D5.1

First version of SIFIS-Home testbed

WP5 – Integration, Testing and Demonstration

SIFIS-Home

Secure Interoperable Full-Stack Internet of Things for Smart Home

Due date of deliverable: 31/05/2022
Actual submission date: 31/05/2022

Responsible partner: SEN
Editor: SEN
E-mail address:
hakan.lundstrom@sensative.com

31/05/2022
Version 1.0

Project co-funded by the European Commission within the Horizon 2020 Framework Programme		
Dissemination Level		
PU	Public	**X**
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

The SIFIS-Home Project is supported by funding under the Horizon 2020 Framework Program of the European Commission SU-ICT-02-2020 GA 952652

Authors: Håkan Lundström (SEN), Marco Tiloca (RISE), Luca Barbato (LUM), Andrea Saracino (CNR), Otto Waltari (F-Sec)

Approved by: Andrea Saracino (CNR), Joni Jämsä (CEN), Tom Tuunainen (CEN)

Revision History

Version	Date	Name	Partner	Section Affected Comments
0.1	13/02/2022	Initial version	SEN	All
0.2	12/04/2022	Restructured version	SEN	All
0.3	27/04/2022	Group OSCORE, ACE, Edhoc	RISE	Implementation
0.4	04/05/2022	Several new chapters added. UX; FIWARE, etc.	SEN	All
0.5	12/05/2022	Several chapters completed	SEN, F-Sec	Validation, Intro, Conclusion, Executive summary
0.9	24/05/22	Ready for review	SEN, CNR, LUM	All
1.0	31/05/2022	Reviewer's comments addressed	SEN	All

Executive Summary

This deliverable describes the design and the deployment of the first version of the SIFIS-Home physical and simulated testbed that has been deployed to allow the partners to develop, deploy and test their applications for the different use cases. The test bed includes live IoT devices as well as simulated ones to perform large scale experiments for stress-testing resilience improving mechanisms.

The test bed was designed and implemented based on the requirements and architectural design from WP1. A key finding is that the API gateway parts of the original architecture had to be revised to make it feasible to implement. The revised architectural proposal that includes a separation of some components between the web parts residing on the API gateway and the device parts residing on the SIFIS-Home smart devices. This change was expected since the original architecture was done early in the project before actual implementation was started by the partners.

Further challenges were overcome in relation to the Cyber-Perimeter and how to manage the communication flow between the smart devices behind the firewall in the home and the API gateway connected to the Internet. We concluded that the only feasible way was to let devices inside the home set up the sessions to the API gateway and subscribe to possible events. By doing so, the traffic will pass through firewalls with standard configurations without problems and the API gateway is able, when required, to communicate with the devices inside the Cyber-Perimeter in a secure way.

With this first version of the test bed, we do not have complete integration between all the SIFIS-Home components, as some are still under development. The focus has been on designing the test bed so that it can be used in a collaborative way by the partners to implement, verify and validate the SIFIS-Home technologies according to the requirements. The test bed is up and running, the UI is working, code is being delivered to GitHub and the SIFIS-Home components are continuously being integrated on the test server.

Table of contents

1 Introduction

In this deliverable we report on the design and implementation of the preliminary version of the SIFIS-Home test bed based on the requirements, architecture and design from WP1. The report goes through the different key steps that were performed with focus on the test bed design aspect and the implementation implications of the requirements.

It starts with an identification of the user roles and the different device types that the test bed needs to support, and discusses if we will have access to live devices or must use simulated devices. Then it moves on to cover the Cyber-Perimeter concept and how the expected firewall at every home put some constraints on the communication flow between the home side of the SIFIS-Home network and the external side.

The actual test bed design considers the requirements defined in WP1 as well as the consortium partners competence and access to assets that can be used in setting up the test bed. Once that is done, implementation aspects of the different main components are analysed one by one, also looking for any specific design considerations that were overlooked during the original architecture design phase.

Next steps walk through the interfaces to key components like the FIWARE Context Broker that holds the state of the system for the web UI and WebThings that is the base for the SIFIS-Home smart devices. There is also a description with screen shots from some of the components in the web UX while the concluding section contains the validation strategy for each of the requirements from D1.2.

2 Actors, devices and the Cyber-Perimeter

2.1 *Analysis of Components and Actors of the SIFIS-Home architecture related to the test bed*

The main components of the SIFIS-Home test bed architecture are the following:

- *Smart Devices:* These are devices that implement a SIFIS Home distributed hash table (DHT) that enables a client to set up a Peer-to-Peer (P2P) logical model. General examples of Smart Devices are Smart TVs, Smart Refrigerators, Laptops/Desktops, Family Hubs. From a test bed point of view, we are planning to both simulate them and use physical devices based on Raspberry PIs.
    - *Internet Connected Smart Devices*: This is a subset of the Smart Devices without a SIFIS-Home client APP, like smart phones and tablets, but located inside the smart home Cyber-Perimeter and equipped with a network interface which enables connectivity outside of the smart home. From a test bed point of view, in SIFIS-Home we have analytics capabilities to analyse the traffic from those devices via a gateway network interface layer with a SIFIS-Home client software.
- *Not So Smart Devices (NSSD):* These are constrained devices that cannot be customized by installing third party software or applications. In the SIFIS-Home test bed, we are using several of these, such as smart sensors, smart cameras, smart lights, smart speakers and smart locks. NSSDs are either using standard protocols like Z-wave, LoRaWAN or are TCP/IP based with Internet connectivity options and are set to connect through their responsible Smart Devices.

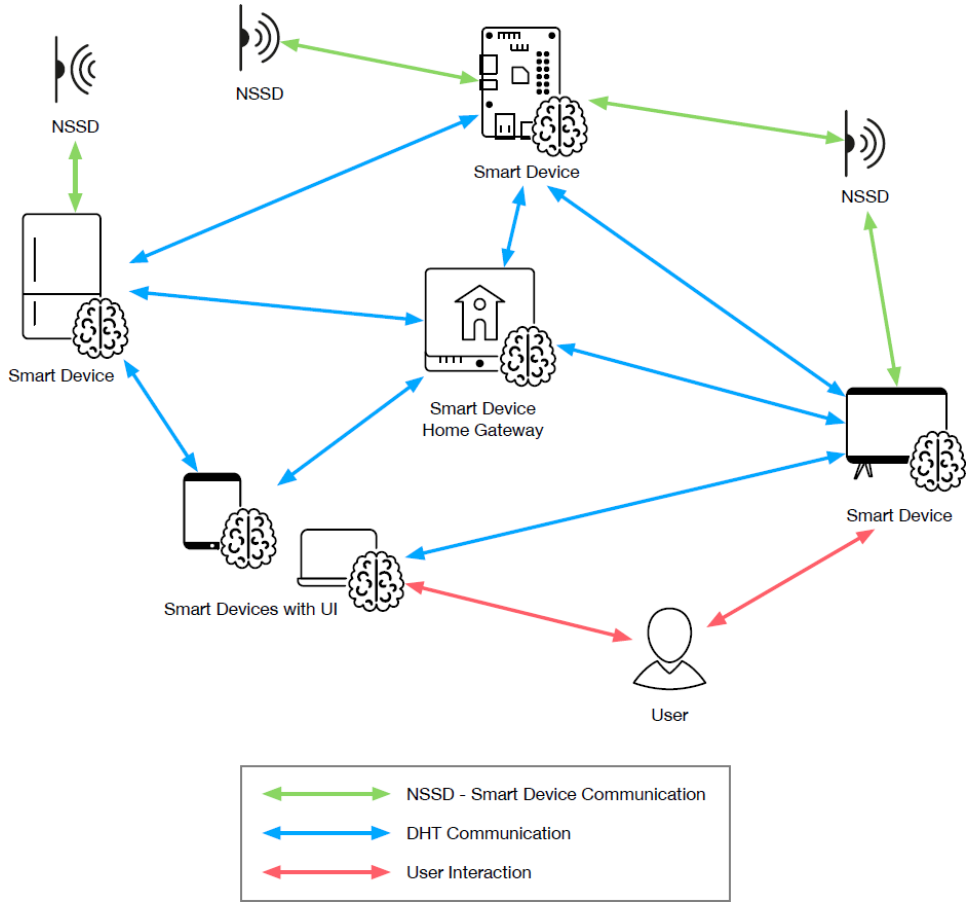


Figure 1: Communication and interaction between components

The actors we have defined for the SIFIS-Home architecture are the following:

- *SIFIS-Home Administrator*: The administrator is a human user who is the owner of an instance of the SIFIS-Home architecture.
- *SIFIS-Home Tenant*: The SIFIS-Home tenant is the standard user of the smart home system.
- *SIFIS-Home Maintainer:* The maintainer is an entity external to the smart home and trusted by the administrator to correctly configure the smart home security, privacy and safety policies.
- *SIFIS-Home Tenant with restrictions*: This user is a smart home tenant with restrictions on the functionalities they can access.
- *Guest*: A guest is a smart home user who is not resident in the smart home but is allowed to access and use the premises and some functionalities for a limited amount of time, upon authorization from the administrator or another tenant.
- *External Operator*: The external operator could be a technician, a plumber, gardener, or house cleaner, allowed to access the smart home premises for a limited amount of time, with the authorization from a tenant.

In the test bed design, all these roles need to be represented and have their distinct usage.

2.2 *Analysis of The Smart Home Cyber-Perimeter*

In order to protect the Smart Home and its users from unintended disclosure of sensitive information, in WP1 we defined a logical distinction between the outside and inside of the Smart Home, based on what we called Smart Home *Cyber-Perimeter*.

