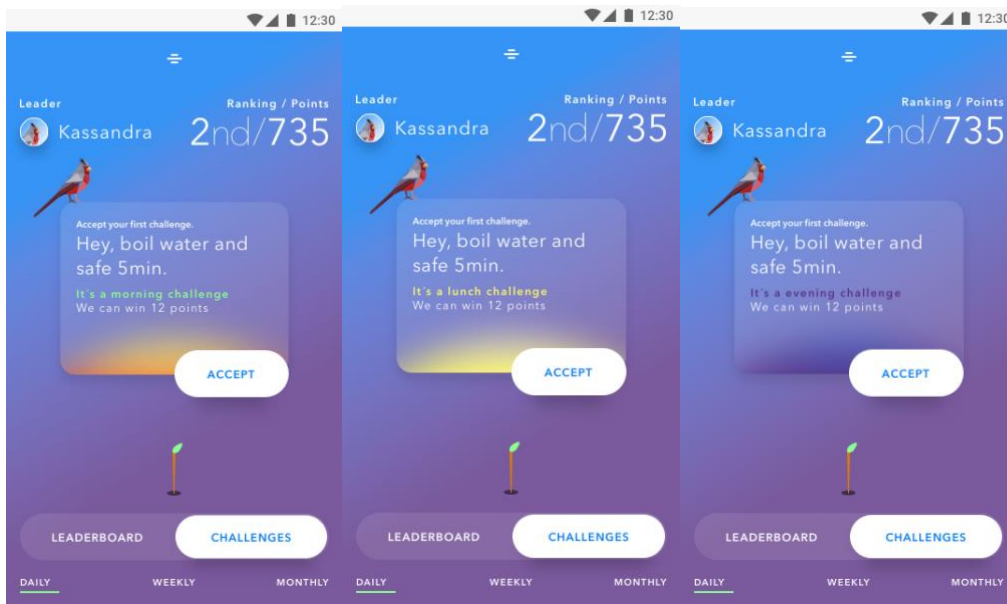


FIGURE 29: AN OVERVIEW OF THE CHARGED MOBILE APP MORNING, LUNCH AND EVENING CHALLENGES

Challenge details can be viewed to see the conditions to be met to win the challenge.

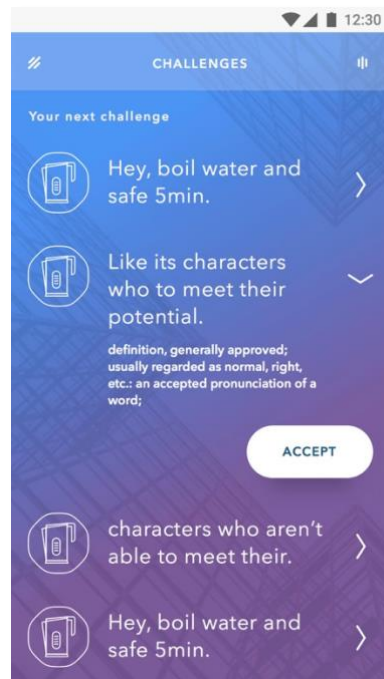


FIGURE 30: THE CHARGED MOBILE APP CHALLENGE SCREENS

From the details, challenges can be accepted and are then displayed in the active challenge overview. Accepting a challenge triggers events in the Mobile-App-Gateway to monitor the user’s specific energy disaggregation until a challenge is won or lost. The amount of challenges will be limited to 5 in the first version of the app.

USER REGISTRATION

Enhancing the current version of the mobile app, users will be able to register using the App by entering their Username, password, name and, if necessary, other information into the registration form to create their own user account.

NFC INTEGRATION

NFC will be used for appliance usage tracking by players but also for administrative purposes:

- **Player usage:** In order to track specific usage of appliances which cannot be solely assigned to specific players (such as desk lamps or the personal computer of the player), NFC will be used to check in at an appliance before usage. By doing this, the energy disaggregation can be easily assigned to a specific user
- **Admin Use:** In order to add new appliances to the SiteWhere system and make these usable within the ChArGED system, the app can be used to scan and add these.

