SOFIE - Secure Open Federation for Internet Everywhere

779984

DELIVERABLE D5.3

End-to-end Platform Validation

Project title	SOFIE – Secure Open Federation for Internet Everywhere
Contract Number	H2020-IOT-2017-3 – 779984
Duration	1.1.2018 – 31.12.2020
Date of preparation	21.7.2020
Author(s)	Dmitrij Lagutin (AALTO), Yki Kortesniemi (AALTO), Tommaso Bragatto (ASM), Francesca Santori (ASM), Francesco Bellesini (EMOT), Gian Marco Silieri (EMOT), Giuseppe Raveduto (ENG), Vincenzo Croce (ENG), Priit Anton (GT), Margus Haavala (GT), Mait Märdin (GT), Antonio Antonino (LMF), Antonis Gonos (OPT), Elias Kanakis (OPT), Asimakis Christodoulopoulos (OPT), David Mason (ROV), Ahsan Manzoor (ROV), Ioannis Oikonomidis (SYN)
Responsible person	Ioannis Oikonomidis (SYN), oikonomidis@synelixis.com
Target Dissemination Level	Public
Status of the Document	Completed
Version	1.00
Project website	https://www.sofie-iot.eu

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 779984.

Table of Contents

1. Introduction 8
1.1 Scope of this document 8
1.2 Structure of the deliverable 8
1.3 Relation to other activities and timeline overview 8

2. SOFIE reference architecture 10

3. Food Supply Chain Pilot 12
3.1 Pilot overview 12
3.2 Pilot platform architecture and services 12
3.3 Validation results 20
3.3.1 Future validation results 26

4. Decentralized Energy Data Exchange Pilot 27
4.1 Pilot overview 27
4.2 Pilot platform architecture and services 28
4.2.1 Introduction 28
4.2.2 Responsibilities of the FA 29
4.2.3 The FA Communication Protocol 30
4.2.3.1 Choosing the Source and Target DID 30
4.2.3.2 Proving the Control of a DID 30
4.2.3.3 Proving the Identity of a DID Owner 31
4.2.3.4 Proving Delegated Authorization 31
4.2.4 FA Internals and the Use of SOFIE Components 33
4.2.5 Proxy Private API 34
4.2.6 Pilot Deployment View 34
4.3 Validation results 35
4.3.1 Validation approaches 35
4.3.2 Validation results 38
4.3.3 Future validation results 40

5. Decentralized Energy Flexibility Marketplace Pilot 41
5.1 Pilot overview 41
5.2 Pilot platform architecture and services 41
5.3 Validation results 44
5.3.1 Future validation results 55

6. Mixed Reality Mobile Gaming Pilot 56
6.1 Pilot overview 56
6.2 Pilot platform architecture and services 58
6.2.1 High-level architecture 58
6.2.2 Flow of the game 59
6.2.3 APIs description 59
6.3 Validation results 61
6.3.1 Validation Results 61
6.3.2 Technical performance tests 66
6.3.2.1 Response Time 66
6.3.2.2 Throughput 68

Fixed Rate 68
Composite rate 68
6.3.2.3 Modeling Active Player Support 69
6.3.2.4 Beacon Detection Time 69
6.3.3 Future plans 70
7. SMAUG 72
7.1 Purpose and use case 72
7.1.1 Use case 72
7.1.1.1 Smart locker management 73
Smart locker registration 73
Request creation 74
Request decision 74
7.1.1.2 Smart locker discovery and access 74
Smart locker discovery 74
Smart locker purchase and access 75
7.2 Architecture 75
7.2.1 MP domain 76
7.2.2 SLO domain 77
7.2.3 SLR domain 77
7.3 SMAUG and SOFIE 77
7.3.1 Usage of SOFIE components 78
7.3.1.1 Marketplace 78
7.3.1.2 Interledger 78
7.3.1.3 Identity, Authentication and Authorisation (IAA) 79
7.3.1.4 Privacy and Data Sovereignty (PDS) 79
7.3.1.5 Provisioning and Discovery (P&D) 79
7.3.1.6 Semantic Representation (SR) 79
8. Cross pilot scenarios and testing plan 80
8.1 Cross pilot data exchange 80
8.1.1 Test cases 81
8.1.1.1 Latency overhead of the FA 81
8.1.1.2 Throughput of the FA 81
8.1.1.3 Integration Effort and Comparison to Current Situation 81
8.2 Cross pilot reward exchange 81
8.2.1 Test cases 82
8.2.1.1 Validation 82
8.2.1.2 Latency 82
9. Conclusions 83
10. References 84
11. Appendix I: Validation of SOFIE components 85

List of Figures

Figure 1: The SOFIE framework architecture 10
Figure 2: High-level pilot architecture 13
Figure 3: FSC Pilot platform Deployment view 14
Figure 4: FSC Pilot component diagram........ 15
Figure 5: API description 19
Figure 6: Steps in QR code creation........ 19
Figure 7: Basic building block of the DEDE pilot 27
Figure 8: Overview of the DEDE platform 29
Figure 9: A connection between two DIDs controlled by different FAs 30
Figure 10: Request to a service that requires proved attributes 31
Figure 11: Request to a service with delegated access rights 32
Figure 12: Internal structure of the FA 33
Figure 13: Deployment view of the DED platform pilot environment 35
Figure 14: Access rights delegation for SOFIE platform in Estfeed user portal 36
Figure 15: SOFIE Estfeed adapter service list 37
Figure 16: High-level architecture of the DEFM pilot........ 42
Figure 17: Message flow in the DEFM pilot 43
Figure 18: DEFM platform deployment view 44
Figure 19: DEFM Pilot Charging Stations Energy Consumption Trend - 1,5 years 50
Figure 20: DEFM Pilot Charging Stations Money Saved Trend - 1 month 50
Figure 21: DEFM Pilot Charging Stations CO2 Saved Trend - 1 month........ 51
Figure 22: Exchanged power between the ASM headquarter and the grid in the ex-ante and first scenario 53
Figure 23: Exchanged power between the ASM headquarter and the grid on 4th April in the ex-ante and first scenarios 53
Figure 24: Distribution of number of EV charging sessions rescheduled 54
Figure 25: Renewable consumed energy rate increase [%]........ 54
Figure 26: Scavenger Hunt game prototype. Starting, playing, and ending a hunt on a mobile client........ 56
Figure 27: Viewing and equipping items in Blockmoji. 57
Figure 28: The high-level architecture of our pilot. 58
Figure 29: Response time for Read requests 67
Figure 30: Response time for Write requests........ 67
Figure 31: There may be multiple instances of marketplaces in the global ecosystem. Suppliers and renters of smart lockers can choose any of them depending on their preferences and the policies implemented by each marketplace. 73
Figure 32: Smart locker owners can rent their smart lockers by publishing their availability on the SMAUG marketplace. On the other hand, potential smart locker renters can discover nearby available smart lockers and pay to purchase access. 73
Figure 33: SMAUG architecture........ 76
Figure 34: Interledger flow for a typical marketplace transaction 78
Figure 35: The setup of cross-pilot testing using the FA from the DEDE pilot........ 80
Figure 36: The setup of cross-pilot testing 82

List of Tables

Table 1: Timeline and main outcomes per reporting period for all SOFIE pilots 8
Table 2: FCP validation results 20
Table 3: Proxy Private API 34
Table 4: DEFM test case 1 44
Table 5: DEFM test case 2 47
Table 6: DEFM test case 3 48
Table 7: Ex-ante Scenario 51
Table 8: Comparison among scenarios 55
Table 9: Requirement validation 62
Table 10: Performance validation – fixed rate 68
Table 11: Performance validation – composite rate 68
Table 12: Beacon performance results. 70

List of Acronyms

AMI	Advanced Metering Infrastructure
AWS	Amazon Web Server
BoEU	Block of Energy Unit
BLE	Bluetooth Low Energy
BP	Business Platform
DEDE	Decentralized Energy Data Exchange
DEFM	Decentralized Energy Flexibility Marketplace
DER	Distributed Energy Resources
DID	Decentralized Identifiers
DLT	Distributed Ledger Technology
DR	Demand Response
DSO	Distribution System Operator
EV	Electric Vehicle
EVSE	Electric Vehicle Supply Equipment
FCP	Food Chain Pilot
FSC	Food Supply Chain
GSM	Global System for Mobile communication
GUI	Graphical User Interface
GW	Gateway
IAA	Identification, Authentication and Authorization
IoT	Internet of Things
ML/VL	Medium Voltage / Low Voltage
MPO	MarketPlace Owners
MRMG	Mixed Reality Mobile Gaming
NORM	Next generation Open Real time smart Meter
PoC	Proof of Concept
PoI	Point of Interest
PV	PhotoVoltaic
QR	Quick Response
RBAC	Role Based Access Control
RES	Renewable Energy Sources
RPF	Reverse Power Flow
SLO	Smart Locker Owners
SLR	Smart Locker Renters
SM	Supermarket

SSEV	Self-Sufficiency for EV
SWS	Supervisor Web Server
TPS	Transactions Per Second
TSO	Transmission System Operator
TR	Transportation
V2G	Vehicular to Grid
WH	Warehouse

1. Introduction

1.1 Scope of this document

This deliverable is a report about the system software architecture and end-to-end validation results for the four SOFIE pilots. An additional use case which serves as a reference implementation of all the components included in SOFIE's architecture and framework is presented. Finally, cross-pilot cases are described in this deliverable, aiming to combine the pilot solutions, highlighting the interoperability aspects of SOFIE. For each pilot, the platform architecture and services are described, including any updates from the previous initial validation phase. Also, the validation end-to-end results are presented, focusing on the users' perspective. The main objective has been to validate the integrated functionality of the SOFIE federation architecture and framework components from a user's point of view. SOFIE components' validation can be found in the document's appendix.

1.2 Structure of the deliverable

We first provide an overview of the SOFIE reference architecture. Then, an individual section is devoted to each of the four SOFIE pilots, containing an overview, the pilot platform architecture updates, and the end-to-end (integration) validation results. These sections are structured in the same way by using the following three subsections:

- Subsection X.1 presents an overview of the pilot, summarizing its application context and any updates from the previously reported version.
- Subsection X.2 reports and presents any updates on the architecture of the pilot platform. In this section, architecture view diagrams (e.g., Component diagram, Deployment diagram, High-level architecture diagram) are included.
- Subsection X.3 presents the integration, end-to-end validation results of the latest pilot platform versions.

Next, a reference implementation (namely SMAUG) that utilizes and demonstrates all SOFIE framework components is presented.

A section is then dedicated to the description of a cross-pilot case that will be realized in the context of the SOFIE pilots.

The final chapter concludes the deliverable.

1.3 Relation to other activities and timeline overview

The table below depicts the timeline and the main outcomes of all SOFIE pilots, mapping activities to deliverables, and giving an overview of what has been done so far and what is expected until the end of the project, pilot-wise.

