D6.9 - Final Interfaces for Professional and Residential Users

Deliverable ID	**D6.9**
Deliverable Title	**Final Interfaces for Professional and Residential Users**
Work Package	**WP6**
Dissemination Level	**PUBLIC**
Version	**1.0**
Date	**16/09/2019**
Status	**final**
Type	**Prototype**
Lead Editor	**LINKS Foundation**
Main Contributors	**LINKS Foundation (Teodoro Montanaro, Orlando Tovar, Michele Ligios), FIT (Vinoth Pandian, Veronika Krauß)**

Published by the Storage4Grid Consortium

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731155.

Document History

Version	Date	Author(s)	Description
0.1	2019-06-26	Teodoro Montanaro, Orlando Tovar, Michele Ligios (LINKS)	Initial TOC
0.2	2019-06-14	Teodoro Montanaro (LINKS)	First draft of the document
0.3	2019-06-28	Teodoro Montanaro (LINKS)	First draft of Section 2
0.4	2019-08-7	Teodoro Montanaro (LINKS)	Update for Section 2
0.5	2019-08-15	Vinoth Pandian, Veronika Krauß (FIT)	Provided content for Section 3 and Section 1.4
0.6	2019-08-26	Teodoro Montanaro, Orlando Tovar, Michele Ligios (LINKS)	Update Section 2 about the Residential Interfaces
0.7	2019-08-29	Orlando Tovar, Michele Ligios (LINKS)	Format and shape control
0.8	2019-08-29	Teodoro Montanaro (LINKS)	Document ready for partners' review
0.9	2019-09-05	Veronika Krauß (FIT), Teodoro Montanaro, Orlando Tovar, Michele Ligios (LINKS)	Updated Sections according to reviewer's comments
1.0	2019-09-16	Teodoro Montanaro (LINKS), Orlando Tovar, Michele Ligios (LINKS)	Version ready for submission

Internal Review History

Review Date	Reviewer	Summary of Comments
2019-09-02 (v0.8)	Vasco Delgado-Gomes (UNINOVA)	Reviewed: • Minor corrections. • Comments and suggestions. • Some clarifications are needed.
2019-09-05 (v0.8)	LIBAL	Approved: Comments and minor corrections
2019-09-14 (v1.0)	Vasco Delgado-Gomes (UNINOVA)	Approved: • Minor corrections and suggestions.

Table of Contents

Executive Summary

This document summarizes the work of Task T6.3 – "Interaction with Residential and Professional users" and describes the developed Graphical User Interfaces (GUIs) software prototypes for professional and residential users. The prototypes are developed by following an iterative approach; the previous versions are D6.7 "Initial Interfaces for Professional and Residential Users" [S4G-D6.7] and D6.8 "Updated Interfaces for Professional and Residential Users" [S4G-D6.8].

The GUIs are developed following the requirements collected during end-user workshops to ensure their relevance as well as their applicability in the corresponding domains and by the High-level use-cases (HLUC) documented in D2.2 "Final Storage Scenarios and Use Cases" [S4G-D2.2].

The Residential GUI is developed as a web application which is mainly displayed on mobile devices. It empowers private house owners to monitor their production, consumption, State of Charge (SOC) of their storage system and, in case they own one, additional information about their Electric Vehicle (EV). Besides, it extends commercial products such as Fronius system interfaces providing additional status information of the household including EV charging units. It also allows prosumers to select the operation mode of their local devices in order to maximize self-consumption and minimizing energy costs. Further information about the Residential GUI is provided in Section 2.

The Professional GUI is intended to be used in a desktop PC. The Professional GUI is detailed in Section 3 and addresses the needs of Distributed System Operator (DSO) grid planners who are investigating in future storage solutions to secure the correct voltage levels on low voltage grids. Those grid planners can simulate the impact of storage, virtually added in the current real grid topology, by running a simulation using the Professional GUI and giving the economic impact of that solution as results. Additionally, the system can propose the optimal positioning of storage for the best possible outcome with respect to voltage impact.

1 Introduction

D6.9 describes the "Final Interfaces for Professional and Residential Users" prototype, developed by the S4G project. Similar to other prototypes, this document provides a minimal technical documentation necessary to understand the functionalities, structure and deployment instructions for the residential and the Professional GUI prototype. More detailed information can be retrieved from related documents summarized in Section 1.2.

The document is organized as follows: Section 2 describes the Residential interfaces while the Section 3 describes the Professional interfaces. Finally, Section 4 provides some conclusions of the developed components.

1.1 Scope

This prototype deliverable has been developed by Task 6.3 – "Interaction with Residential and Professional Users" and updates D6.8 "Updated Interfaces for Professional and Residential Users" [S4G-D6.8].
The implementation is based on the end-user requirements collected by WP2 – "Business Models and Requirements Engineering".

1.2 Related documents

ID	Title	Reference	Version	Date
D2.2	Final Storage Scenarios and Use Cases	[S4G-D2.2]	V1.0	31/07/2018
D3.3	Final S4G Components, Interfaces and Architecture Specification	[S4G-D3.3]	V1.0[1]	31/08/2019
D5.2	Final DSF Hybrid Simulation Engine	[S4G-D5.2]	V1.0[1]	31/08/2019
D6.7	Initial Interfaces for Professional and Residential Users	[S4G-D6.7]	V1.0	31/08/2017
D6.8	Updated Interfaces for Professional and Residential Users	[S4G-D6.8]	V1.0	24/08/2018
D8.1	POPD Requirement No.1	[S4G-D8.1]	V1.0	04/04/2017

1.3 Reference architecture

With the aim of facilitating the interpretation of the information described within the present document, Figure 1 reports the S4G functional architecture that is presented in [S4G-D3.3]. It presents an overview of the S4G infrastructure and highlights the layer domains at which each component operates.

[1] At the time being the deliverable has not been submitted yet. The document version as well as its date may therefore possibly change.

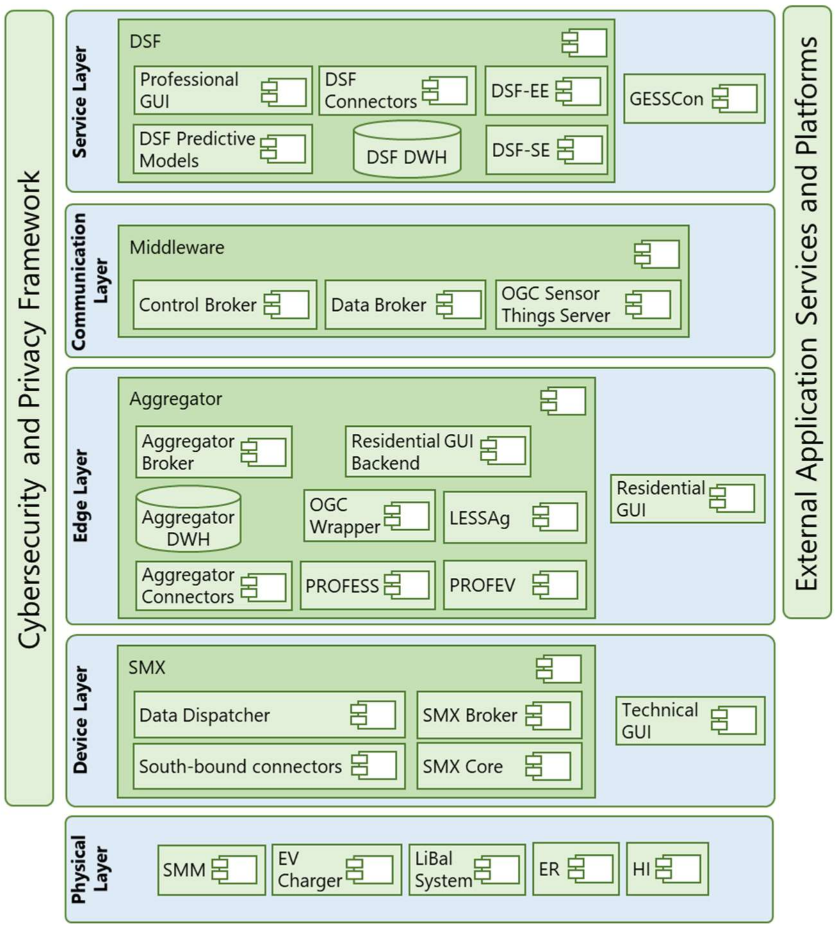


Figure 1 – S4G Functional architecture View [S4G-D3.3]

The Residential GUI is deployed at the *Edge Layer* and interacts with the *Aggregator* to show measurements from the Unbundled Smart Meter (USM) (composed by the Smart Meter eXtension (SMX) and the Smart Metrology Meter (SMM)) devices at test site level (prosumer premises) as well as to set operational modes.
The Professional GUI is deployed at the *Service Layer* and it mainly interacts with Decision Support Framework Simulation Engine (DSF-SE) and Decision Support Framework Economic Engine (DSF-EE) (reported in [S4G-D5.2]) to enable Professional Users (e.g. DSO, Aggregators) storage analysis and planning functionalities.

1.4 Use Cases Overview

Table 1 provides an overview of the actual usage of both the Residential and the Professional Interfaces in existing S4G use cases. Further details are described in [S4G-D2.2].