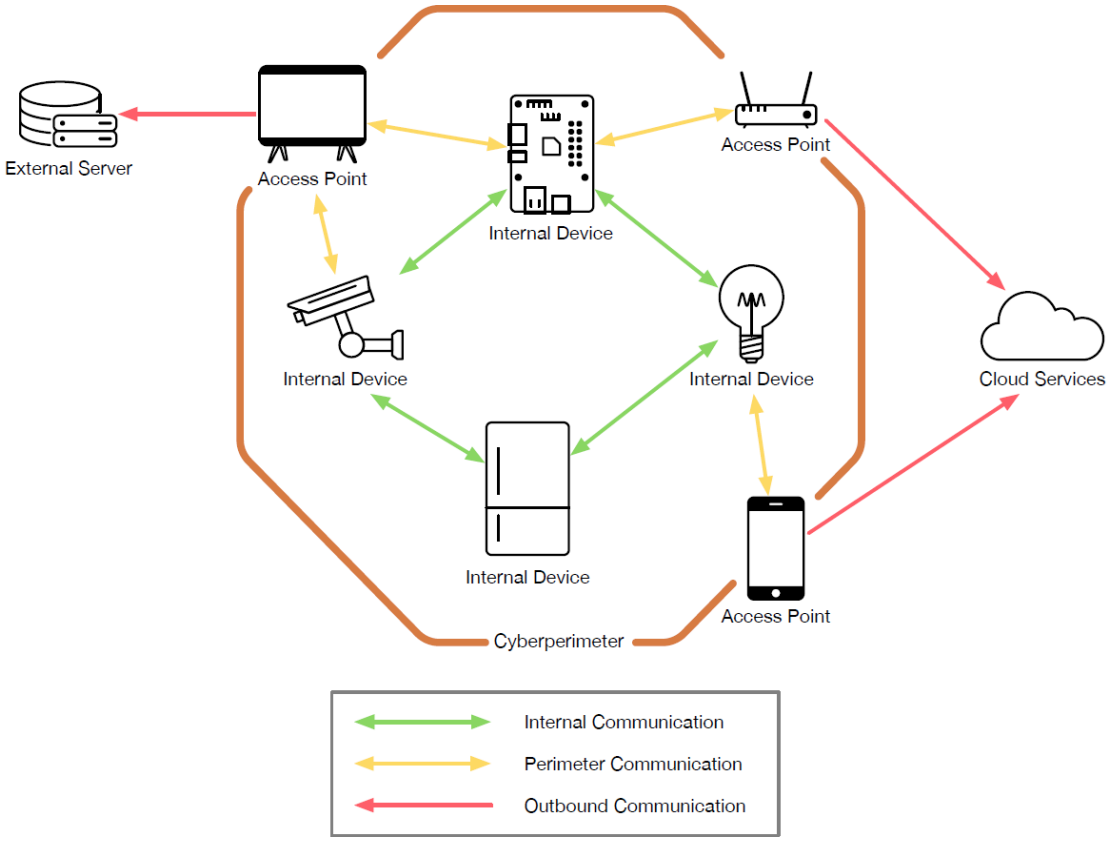


Figure 2: Concept of Smart Home Cyber-Perimeter

From a test bed point of view, the Smart Home Cyber-Perimeter is a requirement that must be fulfilled in the architectural design while also be implemented and verified in the SIFIS- Home test bed. For the time being, we just conclude that the smart home will be protected by a firewall (access point in figure above) that will block all incoming IP traffic, but the devices on the inside are free to set up sessions to outside servers as they like, considering that the relevant ports on the firewall are opened for outgoing traffic.

3 Test bed design

3.1 *Main requirements on the test bed*

When designing the test bed, we considered how to be able to setup, verify and validate that the test bed itself and that the fundamental part of the SIFIS-Home system would work as intended. The conclusion is that it is both the Functional requirements as well the NFR – Non-functional requirements as defined in D1.2 and D2.1 that should be possible to verify and validate when using the test bed. The principal areas that must be covered are as follows.

- Integration feasibility, as a fundamental requirement on the test bed. It must be feasible to integrate all SIFIS-Home software components and analytics into the test bed both live and simulated.

- Communication flow and interfaces between all integrated components like the API Gateway to SIFIS-Home Smart Devices and Not So Smart Devices.
    - This holds both for live smart devices implemented by Raspberry PIs as well as for simulated devices as microservices running in the test bed.
- It is also important to check that every interface is behaving as expected as well as that data structures stored in the DHT – Distributed Hash Tables – and in the data bases in the API gateway are as expected.
- General UI interaction, verification and validation. If things are working well when performing different UI interactions, it is a good indication that the system as such is fundamentally working correctly and as expected.
- Notification and event management verification and validation. These are events that could arise from any device or analytics executing in the network. It must be possible to display these events and present their outcomes to the user, thus enabling decision-making on how to handle the events.
- Performance related verification and validations. The system as such must be reliable and display good performance when different types of user interactions occur.
- Stability related verification and validations. The system must be able to be stable in the long term, and to run for consecutive months without any execution problems.
- Security related verification and validations including intrusion detections and the other analytics defined in WP3 and WP4.
- Privacy related verification and validations, to ensure that no data that could violate the GDPR regulations is collected.
- The Cyber-Perimeter is one of the most difficult requirements to fulfil, since the SIFIS- Home network and its devices must be reachable for the end users not only from inside the home but also from the outside.

As a summary, when all the items above are successfully verified and validated in a SIFIS-Home system run in the test bed, it should also be possible to assess the key metrics like code coverage, unhandled exceptions, security metrics and privacy metrics defined in deliverable D1.2.

3.2 *Allocation of components*

The SIFIS-Home architecture is the representation of the devices and actors interacting with the SIFIS-Home Framework (see deliverable D1.3). In order to design and later implement the test bed, we first found out all partners capabilities, experience, commercial interest and availability of legacy code, tools and devices. Then, the combinations of the partners' expertise and the architecture and requirements of the SIFIS-Home framework led us to identify the optimal allocation of all components to different partners to be able to realize both the live and the simulated test beds. It was early decided that the live and simulated test bed should look alike, with the only difference about connecting either live devices or simulated devices to the SIFIS- Home network.

3.3 *Designing the test bed*

The test bed is built around a server, called "Panarea", which is located at the CNR facilities and can

be reached by the partners via SSH - Secure Shell access – in order to upload and configure code. In the Panarea server, the SIFIS-Home API gateway will run, as well as other SIFIS-Home technologies such as simulated SIFIS-Home smart devices, analytics and network security solutions. In order to validate the generic functionality of the test bed, we also added a few Not So Smart Devices running on standard IoT protocols.

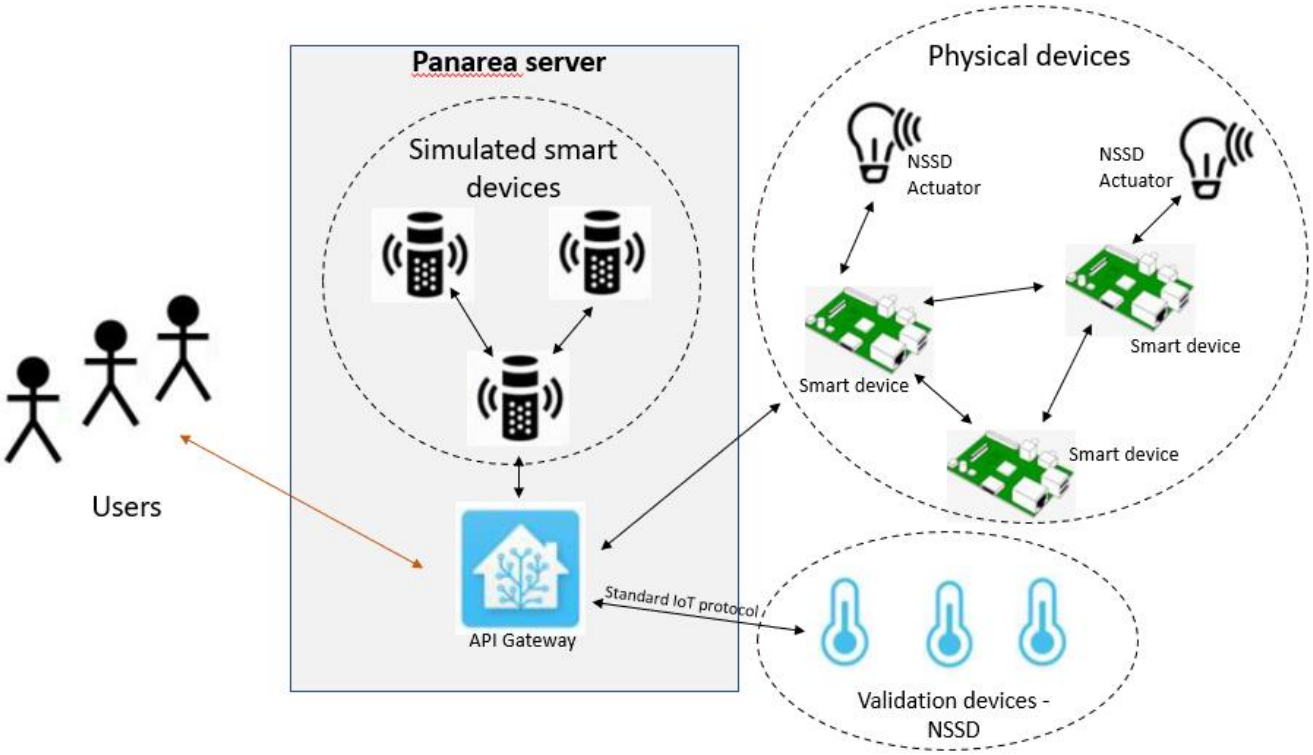


Figure 3: The SIFIS-Home test bed setup

Physical NSSD - Not So Smart Devices as well as SIFIS-Home Smart devices built upon Raspberry PI based upon WebThings technologies will then connect to, authenticate with and register with the API gateway, and thus create the SIFIS-Home network. End users will primarily use the network via the API gateway, which must be also reachable from the Internet, thus enabling end users to check the system status of the smart home from anywhere.