Table 1: Timeline and main outcomes per reporting period for all SOFIE pilots

Period	Achievement	Reported
M1-M6	Definition of the baseline technology, scenarios, and use cases.	D5.1, v1.00
M6-M18	Definition of data management, KPIs and pilot test cases.	D5.1 v1.10

	Definition of pilot software architecture. Initial implementation of supplementary software components. Proof of concept validation of SOFIE federation architecture and components.	D5.2, v1.00
M18-M30	Updates on the implementation of the pilot specific software components. Description of cross-pilot scenarios and testing plan Integration and deployment of pilot software platform (first release). Installation of IoT devices and deployment of IoT platforms Unit/integration tests to verify the functionality of the platform services. Engagement of end users in the improvement of certain platform aspects. End-to-end verification of pilot requirements.	D5.3
M30-M36	Training of end users in using the platform's services. Final release of pilot platform, platform deployment in the production environment. On-site end-to-end validation of platform services, collection of data. Demonstration of cross-pilot scenarios. Overall pilot performance assessment, KPIs evaluation, competitive advantage. Lessons learned, and provision of replication guidelines.	D5.4

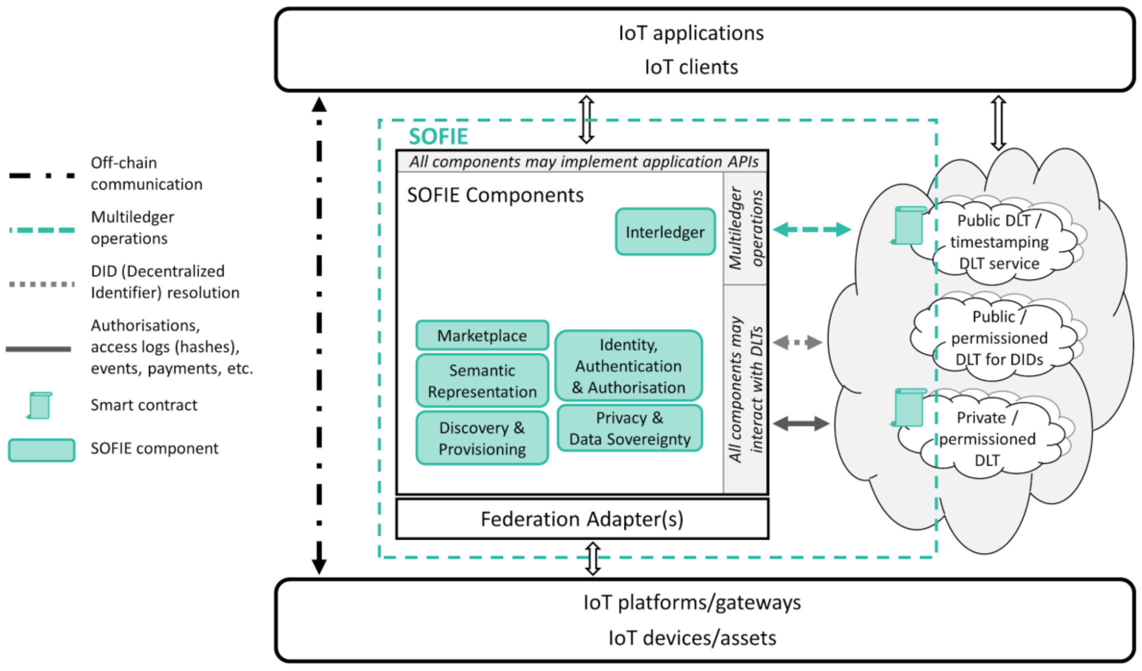
The pilot platforms are continuously integrating any updated versions of the SOFIE framework components. Therefore, we expect that the final version of each pilot platform will have integrated the latest version of any SOFIE components that are being used, and thus any validation will be performed on these versions.

2. SOFIE reference architecture

This section provides an overview of the SOFIE architecture. A full description of the SOFIE architecture is available in [D2.4] while the SOFIE framework is described in [D2.5]. The source code of the SOFIE framework has been released as open-source and is available at: https://github.com/SOFIE-project/Framework.

One of the most fundamental assumptions of SOFIE is that it must be able to support different types of IoT and ledger technologies without requiring changes to those technologies. This is due to the large installed base of existing technologies that do not allow for changes and the fact that different parties and consortia will continue to select their own IoT and distributed ledger technologies based on the different strengths of those technologies. By allowing the federation of such self-selected ledgers, SOFIE enables interoperability across the technology silos created by the manufacturers, who control those silos.

Figure 1 provides a functional overview of the SOFIE architecture. In particular, it depicts the six components that provide the SOFIE functionality (green boxes) and the Federation Adapter(s) used to interact with the IoT platforms and devices.



IoT applications
IoT clients
Off-chain communication
Multiledger operations
DID (Decentralized Identifier) resolution
Authorisations, access logs (hashes), events, payments, etc.
Smart contract
SOFIE component
SOFIE
All components may implement application APIs
SOFIE Components
Interledger
Marketplace
Semantic Representation
Discovery & Provisioning
Identity, Authentication & Authorisation
Privacy & Data Sovereignty
Multiledger operations
All components may interact with DLTs
Federation Adapter(s)
Public DLT / timestamping DLT service
Public / permissioned DLT for DIDs
Private / permissioned DLT
IoT platforms/gateways
IoT devices/assets

Figure 1: The SOFIE framework architecture

A key element of the SOFIE architecture is that it is a *framework architecture* that defines the types of functionalities provided by the components and adapter, but not an exhaustive list of supported functions. This is due to the fact that SOFIE is intended to support IoT federation in many application areas and it is infeasible to define a set of functions that would encompass all the needs (including future needs) of the different application areas. Instead, SOFIE defines types of functionalities and provides example implementations of each component and adapter in the SOFIE Framework. The provided examples are based on the pilots in the SOFIE project and they can be freely adapted and expanded to suit the needs of other applications.

The lowest level of the architecture contains *IoT assets* (or resources), that include, e.g., IoT sensors for sensing the physical environment, actuators for acting on the physical environment, and boxes with RFID tags that are used to transport products. IoT assets can be connected to or be integrated in actual devices. *IoT platforms* include platforms with data stores, where the measurements from sensors are collected and made available to third parties, as well as servers providing IoT services.

The *federation adapter(s)* are used to interface the IoT platforms with the SOFIE framework. This allows the IoT platforms to interact with SOFIE *without requiring any changes to the IoT platforms* themselves. Different scenarios and pilots can utilise different types of federation adapters, which expose only the required parts of the SOFIE functionality to the IoT platform.

Of the six components, the architecture emphasises the *interledger component* responsible for interconnecting the different types of DLTs, which can have quite different features and functionality. Public (or permissionless) DLTs offer wide-scale decentralised trust and immutability, but this necessitates a large network with many peers and/or a more demanding consensus mechanism, thereby incurring a higher overall computation cost that will lead to longer transaction confirmation times. On the other hand, permissioned or consortium DLTs have a lower, or even zero, transaction cost and low latency; however, trust is determined by the peers in the set of permissioned nodes that participate in the DLT's consensus mechanism. Moreover, the level of privacy afforded also differs: the transactions and data on public/permissionless blockchains are completely open to everyone, which is necessary to achieve wide-scale decentralised trust and transparency but forgoes any privacy. On the other hand, private/permissioned DLTs involve the collaboration of peers that belong to a specific permissioned set and can arrange for their records to be opaque to others (private), or public (but only allowing the permissioned set to contribute to the DLT). Thus, permissioned blockchains can support different levels of write and read access, which allows them to support different levels of privacy. DLTs can also differ in the functionality they provide: a DLT can focus, e.g., on cryptocurrency payments, recording of IoT events, access authorisation, or providing resolution of *Decentralised Identifiers* (DIDs). Utilising multiple ledgers that are interconnected through interledger functionality, instead of a single DLT, provides the flexibility to exploit the trade-offs. Finally, providing interledger mechanisms to interconnect different DLTs allows companies and consortiums to select private/permissioned distributed ledgers based on their requirements and constraints. Hence, interledger mechanisms can enhance interoperability across different IoT platforms that utilise different distributed ledger technologies.

The other SOFIE framework components are: *Identity, Authentication, and Authorisation (IAA)*, which provides identity management and supports multiple authentication and authorisation techniques; *Privacy and data sovereignty*, which provides mechanisms that enable data sharing in a controlled and privacy preserving way; *Semantic representation*, which provides tools for describing services, devices, and data in an interoperable way; *Marketplace*, which allows participants to trade resources by placing bids and offers in a secure, auditable, and decentralised way; and *Discovery & provisioning*, which provides functionality for the discovery and bootstrapping of services.

Finally, all the components can expose *application APIs*, which provide the interfaces for IoT clients and applications to interact with the SOFIE components. In Figure 1, the multiledger operations are positioned next to the Interledger component as it is mostly using that functionality, but any of the other components can also utilise multiledger operations when required. Also, the framework adapters and IoT applications can directly interact with the DLTs, but for simplification this is not shown in the figure. The figure also does not show the interactions between the components – these are described in more detail in deliverable D2.5.

The interactions with the DLTs that support DIDs can include DID document creation/modification, DID resolution, credential recording/revocation, etc. The format and information contained in the transactions is described in deliverable D2.5.

The architecture also illustrates the separation of data transfer and control message exchanges. Some IoT data can be transferred directly between the IoT platforms and IoT clients. Control messages related to authorisation logs, events, payments, etc. go through the SOFIE framework. IoT data or hashes of data can also be handled by the SOFIE framework.

3. Food Supply Chain Pilot

3.1 Pilot overview

The objective of the *Food Supply Chain* (FSC) pilot is to demonstrate the use of the SOFIE architecture and framework components in the product (grapes) supply chain and validate a provenance *Business Platform* (BP) that offers the following two main important services:

1. a traceability service used by the consumers to access the full history of grapes from the field to the supermarket shelf.
2. an audit service used by the supermarket company to verify the integrity of data which is collected as grapes are transferred over the supply chain as well as relevant business rules (driven by this data) which have been agreed with the suppliers.

The architectural design of the FSC software platform, the considered scenarios/use cases, the types and the roles of the involved actors, as well as the added value of the pilot BP into the supply chain business can be found in detail in Deliverable 5.2 (Initial Platform Validation). At this stage, the first integration of the FSC pilot platform has been completed and the first round of end-to-end validation results have been performed to verify the platform's operation with respect to the defined requirements. More details about the platform's implementation view and validation results are discussed in §3.2 and §3.3, respectively. Overall, the main achievements of the FSC pilot during the period M18-M30 and the activities which are scheduled for the last period of the project are summarized in Table 1 as part of the general pilot timeline.