Building up on functionality of the initial prototypes, NFC support will be added for more brands as we progress, in case there are bugs/issues with some of them, to make scanning smoother on a wider area of phones, i.e. Samsung, and make the gaming experience smoother.

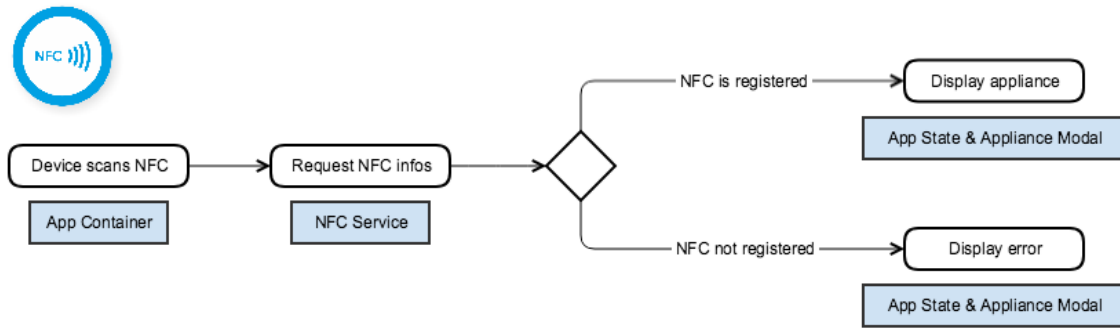


FIGURE 31: NFC INTEGRATION WITHIN THE MOBILE APP

BLE INTEGRATION

Bluetooth Low Energy (Bluetooth LE, BLE) is a wireless personal area network technology which allows to use various applications within this standard to be used in Mobile Apps. Within the project we use “Proximity sensing” for location tracking of the users within the project sites in rooms and floors.

Specifically this means we make use of PXP — the proximity profile — which allows a proximity monitor to detect whether a proximity reporter is within a close range. Physical proximity can be estimated using the radio receiver’s RSSI value, although this does not have absolute calibration of distances.

BLE is integrated in order to track presence of users within a specific part of a building in order to share energy consumption from appliances which are shared between different users (ceiling lights, a beamer during a presentation, etc.).

BLE integration is currently supported and will constantly enhanced during trials once we better understand the location specifics like rooms, other beacon interferences, wall thickness etc., to better track the users current position within a room or floor, in order to better understand how to incorporate that data into running challenges.

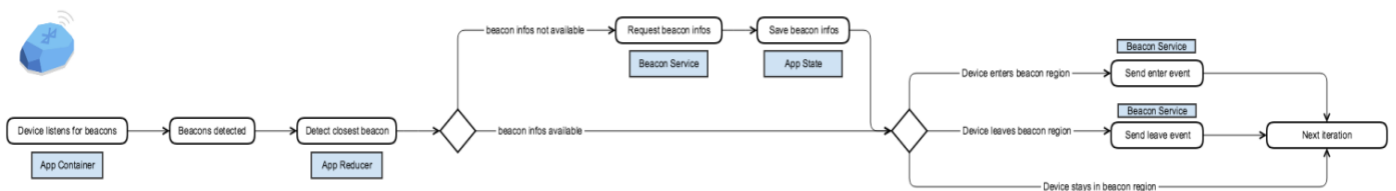


FIGURE 32: BLE INTEGRATION WITHIN THE MOBILE APP

REQUEST - RESPONSE FLOW AND GAME BACKEND INTEGRATION

In order to implement the gamification approach, the mobile app must communicate with different systems. For encapsulating the communication, the Communication Backend is the sole communication partner for the mobile app. The communication backend will in turn communicate with the rest of the charged backend infrastructure. Within this range, other systems may be requested. This can be done synchronously or asynchronously.

Error! Reference source not found. describes the general communication procedure. In this case, the handling of a NFC swipe is visualized.