3.4 *Panarea server*

The SIFIS-Home testbed is built around the Panarea server, hosted at the CNR premises, in a dedicated virtual machine. The VM exploits Docker for deploying a lightweight testbed. In particular, through each Docker container we instantiate both the API gateway as well as emulated SIFIS-Home smart devices. Thus, each Docker can be viewed as a Linux-based device, able to install the software of the SIFIS-Home framework. To complete the test bed, both live Smart Devices and Not So Smart Devices will get connected to the API gateway and the emulated Smart Devices on the Panarea server. In Figure 4 we depict the logical structure of the emulated testbed. As shown in the figure, the testbed exploits Docker as a virtualization environment, on top of an ESXi-based VM. Each Docker container represents a Smart Device, thus being able to run the SIFIS-Home framework as a service on top of it. A further container, in the current testbed, runs Ratatosk as an external component, whose functionalities can be queried via API.

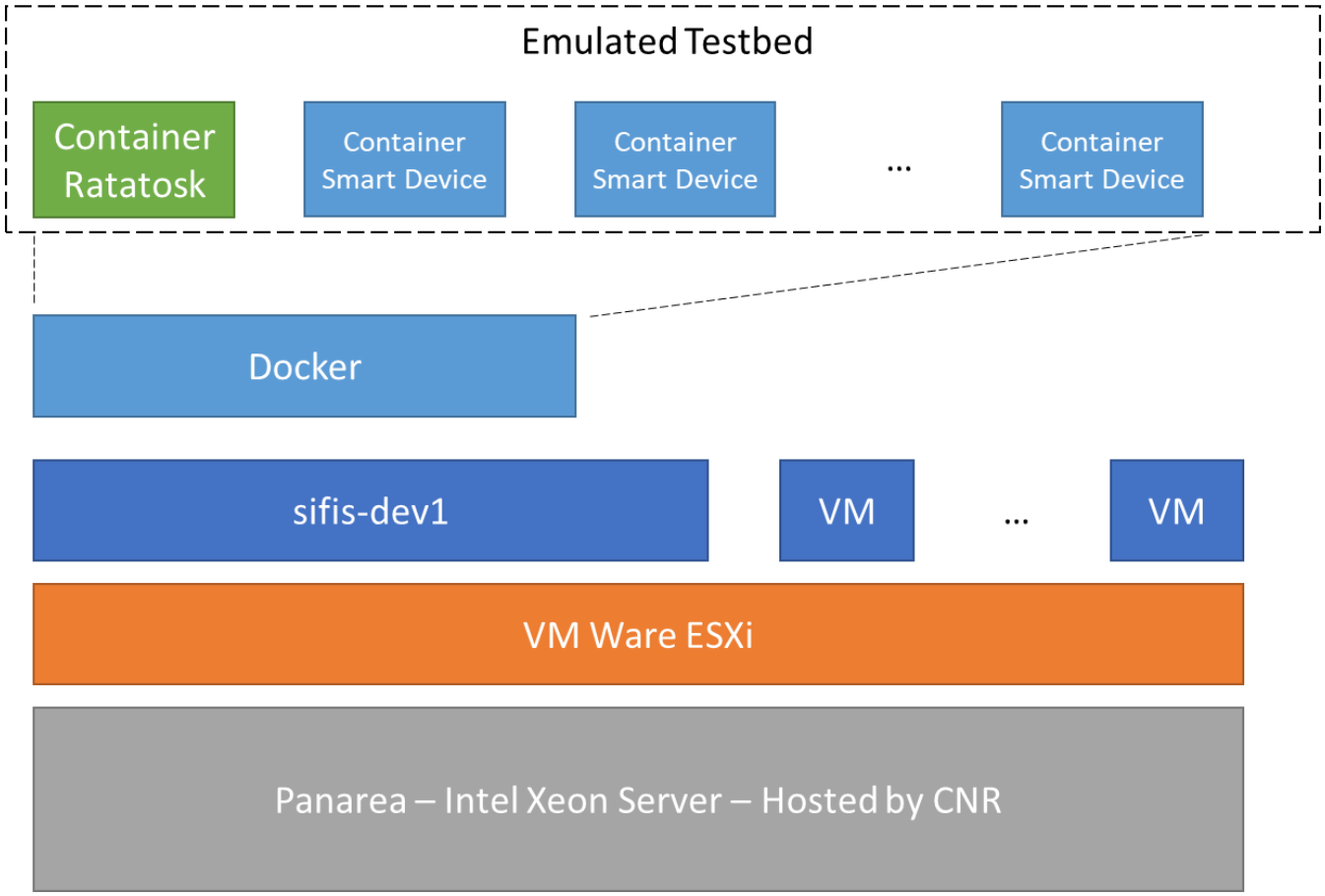


Figure 4: Architecture of the SIFIS-Home emulated testbed

4 Implementing the test bed

4.1 *Original architecture*

The design of the SIFIS-Home framework in WP1 is based on Docker containers with *microservices* design pattern. Each microservice defines a Rest API for interfacing with it. This design pattern applies to the test bed design as well, but in the test bed some partners legacy code has been added to be able to quickly manage integration of components and to enable us to verify and validate the system.

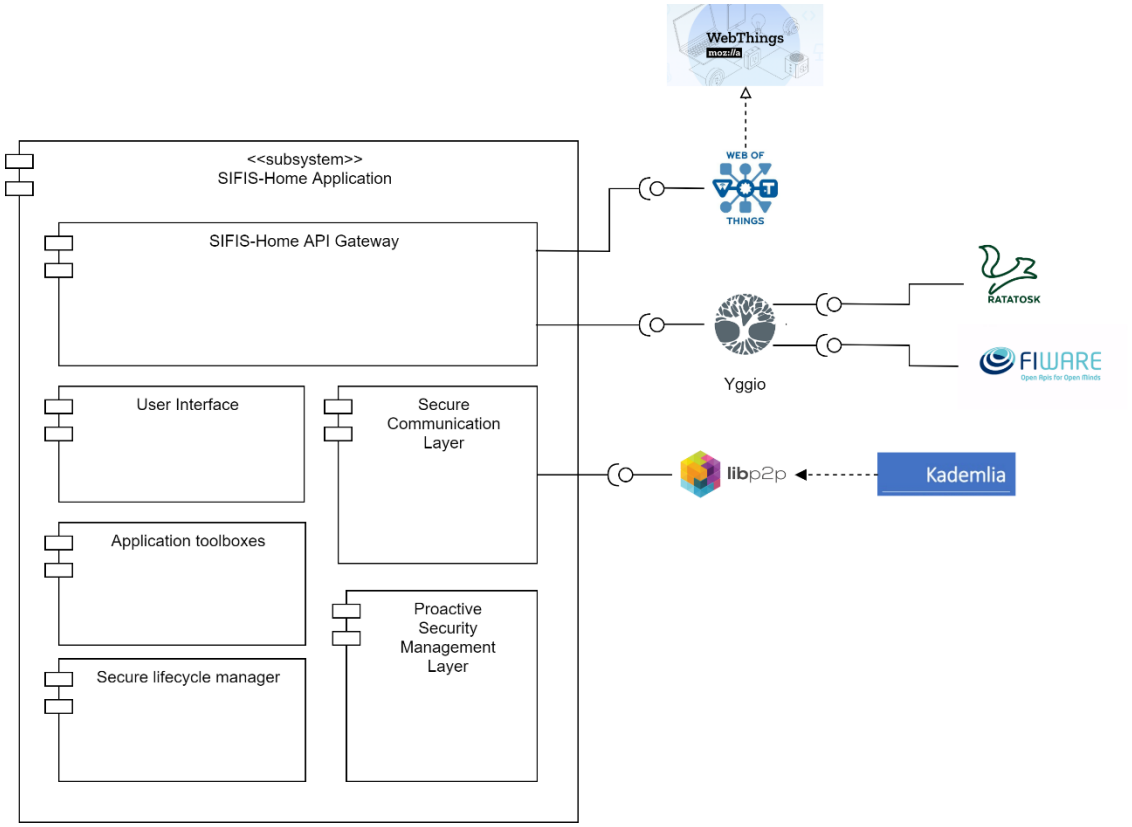


Figure 5: Original high-level architecture of the SIFIS-Home framework as defined in D1.3

The high-level architecture of the SIFIS-Home framework defined in deliverable D1.3 is the basis for the test bed design and is visualized in Figure 5. The architecture comprises six main building blocks that together constitute a SIFIS-Home application. When we started the in-depth technical design and actual implementation of the system and the test bed, we realized that the API gateway and the use of Web-Of-Things as defined in the architecture from deliverable D1.3 were not feasible to implement. This is not an unexpected finding, since that was a theoretical design made early in the project before all technical aspects had been considered.

4.2 *Implementation*

The following provides implementation considerations for each main building block.

- **SIFIS-Home API Gateway**: this component includes a set of high-level APIs and is used to access SIFIS-Home networks and their UI both internally from inside the smart home network and externally. The core part of the API gateway is the FIWARE Context Broker Ratatosk on top of which its UI is built. The Ratatosk Context broker will contain the full description of the SIFIS-Home devices that are connected to the gateway.