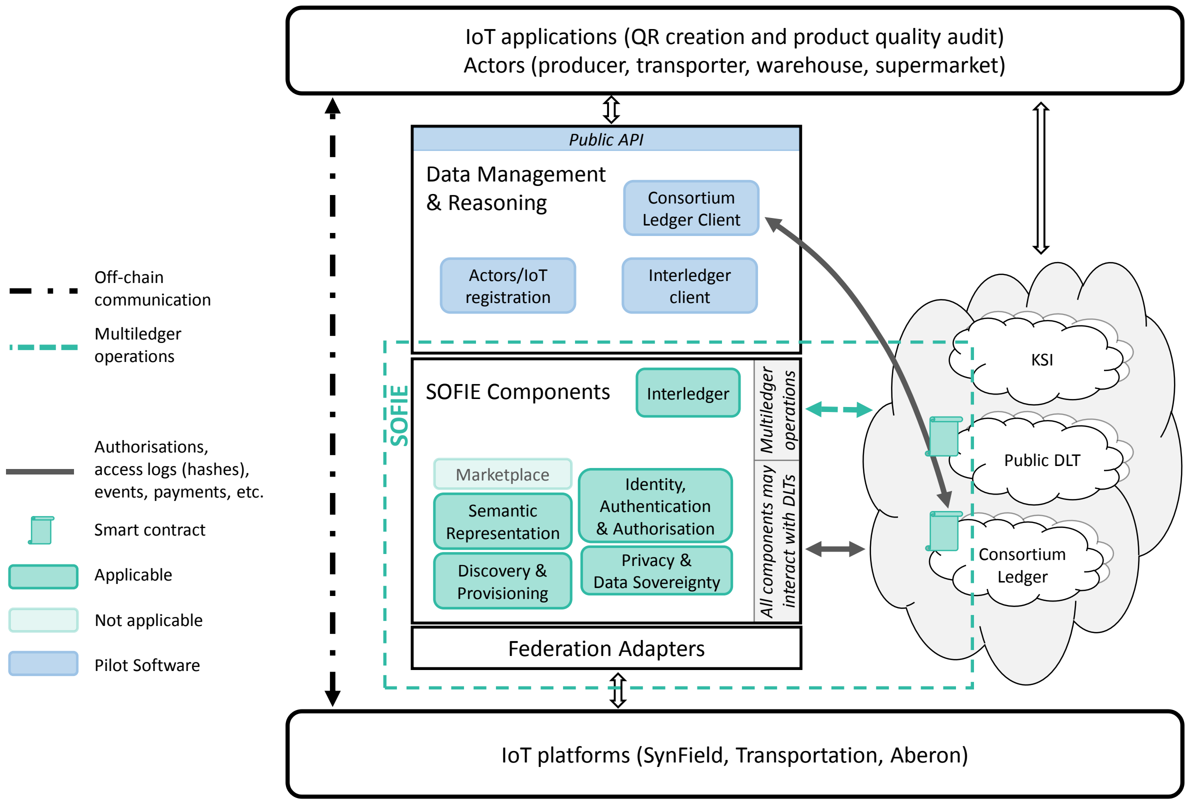
3.2 Pilot platform architecture and services

In this section we describe the updated pilot platform architecture. This version of the platform has been used for the validation results described in the next section.

Figure 2 depicts the platform's high-level architecture. The SOFIE components that have been utilized are shown in this architecture view and are listed below:

- Federation Adapters
- Identity, Authentication, Authorization
- Semantic Representation
- Discovery & Provisioning
- Interledger

This figure depicts pilot-specific software as well. This includes the Supervisor Web Server component, which offers a public API for the internal services provided by the Data management and the reasoning sub-component (i.e., Actors/IoT registration, Interledger client, Consortium Ledger Client). More details have been included in D5.2 (Initial Platform Validation)



IoT applications (QR creation and product quality audit)
Actors (producer, transporter, warehouse, supermarket)
Public API
Data Management & Reasoning
Consortium Ledger Client
Actors/IoT registration
Interledger client
SOFIE
SOFIE Components
Interledger
Marketplace
Semantic Representation
Identity, Authentication & Authorisation
Discovery & Provisioning
Privacy & Data Sovereignty
Multiledger operations
All components may interact with DLTs
Federation Adapters
KSI
Public DLT
Consortium Ledger
IoT platforms (SynField, Transportation, Aberon)
Off-chain communication
Multiledger operations
Authorisations, access logs (hashes), events, payments, etc.
Smart contract
Applicable
Not applicable
Pilot Software

Figure 2: High-level pilot architecture

In Figure 3, the Deployment view of the FSC Pilot platform is presented. Starting from the bottom layer, the Federation Adapter components are deployed on the IoT platform premises, since this is where the adaptation to the pilot needs to take place. The adaptation, as already mentioned in previous deliverables, does not require any modifications on the IoT platform side but it adds the functionality required by the IoT platform to connect to the Supervisor Web Server component on top, in a separate component (as a (micro)service).

The Supervisor Web Server component along with the SOFIE components are all combined to offer an API to the user's application, residing on the pilot's cloud that has been setup for this purpose. This cloud is also where the Consortium ledger resides. As expected, the Public ledger is external to the pilot's cloud. To connect between the two ledgers, SOFIE components are utilized (including clients that serve as connection libraries).

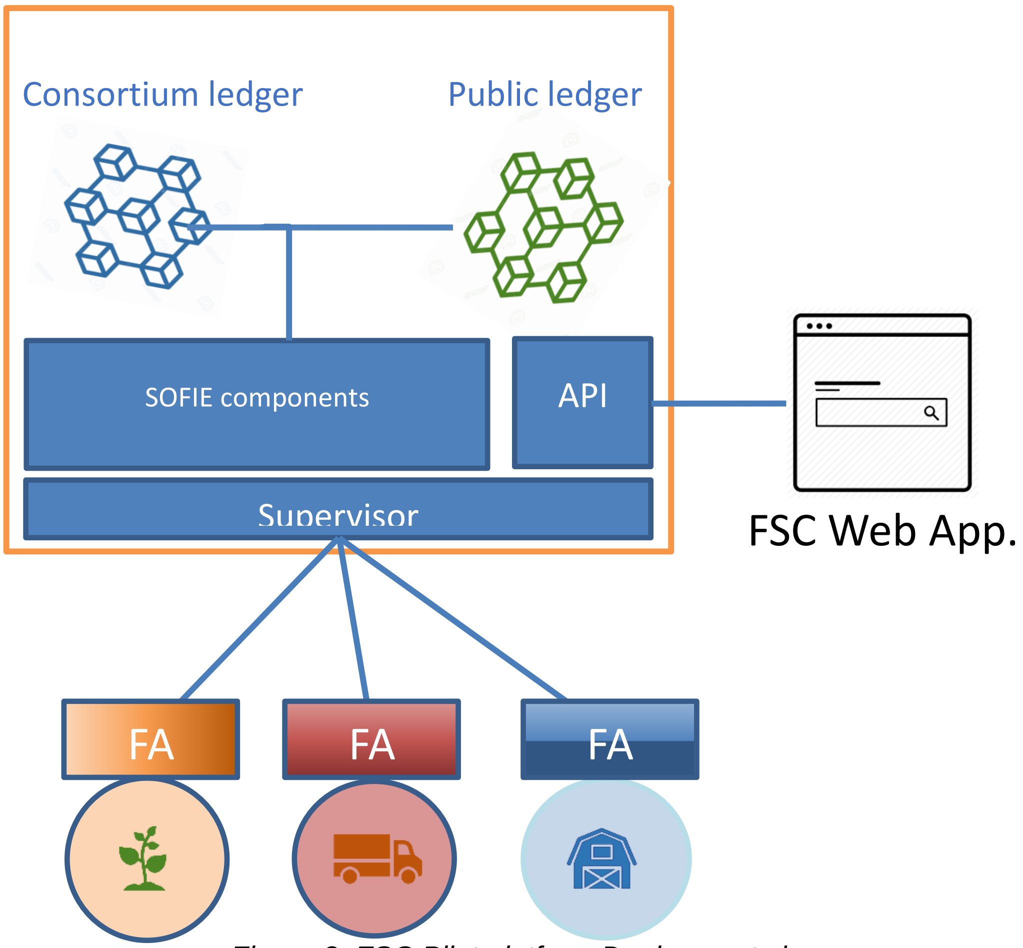


Figure 3: FSC Pilot platform Deployment view

The component diagram of the FSC Pilot platform is in line with the description of the architecture of the platform presented in D5.2 and is depicted in Figure 4.

Federation Adapters: the components that provide the adaptation functionality to the IoT platforms.

Supervisor Web Server: the component that provides the backend functionality of the platform.

Web application: the component that provides the user interface of the platform.

Ledgers (private and public): the components that represent the private and public ledgers used within the pilot platform.

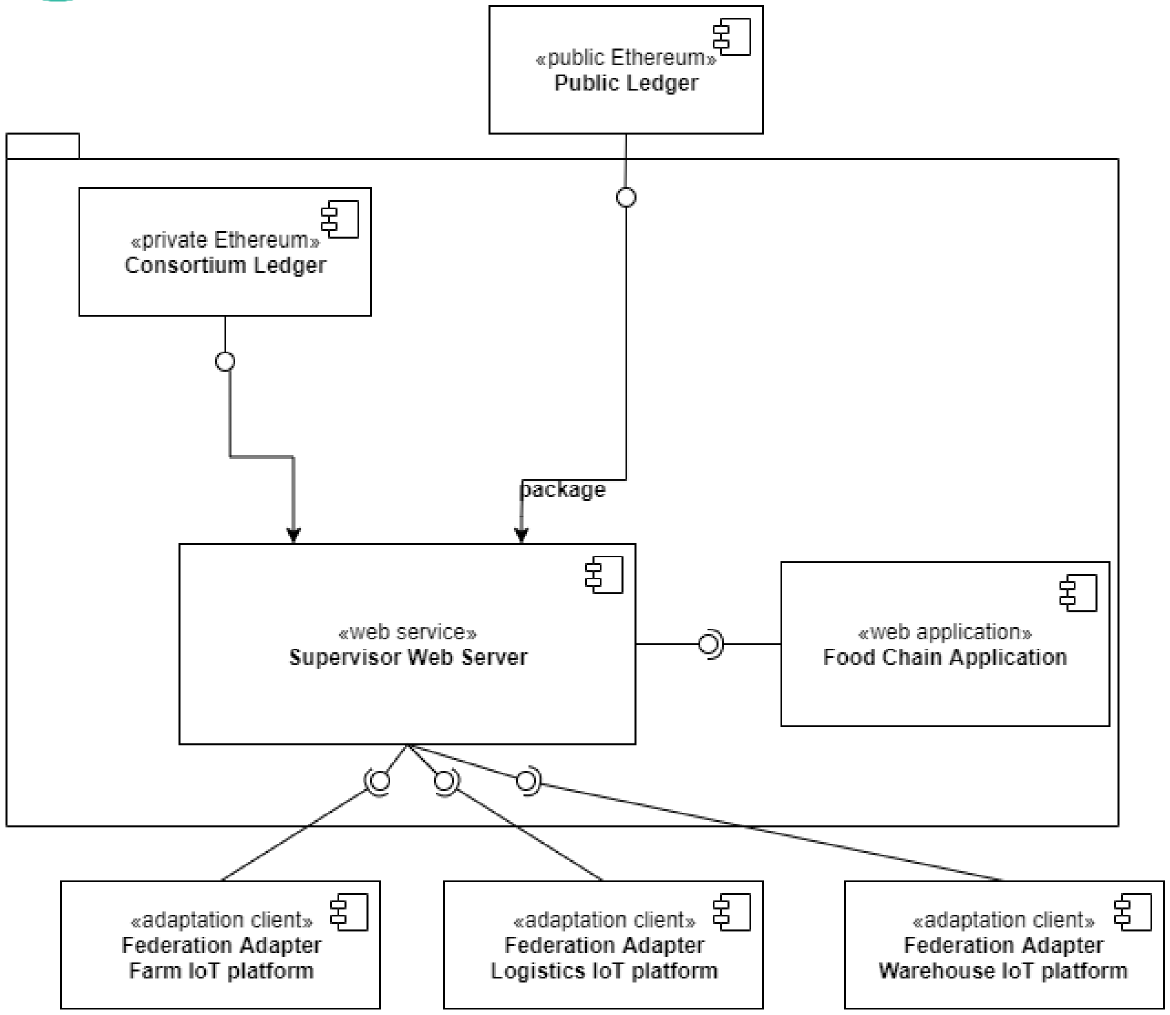
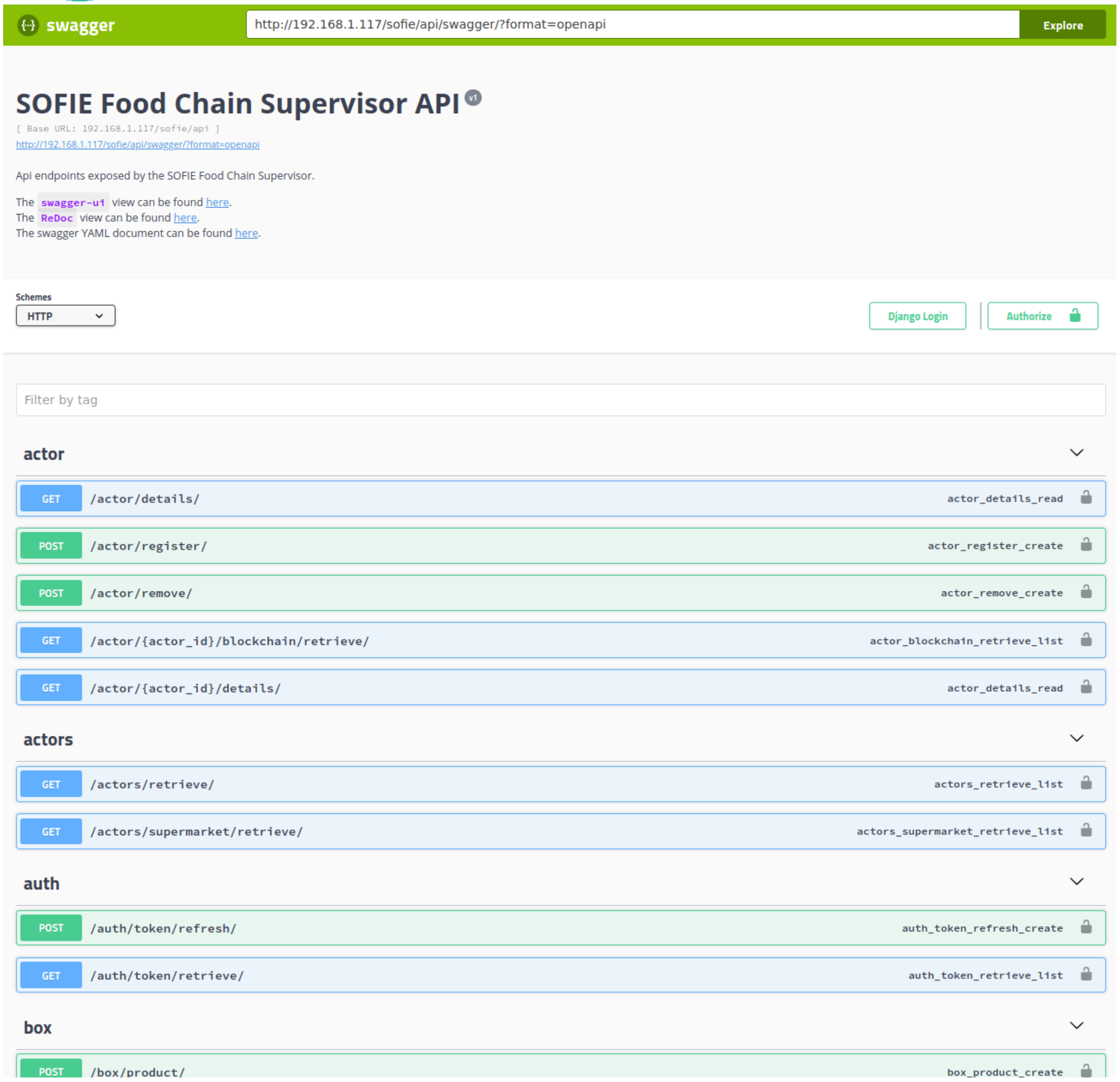