Table 1 - Use Cases and their relation to the Professional and the Residential GUI

GUI	Use Case ID	Use Case Title
Professional GUI	HLUC-2-PUC-3	Simulation of high penetration of EV chargers and of prosumers with storage and residential EV charging
	HLUC-3-PUC-1	Support for analysing storage dimensioning and positioning in the low-voltage grid
	HLUC-3-PUC-4	Coordinated Distributed storage in the grid
Residential GUI	HLUC-2-PUC-1	Residential Prosumer with Storage and EV

2 Residential Interfaces

2.1 Overview

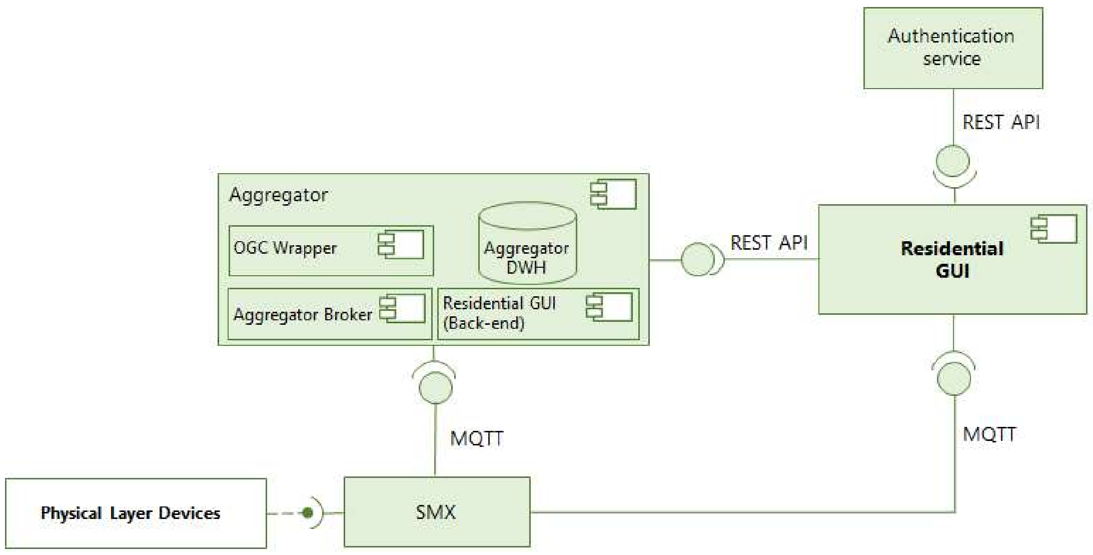


Figure 2 – The prototypal structure of the system that enables the Residential GUI

The Residential Graphical User Interface (GUI) enables users who are hosting a residential PV, ESS and/or a controllable EV charging system to monitor the behaviour of their smart energy management system through a local (i.e., installed and accessible only within their private Local Area Network) dedicated graphical interface.

Figure 2 summarizes the high-level structure of the prototype of the system that provides the Residential GUI to end-users. As depicted in Figure 2, various types of sensors continuously send updated data to the Smart Meter eXtension (SMX) framework that exposes such data through some dedicated Message Queue Telemetry Transport (MQTT) topics.

The Aggregator software components and the Residential GUI are then registered to dedicated MQTT topics to receive real-time data with the aim of providing various services (e.g., data extraction from local and remote database, data fusion algorithms). One goal of the Aggregator is to collect sensors data for later uses. It exploits a time-series database accessible only from the prosumer Local Area Network (LAN). The historical data collected are consequently provided locally for Residential GUI purposes.

The main sub-components of the prototype are shortly summarized in the following subsections.

2.1.1 OGC Wrapper

As already discussed, various types of sensors continuously send updated data exploiting the SMX that exposes such data through some dedicated MQTT topics.
Therefore, the SMX "MQTT topic" are intercepted by the Open Geospatial Consortium (OGC) Wrapper tool that catch, verify, elaborate and then forward the data towards multiple destinations. It feeds the local database instance, the DSF-DWH and the Residential GUI.
The fixed MQTT topics exploited in real-time by the GUI are: "RESIDENTIAL/GUI" and "RESIDENTIAL/AGGREGATED". The first one is used to receive the data produced by each sensor, while the second one is used to receive aggregated data, such as the data coming from all the available sensors in each specific moment. In order to accomplish this task, the current tool needs to align the received messages

exploiting the embedded timestamp and, under proper conditions, to build and forward a single updated message containing the overall measurements produced by all the locally placed sensors at that time.
No special requirements are specified by the MQTT protocol for the data format; thus, the following text format is used.

Topic "RESIDENTIAL/GUI"

```
<SENSOR_ID> KEY1=<value1>, KEY2=<value2>, ..., KEYn=<valuen>
datatype=<dataType> <timestamp>
```

Figure 3 – Data format used on the "RESIDENTIAL/GUI" MQTT topic

Topic "RESIDENTIAL/AGGREGATED"

```
<SENSOR_ID> <groupOfData1> <groupOfData2> <groupOfData3>
<groupOfData4> <timestamp>
```

Figure 4 – Data format used on the "RESIDENTIAL/AGGREGATED" MQTT topic

```
KEY1=<value1>, KEY2=<value2>, ..., KEYn=<valuen>, type=<dataType>
```

Figure 5 – Content of <groupOfData> on the "RESIDENTIAL/AGGREGATED" MQTT topic

Possible values for "SENSOR_ID" (with respect to the Bolzano infrastructure) are:

- S4G-GW-EDYNA-0014 -> 'EV'
- S4G-GW-EDYNA-0015 -> 'PV'
- S4G-GW-EDYNA-0016 -> 'ESS'
- S4G-GW-EDYNA-0017 -> 'PCC'
- AGGREGATED (only available in Topic "RESIDENTIAL/AGGREGATED")

Possible values for "dataType" are:

- 0 -> 'PCC' (Point of Common Coupling)
- 1 -> 'PV' (Photovoltaic)
- 2 -> 'EV' (Electric Vehicle)
- 3 -> 'ESS' (Energy Storage System)
- 7 -> 'Fronius'

The following list summarizes the most important values for the KEY field used by the GUI:

- Processed_P: Power measured by the SMX
- P_Grid: Power being taken from the grid (negative means that power is being fed in the grid) (available only if dataType = 7)
- P_Load: Home load (negative means that house is consuming energy) (available only if dataType = 7)
- P_Akku: Power being injected in home by battery (negative means charging, positive means discharging) (available only if dataType = 7)
- P_PV: Power being generated by the PV (always positive) (available only if dataType = 7)
- SOC: percentage of power available in the battery (available only if dataType = 7)

2.1.2 Residential GUI backend

When a new OGC formatted data (e.g., the current power consumption) is received by the “Aggregator Broker”, the Aggregator parses it through the “OGC Wrapper” and stores the contained information into the “Aggregator Data Warehouse (DWH)”.
Consequently, the received information is exposed for later uses though dedicated Representational State Transfer (REST) Application Programming Interface (API).
An overview of the API exposed by the Residential GUI backend prototype built on the Aggregator is depicted in Table 2.

Table 2 – API exposed by the aggregator

Resource Path	Available methods
LOCALENERGY/{fromDate}/{toDate}/{SMXmeasurement}	**GET:** Energy evaluated from Power measurements: interact with Local DWH. Possible values for "SMXmeasurement": • "S4G-GW-EDYNA-0014" • "S4G-GW-EDYNA-0015" • "S4G-GW-EDYNA-0016" • "S4G-GW-EDYNA-0017" **e.g.** LOCALENERGY/2018-12-24/2018-12-25/S4G-GW-EDYNA-0015
ENERGY/{fromDate}/{toDate}/{SMXmeasurement}	**GET:** Energy evaluated from Power measurements: interact with Global DWH. Possible values for "SMXmeasurement": • "S4G-GW-EDYNA-0014" • "S4G-GW-EDYNA-0015" • "S4G-GW-EDYNA-0016" • "S4G-GW-EDYNA-0017" **e.g.** ENERGY/2018-12-24/2018-12-25/S4G-GW-EDYNA-0015
ENERGY/{fromDate}/{toDate}/{FroniusMeasurement}/{FieldOfInterest}	**GET:** Manage the Fronius-Entries in InfluxDB Possible values for "FroniusMeasurement": • "InstallationHouse20" • "InstallationHouse24" • "InstallationHouse25" • "InstallationHouse26" • "InstallationHouse27" • "InstallationHouseBolzano" Possible values for "FieldOfInterest": • "photovoltaic" • "load" • "battery" • "grid"

Resource Path	Available methods
	Result is a number! **e.g.,** ENERGY/2018-12-24/2018-12-25/InstallationHouse20/photovoltaic
ENERGY/{fromDate}/{toDate}/{measurement}/{Field}/{FILTER}/{OPERATION}/{VALUE}	**GET:** Filtered **Energy** Estimations (exploiting signed values and operations such as: GROUPBY) Possible values for "**measurement**": • "S4G-GW-EDYNA-0014" • "S4G-GW-EDYNA-0015" • "S4G-GW-EDYNA-0016" • "S4G-GW-EDYNA-0017" • "InstallationHouse20" • "InstallationHouse24" • "InstallationHouse25" • "InstallationHouse26" • "InstallationHouse27" • "InstallationHouseBolzano Possible values for "**Field**": • If "measurement" starts with "S4G-GW" (the list only reports the most important values): ○ P ○ PROCESSED_P • If "measurement" starts with "Installation": ○ "photovoltaic" ○ "load" ○ "battery" ○ "grid" Possible values for "**FILTER**": • "POSITIVE" • "NEGATIVE" • "ALL" Possible values for "**OPERATION**": • "GROUPBY" Possible values for "**VALUE**": • If "OPERATION" is equal to GROUPBY ○ Delay in minutes DEFAULT (FIXED) = GROUP BY 30 minutes **Result is a number!** e.g., ENERGY/2018-12-24/2018-12-25/S4G-GW-EDYNA-0015/P/POSITIVE/GROUPBY/30
INFLUXDB/{fromDate}/{toDate}/{measurement}/{	**GET:** Filtered **Data** (plus operations such as: GROUPBY)

Resource Path	Available methods
Field}/{FILTER}/{OPERATI ON}/{VALUE}	Possible values for "**measurement**": • "S4G-GW-EDYNA-0014" • "S4G-GW-EDYNA-0015" • "S4G-GW-EDYNA-0016" • "S4G-GW-EDYNA-0017" • "InstallationHouse20" • "InstallationHouse24" • "InstallationHouse25" • "InstallationHouse26" • "InstallationHouse27" • "InstallationHouseBolzano Possible values for "**Field**": • If "measurement" starts with "S4G-GW" (the list only reports the most important values): ○ P ○ PROCESSED_P • If "measurement" starts with "Installation": ○ "photovoltaic" ○ "load" -> Result is a **Json** ○ "battery" ○ "grid" ○ direct_consumption -> Result is a **Json** Possible values for "**FILTER**": • "POSITIVE" • "NEGATIVE" • "ALL" Possible values for "**OPERATION**": • "GROUPBY" Possible values for "**VALUE**": • If "OPERATION" is equal to GROUPBY ○ Delay in minutes DEFAULT (FIXED) = GROUP BY 30 minutes Result is an **array**, except for **load** and **consumption_direct** -> **JSON!** e.g., • Extract Battery Consumption (the result is negative): -> Consumption Battery = (-P_Akku) if(P_Akku<0): INFLUXDB/2019-03-25/2019-03-27/InstallationHouseBolzano/battery/NEGATIVE/GROUPBY/30 • Extract Over Production (the result is negative): -> Over Production = (-P_Grid) if(P_Grid<0): INFLUXDB/2019-03-25/2019-03-27/InstallationHouseBolzano/grid/NEGATIVE/GROUPBY/30 • Extract Power from Battery (the result is positive):