1. The mobile device detects an NFC swipe
2. It sends an alert to the Communication Backend
3. The alert is then send to SiteWhere where it is stored and forwarded to the Energy Analytics and Game backends .
4. The alert is processed by the backends. Past data are retrieved as needed during this process from SiteWhere.
5. The energy savings of the user as well as the total game progress (scores,challenge updates) are generated and forwarded back to SiteWhere where they are stored.
6. A message is send to the mqtt broker to inform the mobile app that there is an new update in the server.
7. The mobile app gets the new data from SiteWhere and updates the user interface.

6.2 GAME BACKEND / GAME MECHANICS

6.2.1 DESCRIPTION AND OPERATION

The Game Backend implements the game rules logic that is going to be used in order to decide the progress of a user in the game, updates the user scores and leader board, and keeps track of the currently accepted/available challenges of a user.

The Game Backend is implemented in Java and interfaces with SiteWhere via MQTT and REST. All the communication between the software components happens through SiteWhere. Whenever an event is sent to SiteWhere from a device (i.e. a measurement or an alert), the event is stored and forwarded to a predefined MQTT topic which is listened by the other software. The Game Backend listens to events sent to SiteWhere that describe the users' behavior and its results (such as NFC swipes, user location updates and energy updates), processes the data and determines the user progress with respect to the challenges that have been accepted or schedules delayed or recurrent processing. The processing performed by the game backend is not necessarily tied with the specific time an alert has arrived. Separate logic can also be triggered or executed at a different time to check/update the game progress and provide updates to the other system components.

When a challenge is completed the scores and the game progress for each user are updated and the updates are sent to SiteWhere and to the mobile app. The mobile app frontend is notified when updates become available via MQTT which in turn retrieves the data and displays them to the user. Initially the possibility of using WSO2 Complex Event processor (CEP) has been investigated but it was replaced with a low level game engine implemented in Java. This was done for the following reasons

1. WSO2 CEP does not officially support HTTP requests from within a rule to Sitewhere's API nor officially supports communication with InfluxDB

This provides us with the problem that we cannot query SiteWhere and get information about the system because all the information about our system i.e. devices, locations, NFC associations, user scores are stored in SiteWhere. Those information are important as they are used as input in the different rules that are implemented in the Game Backend.

2. WSO2 CEP cannot implement functionality not tied to a specific alert

In the game backend we need to periodically check the process of all users regarding the challenges they have accepted and determine if a challenge should be terminated or the progress updated. For example at the end of each week we will need to check the progress that is made for each user and determine if the challenge has been completed. The WSO2 CEP is primarily event driven so this functionality cannot be implemented as easily. Java provides more straightforward implementation options that reduce complexity and improve the efficiency of the whole system.

3. Development and Debugging tools

Java is a widely used programming language that provides advanced tools for debugging and testing. Similar functionality is not provided from WSO2 CEP or is not as advanced something that increases development time and decreases software quality the more complex the software gets.

6.2.2 IMPLEMENTATION

6.2.2.1 ARCHITECTURE

The game backend gets notifications for the events that are arriving in SiteWhere (measurements from sensors and alerts from the mobile app and energy analytics backend) via MQTT. It also uses SiteWhere's Rest API to request specific information or make updates. The way it communicates with the different software components is described in Figure 33.

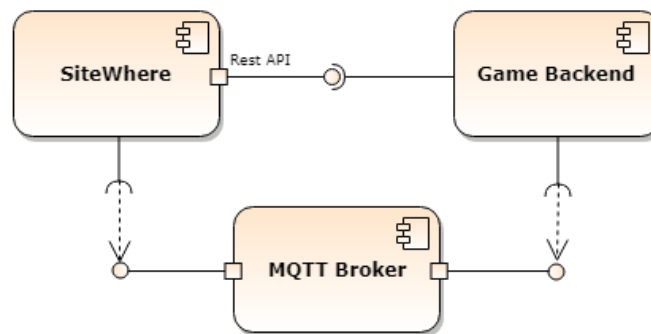


FIGURE 33: THE GAME BACKEND INTERFACES

Internally the Game Backend consists of three different subcomponents. The first one is responsible for interfacing the backend with the rest of the system through MQTT and REST. This interface is used to receive/request and update data according to the results of the rules. The second sub component is the core engine that implements the game logic. It consists of a preprocessing component which handles the receiving and identification of the incoming messages, a rule selection component which deduces which rule should be used depending on the input and the rules which describe a unique challenge in our system or a unique action that should be performed (for example at the end of the week check which challenges have not been completed and release them if they have been assigned to a user). The third subcomponent is the scheduler. Its main use it to schedule delayed rules that should be executed in the future once or executed at specific time intervals (i.e. every day, every week etc).