Figure 4: FSC Pilot component diagram

The components of the platform provide API endpoints for communication with other components of the platform. The Supervisor component that was implemented and plays a key role in the Food Chain pilot offers an API which the Federation Adapters and the Web application are using. The API endpoints of the Supervisor Web Server component are included in Figure 5.



SOFIE Food Chain Supervisor API v1

[Base URL: 192.168.1.117/sofie/api]
http://192.168.1.117/sofie/api/swagger/?format=openapi

Api endpoints exposed by the SOFIE Food Chain Supervisor.

The `swagger-ui` view can be found here.
The `ReDoc` view can be found here.
The swagger YAML document can be found here.

actor

```
GET   /actor/details/                          actor_details_read
POST  /actor/register/                         actor_register_create
POST  /actor/remove/                           actor_remove_create
GET   /actor/{actor_id}/blockchain/retrieve/   actor_blockchain_retrieve_list
GET   /actor/{actor_id}/details/               actor_details_read
```

actors

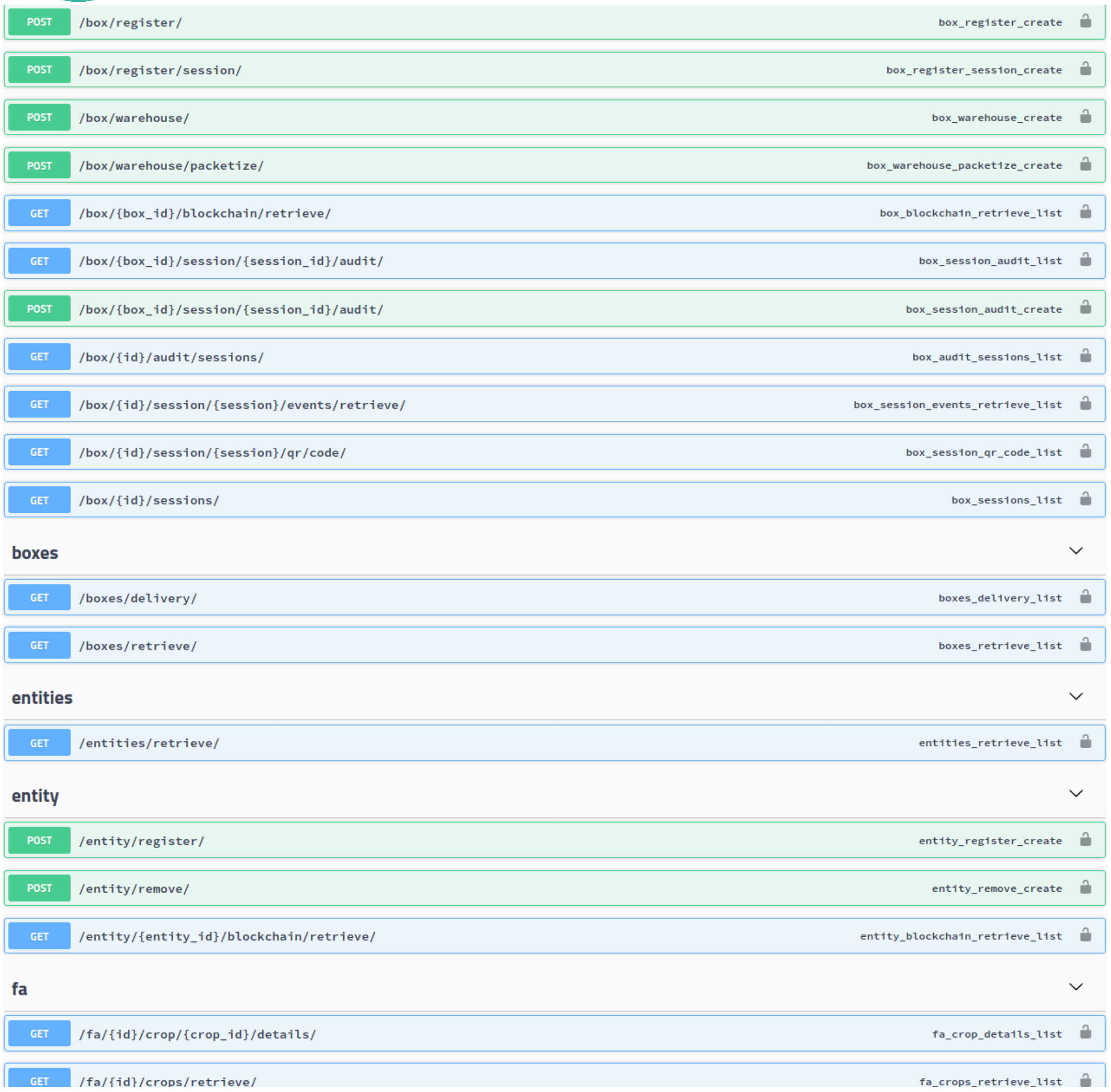
```
GET   /actors/retrieve/                        actors_retrieve_list
GET   /actors/supermarket/retrieve/            actors_supermarket_retrieve_list
```

auth

```
POST  /auth/token/refresh/                     auth_token_refresh_create
GET   /auth/token/retrieve/                    auth_token_retrieve_list
```

box

```
POST  /box/product/                            box_product_create
```



POST /box/register/ box_register_create
POST /box/register/session/ box_register_session_create
POST /box/warehouse/ box_warehouse_create
POST /box/warehouse/packetize/ box_warehouse_packetize_create
GET /box/{box_id}/blockchain/retrieve/ box_blockchain_retrieve_list
GET /box/{box_id}/session/{session_id}/audit/ box_session_audit_list
POST /box/{box_id}/session/{session_id}/audit/ box_session_audit_create
GET /box/{id}/audit/sessions/ box_audit_sessions_list
GET /box/{id}/session/{session}/events/retrieve/ box_session_events_retrieve_list
GET /box/{id}/session/{session}/qr/code/ box_session_qr_code_list
GET /box/{id}/sessions/ box_sessions_list

boxes

GET /boxes/delivery/ boxes_delivery_list
GET /boxes/retrieve/ boxes_retrieve_list