Resource Path	Available methods
	-> Power From Battery = (P_Akku) if(P_akku>0): INFLUXDB/2019-03-25/2019-03-27/InstallationHouseBolzano/battery/POSITIVE/GROUPBY/30 • Evaluate Direct house consumption (the result is negative): -> consumption_direct = (-P_Load) if(P_Grid < 0) INFLUXDB/2018-12-24/2018-12-25/InstallationHouseBolzano/direct_consumption/GROUPBY/30 • Evaluate house overall consumption (the result is negative): -> Consumption_house = (-P_Load) INFLUXDB/2019-03-25/2019-03-27/InstallationHouseBolzano/load/GROUPBY/30

LOCALINFLUXDB/{fromDate}/{toDate}/{measurement}
/{OPERATION}/{VALUE}
	GET:
	Interact with Local DWH -> Filtered **Data**
	
	Possible values for "**measurement**":
	• "S4G-GW-EDYNA-0014"
	• "S4G-GW-EDYNA-0015"
	• "S4G-GW-EDYNA-0016"
	• "S4G-GW-EDYNA-0017"
	• "InstallationHouse20"
	• "InstallationHouse24"
	• "InstallationHouse25"
	• "InstallationHouse26"
	• "InstallationHouse27"
	• "InstallationHouseBolzano
	Possible values for "**OPERATION**":
	• "GROUPBY"
	• empty
	Possible values for "**VALUE**":
	• If "OPERATION" is equal to GROUPBY
	  ○ Delay in minutes
	  DEFAULT (FIXED) = GROUP BY 30 minutes
	
	Correct Result is a **Json**

Battery/cycles	**GET:** Retrieve Number of battery cycles Correct Result is a number.
OPMODE	**GET:** Get Operational Mode Correct Result is a **number**. **PUT:** Set Operational Mode + Integer as payload: {0,1,2,3,4}

Resource Path	Available methods
	The meaning of the values is: • 0:"maximizeSelfConsumption" • 1:"maximizeSelfProduction" • 2:"minimizeCosts" • 3:"MaximizeBatteryHealth" • 4:"None" The above API will trigger the interaction with the PROFESS tool. The required steps to set a specific operational mode over PROFESS are the following: first, send a request towards it by requesting its current state. Then, it is mandatory to send a message to stop the current optimization mode. After the success of the above jobs it will finally send OPMODE SET message with the value provided by the user through the GUI and will trigger the status change on PROFESS.

2.1.3 Authentication service

The user authentication on the Residential GUI is guaranteed by the Firebase Auth service through its exposed API. An overview of the API exposed by the Firebase service is available at the following URI: https://firebase.google.com/docs/reference/rest/auth/

2.1.4 Residential GUI

The Residential GUI gets real-time data and historical data from the OGC Wrapper and from the Residential Backend respectively, by acting as dashboard for users. Specifically, it provides a responsive website (i.e., that can be used on different devices and screen sizes) to:

a) Log in into the system through personal credentials;
b) Create a new user;
c) Show real time information about the system:
   - The current health status of the system;
   - The today's consumption, by distinguishing the total consumption from the one taken from the grid and the one that is self-produced;
   - The current charging status of the household battery (i.e., charging, discharging, hold) and the ESS State of Charge (SOC);
   - The current charging status of the available car (i.e., plugged/unplugged/no status);
   - The current behavioural mode selected by the user (e.g., "maximize the battery life" mode);
d) Present an overview of the current status of the system;
e) Present a data analysis of consumption over various time frames in a dedicated view with the possibility to export data in JSON or CSV;
f) Present a data analysis of production over various time frames in a dedicated view with the possibility to export data in JSON or CSV;
g) Set and update user personal information (e.g., username and password).

2.2 Residential GUI design

Starting from the high-level requirements exposed in section 2.1.4, this section presents all the details of the designed Residential GUI.

2.2.1 Project assumption

The following list summarizes the assumptions that were considered in the design of the Residential GUI to both facilitate the development and, also, to avoid system feasibility issues:

- The Residential GUI will be accessed only in the LAN present in the household to ensure privacy and security related aspects;
- The LAN gateway provides internet connectivity with all the necessary security measures to avoid unwanted accesses from unauthorized users/systems/services.

2.2.2 Main Stakeholders

The following list presents all the main stakeholders of the Residential GUI:

- Generic user, it is a user who did not yet logged in the system;
- Logged user, it is a user who already entered the system through her personal credentials.

2.2.3 Functional requirements

- **Generic user**
  1. Log in to the Residential GUI through her/his credentials;
  2. Recover lost username and password through her/his email;
  3. Register to the Residential GUI.

- **Logged user**
  1. Have an overview of the current situation of the household and specifically the following information:
     - The current health status of the system;
     - The today's consumption, with a distinction among the total consumption, the consumption taken from the grid and the one that is self-produced;
     - The current charging status of the household battery (i.e., charging, discharging, hold) and the ESS SOC;
     - The current charging status of the available car (i.e., plugged/unplugged/no status);
     - The current behavioural mode selected by the user (e.g., “maximize the battery life” mode).
  2. Switch behavioural mode from the overview of the current situation of the household;
  3. Consult and export (csv/txt/json) a data analysis of consumption over various time frames in a dedicated view;
  4. Consult and export (csv/txt/json) a data analysis of production over various time frames in a dedicated view;
  5. Consult an overview of the current status of the system;
  6. Set and update user personal information (e.g., username and password).

2.2.4 GUI Navigation Map

Considering that the Navigation Map for the "Generic User" only contains the Login View, the Navigation Map for the "Logged User" is reported in Figure 6.

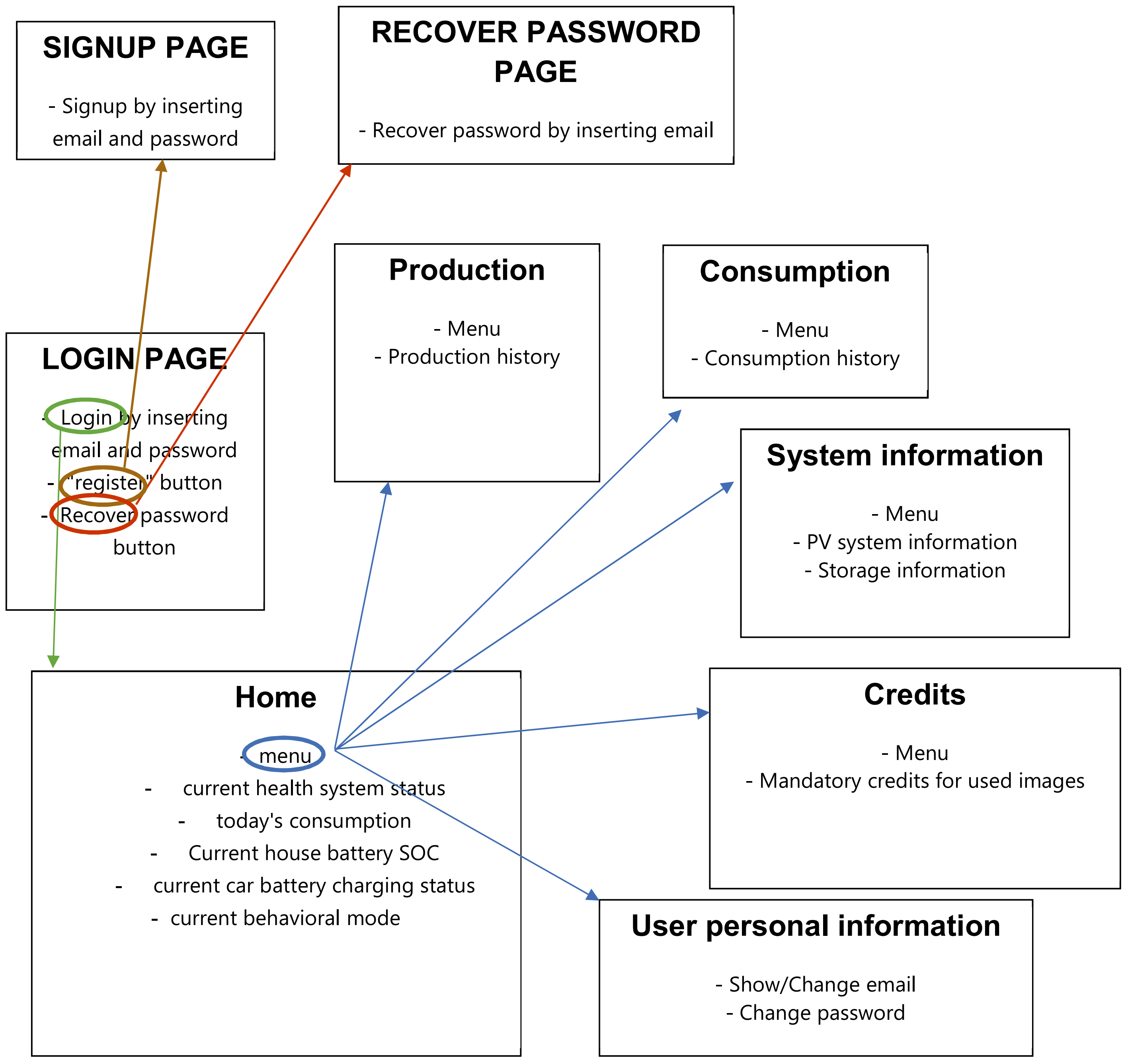

SIGNUP PAGE
- Signup by inserting email and password
RECOVER PASSWORD PAGE
- Recover password by inserting email
Production
- Menu
- Production history
Consumption
- Menu
- Consumption history
LOGIN PAGE
- Login by inserting email and password
- "register" button
- Recover password button
System information
- Menu
- PV system information
- Storage information
Home
- menu
- current health system status
- today's consumption
- Current house battery SOC
- current car battery charging status
- current behavioral mode
Credits
- Menu
- Mandatory credits for used images
User personal information
- Show/Change email
- Change password

Figure 6 – Navigation Map for "Logged User"

2.2.5 Pages specification

- **LOGIN view:** Figure 7 and Figure 8 report the desktop and mobile view of the login page.