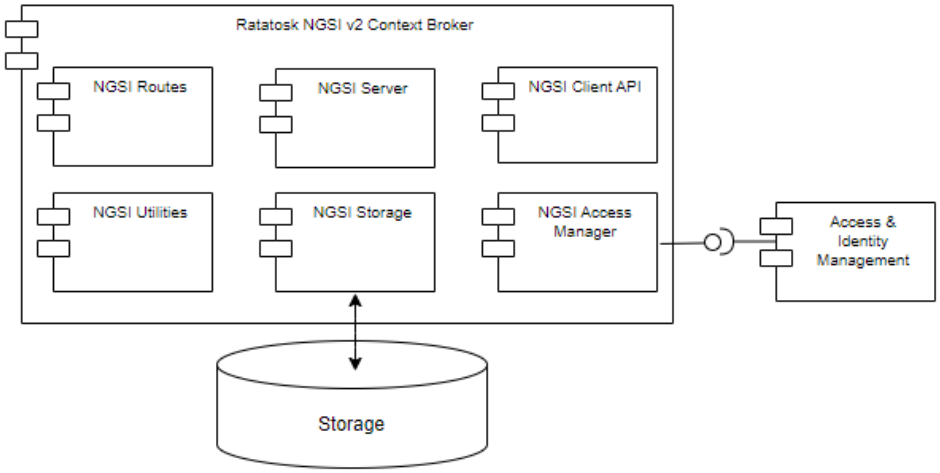


Figure 6: Ratatosk FIWARE Context Broker as the core of the API gateway

We decided to base the API gateway on the partner Sensative's horizontal IoT integration platform Yggio, which also implements the FIWARE Ratatosk Context broker. This also means that, from an architecture design point of view, the API Gateway does not belong to the SIFIS-Home application and is instead an external component.

- **User Interface:** this component provides the Graphical User Interfaces to all the different kinds of users that use the SIFIS-Home Framework.

  The web part of the user interface resides on the API Gateway, since it needs to be accessible from both inside the SIFIS-Home network as well as externally from the Internet when the home users are outside of the smart home. The web UI on the API Gateway is running on top of the Ratatosk Context broker, but FIWARE does not implement all the APIs required by the UI. Thus, Sensative Yggio's legacy proprietary API's and security APIs are used as well, in order to make it all work well in the test bed.

  The device part of the UI that executes on the SIFIS-Home Smart Devices residing inside the smart home will comprise, as intended to the device, the relevant subset of the web interface. This UI is still under design phase.

- **Secure Lifecycle Manager**: this module handles the standard workflow of the SIFIS-Home framework. This module acts as orchestrator of the framework lifecycle, regulating presence and behaviour of both Smart Devices and applications. In particular, the Secure Lifecycle Manager enables and handles new device registration and deregistration, as well as management/provisioning of device-to-device keying material and management/enforcement of access rights for device-to-device resource access. Moreover, it manages installation and removal of third-party applications, according to the received instructions from the user or from the policy engine and the intrusion detection system.

  The following security solutions developed in WP3 pertain to the "Security Lifecycle Manager" module and a link to their implementation from RISE is also provided. These implementations build on the open-source Eclipse Californium CoAP framework available at [CALIFORNIUM], which provides the CoAP protocol and the OSCORE security protocol. A single codebase collecting these implementations is accessible at [WP3-CODEBASE], as

available for use, integration and testing within the SIFIS-Home project and especially in the interest of the WP5 testbed.

[CALIFORNIUM] https://github.com/eclipse/californium
[WP3-CODEBASE] https://github.com/sifis-home/wp3-solutions

- **OSCORE profile of the ACE framework**, as documented in Section 5.1 of deliverable D3.2. Together with the main ACE framework, the implementation is available at https://bitbucket.org/marco-tiloca-sics/ace-java
- **Key provisioning for Group OSCORE communication using the ACE framework**, as documented in Section 6.1 of deliverable D3.2. Together with the main ACE framework, the implementation is available at https://bitbucket.org/marco-tiloca-sics/ace-java
- **EDHOC key establishment**, as documented in Section 6.3 of deliverable D3.2, including specific profiling for CoAP and OSCORE. The implementation is available at https://github.com/rikard-sics/californium/tree/edhoc

- **Secure Communication Layer**: this module is responsible for the secure communication between the devices of the SIFIS-Home architecture. The testbed that is to be created should be able to cover multiple scenarios of the Secure Communication Layer. The testbed serves as an environment for testing the efficiency of the solution to be created and it must be a controlled environment where actions are carried out in a secure and isolated manner, without influencing the process being controlled. The secure communication layer includes mechanisms to ensure standard encryption features on the communication associations to be used among the devices. In particular, the component handles security of communication between Smart Devices and among Smart Devices and NSSDs. In the implemented testbed, we will consider different communication models, implementing different possible topologies: in particular, we are going to consider a first emulated testbed where all devices are supposed to be connected through a Wi-Fi access point and another configuration where devices will be connected via Ad-Hoc Wi-Fi or an alternative point-to-point physical protocol. This last configuration will be particularly needed to test the DHT-based routing of SIFIS-Home messages among Smart Devices and to model possible issues of network partitioning.

  The following security solutions developed in WP3 pertain to the “Secure Communication Layer” module and a link to their implementation from RISE is also provided. These implementations build on the open-source Eclipse Californium CoAP framework available at [CALIFORNIUM], which provides the CoAP protocol and the OSCORE security protocol. A single codebase collecting these implementations is accessible at [WP3-CODEBASE], as available for use, integration and testing within the SIFIS-Home project and especially in the interest of the WP5 testbed.

  [CALIFORNIUM] https://github.com/eclipse/californium
  [WP3-CODEBASE] https://github.com/sifis-home/wp3-solutions

  - **Group OSCORE**, as documented in Section 4.1 of deliverable D3.2. The implementation is available at https://github.com/rikard-sics/californium/tree/group_oscore
  - **OSCORE profile of the ACE framework**, as documented in Section 5.1 of deliverable D3.2. Together with the main ACE framework, the implementation is available at

https://bitbucket.org/marco-tiloca-sics/ace-java

- **Key provisioning for Group OSCORE communication using the ACE framework**, as documented in Section 6.1 of deliverable D3.2. Together with the main ACE framework, the implementation is available at https://bitbucket.org/marco-tiloca-sics/ace-java
- **EDHOC key establishment**, as documented in Section 6.3 of deliverable D3.2, including specific profiling for CoAP and OSCORE. The implementation is available at https://github.com/rikard-sics/californium/tree/edhoc

- **Application toolboxes**: this component collects related and interconnected sub-components that are all services inside the SIFIS-Home infrastructure.
  - Alarm / Log: The log storage component was decided to be based on the well proven standard UNIX SYSLOG format (https://datatracker.ietf.org/doc/html/rfc5424) and then adapt it to work with modern REST API with a JSON payload containing the actual log event. The log storage component cannot be physically located in the SIFIS-Home Application and must instead reside on the API Gateway. This will enable all SIFIS-Home devices and analytics from any SIFIS-Home network to create log events that can then be viewed and acknowledged in the UI by authorized users in the smart home where the log events were generated.
  - Analytics toolbox: the component of the analytics toolboxes will be used by the testbed to support intrusion detection tests. As this testbed is, in fact, focused on verifying the fulfillment of non-functional requirements, the analytic toolbox does not play a key role. Still, the testbed will verify the possibility of correctly invoking the analytics, by using synthetic data for functionality verification.
  - Anonymization Toolbox: this component will be included in the testbed and its functionality verified on top of synthetic data.
- **Proactive security management layer**: This component is responsible for proactively preserving the security properties of the SIFIS-Home infrastructure. Proactivity implies taking preemptive measures before an incident occurs.

  The following proactive security measures have been developed as part of SIFIS-Home project and will be deployed on the WP5 testbed:
  - **AUD Manager** is a proactive security measure for IoT anomaly detection in a consumer environment. The AUD manager operates on information from the network and transport layers (L3 & L4), as underlying the session and application layers where vulnerabilities are often exploited, e.g., through session hijacking and man-in-the-middle attacks. The analytic developed in WP4 strives to spot the anomalous network events before data packets reach their destinations, i.e., the target devices.
  - **Manufacturer Usage Description** (MUD, RFC 8520) mimicking support for devices without MUD support from the vendor. MUD is a proposed standard for describing the expected network behavior of an IoT device. The usage description itself is an access control list defined by the vendor. It explicitly defines protocols, ports and endpoints with which a particular IoT device is allowed to communicate with. However, as of today, vendors that provide MUD support are quite uncommon. By leveraging the usage description models created by the AUD manager described above, we can create an access control list for any local IoT device. By enforcing such an access control measure for a home IoT-device, one would proactively block any incoming harmful

network intrusion attempts.

4.3 *Authorization and Access management integration*

When looking at the challenges of the authorization and access management integration related to the API Gateway that is powered by Sensative Yggio horizontal IoT integration platform, there are some aspects that we need to take into consideration when designing the test bed.
Yggio is using the well-known open-source component KeyCloak for access and authorization management, which is a well proven industry standard component that meets the requirements of the API Gateway. Within the SIFIS-Home network, we use DHT – Distributed Hash Table – for both device and user management. Hence, the key integration activity to make the SIFIS-Home network work with the API gateway was to identify how to connect them through the firewall that every smart home will be supposed to have. We decided to adopt the following solution.

- Users, access and authorization: Let the DHT inside the SIFIS-Home network "Create the smart home", by activation of the home admin user, and then create the default smart home users in the API Gateways KeyCloak component via its API. This is possible since the API gateway is IP addressable from inside the SIFIS-Home network. This is effectively a DHT to KeyCloak integration.
- Devices: In order to achieve secure communication across the Cyber-Perimeter, we decided to start with using MQ Telemetry Transport (MQTT) with TLS 1.3 and utilize the publish/subscribe mechanism to synchronize the devices between Ratatosk and the DHT within the smart home. MQTT communication can pass through the firewall and makes it feasible to use the UI on the API gateway to control both Smart Devices and Not So Smart Devices inside the SIFIS-Home network. This is effectively a DHT to MQTT integration.

The WP3 security services based on Group OSCORE, ACE framework and EDHOC (see Section 4.2) will be integrated into the Panarea server. The technical feasibility of how to integrate them with the API Gateway is being studied.