entities

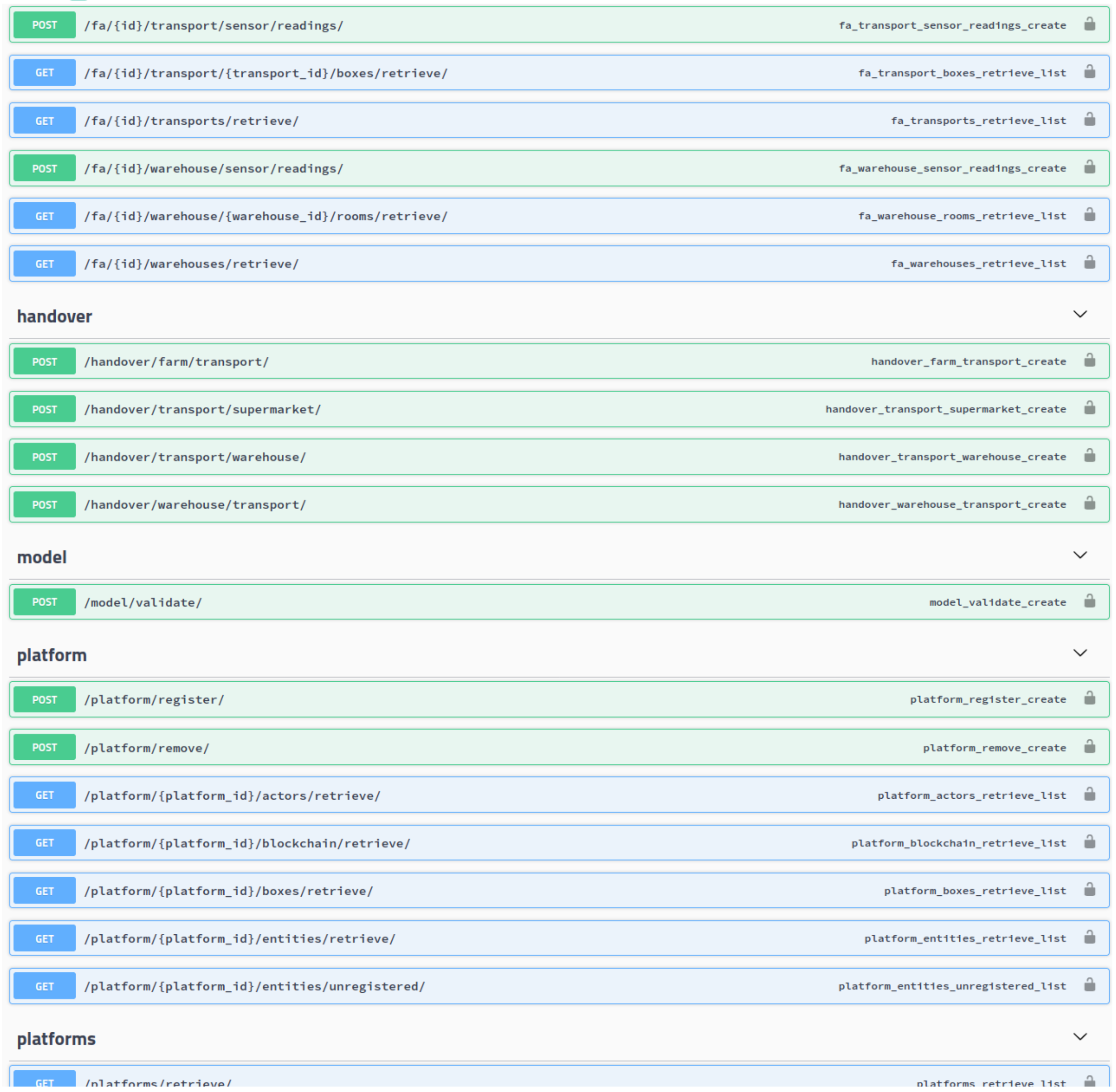
GET /entities/retrieve/ entities_retrieve_list

entity

POST /entity/register/ entity_register_create
POST /entity/remove/ entity_remove_create
GET /entity/{entity_id}/blockchain/retrieve/ entity_blockchain_retrieve_list

fa

GET /fa/{id}/crop/{crop_id}/details/ fa_crop_details_list
GET /fa/{id}/crops/retrieve/ fa_crops_retrieve_list



POST /fa/{id}/transport/sensor/readings/ fa_transport_sensor_readings_create

GET /fa/{id}/transport/{transport_id}/boxes/retrieve/ fa_transport_boxes_retrieve_list

GET /fa/{id}/transports/retrieve/ fa_transports_retrieve_list

POST /fa/{id}/warehouse/sensor/readings/ fa_warehouse_sensor_readings_create

GET /fa/{id}/warehouse/{warehouse_id}/rooms/retrieve/ fa_warehouse_rooms_retrieve_list

GET /fa/{id}/warehouses/retrieve/ fa_warehouses_retrieve_list

handover

POST /handover/farm/transport/ handover_farm_transport_create

POST /handover/transport/supermarket/ handover_transport_supermarket_create

POST /handover/transport/warehouse/ handover_transport_warehouse_create

POST /handover/warehouse/transport/ handover_warehouse_transport_create

model

POST /model/validate/ model_validate_create

platform

POST /platform/register/ platform_register_create

POST /platform/remove/ platform_remove_create

GET /platform/{platform_id}/actors/retrieve/ platform_actors_retrieve_list

GET /platform/{platform_id}/blockchain/retrieve/ platform_blockchain_retrieve_list

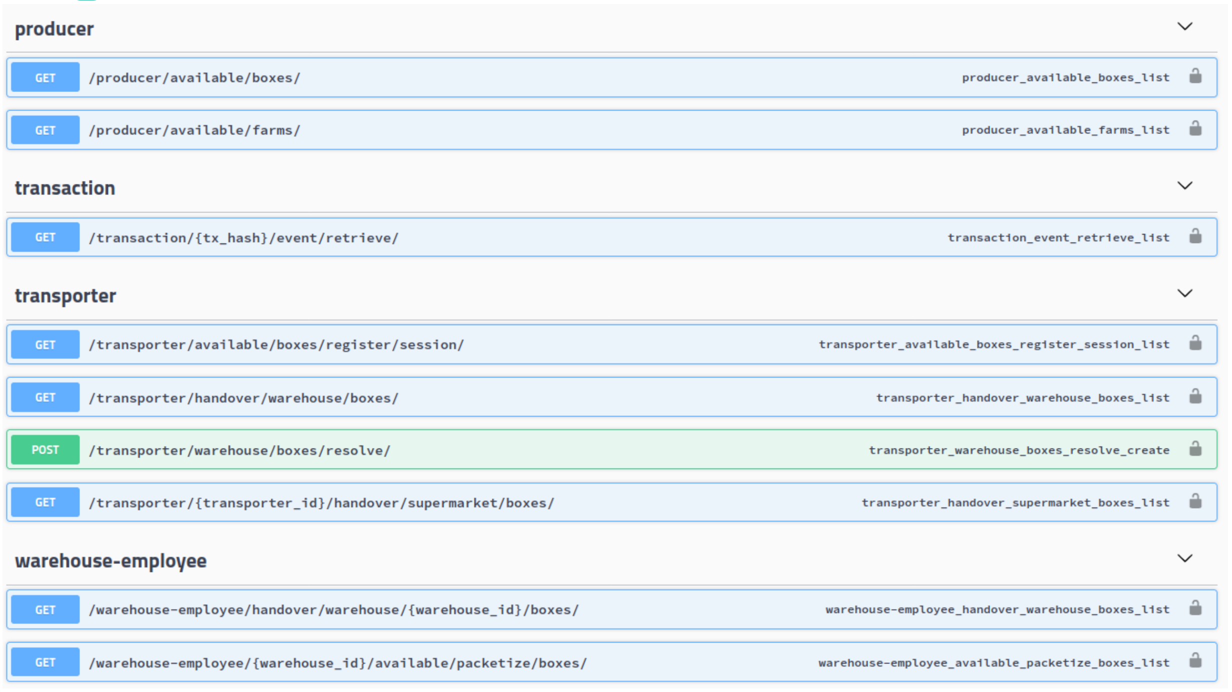
GET /platform/{platform_id}/boxes/retrieve/ platform_boxes_retrieve_list

GET /platform/{platform_id}/entities/retrieve/ platform_entities_retrieve_list

GET /platform/{platform_id}/entities/unregistered/ platform_entities_unregistered_list

platforms

GET /platforms/retrieve/ platforms_retrieve_list



producer

GET /producer/available/boxes/ producer_available_boxes_list

GET /producer/available/farms/ producer_available_farms_list

transaction

GET /transaction/{tx_hash}/event/retrieve/ transaction_event_retrieve_list

transporter

GET /transporter/available/boxes/register/session/ transporter_available_boxes_register_session_list

GET /transporter/handover/warehouse/boxes/ transporter_handover_warehouse_boxes_list

POST /transporter/warehouse/boxes/resolve/ transporter_warehouse_boxes_resolve_create

GET /transporter/{transporter_id}/handover/supermarket/boxes/ transporter_handover_supermarket_boxes_list

warehouse-employee

GET /warehouse-employee/handover/warehouse/{warehouse_id}/boxes/ warehouse-employee_handover_warehouse_boxes_list

GET /warehouse-employee/{warehouse_id}/available/packetize/boxes/ warehouse-employee_available_packetize_boxes_list

Figure 5: API description

In Figure 6 we show the sequence of steps that must be performed by the pilot platform to achieve the QR creation result. The flow presented follows the several actions taken by the actors (producer, transporters, employees) during the transportation of an asset from the field to the end destination (supermarket).

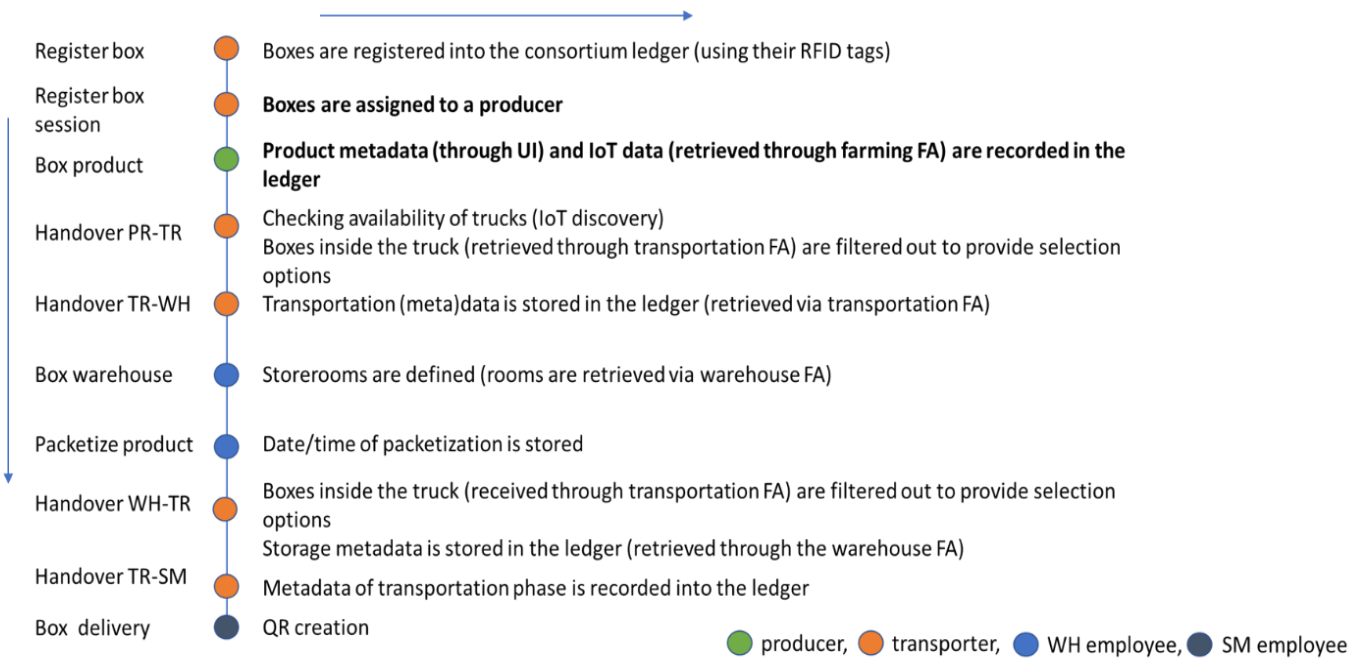

Register box
Boxes are registered into the consortium ledger (using their RFID tags)
Register box session
Boxes are assigned to a producer
Box product
Product metadata (through UI) and IoT data (retrieved through farming FA) are recorded in the ledger
Handover PR-TR
Checking availability of trucks (IoT discovery)
Boxes inside the truck (retrieved through transportation FA) are filtered out to provide selection options
Handover TR-WH
Transportation (meta)data is stored in the ledger (retrieved via transportation FA)
Box warehouse
Storerooms are defined (rooms are retrieved via warehouse FA)
Packetize product
Date/time of packetization is stored
Handover WH-TR
Boxes inside the truck (received through transportation FA) are filtered out to provide selection options
Storage metadata is stored in the ledger (retrieved through the warehouse FA)
Handover TR-SM
Metadata of transportation phase is recorded into the ledger
Box delivery
QR creation
producer, transporter, WH employee, SM employee