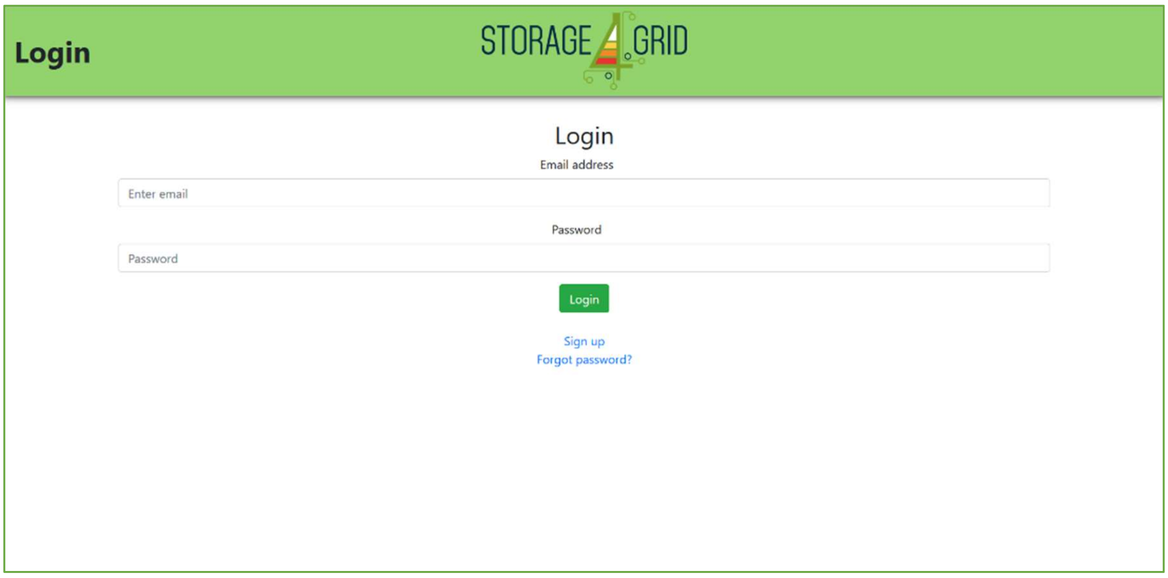


Figure 7 – Desktop Login view

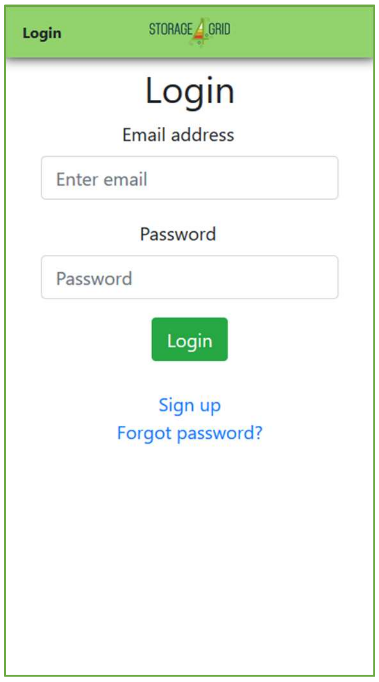
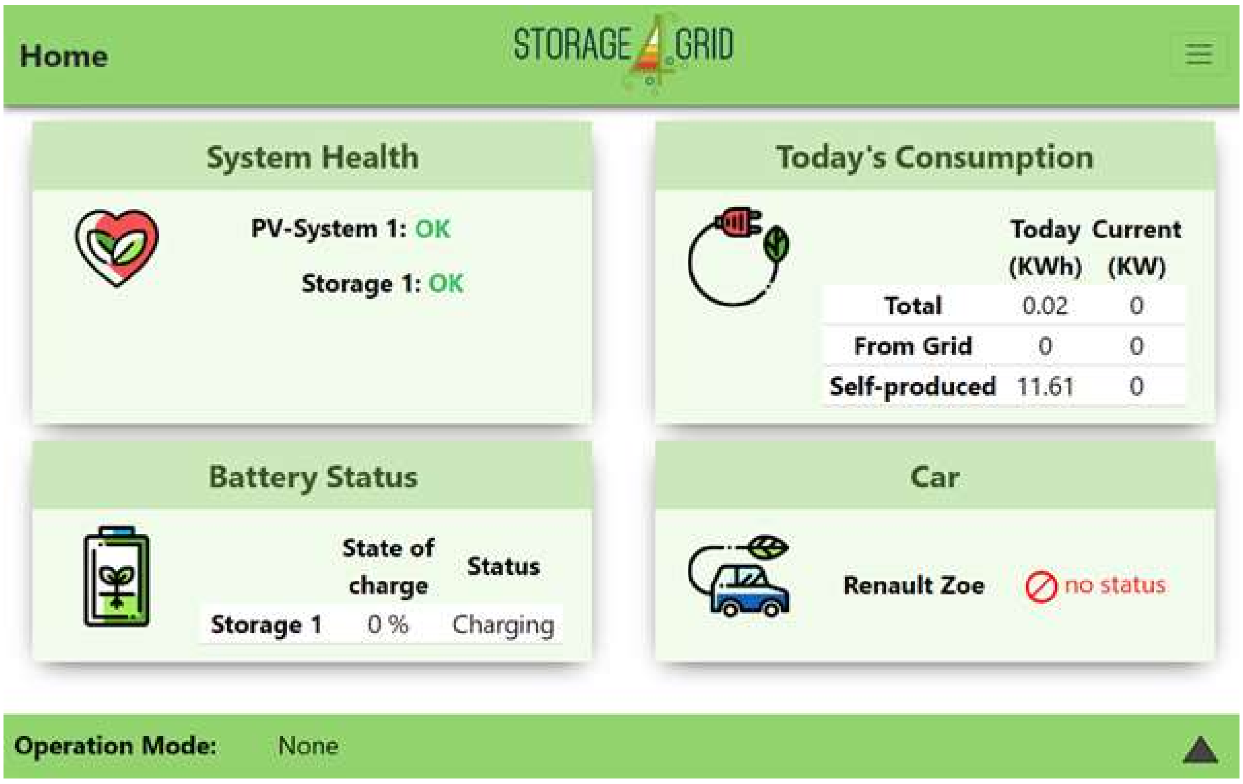


Figure 8 – Mobile Login view

- **HOME view:** Figure 9 and Figure 10 report the desktop and mobile view of the login page.



Home

System Health

PV-System 1: OK

Storage 1: OK

Today's Consumption

	Today (KWh)	Current (KW)
Total	0.02	0
From Grid	0	0
Self-produced	11.61	0

Battery Status

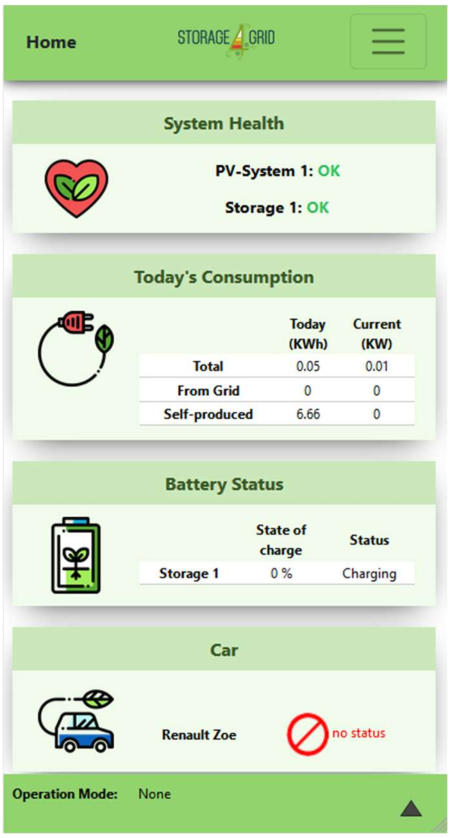
	State of charge	Status
Storage 1	0 %	Charging

Car

Renault Zoe no status

Operation Mode: None

Figure 9 – Desktop view home Page



Home

System Health

PV-System 1: OK

Storage 1: OK

Today's Consumption

	Today (KWh)	Current (KW)
Total	0.05	0.01
From Grid	0	0
Self-produced	6.66	0

Battery Status

	State of charge	Status
Storage 1	0 %	Charging

Car

Renault Zoe no status

Operation Mode: None

Figure 10 – Mobile view home Page

The home page contains the following information:

- The current health status of the system divided into "PV-System 1" and "Storage 1". Its status is calculated by using the following conditions:
  - **PV System 1:**

```
if (time passed since the last P_PV or P_ESS >20 sec)
{
   Current P_PV System1 Status= "Offline"
}
else
{
         if (time passed since the last P_PV >20 sec)
         {
            Current P_PV System1 Status = "Offline"
         }
         else
         {
            Current P_PV System1 Status = "OK"
         }
}
```

  - **Storage 1:**

```
if (time passed since the last P_PV or P_ESS>20 sec)
{
   Current Storage1 Status = "Offline"
}
else
{
         if (time passed since the last P_ESS >20 sec)
         {
            Current Storage1 Status = "Offline"
         }
         else
         {
            Current Storage1 Status = "OK"
         }
}
```

- The Today's consumption of the household with a distinction among the total consumption, the consumption taken from the grid and the one that is self-produced. It is calculated as presented in Table 3.

Table 3 – Today's consumption evaluation

	Today (Energy KWh)	Current (Power KW)

Total	if (Energy P_Akku<0)
	{
	  Total Consumption = Energy P_Load + Energy P_EV + Energy P_ESS;
	}
	else
	{
	  Total Consumption = Energy P_Load + Energy P_EV;
	}
		if (P_Akku<0)
		{
		  Consumption = P_Load + P_EV + P_ESS;
		}
		else
		{
		  Consumption = P_Load + P_EV;
		}

From Grid	if (Energy P_Grid>0)
	{
	  Consumption From Grid = Energy P_Grid;
	}
	else
	{
	  Consumption From Grid= 0;
	}
		if (P_Grid>0)
		{
		  Consumption From Grid = P_Grid;
		}
		else
		{
		  Consumption From Grid = 0;
		}

Self-produced	if (Energy P_Akku>0)
	{
	  Consumption Self = Energy P_PV + Energy P_ESS;
	}
	else
	{
	  Consumption Self = Energy P_PV;
	}
		if (P_Akku>0)
		{
		  Consumption Self = P_PV + P_ESS ;
		}
		else
		{
		  Consumption Self = P_PV;
		}

  - The current charging status of the household battery (i.e., charging/discharging) and the ESS SOC
  - The current charging status of the available car (i.e., Not charging/Charging)
  - The current behavioural mode selected by the user (e.g., "maximize the battery life" mode). The possible values are followed listed:
    - **Max Self Cons:** maximize Self Consumption
    - **Max Self Prod:** maximize Self Production
    - **Min Price:** minimize the price
    - **None**

- **Production view:** Figure 11 and Figure 12 contain the desktop and mobile view of the Production.