4.4 *Revised proposed architecture*

The implementation of the test bed has revealed some weaknesses in the original architecture that had to be analyzed and revised. The architecture in the image below is what is proposed to be used as input for further studies in WP1. The target is to finalize it in deliverable D1.4.

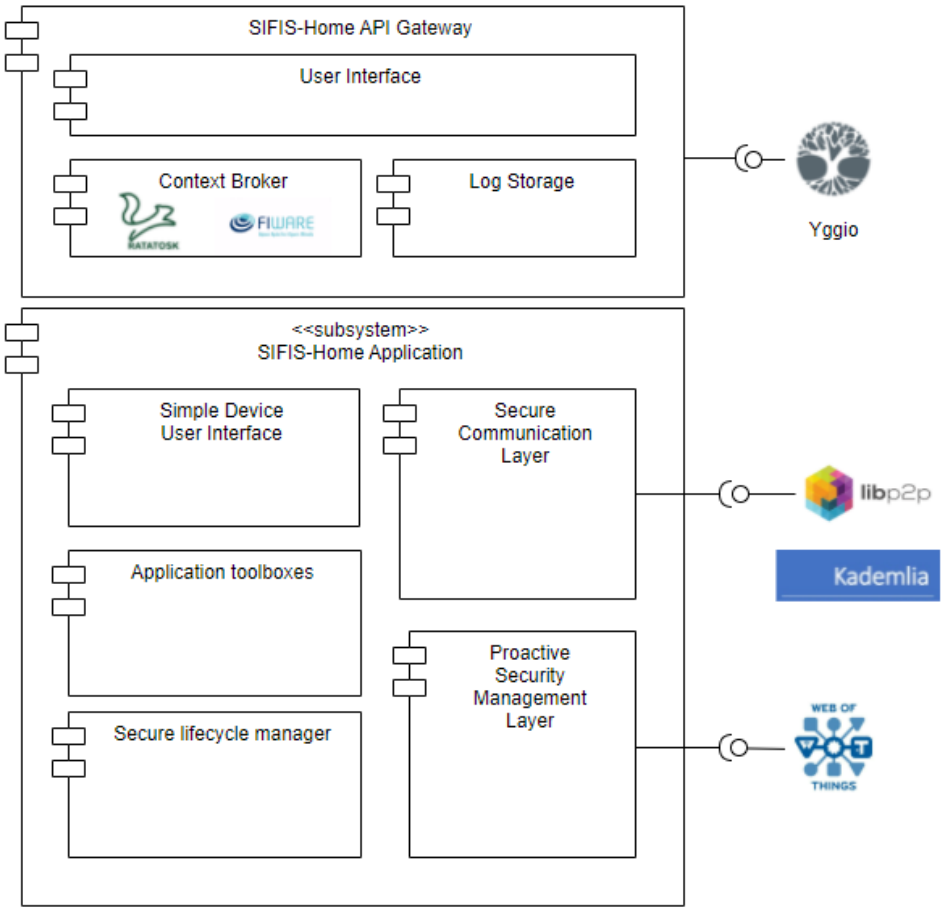


Figure 7: Revised SIFIS Home architecture with some components moved to the API Gateway

As a summary of the test bed design considerations, the overall architecture from deliverable D1.3 is mostly unchanged. A few components have been moved to the API Gateway as described in the revised architecture above. Sensative Yggio also provides legacy code, API's and components like KeyCloak that enables the test bed to be implemented with a nice and functional UI in an efficient way. Via the test bed special integration of the DHT with the API Gateway over MQTT and with the KeyCloak API, the full system is expected to work in line with the vision of SIFIS- Home.

5 Interfaces of the test bed

5.1 *Ratatosk, the FIWARE Context Broker*

FIWARE NGSI v2 [FIWARE, 2021] Ratatosk is a publish/subscribe Context Broker that holds a state of a system via FIWARE entities. At the moment, the Context Broker implements FIWARE NGSI v2 APIs as defined here https://swagger.lab.fiware.org/ , and we are investigating the feasibility to upgrade it to NGSI-LD API's during the SIFIS Home project. However, it is not a priority requirement.

Each FIWARE entities are described in JSON via a data model. FIWARE defines recommended data models to simplify interoperability between systems at https://github.com/smart-data-models , but, if none fits, it is also possible to define one's own data models or use a subset of an existing data model to represent the type of object one wants to describe. This is an example of an NGSI-LD Air Quality entity in so called *normalized* format:

```
{
  "id": "urn:ngsi-ld:AirQualityObserved:SE:Id235",
  "type": "AirQualityObserved",
  "dateObserved": {
    "type": "Property",
    "value": {
      "@type": "DateTime",
      "@value": "2022-04-28T11:40:00Z"
    }
  },
  "CO2": {
    "type": "Property",
    "value": 314,
    "unitCode": "PPM"
  },
  "refPointOfInterest": {
    "type": "Relationship",
    "object": "urn:ngsi-ld:PointOfInterest:SE:Parkingplace"
  },
  "@context": [
    "https://schema.lab.fiware.org/ld/context",
    "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld"
  ]
}
```


The data models can also be represented in simple KeyValue format in a flat structure. In the SIFIS-Home project, we still have to define data models for the devices we will use in the test bed, and it is currently not decided how to merge data models defined by devices in discoverable WebThings format with the NGSI format used by the Context Broker.

An important distinction with Ratatosk is that it is a secure FIWARE Context Broker relying on the KeyCloak open-source component as a security plug-in, and that it always requires a valid authentication token to accept a command. This will be used to make sure that a user in the smart home that attempts to perform some actions has the required authorization to do so. These are some Rest API examples of how to interface the Context Broker that will be used in the project:

Command	Modify	Syntax

Update description of existing node	<id>
	<token> hex string
	<user id> Yggio user id
		curl -sS -X POST https://yggio.sifis-home.eu/ngsi/v2/entities/<id>/attrs?type=Device -H "Fiware-UserToken:<token>" -H "content-type: application/json" -H "fiware-service: yggio" -H "fiware-servicepath: /" -H "fiware-userid: <user id>" -d @- <<EOF
		{
		"description": "My new description - FIWARE"
		}
		EOF

Create entity	<token> hex string <id> 24 char hex <type> string	curl -sS -X POST https://yggio.sifis-home.eu/ngsi/v2/entities -H "Fiware-UserToken: <token>" -H 'Content-Type: application/json' -d @- <<EOF {"id":"<id>","type":<type>,"name":{"value":"myNewRoom","type":"String"}, "temperature":{"value":23,"type":"Float"},"humidity":{"value":45,"type":"Float"}} EOF

Update values of existing entity	<token> hex string <id>	curl -sS -X POST https://yggio.sifis-home.eu/ngsi/v2/entities/<id>/attrs -H "Fiware-UserToken: <token>" -H 'Content-Type: application/json' -d @- <<EOF {"temperature":{"value":28,"type":"Float"},"humidity":{"value":48,"type":"Float"},"lux":{"value":1245,"type":"Float"}} EOF
Get entities of certain type	<token>	curl https://yggio.sifis-home.eu/ngsi/v2/entities?type=<type> -H "Fiware-UserToken: <token>"
Get entities that fulfill "q query"	<token> <root key> <key> <value>	curl "https://yggio.sifis-home.eu/ngsi/v2/entities?type=Device&q=<root key>.<key><condition>%27<value>%27" -H "Fiware-UserToken: <token>" -H "fiware-service: yggio" -H "fiware-servicepath: /"

Create NGSI subscription	<token>
	<id>
	<url>
	<type>
		curl -sS -X POST https://yggio.sifis-home.eu/ngsi/v2/subscriptions -H "Fiware-UserToken: <token>" -H 'Content-Type: application/json' -d @- <<EOF
		{
		"description": "Initiate NSGI subscription",
		"subject": {
		"entities": [
		{
		"id": <id>,
		"type": <type>
		}]},
		"notification": {
		"http": {
		"url": <url>
		}}}
		EOF

List NGSI subscriptions	<token>	curl https://yggio.sifis-home.eu/ngsi/v2/subscriptions -H "Fiware-UserToken: <token>"
Delete NGSI subscription	<id> <token>	curl -sS -X DELETE https://yggio.sifis-home.eu/ngsi/v2/subscriptions/<id> -H "Fiware-UserToken: <token>"

A design aspect of SIFIS-Home is that the API Gateway that implements RATATOSK must be able to send a command through a firewall to the DHT based network inside a house and then execute some command, like turning on a lamp. The natural solution to use NGSI subscriptions will unfortunately not work, since subscription requests are required to be IP addressable and will get blocked by the smart home firewall. The solution we identified was to develop a MQTT to DHT bridge, i.e., the devices and analytics in the smart home will both publish events and subscribe to RATATOSK events not via the API and NGSI subscriptions, but rather via an MQTT broker that integrates with Ratatosk. The Ratatosk events will then be triggered either by the user via the UI or by other SIFIS-Home devices.

5.2 *WebOfThings Smart Devices*

The WebOfThings standards focus on the key concept of enabling the devices to self-describe. The Thing Description is the cornerstone of it.
Within SIFIS-Home, we augment the Thing Description with a custom ontology to deliver information regarding the risks involved in interacting with the device.
We contributed to the Webthings.io implementations with several changes to validate our model, and we are working on a full refactor our initial proof of concept to provide a virtualized and Raspberry-Pi-simulated implementation of a Lamp device, an Oven device, a Cooker device and a Fridge device to simulate a kitchen.

A minimalistic WoT-Discovery will bridge the WoT-Things with the FIWARE broker.

5.3 *User Interface*

The user interface is the main interaction element between the SIFIS-Home framework and the tenants and other users (e.g., administrator, maintainer…). The user interface is used to configure user preferences, interact with GUI capable applications, install and remove applications, as well as set-up usage, safety and security policies.