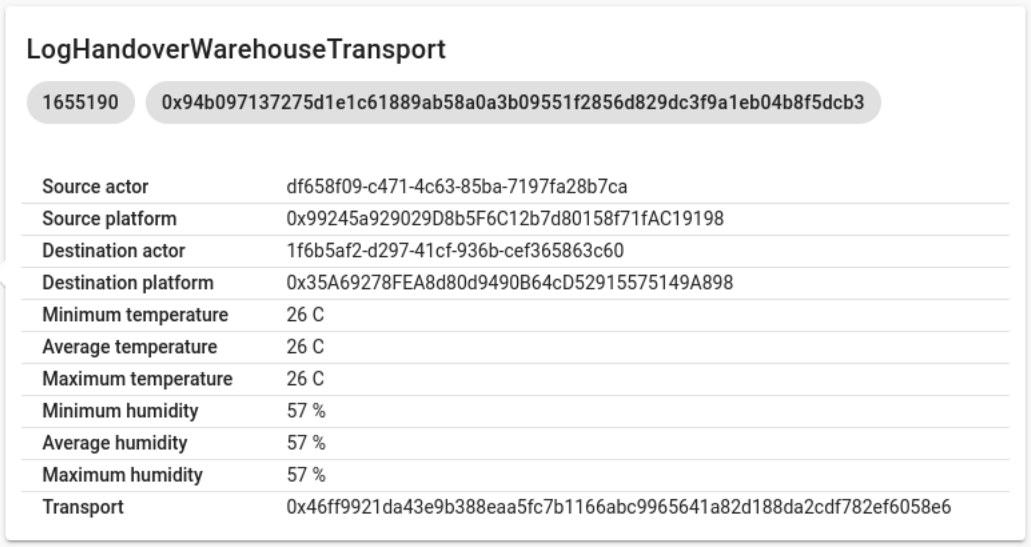
Figure 6: Steps in QR code creation

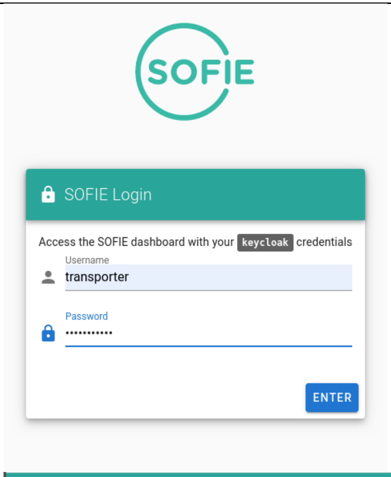
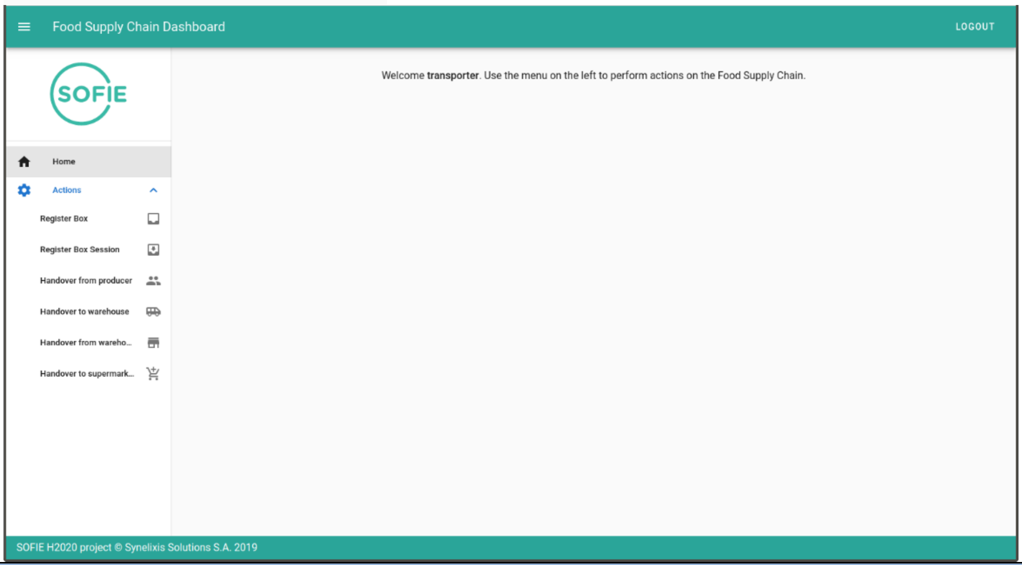
Following the overview of the architecture of the FSC Pilot platform, we present the validation of the implemented platform in the next section.

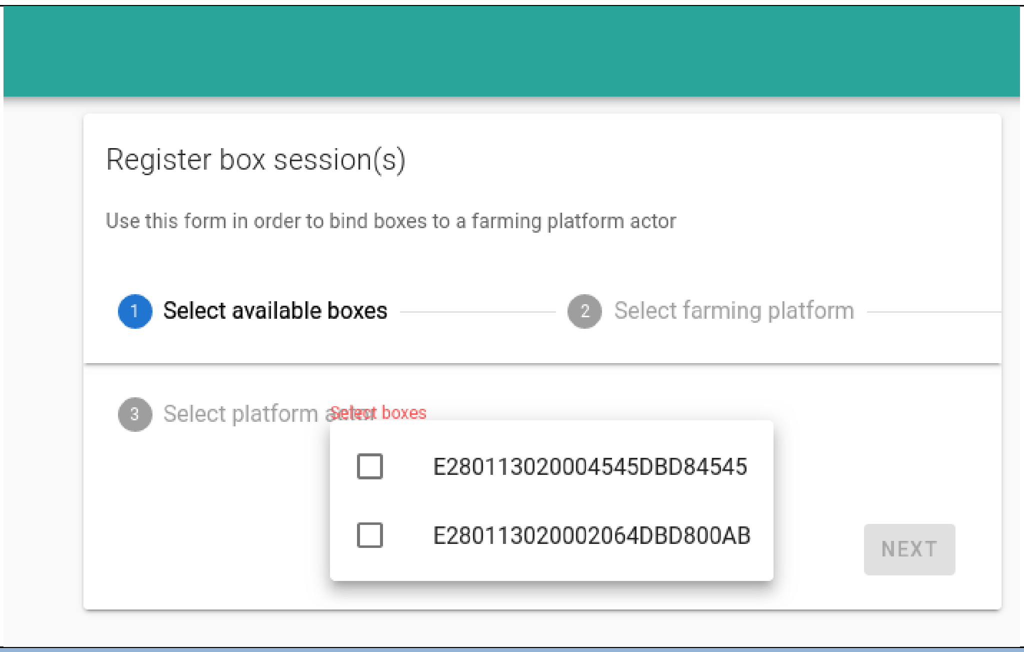
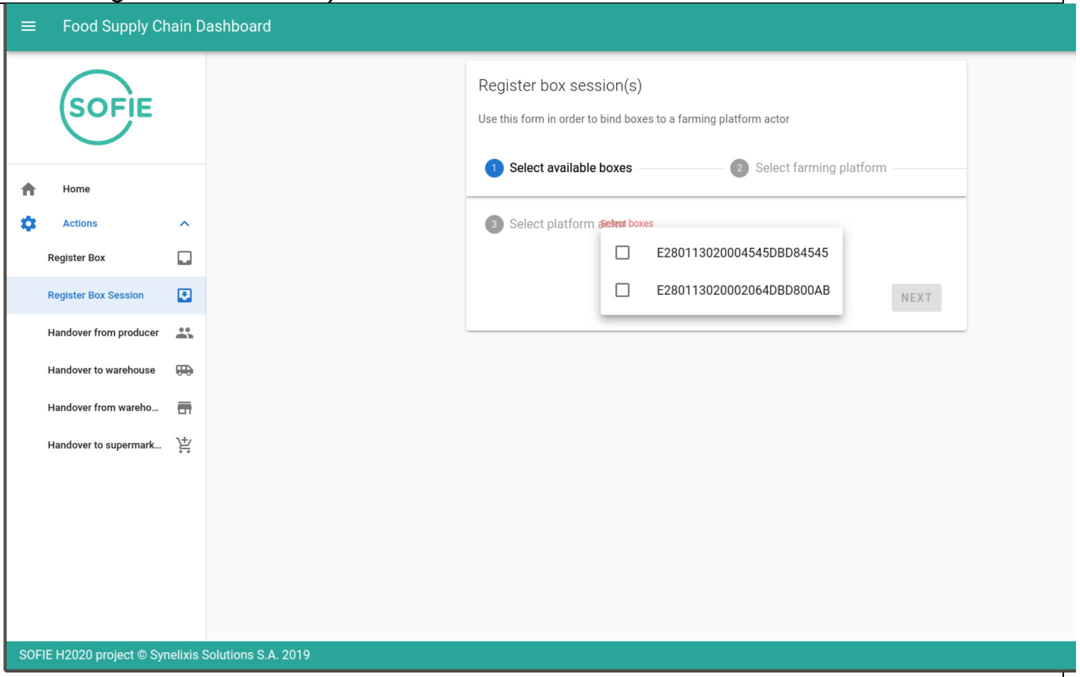
3.3 Validation results

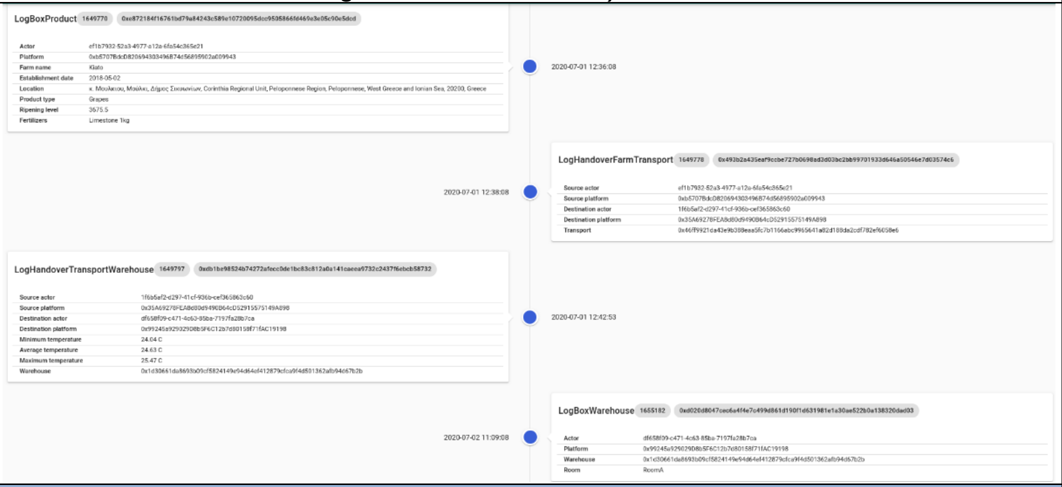
In the tables below we present the test cases defined in D5.1 (Baseline system and measurements) and the related results from the validation process:

Table 2: FCP validation results

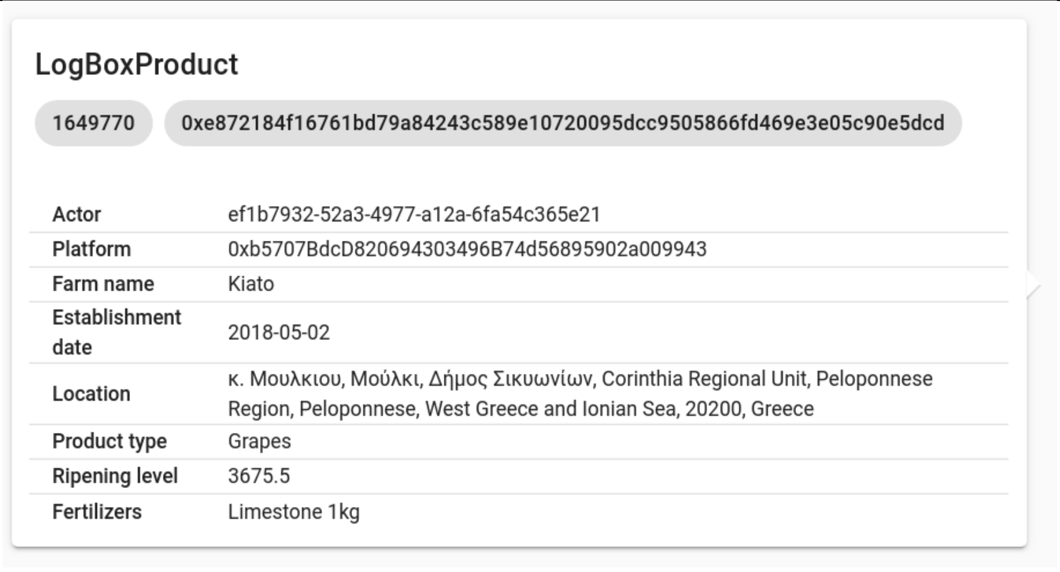
Test ID	**FSC_TC01**
Test description	Measurements from each deployed sensing device are collected by the corresponding IoT platform and they are properly stored in its database system.
Steps	1. Sensing devices are deployed on site and they are properly configured to communicate and send data to the corresponding IoT platform. 2. Collect data from a given period of time (e.g. few days) 3. Use IoT platform API to retrieve data from each integrated sensing devices within a specific time period.
Pass criteria	All relevant measurement values are properly retrieved.
Result	 LogHandoverWarehouseTransport 1655190 0x94b097137275d1e1c61889ab58a0a3b09551f2856d829dc3f9a1eb04b8f5dcb3 Source actor: df658f09-c471-4c63-85ba-7197fa28b7ca Source platform: 0x99245a929029D8b5F6C12b7d80158f71fAC19198 Destination actor: 1f6b5af2-d297-41cf-936b-cef365863c60 Destination platform: 0x35A69278FEA8d80d9490B64cD52915575149A898 Minimum temperature: 26 C Average temperature: 26 C Maximum temperature: 26 C Minimum humidity: 57 % Average humidity: 57 % Maximum humidity: 57 % Transport: 0x46ff9921da43e9b388eaa5fc7b1166abc9965641a82d188da2cdf782ef6058e6
Test ID	**FSC_TC02**
Test description	Each registered actor of any type (e.g. producer, transporter, warehouse, supermarket employee) can access and perform all the services provided by the FSC web application based on its role.
Steps	1. The actor initiates an HTTPS session to the FSC web application login page. 2. The HTTPS traffic is intercepted and the authorization is initiated by the Authentication Server (AS) of the SOFIE platform. The login page is sent to the actor. 3. The actor enters a username and password, which are sent to the AS of the SOFIE platform. 4. The OAuth2.0 server authenticates the actor and creates a unique token that is used to enable role-based access to FSC web application resources.
Pass criteria	Actor's access policy is activated. The actor is able to access the FSC web application's resources.

Result	 SOFIE Login Access the SOFIE dashboard with your keycloak credentials Username: transporter Password: ••••••••••••  Welcome **transporter**. Use the menu on the left to perform actions on the Food Supply Chain. SOFIE H2020 project © Synelixis Solutions S.A. 2019 Food Supply Chain Dashboard LOGOUT Home Actions Register Box Register Box Session Handover from producer Handover to warehouse Handover from wareho... Handover to supermark...
Test ID	**FSC_TC03**
Test description	Test that box reuse is possible (after its release) and that registration of a box with an ID that is already used by another box is impossible (box unique identifier).
Steps	1. An actor (transporter) enters its profile in the FSC web applications and activates register box action. 2. The actor provides as input to the action a box ID which has been already registered in the used DLT. 3. The actors provides as input the ID of a released box.
Pass criteria	Registration of a box with an already used ID (by another box) is prohibited. Reuse of a released box is possible.

Result	 Register box session(s) Use this form in order to bind boxes to a farming platform actor 1 Select available boxes 2 Select farming platform 3 Select platform actor Select boxes E280113020004545DBD84545 E280113020002064DBD800AB NEXT
Test ID	**FSC_TC04**
Test description	Presence of a group of boxes (RFID tags) is detected as they are placed/removed in/from the truck.
Steps	1. A number of boxes are placed inside the truck at a certain time instant. 2. Some of the boxes are removed from the truck at a certain time instant. 3. The boxes which were removed in step 2) are placed again inside the truck at a certain time instant but in a different location (inside the RFID range)
Pass criteria	The presence of all the boxes inside the truck is properly detected by the transportation IoT platform all times (taking also into account the delay in collecting measurements).
Result	 Food Supply Chain Dashboard Home Actions Register Box Register Box Session Handover from producer Handover to warehouse Handover from wareho... Handover to supermark... Register box session(s) Use this form in order to bind boxes to a farming platform actor 1 Select available boxes 2 Select farming platform 3 Select platform actor Select boxes E280113020004545DBD84545 E280113020002064DBD800AB NEXT SOFIE H2020 project © Synelixis Solutions S.A. 2019

Test ID	**FSC_TC05**
Test description	The SOFIE platform receives data from the transportation GW deployed in the truck i) as the vehicle moves, and ii) as the vehicle engine is turned off.
Steps	1. A group of boxes is placed inside the truck. 2. At a certain time the truck starts to move from site A to site B. 2. Before reaching its destination, the truck stops for a certain period of time and its engine is turned off for a certain period of time (a few minutes). 3. The engine is turned on and the truck moves to reach site B.
Pass criteria	The presence of the boxes inside the truck is continually detected (given the used time resolution in collecting data from the truck) from A to B.
Result	 LogBoxProduct 1649770 0xe872184f16761bd79a84243c589e10720095dcc9505866fd469e3e05c90e5dcd Actor: ef1b7932-52a3-4977-a12a-6fa54c365e21 Platform: 0xb5707BdcD820694303496B74d56895902a009943 Farm name: Kiato Establishment date: 2018-05-02 Location: κ. Μουλκιου, Μούλκι, Δήμος Σικυωνίων, Corinthia Regional Unit, Peloponnese Region, Peloponnese, West Greece and Ionian Sea, 20200, Greece Product type: Grapes Ripening level: 3675.5 Fertilizers: Limestone 1kg 2020-07-01 12:36:08 LogHandoverFarmTransport 1649778 0x493b2a435eaf9ccbe727b0698ad3d03bc2bb99701933d646a50546e7d03574c6 Source actor: ef1b7932-52a3-4977-a12a-6fa54c365e21 Source platform: 0xb5707BdcD820694303496B74d56895902a009943 Destination actor: 1f6b5af2-d297-41cf-936b-cef365863c60 Destination platform: 0x35A69278FEA8d80d9490B64cD52915575149A898 Transport: 0x46ff9921da43e9b388eea5fc7b1166abc9965641a82d188da2cdf782ef6058e6 2020-07-01 12:38:08 LogHandoverTransportWarehouse 1649797 0xdb1be98524b74272afecc0de1bc83c812a0a141caeea9732c2437f6ebcb58732 Source actor: 1f6b5af2-d297-41cf-936b-cef365863c60 Source platform: 0x35A69278FEA8d80d9490B64cD52915575149A898 Destination actor: df658f09-c471-4c63-85ba-7197fa28b7ca Destination platform: 0x99245a929029D8b5F6C12b7d80158f71fAC19198 Minimum temperature: 24.04 C Average temperature: 24.63 C Maximum temperature: 25.47 C Warehouse: 0x1d30661da8693b09cf5824149e94d64ef412879cfca9f4d501362afb94d67b2b 2020-07-01 12:42:53 LogBoxWarehouse 1655182 0xd020d8047cec6a4f4e7c499d861d190f1d631981e1a30ae522b0a138320dad03 Actor: df658f09-c471-4c63-85ba-7197fa28b7ca Platform: 0x99245a929029D8b5F6C12b7d80158f71fAC19198 Warehouse: 0x1d30661da8693b09cf5824149e94d64ef412879cfca9f4d501362afb94d67b2b Room: RoomA 2020-07-02 11:09:08
Test ID	**FSC_TC06**
Test description	Data and metadata provided by the actors through the FSC web application are recorded in DLTs. The payload of any transaction is verified.
Steps	1. An actor accesses the FSC web application and activates an action. 2. The actor performs any operation in the physical world requested to complete the action (e.g., boxes onboarding in the truck) and inputs the necessary (meta)data. 3. The actor completes the action (thus data is recorded in the used DLTs) 4. The actor accesses the logs of the performed operation and verifies that information recorded in the DLTs is correct.
Pass criteria	Data of the transaction which is stored in the DLTs matches the relative activity and metadata.

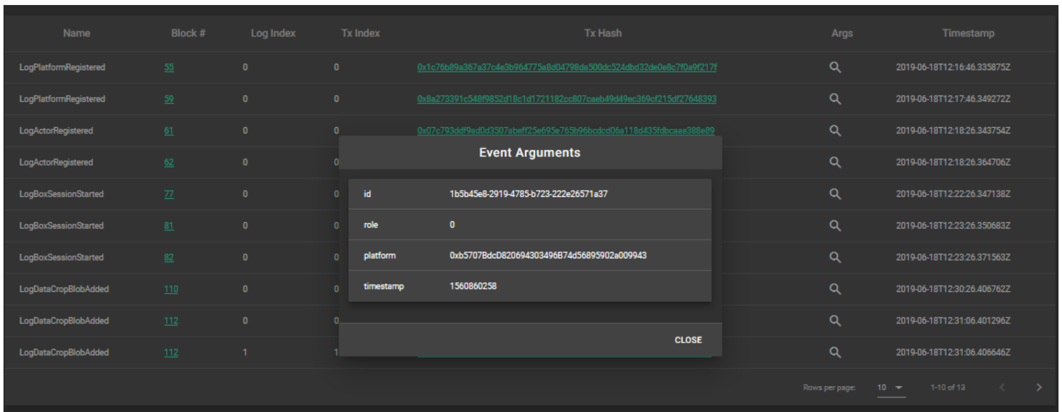
Result



LogBoxProduct

1649770 0xe872184f16761bd79a84243c589e10720095dcc9505866fd469e3e05c90e5dcd

Actor	ef1b7932-52a3-4977-a12a-6fa54c365e21
Platform	0xb5707BdcD820694303496B74d56895902a009943
Farm name	Kiato
Establishment date	2018-05-02
Location	κ. Μουλκιου, Μούλκι, Δήμος Σικυωνίων, Corinthia Regional Unit, Peloponnese Region, Peloponnese, West Greece and Ionian Sea, 20200, Greece
Product type	Grapes
Ripening level	3675.5
Fertilizers	Limestone 1kg