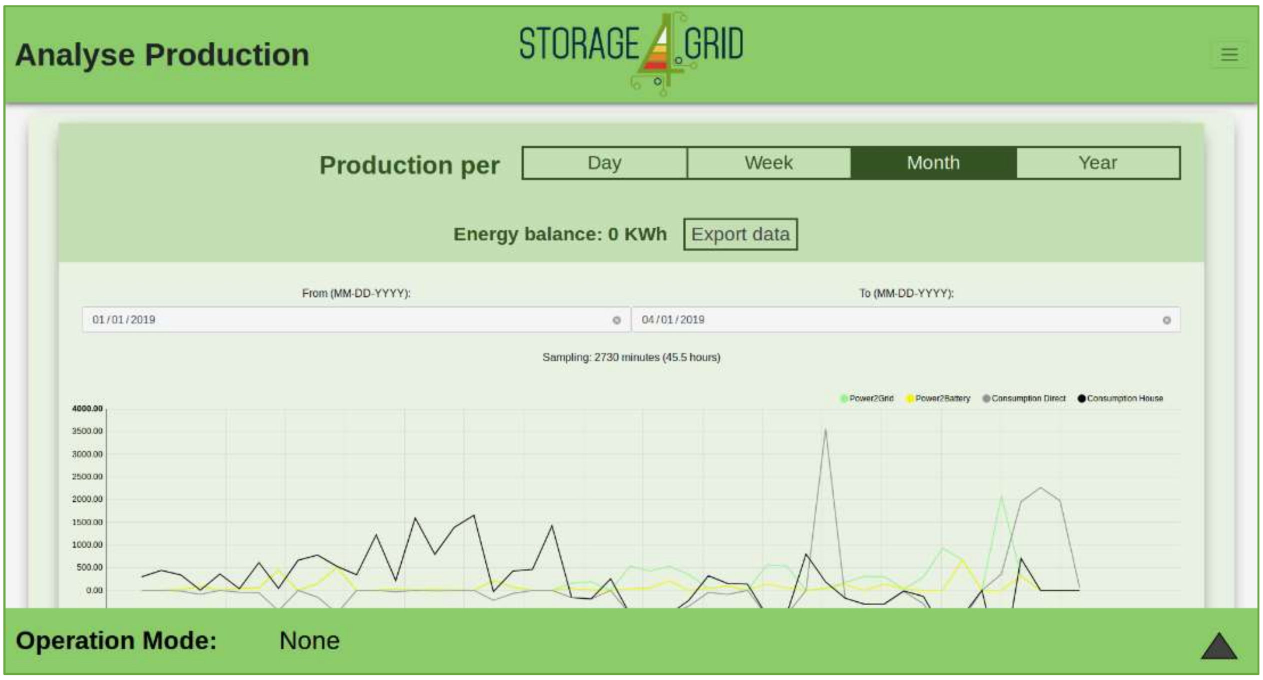


Figure 11 – Desktop production view

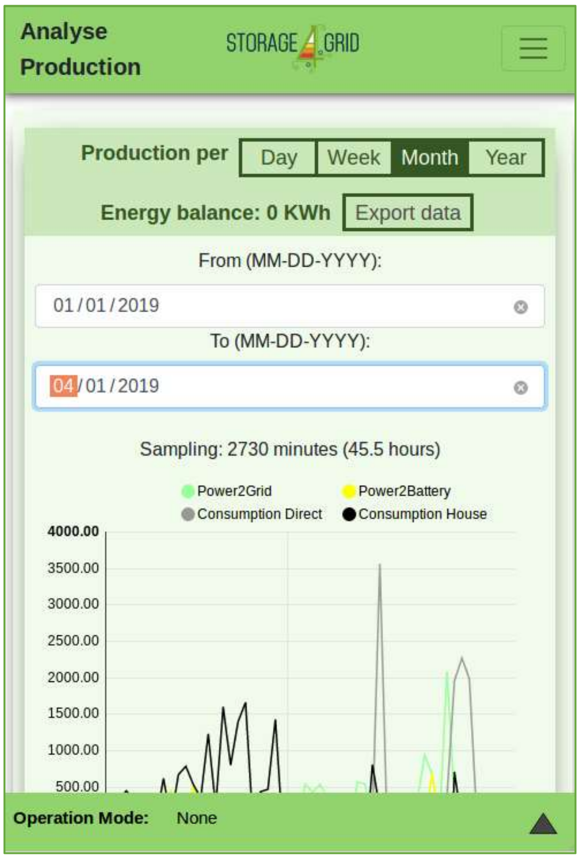


Figure 12 – Mobile production view

It allows to visualize the current energy Balance and the historical information related to:

- **Power 2 Grid**

```
Power2Grid = over_production + consumption_battery

Where:
over_production = if(P_Grid<0) then (-P_Grid) else 0
consumption_battery = if (P_Akku<0) then (-P_Akku) else 0
```

- **Power 2 Battery**

```
power2battery = if (P_Akku<0) then (-P_Akku) else 0
```

- **Consumption Direct**

```
consumption_directly = P_PV - consumption_battery - over_production

Where:
consumption_battery = if (P_Akku<0) then (-P_Akku) else 0
over_production = if(P_Grid<0) then (-P_Grid) else 0
```

- **Consumption House**

```
consumption_house = -P_Load
```

Data is shown depending on the dates that are selected on the top of the page. The granularity of shown data depends on the following formula:

```
diffAmongDates = endDate – startDate;
if (diffAmongDates <= 1 day)
{
        frequencyInMinutesForChart = 30*1;
}
else
{
        frequencyInMinutesForChart = 30*diffInDays;
}
```

Finally, the "Export Data" button allows to export all the available data, also the ones that are not shown in the chart, but that were used to calculate the shown information.

- **Consumption view:** Figure 13 and Figure 14 contain the desktop and mobile view of the Consumption.

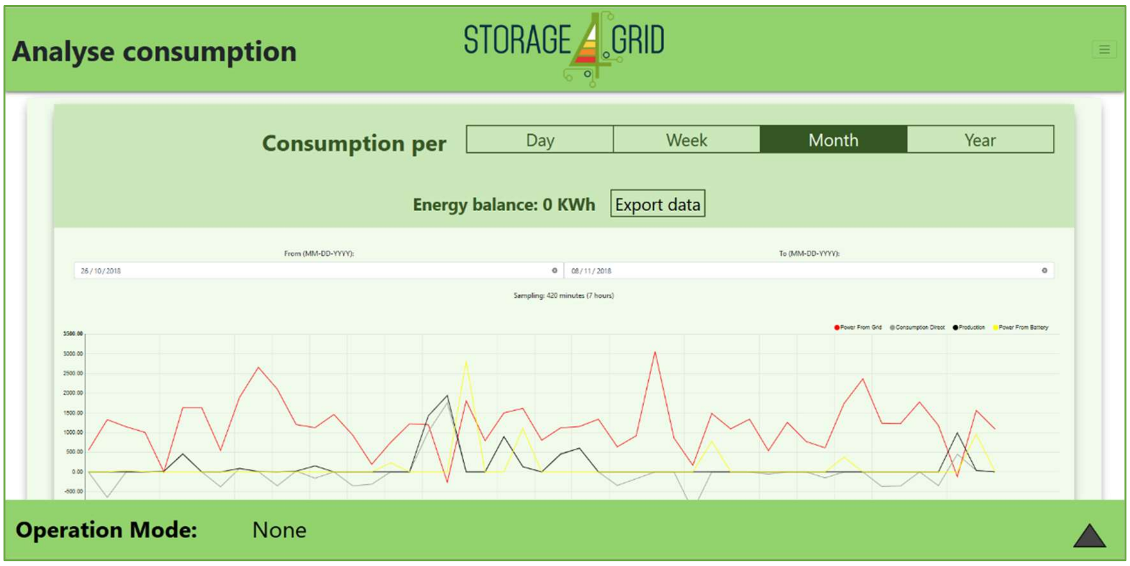


Figure 13 – Desktop consumption view

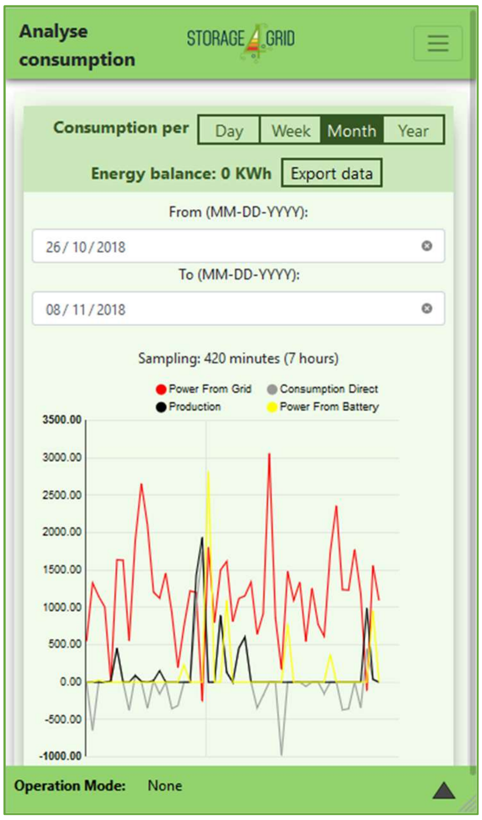


Figure 14 – Mobile consumption view

It allows to visualize the current energy Balance and the historical information related to:

- **Power From Grid**

```
power_from_grid = - P_Load
```

- **Power From Battery**

```
power_from_battery = if (P_Akku>0) then (P_Akku) else 0
```

- **Consumption Direct**

```
consumption_directly = P_PV - consumption_battery - over_production

Where:
consumption_battery = if (P_Akku<0) then (-P_Akku) else 0
over_production = if(P_Grid<0) then (-P_Grid) else 0
```