Since the UI should be reachable both from inside the SIFIS-Home network and externally, the web parts of the UI will be placed on the API Gateway and thus run on top the RATATOSK FIWARE Context Broker complemented with additional APIs to support all the functionality required in SIFIS-Home. From a test bed point of view, we have leveraged existing functionality in Sensative Yggio to be able to implement and get the web UI working well within the scope of the project.

The actual UI for SIFIS-Home Smart Devices is still under design phase and will be specific for the device type in question.

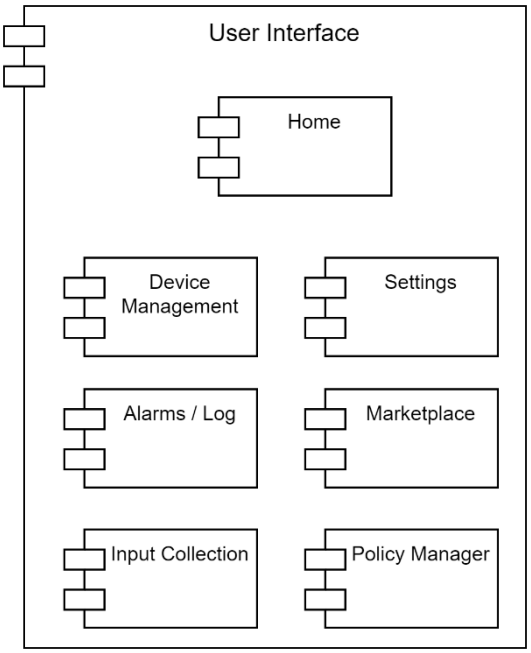


Figure 8: Components of the User Interface module as defined in deliverable D1.3

The diagram in Figure 8 shows the structure of the User Interface component. The User Interface module includes the following components that are further elaborated in six components.

- **Create Home:** This component makes it possible to perform a sequence of events triggered by the end user towards activating a first SIFIS-Home smart device in their smart home. The main account for the smart home will be created on the API gateway together with the standard users and their access rights, while the first device user used to initiate the registration will also be onboarded. From a test bed point of view, we will manually initiate the ”Create Home” activity and user registration from the UI.

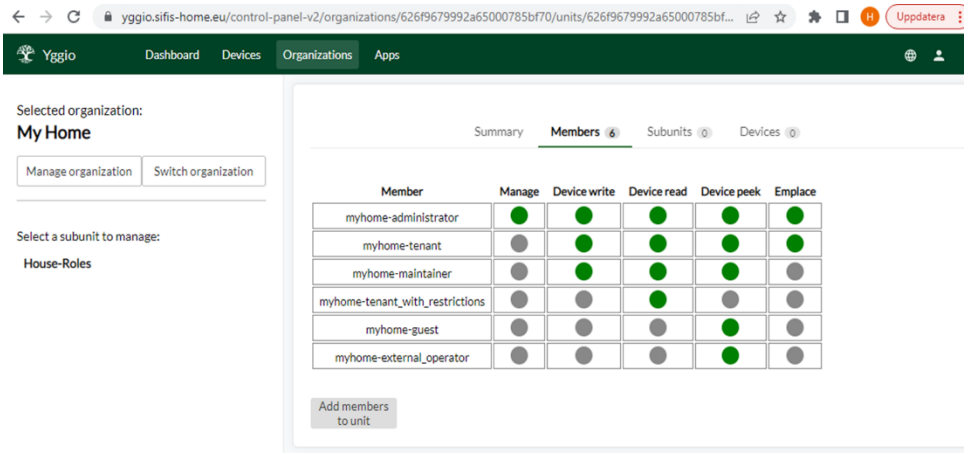


Figure 9: The home main account together with standard users were created.

- **Home:** This is the main component of the User Interface and is used to launch other applications installed in the system for a smart home. Our plan is also to add a simple dashboard that shows some key metric of the system like number of devices and installed third party applications.

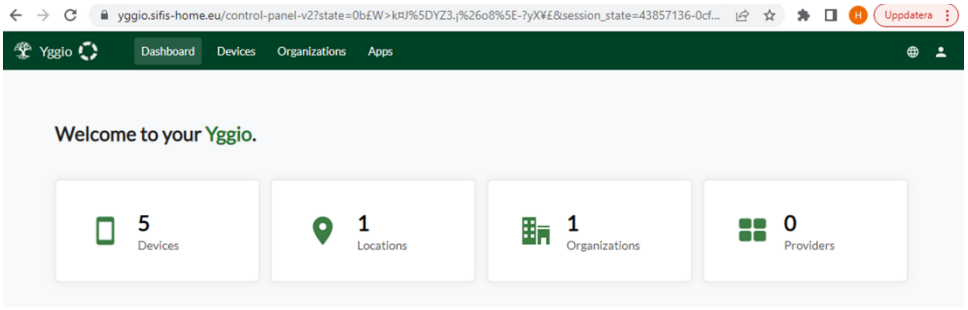


Figure 10: Home screen with a simple dashboard

- **Device management:** This component enables the configuration of the devices in the SIFIS-Home network. This is a core component in the system that is managing onboarding, configuration, displaying of device status and other functionalities related to the devices added to the smart home system. Depending on what role the logged in user has, different activities like read or write data will be allowed.

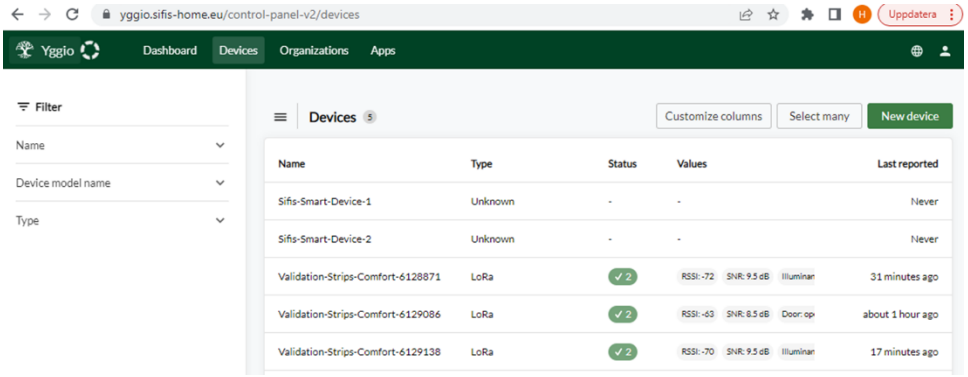


Figure 11: Device Manager with a list of devices

- **Settings:** This component provides user interfaces for the configuration of the SIFIS-Home infrastructure and most items in the API Gateway, like devices, analytics, etc, in the SIFIS

Home network will from a SW architectural point of view be represented by FIWARE entities in the Ratatosk Context broker. . Then this component provides a viewer and editor of applicable values related to FIWARE entities.

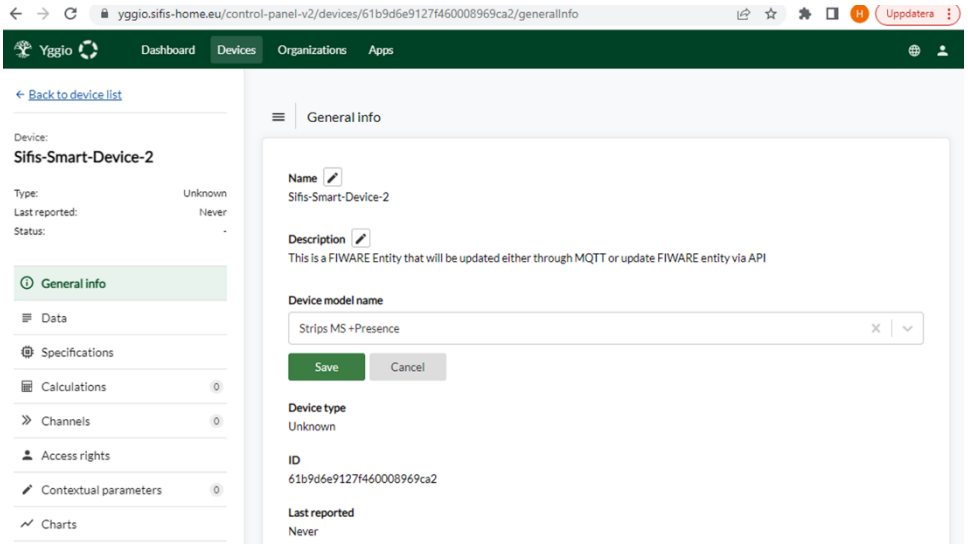


Figure 12: View status and edit settings for a SIFIS Home Smart Device

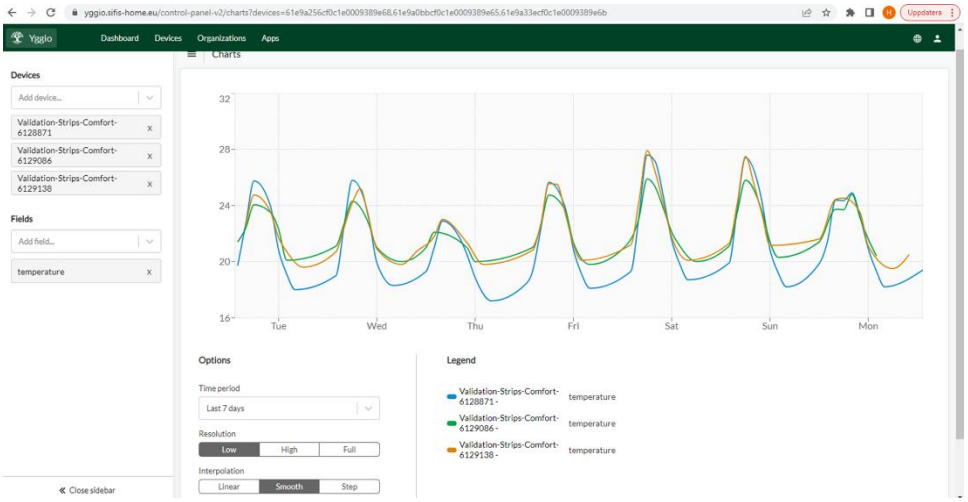


Figure 13: Compare time series data of 3 devices used for validation

- **Alarms/Log:** This component provides features to show alarms to the user and to gather logs of the functioning of the SIFIS-Home infrastructure. At the time of writing, this component is still under architectural definition and will be further defined in deliverable D1.4. Its implementation is not started yet.
- **Marketplace:** This component provides graphical user interfaces through which the user can visualize available applications to be installed on the SIFIS-Home system, and then download, install and update them if new versions are available. This component is still under development and a first version based on only a front end is already available.