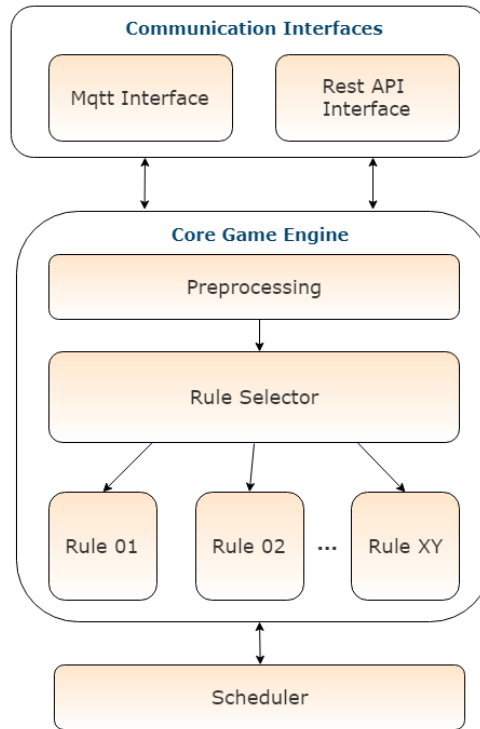


FIGURE 34: THE GAME BACKEND INTERNAL ARCHITECTURE

New rules can be added as needed to incorporate new challenges. Each rule is contained in one java file and can be added separately from the rest. The architecture has been designed with the scalability and modularity aspects in mind, i.e. we want to be able to add new rules independently from the rest of the system as needed in case the game changes or is enhanced. For this reason the rules are organized in a specific structure by inheriting from an abstract class Rule which defines a common interface as well as implements common functionality.

2. The Alert is forwarded to the game backend and the event is matched to the device - NFC pair that it corresponds to inside SiteWhere.

Inside SiteWhere we have a Device assigned to an NFC where all the measurement of this device are sent and an NFC assigned to a device that inside its metadata it contains the device_id of the device that it corresponds to. The nfc_id provided by the app is used to get the device_id from the metadata of the NFC. If we are close to the end of the working hours and the challenge has not been yet completed today a new alert named user_action_update is generated that is listened by the energy analytics backend. The swipe is also matched to the specific challenge and a unique challenge identifier is added in the alert which is labelled user_left_for_the_day.

3. The energy analytics backend retrieves the measurements, processes the data and produces a new power_updatealert that is sent back to SiteWhere, which contains useful information about how the user's action affected the power consumption. This alert is gathered by the game backend.

4. Using the information contained in the alert the game backend then matches the alert to the specific challenge. The power update (sent by the energy analytics) is checked against a threshold to determine if

the user closed the pc. We are looking for a drop in the consumption and a change greater than a threshold (e.g. 15W).. If the previous requirements are met the challenge is considered completed and points are assigned to the user's score in his/hers smartphone device in SiteWhere. The device's metadata are also updated with other useful information such as information on the last completed challenge, points awarded for the last completed challenge, points gained the previous week and more.

5. The Game Backend finally sends an alert to SiteWhere that the challenge has been completed and a notification on a separate MQTT topic listened by the mobile app in order to communicate that there is an update available for the specific user. Then the mobile app parses these data from SiteWhere and displays them in the front end.

8 CONCLUSIONS

This report describes the final version of the ChArGED platform architecture and its software implementation. It is the result of a cooperation work between all project partners, driven by requirements gathered at the three pilot sites.

The initial system architecture has been enriched after a number of challenges appeared during the implementation and evaluation phase, mostly to do with communication between components and location of functionality within the software system.

All functional blocks are described with information on both functionality and implementation, for exploitation by the project's stakeholders.