- **Production**

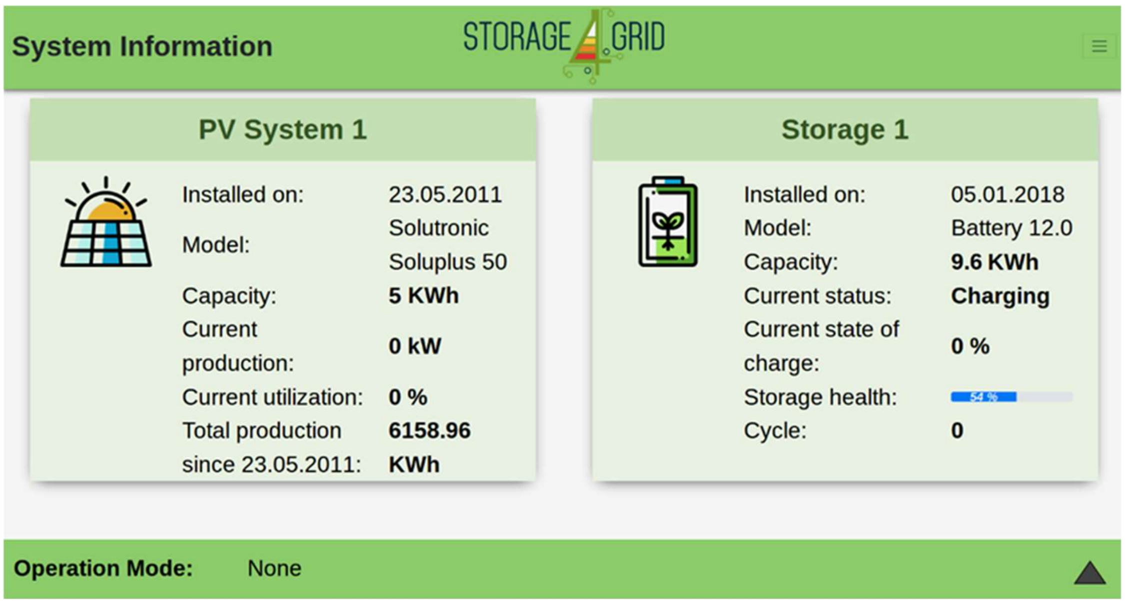
```
production = P_PV - over_production

Where:
over_production = if(P_Grid<0) then (-P_Grid) else 0
```

Data is shown depending on the dates that are selected on the top of the page. The granularity of shown data depends on the following formula:

```
diffAmongDates = endDate – startDate;
if (diffAmongDates <= 1 day)
{
        frequencyInMinutesForChart = 30*1;
}
else
{
        frequencyInMinutesForChart = 30*diffInDays;
}
```

Finally, the "Export Data" button allows to export all the available data, also the ones that are not shown in the chart, but that were used to calculate the shown information.

- **System information view:** shows the System information view that is divided into 2 main tabs:
    - **PV system information:** It contains the current status of the PV system, i.e., the date in which the PV system was installed, the model of the PV system, its capacity, its current production and utilization and the total production since the date of installation.
    - **Storage information:** It contains the current status of the Storage system, i.e., the date in which the ESS was installed, the model of the ESS, its capacity, its current status, its current SOC, the overall ESS health and the total cycles.



System Information

PV System 1

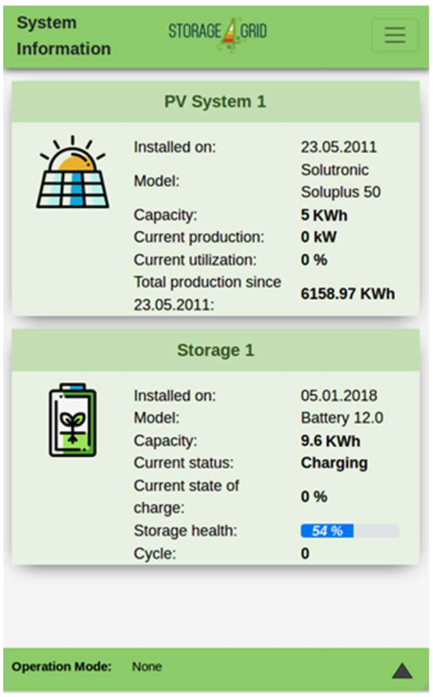
Installed on:	23.05.2011
Model:	Solutronic Soluplus 50
Capacity:	5 KWh
Current production:	0 kW
Current utilization:	0 %
Total production since 23.05.2011:	6158.96 KWh

Storage 1

Installed on:	05.01.2018
Model:	Battery 12.0
Capacity:	9.6 KWh
Current status:	Charging
Current state of charge:	0 %
Storage health:	54 %
Cycle:	0

Operation Mode: None

Figure 15 – Desktop System Information view



System Information

PV System 1

Installed on:	23.05.2011
Model:	Solutronic Soluplus 50
Capacity:	5 KWh
Current production:	0 kW
Current utilization:	0 %
Total production since 23.05.2011:	6158.97 KWh

Storage 1

Installed on:	05.01.2018
Model:	Battery 12.0
Capacity:	9.6 KWh
Current status:	Charging
Current state of charge:	0 %
Storage health:	54 %
Cycle:	0

Operation Mode: None

Figure 16 – Mobile System Information view

- **Account view:** The current page contains all the user personal information. Specifically, it contains (and allows to modify):
    - Email
    - password
- **Credits view:** The current page contains some information about the project, contact information in case of problems and all the references to satisfy Creative Common agreement.

2.3 Installation/Deployment instructions

The Residential GUI can be served on a simple http-server e.g. Apache[i]. It should be installed on the same LAN in which the aggregator is placed to receive the data through the MQTT topics.
The Residential GUI Backend has been built as a single python3 application installed as system autonomous service and instantiated during boot on the Aggregator. It relies mainly on the urllib and HyperText Transfer Protocol (HTTP) server python packages.

2.4 Software dependencies and requirements

No special hardware requirements are needed to use and/or deploy the first prototype of the Residential GUI. The prototype is based on the JavaScript framework AngularJS[ii] and uses the Firebase Auth[iii] service offered by Firebase[iv] (a cloud mobile[v] and web application[vi] development platform) for authentication.

Table 4 – Software dependencies of the Residential GUI

Dependency	License	Role
Apache httpd[vii], version 2.4.26	Apache version 2.0	Used to register and expose webpages for the Residential GUI
AngularJS[ii] V. 1.5.5	MIT License	Framework on which the Residential GUI is based
Firebase Auth[iii],	Google Cloud Platform License Agreement	Authentication service

3 Professional GUI

3.1 Overview

The overall, high-level structure of the Professional GUI is summarized in Figure 17.

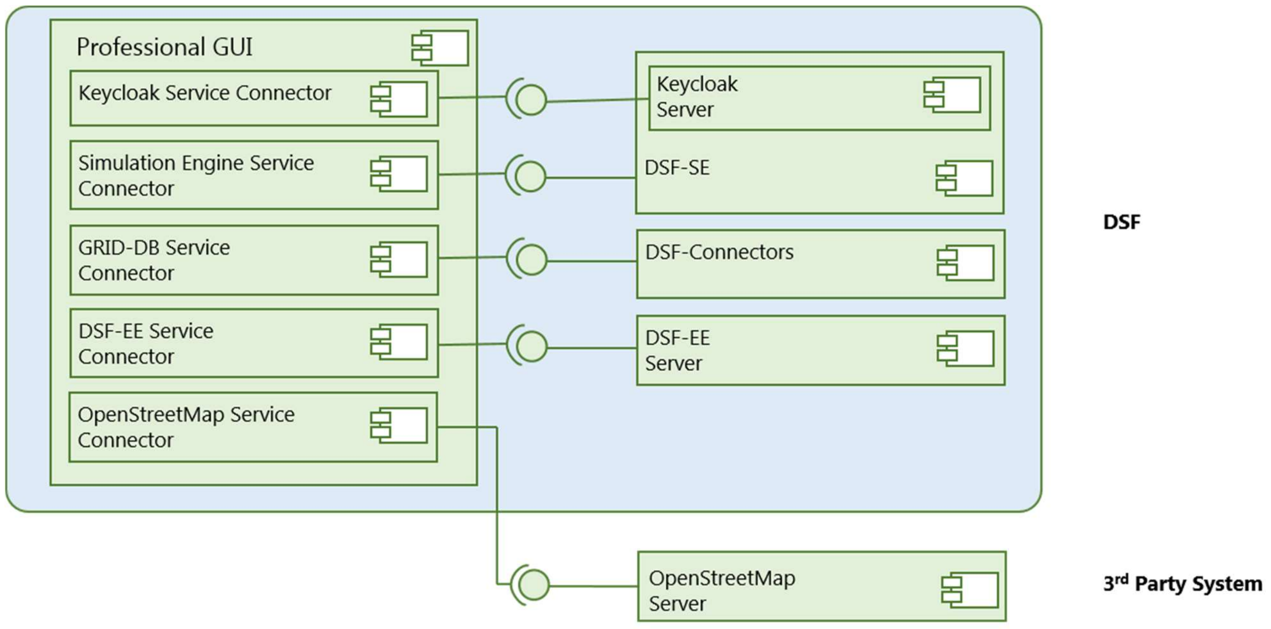
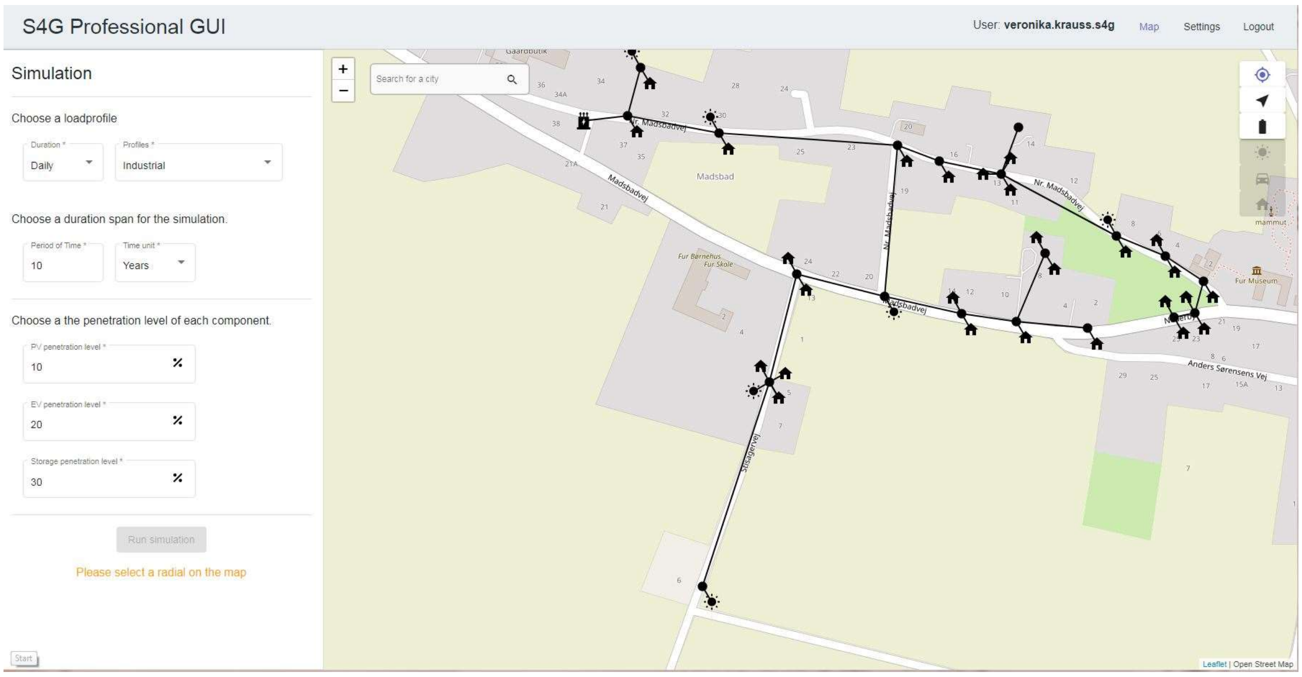