Figure 14: UI how an application is visualized to the user.

5.4 *GitHub*

The code used in WP5 is slowly being uploaded to GitHub and reviewed by the partners as part of the WP2 activities. The SIFIS-generate tool is made available to streamline the process of adding GitHub Actions to the projects to have a working Continuous Integration.

sifis-generate

This tool generates either a new project for some build systems or configuration files for some Continuous Integration with the use of templates.

Templates define the layout for a project and allow developers to insert data at runtime.

Each template contains all files necessary to build a project with a build system, in addition to Continuous Integration and Docker files used to run tests and implement further checks.

Supported build systems

- ☑ meson
- ☑ poetry
- ☑ maven

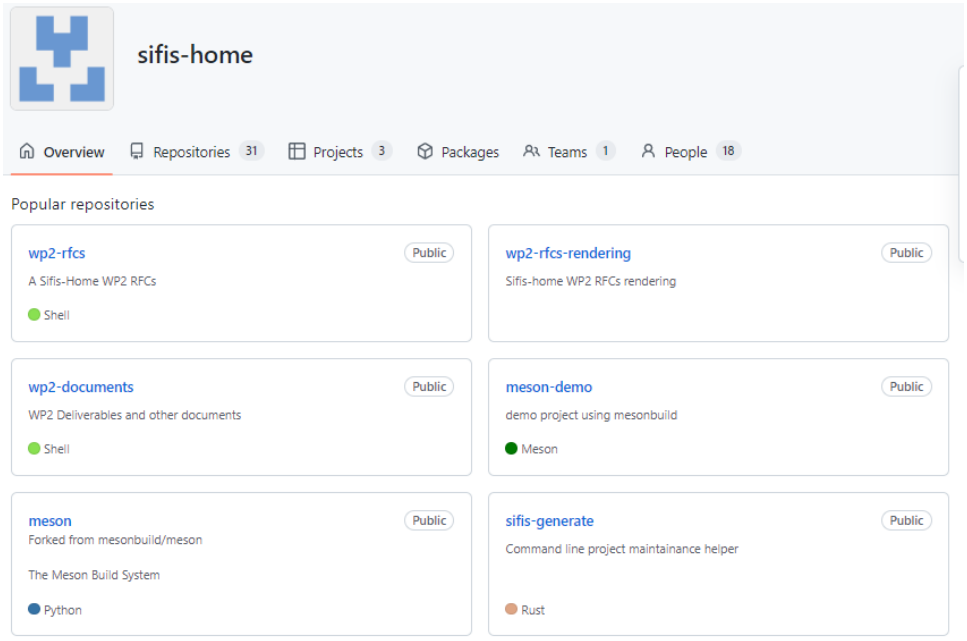
Build systems CI files

- ☑ cargo
- ☑ yarn

A checklist had been provided to the parners to make them aware of the status of their code and self-assess what is missing.

Code upload checklist

- ☐ Does the project have a `README.md` ?
- ☐ Does the project have a `LICENSE` file?
  - If possible use either MIT or other compatible licenses.
- ☐ Does the project have a GitHub Action?
  - ☐ Does it upload the code coverage somewhere?
  - ☐ Does it manage the release of the code? (Continuous Delivery)
- ☐ Does it have a DockerFile or does it need it?



The FIWARE Context Broker RATATOSK is available as open source and is getting continuously updated. It is the core of the SIFIS-Home API Gateway and handles device management as well as identity and access management with help of other modules.

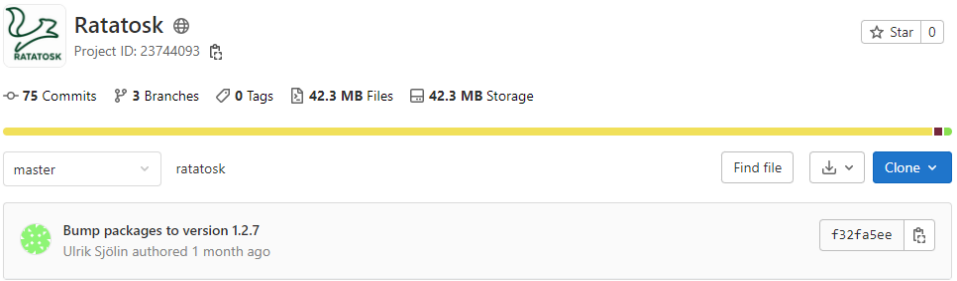


Figure 15: The Ratatosk FIWARE Context Broker is available as open source

Partners are in the process of uploading their code to the SIFIS-Home GitHub. We expect that, in the next few months, a lot of new code will get published and later maintained/updated.

6 Validation of implementation and of the test bed

This section reports the set of use cases of the SIFIS-Home framework that we used in order to design

and verify the test bed. In the previous section, it was stated that it should be possible to verify and validate the NFRs – Non-Functional Requirements – as defined in deliverables D1.2 and D2.1 by using the test bed.

Independent of the programming language used to develop the different components, their integration and execution should be feasible, as well as the continuous integration and continuous deployment of related new code uploaded to the SIFIS-Home GitHub. Once the system is up and running, it is required that the communication flow between all devices (e.g., from Smart Device to Smart Device, from Smart Device to API gateway, from API gateway to all devices, etc.) work reliably. Also, accesses to services and resources have to be secured and verified in the light of access policies to enforce.

Req. ID	Req. description	FR	Priority	Validation strategy
PE-01	The user authentication shall happen in less than 2s.	F-02 F-03	Critical	UI performance test
PE-02	The user recognition (identification/ biometric-based) shall happen in less than 5s.	F-06	Critical	UI performance test
PE-03	Biometric-based authentication should be performed in less than 5 seconds.	F-03	Standard	Under consideration
PE-04	Activation of features based on user identity (biometric recognition) should be performed in less than 5 seconds.	F-04 F-05	Standard	UI performance test
PE-05	Recognition of the start of an interaction through voice command should be performed in less than 2 seconds.	F-06	Standard	Under consideration
PE-06	The interpretation of the voice commands provided by the user should be performed in less than 2 seconds.	F-07	Standard	Under consideration
PE-07	A command should be invoked within 5 seconds from the event that triggered its execution	F-08	Standard	UI performance test
PE-08	The maintainer must be able to access and watch a recording in less than one minute.	F-13	Standard	Under consideration
PE-09	If requested to, the SIFIS-Home system shall contact law enforcement or private surveillance services to receive assistance in less than 30 seconds.	F-14	Optional	Event performance test. Use test mobile as receiver.
PE-10	An abnormal (suspicious) behavior caused by malware shall be identified and notified within 60 seconds	F-19	Optional	Event performance test.
PE-11	The user should be informed of the presence of malware no later than 5 seconds after the malware is recognized.	F-20	Standard	Event performance test.
PE-12	Self-healing algorithms should be started in less than 60 seconds if available when malware is recognized.	F-21	Critical	Event performance test.
PE-13	The registration of a new device should be completed in less than 30 seconds.	F-23	Standard	Event performance test.
PE-14	The list of registered devices shall be shown by the SIFIS-Home system in less than 30 seconds.	F-24	Standard	UI performance test
PE-15	The de-registration of a device should be completed in less than 30 seconds.	F-25	Standard	UI performance test
PE-16	The correct configuration changes should be	F-26	Critical	

	propagated successfully in less than 30 seconds.			UI performance test
PE-17	The current configuration of a device should be retrieved in less than 10 seconds.	F-26	Standard	UI performance test
PE-18	The marketplace should be accessible in less than 60 seconds.	F-28	Standard	UI performance test
PE-19	The configuration of policies for groups of users should be applied and enforced in less than 60 seconds.	F-32	Critical	UI performance test
PE-20	The configuration of policies for groups of devices should be applied and enforced in less than 60 seconds.	F-33	Critical	Event performance test
PE-21	The list of policies should be retrieved in less than 30 seconds.	F-30	Standard	Event performance test
PE-22	The configuration of profiles should be applied and enforced in less than 60 seconds.	F-37	Critical	Event performance test
PE-23	The change of current profile should be performed in less than 60 seconds.	F-38	Critical	Event performance test
PE-24	The statistics about usage and behavior of devices should be presented to the administrator in less than 30 seconds.	F-41	Standard	UI performance test
PE-25	The statistics about usage of profiles should be presented to the administrator in less than 30 seconds.	F-42	Standard	UI performance test
PE-26	Remote log-in should be performed in less than 60 seconds.	F-43	Critical	Event performance test
PE-27	In case of an incomplete or unsuccessful command execution, an error response should be sent within 5 seconds	F-08	Standard	Event performance test
PE-28	The used solutions for communication and system security shall be as much as possible lightweight to enforce in terms of performance and especially feasible also for resource-constrained devices.	All	Critical	By design
PE-29	The performance impact due to communication and system security shall not result in unacceptable impact on the user experience.	All	Critical	UI performance test
PE-30	The network infrastructure shall provide means also for one-to-many message delivery, e.g., over IP multicast.	F-47 F-48 F-49 F-50	Critical	By design
PE-31	It must be possible to have multiple security groups simultaneously active in the system.	F-47 F-48 F-49 F-50	Critical	By design
PE-32	When relevant, support shall be ensured for communication intermediaries performing, e.g., message forwarding and/or (transport-) protocol translation. This applies also in secure scenarios and in (secure) group communication scenarios.	All	Critical	By design