Figure 17 – The structure of the Professional GUI prototype

The Professional GUI enables grid planners to observe the impact of storage in the grid by simulating the voltage levels over a user-defined time frame in a radial selected by the grid planner. By interfacing with the DSF-EE, it also allows the professional user to analyse grid strengthening vs. storage installation costs. Therefore, a connection to the available data sources such as a dedicated entity storing and providing information about the grid models, as well as the Decision Support Framework Simulation Engine (DSF-SE) for running the simulation of storage elements etc. needs to be established.

The Professional GUI inclusion in the overall S4G software environment is further being described in D3.3 [S4G-D3.3]. The screenshots presented in Figure 18 and Figure 19 show how the GUI supports the grid planner by displaying the grid topology and allowing the installation of new storage models to set up the simulation environment. Figure 19 depicts the results presentation after a simulation took place. The configuration of the simulation details is done by providing additional information using the side panel on the left. Critical nodes are indicated using the color red. Critical nodes can be further investigated by clicking on the respective house symbols. Simulation results are listed as depicted in Figure 20.


S4G Professional GUI

User: veronika.krauss.s4g Map Settings Logout

Simulation

Choose a loadprofile

Duration *: Daily
Profiles *: Industrial

Choose a duration span for the simulation.

Period of Time *: 10
Time unit *: Years

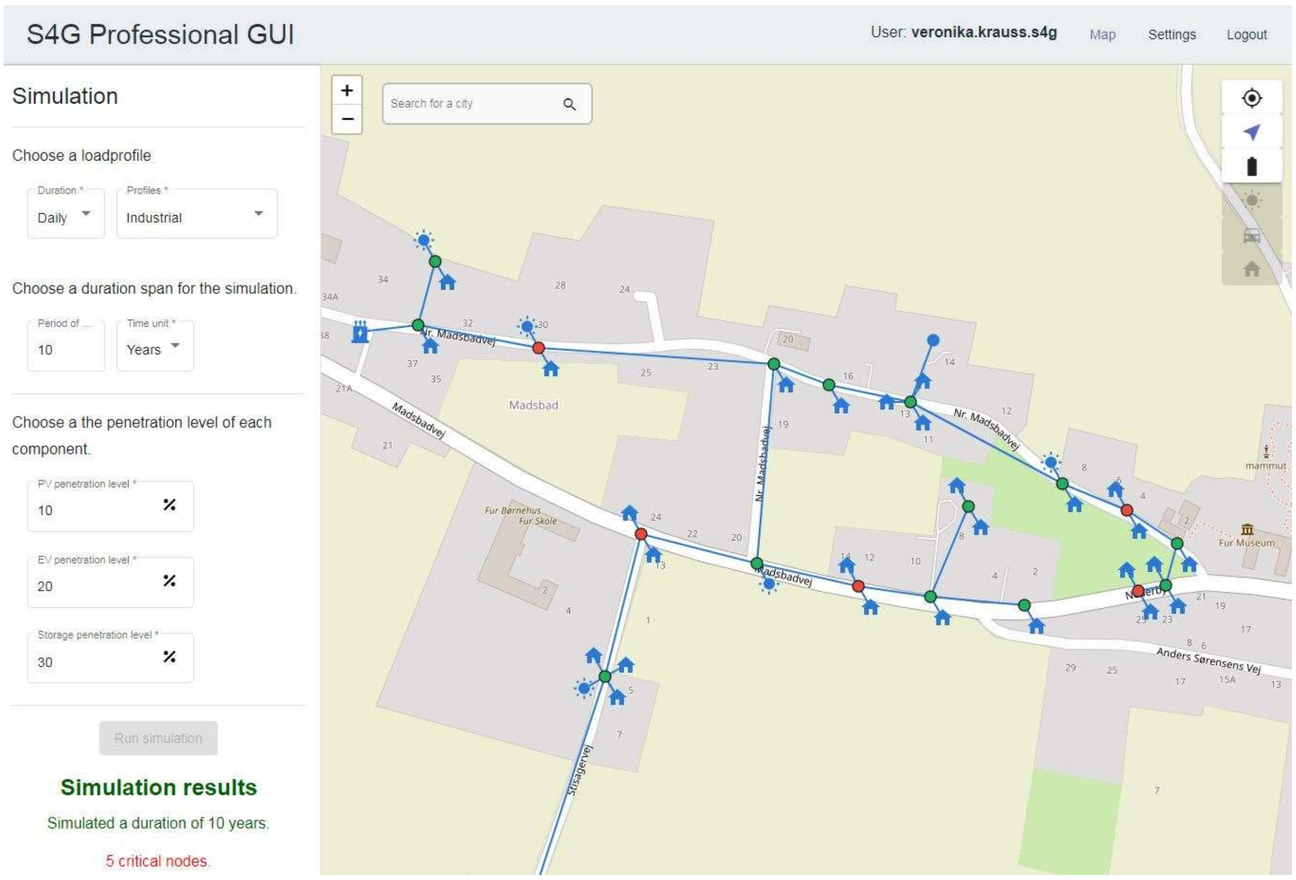
Choose a the penetration level of each component.

PV penetration level *: 10 %
EV penetration level *: 20 %
Storage penetration level *: 30 %

Run simulation

Please select a radial on the map

Figure 18 – Screenshot of the Professional GUI displaying the grid topology of Fur


S4G Professional GUI

User: veronika.krauss.s4g Map Settings Logout

Simulation

Choose a loadprofile

Duration *: Daily
Profiles *: Industrial

Choose a duration span for the simulation.

Period of ...: 10
Time unit *: Years

Choose a the penetration level of each component.

PV penetration level *: 10 %
EV penetration level *: 20 %
Storage penetration level *: 30 %

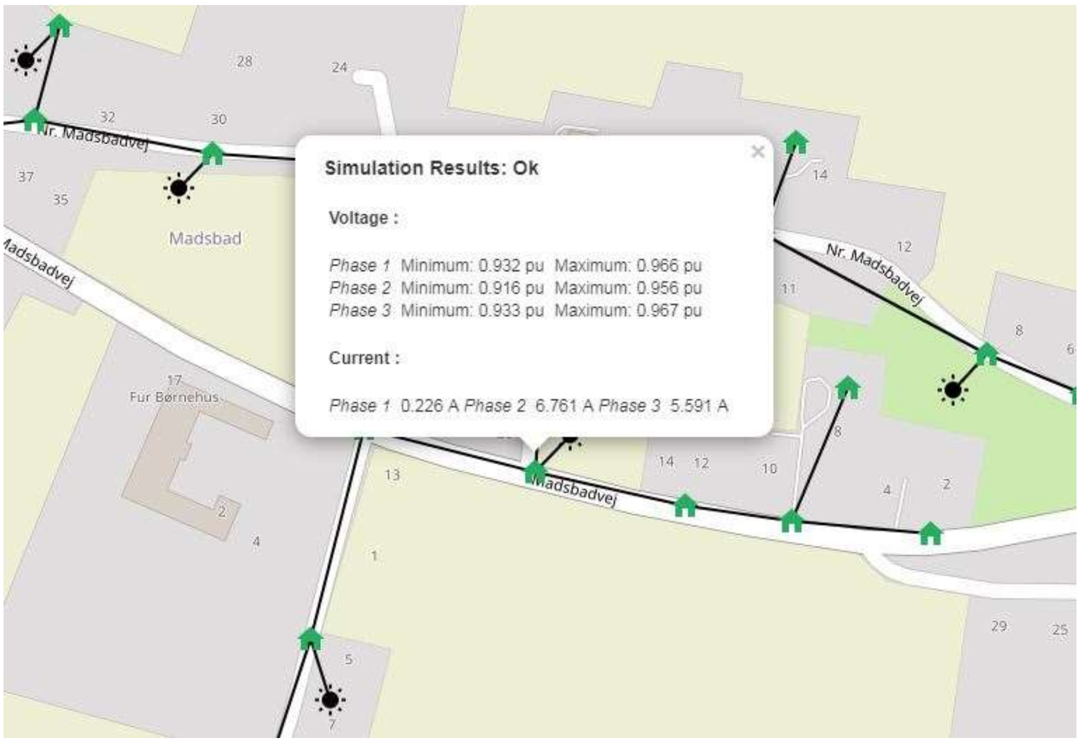
Run simulation

Simulation results

Simulated a duration of 10 years.

5 critical nodes.

Figure 19 – Results presentation after a simulation happened



Simulation Results: Ok

Voltage :

Phase 1 Minimum: 0.932 pu Maximum: 0.966 pu
Phase 2 Minimum: 0.916 pu Maximum: 0.956 pu
Phase 3 Minimum: 0.933 pu Maximum: 0.967 pu

Current :

Phase 1 0.226 A *Phase 2* 6.761 A *Phase 3* 5.591 A

Figure 20 – displaying simulation results per node by clicking on the respective house symbol

The sub-components of the prototype are shortly summarized in the following subsections.

3.1.1 Keycloak Service Connector

The Keycloak Service connector is established based on the Keycloak JavaScript Adapter listed in Table 5. It allows for end-user authentication using a REST API and an access/ ID token approach.

3.1.2 Simulation Engine Service Connector

The Simulation Engine Service connector establishes data exchange between the Professional GUI and the DSF-SE based on a REST API. Hereby, the GUI acts as visual component for operating the DSF-SE by enabling the end-user to specify data that should be used in the simulation, start/stop the simulation as well as manipulate the current grid topology and query/download simulation results.