PE-33	When relevant, it shall be possible to enable one-to-many response messages, sent at once to multiple requesters. This applies also to secure communication scenarios and in presence of communication intermediaries.	All	Critical	By design
PE-34	When relevant and limited to read-only operations, it shall be possible to enable cache ability of response messages at communication intermediaries, also when protected end-to-end.	All	Critical	By design
PE-35	Devices should, if available, utilize low-power modes of operation to further mitigate the performance impact of ongoing (D)DoS attacks.	All	Standard	By design
PE-36	There should be a means to enable an optimized, combined establishment of a cryptographic secret with a first message protected with key material derived from that secret.	All	Standard	By design
PE-37	In case of an incomplete or unsuccessful configuration change, an error message should be returned within 5 seconds	F-26	Standard	UI performance test
RE-01	The system shall not fail more than once a week (on average).	All	Critical	Event performance test
RE-02	The system shall not take more than one day to be repaired (on average).	All	Critical	Event performance test
AV-01	The SIFIS-Home system services and devices shall be available 99% of the time	All	Critical	Event performance test
AV-02	The SIFIS-Home system shall ensure basic services availability in case of system failures.	All	Critical	Event performance test
AV-03	Support should be ensured for devices to dynamically react to (D)DoS attacks, by gradually adapting their availability. This includes relying on communication intermediaries for traffic offloading during intense (D)DoS attacks.	All	Standard	By design
AV-04	Devices under (D)DoS attacks should be able to continue providing a (best-effort) service to legitimate requests, i.e., by displaying a graceful degradation of quality of service.	All	Standard	Event performance test
US-01	The system shall be easy to use for users with no technical background	All	Critical	UI performance test
US-02	The SIFIS-Home system shall be autonomous and learn based on the users' habits, still according to defined privacy policies.	All	Critical	By design
US-03	The SIFIS-Home system shall consider special cases in its design, such as color blindness.	All	Optional	By design
US-04	The SIFIS-Home system shall preserve consistency among all devices, related databases and constraints.	All	Critical	By design
US-05	The SIFIS-Home hardware components should be easy to use for the elderly and users with no engineering background.	All	Optional	Not Verifiable
US-06	The SIFIS-Home system shall have an explorable interface.	All	Standard	By design

US-07	Proper and easy hardware installation should be considered.	All	Standard	By design
US-08	The image-based identification through biometrics in a room (interior) or in an open space (exterior), without obstacles or face covering elements, it should be performed by the system in a radius of at least 10 meters from the device.	F-01	Standard	Event performance test
US-09	An untrained user should be able to understand that an attack is ongoing in less than a minute from reading the SIFIS-Home alert or notification.	F-09 F-13	Critical	UI performance test
US-10	An untrained user should be able to recognize a software intrusion in less than one minute.	F-19 F-20	Critical	UI performance test
US-11	An untrained user should be able to perform the device registration procedure in less than 5 minutes.	F-23	Standard	UI performance test
US-12	An untrained user should be able to perform the device de-registration procedure in less than 5 minutes.	F-26	Standard	UI performance test
US-13	An untrained user should be able to perform the configuration of devices in less than 5 minutes.	F-26	Standard	UI performance test
US-14	An untrained user should be able to perform the installation of an application in less than 5 minutes.	F-28	Standard	UI performance test
US-15	An untrained user should be able to complete the configuration of policies for groups of users in less than 5 minutes.	F-32	Standard	UI performance test
US-16	An untrained user should be able to complete the configuration of policies for groups of devices in less than 5 minutes.	F-33	Standard	UI performance test
US-17	An untrained user should be able to complete the configuration of profiles in less than 5 minutes.	F-37	Standard	UI performance test
US-18	An untrained user should be able to perform a profile change in less than 30 seconds.	F-38	Standard	UI performance test
US-19	An untrained user should be able to access the statistics for visualizing and interpreting them in less than 5 minutes.	F-41	Standard	UI performance test
US-20	The Multi-Level Anomaly Detection system (MLADS) must monitor network traffic provided by several input sources and several locations.	F-15 F-16 F-17 F-18	C	By design
US-21	The workload of the devices should be available to the MLADS.	F-15 F-16 F-17 F-18	C	By design
US-22	The list of applications running on each device should be available to MLADS.	F-15 F-16 F-17 F-18	C	By design
US-23	Raw sensor data must be available to be analyzed by MLADS.	F-15 F-16 F-17	C	By design

		F-18		
US-24	Features from different devices should be aggregable directly or by means of pre-processing through specific analysis tools.	F-15 F-16 F-17 F-18	C	By design
US-25	When possible, a dataset should not be present as a whole on a single device for analysis.	All	S	By design
US-26	The presence of a GPU is needed to perform DL-based analysis.	F-15 F-16 F-17 F-18	S	By design
DE-01	Identification through biometrics should be performed correctly in more than 95% of cases.	F-01	Critical	Test on public datasets
DE-02	The start of an interaction command should be recognized properly and correctly in more than 99% of cases.	F-06	Critical	Event performance test
DE-03	The commands to execute should be recognized properly and correctly in more than 95% of cases.	F-06 F-07	Critical	Event performance test
DE-04	Record of intrusions must be available for a configurable time (default six months) after the recording.	F-10	Standard	By design
DE-05	Identity of the successfully recognized intruders must be available for a configurable time (default six months) after the recording.	F-12	Standard	By design
DE-06	Core functionalities should be replicated on multiple devices to avoid single points of failure.	F-21	Critical	By design
DE-07	The registration of a new device should be successful in at least 99% of the cases.	F-23	Critical	Event performance test
DE-08	The de-registration of a new device should be successful in at least 99% of the cases	F-25	Critical	Event performance test
DE-09	The configuration changes should be propagated successfully to the devices more than 99% of the time.	F-26	Critical	Event performance test
DE-10	The SIFIS-Home system should be able to restore the previous configurations if there are errors in applying configuration changes.	F-26	Standard	Event performance test
DE-11	The installation of the selected app should be completed successfully in at least 95% of cases.	F-28	Critical	Event performance test
DE-12	The application of policies should always be completed successfully.	F-31 F-34 F-33	Critical	Event performance test
DE-13	The configuration of profiles should be completed successfully in at least 99% of cases.	F-37	Critical	Event performance test
DE-14	The change of current profile should be completed successfully in at least 99% of cases.	F-38	Critical	Event performance test
DE-15	The statistics must be shown correctly in at least 99% of cases.	F-41	Critical	Event performance test
DE-16	Remote log-in for the configure should be successful in at least 99% cases.	F-43	Critical	Event performance test
DE-17	The SIFIS-Home system should be able to distribute the processing among multiple machines in separate places if required.	All	Critical	Event performance test

DE-18	The SIFIS-Home system is required to be fault tolerant, it should continue to operate, even if one or more of the nodes fail.	All	Critical	Event performance test
DE-19	The SIFIS-Home system is required to be scalable dynamically by adding or removing nodes according to demand.	All	Critical	Event performance test
TE-01	The SIFIS-Home system needs Java version 8 or higher to interact with the ontology.		Critical	By Design
TE-02	The process for getting and inserting information into the ontology will be through APIs provided via HTTP(S).		Critical	By Design
TE-03	The software for handling the ontology should be hosted on a high-availability server.		Critical	By Design
TE-04	Internet connectivity should be present.		Standard	By Design

Table 1. Finalized list of Non-Functional requirements for the SIFIS-Home framework

7 Conclusion

In this deliverable, we have presented how we analysed the key SIFIS-Home requirements and the consortium partners assets and expertise to design and setup the SIFIS-Home test bed built around the Panarea server provided by CNR.
During the integration of software components, we also found out that the original architecture from deliverable D1.3 was not feasible to instantiate and that there must be a clear separation between the API gateway and the SIFIS- Home application designed to be installed and to run on the Smart Devices in the smart home. A proposal of a revised architecture has been sent provided to WP1 for further studies.

As visible from the UX snapshots, the preliminary version of test bed with the SIFIS-Home software is up and running on the Panarea server. The partners are providing both code to the SIFIS-Home GitHub as well as integrating code into the test bed. This is done mostly in accordance with processes defined by WP2, even if some partners still have some way to go in order to be fully compliant. The test bed is still in a preliminary phase and there is ongoing work, for example with implementing and integrating network and security solutions as well as analytics.

From a validation point of view, deliverable D1.2 provided clear requirements and the validation strategy for most of the requirements has been set.

8 References

[WoT, 2020] Web Of Things (WoT) Architecture, W3C recommendation 9 April 2020, https://www.w3.org/TR/wot-architecture/

[FIWARE, 2021] What is FIWARE?, https://www.fiware.org/developers/

[YGGIO, 2021] Yggio DiMS, Digitalization infrastructure Management System, https://sensative.com/yggio/

Glossary

Acronym	Definition
AODV	Ad-hoc On-demand Distance Vector
CH	Context Handler
DHT	Distributed Hash Table
FR	Functional Requirements
NFR	Non-functional requirement
NSSD	Not So Smart Device
OS	Operative System
P2P	Peer to Peer
PAP	Policy Administration Point
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PIP	Policy Information Point
PTP	Policy Translation Point
SD	Smart Device
SIFIS-Home	Secure Interoperable Full Stack Internet of Things for Smart Home
UC	Use case
US	User story
XACML	eXtensible Access Control Markup Language