3.1.3 GRID-DB Service Connector

A REST API connector for communicating with the DSF Connectors; which provide the grid models of interest.

3.1.4 OpenStreetMap Service Connector

A connector implemented using Leaflet, which connects to OpenStreetMap via REST API to get map data.

3.1.5 DSF-EE Connector

DSF-EE connector provides access to the DSF-EE and simulates the economic model of the grid. The Professional GUI sends the grid data and the simulation results acquired from Simulation Engine to DSF-EE Connector via a REST API HTTP request. Once, the DSF-EE responds with the economic model output, Professional GUI displays this result.

3.2 Installation/Deployment instructions

The application can be served on a simple http-server e.g. Apache[vii]. The Professional GUI code base contains a docker configuration file and a docker compose configuration file to assist in quick deployment using docker containers.

3.3 Software dependencies and requirements

No special hardware requirements are needed to use and/or deploy the Professional GUI. The prototype is based on the JavaScript framework Angular[viii] and uses Keycloak[ix] for authentication.

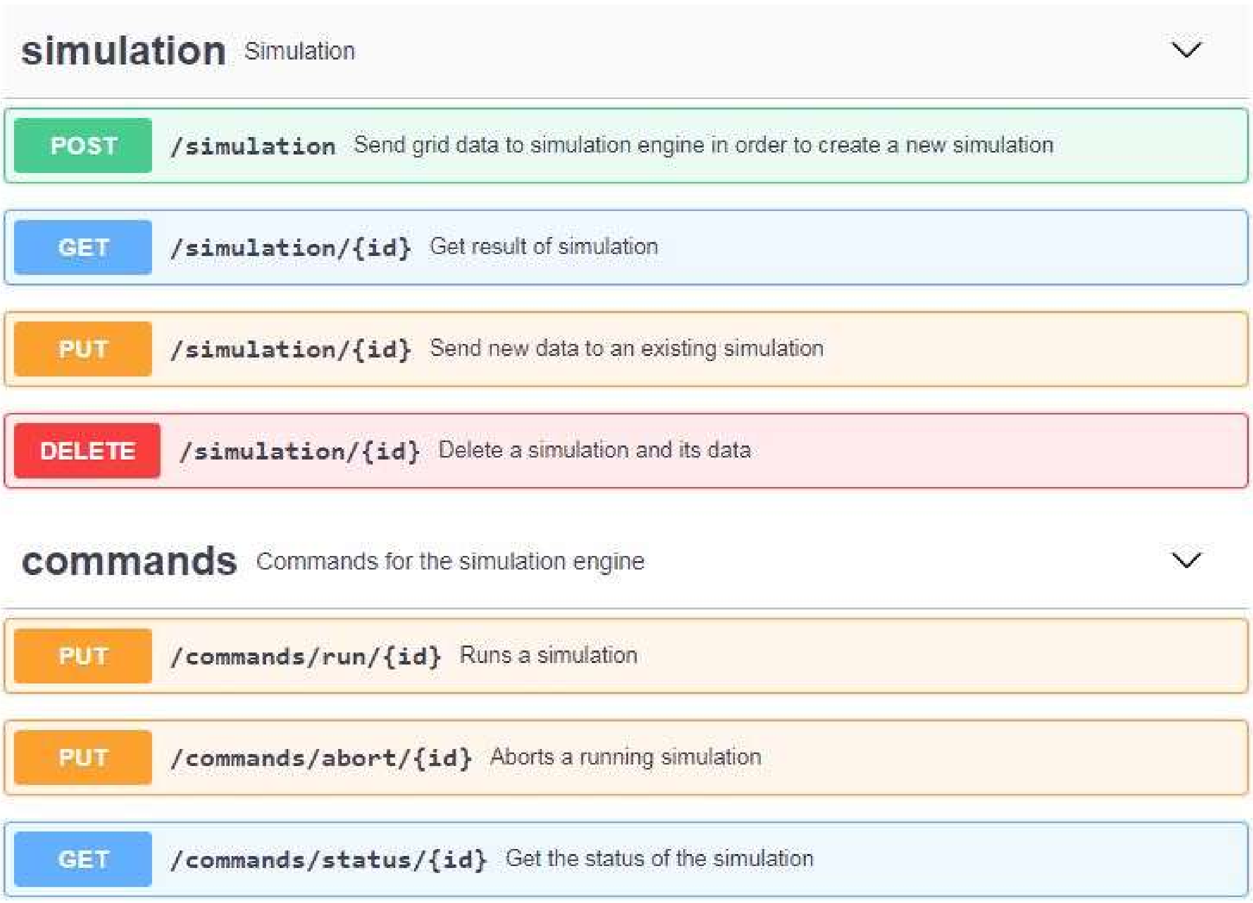
Table 5 – Software Dependencies of the Professional GUI

Dependency	License	Role
Angular V. 7.2.6 [viii]	MIT License	Framework on which the Professional GUI is based
Keycloak V. 5.0.0[ix]	Apache version 2.0	Authentication service
Keycloak JavaScript Adapter V. 4.1.0[ix]	Apache version 2.0	Connects the Professional GUI with the Keycloak authentication service
Leaflet V. 1.4.0[x]	2-Clause BSD	Communicates with OpenStreetMap and enhances the visualization of map data
Node Package Manager npm[xi]	Artistic License 2.0	Package manager

3.4 Server connection

The Professional GUI prototype connects to both the DSF-Connectors providing grid-relevant data such as grid models as well as the DSF-SE via a dedicated REST API specified for each component. More information about the DSF-Connectors and the DSF-SE is documented in the related documents D3.3 [S4G-D3.3] and D5.2 [S4G-D5.2] [M34]. An overview of the DSF-SE API is depicted in Figure 21.
For accessing data, user authentication is required using a Keycloak-based service, which offers a dedicated API for authentication requests. The authentication is established via access & ID token with a validity of 30 minutes.
Map data is provided via OpenStreetMap[xii] and accessed using the API offered by the library Leaflet listed in Table 5.

The DSF-EE provides its own API for interfacing with the Professional GUI. The GUI itself does not offer an API.



simulation Simulation

POST /simulation Send grid data to simulation engine in order to create a new simulation

GET /simulation/{id} Get result of simulation

PUT /simulation/{id} Send new data to an existing simulation

DELETE /simulation/{id} Delete a simulation and its data

commands Commands for the simulation engine

PUT /commands/run/{id} Runs a simulation

PUT /commands/abort/{id} Aborts a running simulation

GET /commands/status/{id} Get the status of the simulation

Figure 21 – API for the communication between the Professional GUI and the DSF-SE

4 Conclusions

This document presents the final version of the S4G GUIs developed within Task 6.3 - "Interaction with Residential and Professional users". The prototypes have been developed by following an iterative approach; the previous versions are D6.7 [S4G-D6.7] and D6.8 [S4G-D6.8].

The GUIs design and development were driven by the Storage4Grid high-level and primary use cases documented in D2.2 [S4G-D2.2]. The requirements creations and execution followed the S4G Agile Methodology, user needs and requirements were documented in the issue-tracking software JIRA. The GUIs requirements were based on scenario thinking, on the definition of persona, prototyping, as well as on interviews and user questionnaires.

The deliverable described the prototype interfaces that allow residential and professional users to interact with the S4G systems. The GUIs enable user to provide preferences, configuration options, access information about S4G systems operations and receive real-time and historical measurements.

The Residential GUI extended commercial products such as Fronius system interfaces providing additional status information of the household including EV charging units. Furthermore, it allows prosumers to select the operation mode of their local devices in order to maximize self-consumption or minimize costs.

The Professional GUI enables grid planners to analyze the techno-economic impact of storage system in the (low to medium voltage) grid. Additionally, the system can propose the optimal positioning of storage for the best possible outcome with respect to voltage impact. Grid planners can select the radial of the grid which is interesting for their analysis.

Finally, the developed prototypes have been evaluated with end-users to ensure a high quality and applicability of the proposed solutions in the problem domain addressed by S4G. The evaluation will be documented in D6.12 "*Phase 3 Evaluation Report*" in M39.

Acronyms

Acronym	Explanation
API	Application Programming Interface
CSV	Comma-Separated Values
DSF-DWH	Decision Support Framework Data Warehouse
DSF-EE	Decision Support Framework Economic Engine
DSF-SE	Decision Support Framework Simulation Engine
DSO	Distributed System Operator
ESS	Energy Storage System
EV	Electric Vehicle
GUI	Graphical User Interface
HTTP	HyperText Transfer Protocol
JSON	JavaScript Object Notation
LAN	Local Area Network
MQTT	Message Queue Telemetry Transport
OGC	Open Geospatial Consortium
P_Akku	Power being injected in home by battery (negative means charging, positive means discharging) acquired through the Fronius system
P_Grid	Power being taken from the grid (negative means power is being fed in the grid) and acquired through the Fronius system
P_Load	Home load (Negative means: house is consuming energy) acquired through the Fronius system
P_PV	Power being generated by the PV (always positive) acquired through the Fronius system
PCC	Point of Common Coupling
PV	Photovoltaic
REST	Representational State Transfer
S4G	Storage4Grid
SMM	Smart Metrology Meter
SMX	Smart Meter eXtensions
SOC	State of charge
USM	Unbundled Smart Meter

List of figures

List of tables

References

[i] Apache HTTP Server Project, https://httpd.apache.org/, last accessed 04 September 2019
[ii] AngularJS version 1.5.5, https://angularjs.org/, last accessed 04 September 2019
[iii] Firebase Auth service, https://firebase.google.com/docs/auth/, last accessed 04 September 2019
[iv] Firebase, https://firebase.google.com/, last accessed 04 September 2019
[v] Mobile App, https://en.wikipedia.org/wiki/Mobile_app, last accessed 04 September 2019
[vi] Web Application, https://en.wikipedia.org/wiki/Web_application, last accessed 04 September 2019
[vii] Apache Server, https://httpd.apache.org/, last accessed 04 September 2019
[viii] Angular JS V. 7.2.6, https://angular.io/, last accessed 04 September 2019
[ix] Keycloak, https://www.keycloak.org/index.html, last accessed 04 September 2019
[x] Leaflet, https://github.com/Leaflet/Leaflet, last accessed 04 September 2019
[xi] Node Package Manager, https://www.npmjs.com/, last accessed 04 September 2019
[xii] OpenStreetMap, https://www.openstreetmap.de/, last accessed 04 September 2019