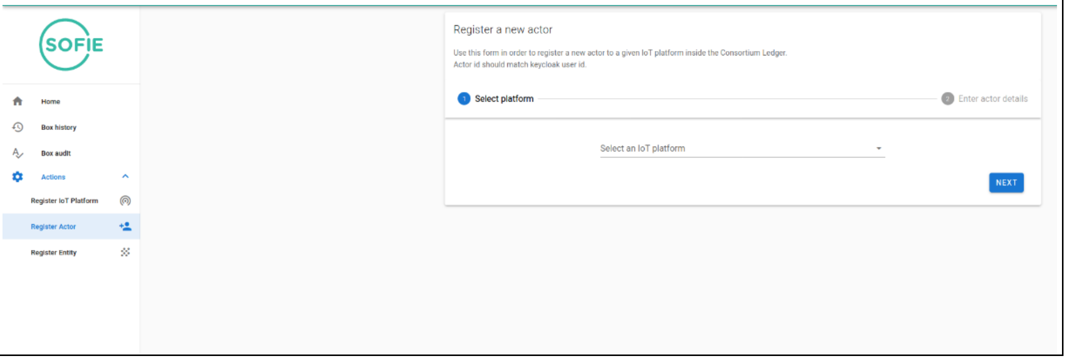
Name	Block #	Log Index	Tx Index	Tx Hash	Args	Timestamp
LogPlatformRegistered	55	0	0	0x1c76b89a367a37c4e3b964775a8d04798da500dc524dbd32de0e8c7f0a9f217f	🔍	2019-06-18T12:16:46.335875Z
LogPlatformRegistered	59	0	0	0x8a273391c548f9852d18c1d1721182cc807caeb49d49ec369cf215df27648393	🔍	2019-06-18T12:17:46.349272Z
LogActorRegistered	61	0	0	0x07c793ddf9ad0d3507abeff25e695e765b96bcdcd06a118d435fdbcaae388e89	🔍	2019-06-18T12:18:26.343754Z
LogActorRegistered	62	0	0		🔍	2019-06-18T12:18:26.364706Z
LogBoxSessionStarted	77	0	0		🔍	2019-06-18T12:22:26.347138Z
LogBoxSessionStarted	81	0	0		🔍	2019-06-18T12:23:26.350683Z
LogBoxSessionStarted	82	0	0		🔍	2019-06-18T12:23:26.371563Z
LogDataCropBlobAdded	110	0	0		🔍	2019-06-18T12:30:26.406762Z
LogDataCropBlobAdded	112	0	0		🔍	2019-06-18T12:31:06.401296Z
LogDataCropBlobAdded	112	1	1		🔍	2019-06-18T12:31:06.406646Z

Rows per page: 10 1-10 of 13

Event Arguments

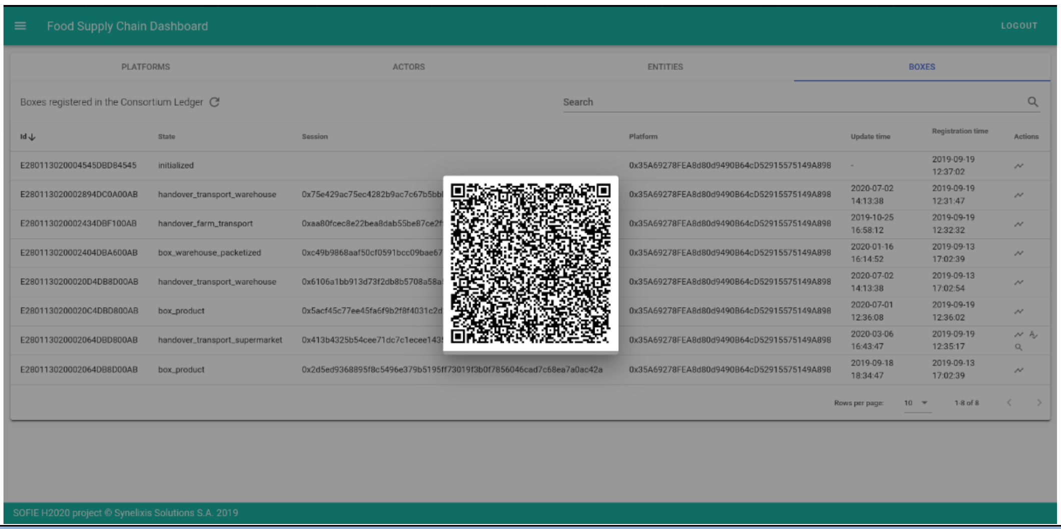
id	1b5b45e8-2919-4785-b723-222e26571a37
role	0
platform	0xb5707BdcD820694303496B74d56895902a009943
timestamp	1560860258

Test ID	FSC_TC07
Test description	Metadata related to an actor's activity (in the FSC app) is accessible by that actor at any time and is invisible to any other actor.
Steps	1. An actor logs in using his profile in the FSC web application. 2. The actor performs a number of actions. 3. The actor confirms that he can access the logs of all performed actions and that recorded information per (trans)action is correct. 4. The actor tries to access a view/endpoint for which he does not have the authority (based on his role).
Pass criteria	Access of each actor to its own resources is allowed, while access to other resources is prohibited.
Result	



Register a new actor

Use this form in order to register a new actor to a given IoT platform inside the Consortium Ledger.
Actor id should match keycloak user id.

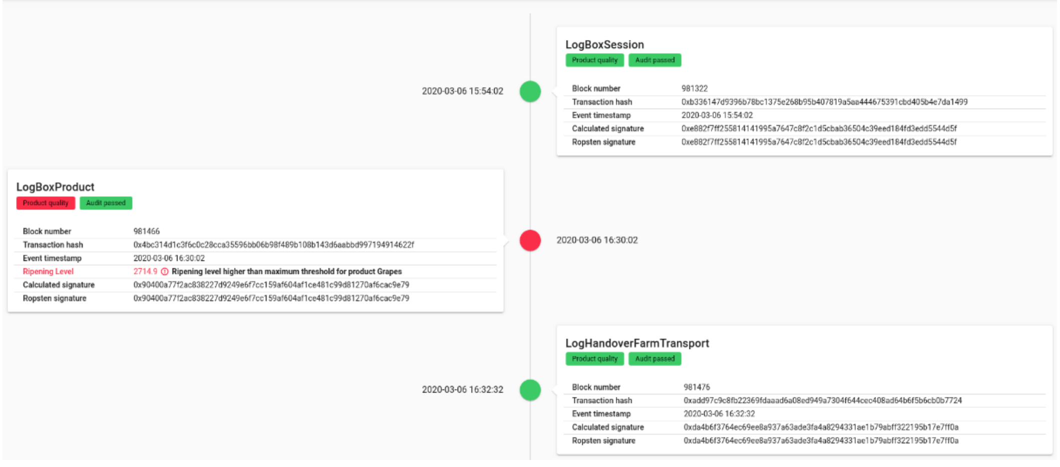
Test ID	**FSC_TC08**
Test description	A QR code which is created by the supermarket employee using the FSC web application can be read offline by using different smartphone devices. Readability of all included information is confirmed.
Steps	1. The customer uses a smartphone to read information encoded in the QR code of the package. 2. The action is repeated by using five different smartphone devices/QR reading applications.
Pass criteria	The revealed information includes (at least) the following information: farm location, type of product, harvesting date, used fertilizers, packetizing date, ID of used box and session ID.
Result	

Boxes registered in the Consortium Ledger

Id	State	Session	Platform	Update time	Registration time	Actions
E280113020004545DBD84545	initialized		0x35A69278FEA8d80d9490B64cD52915575149A898	-	2019-09-19 12:37:02	
E2801130200002894DC0A00AB	handover_transport_warehouse	0x75e429ac75ec4282b9ac7c67b5bb	0x35A69278FEA8d80d9490B64cD52915575149A898	2020-07-02 14:13:38	2019-09-19 12:31:47	
E28011302000 2434DBF100AB	handover_farm_transport	0xaa80fcec8e22bea8dab55be87ce2f	0x35A69278FEA8d80d9490B64cD52915575149A898	2019-10-25 16:58:12	2019-09-19 12:32:32	
E2801130200002404DBA600AB	box_warehouse_packetized	0xc49b9868aaf50cf0591bcc09bae67	0x35A69278FEA8d80d9490B64cD52915575149A898	2020-01-16 16:14:52	2019-09-13 17:02:39	
E280113020020D4DB8D00AB	handover_transport_warehouse	0x6106a1bb913d73f2db8b5708a58a	0x35A69278FEA8d80d9490B64cD52915575149A898	2020-07-02 14:13:38	2019-09-13 17:02:54	
E280113020020C4DBD800AB	box_product	0x5acf45c77ee45fa6f9b2f8f4031c2d	0x35A69278FEA8d80d9490B64cD52915575149A898	2020-07-01 12:36:08	2019-09-19 12:36:02	
E28011302000 2064DBD800AB	handover_transport_supermarket	0x413b4325b54cee71dc7c1ecee143	0x35A69278FEA8d80d9490B64cD52915575149A898	2020-03-06 16:43:47	2019-09-19 12:35:17	
E28011302000 2064DB8D00AB	box_product	0x2d5ed9368895f8c5496e379b5195ff73019f3b0f7856046cad7c68ea7a0ac42a	0x35A69278FEA8d80d9490B64cD52915575149A898	2019-09-18 18:34:47	2019-09-13 17:02:39	

Rows per page: 10 1-8 of 8

SOFIE H2020 project © Synelixis Solutions S.A. 2019

Test ID	**FSC_TC9**
Test description	Test that the audit service can access/process data streams containing relevant information and discard requests containing irrelevant information, e.g., improper box ID and session ID.
Steps	1. The supermarket employee scans the QR code attached on the product. 2. The supermarket employee requests an audit by accessing the corresponding service in the FSC web application and providing box ID and session ID values.
Pass criteria	Audit services are properly executed once relevant data is provided whereas they are aborted in cases of irrelevant data.
Result	

2020-03-06 15:54:02

LogBoxSession

Block number	981322
Transaction hash	0xb336147d9396b78bc1375e268b95b407819a5aa444675391cbd405b4e7da1499
Event timestamp	2020-03-06 15:54:02
Calculated signature	0xe882f7ff255814141995a7647c8f2c1d5cbab36504c39eed184fd3edd5544d5f
Ropsten signature	0xe882f7ff255814141995a7647c8f2c1d5cbab36504c39eed184fd3edd5544d5f

2020-03-06 16:30:02

LogBoxProduct

Block number	981466
Transaction hash	0x4bc314d1c3f6c0c28cca35596bb06b98f489b108b143d6aabbd997194914622f
Event timestamp	2020-03-06 16:30:02
Ripening Level	2714.9 Ripening level higher than maximum threshold for product Grapes
Calculated signature	0x90400a77f2ac838227d9249e6f7cc159af604af1ce481c99d81270af6cac9e79
Ropsten signature	0x90400a77f2ac838227d9249e6f7cc159af604af1ce481c99d81270af6cac9e79

2020-03-06 16:32:32

LogHandoverFarmTransport

Block number	981476
Transaction hash	0xadd97c9c8fb22369fdaaad6a08ed949a7304f644cec408ad64b6f5b6cb0b7724
Event timestamp	2020-03-06 16:32:32
Calculated signature	0xda4b6f3764ec69ee8a937a63ade3fa4a8294331ae1b79abff322195b17e7ff0a
Ropsten signature	0xda4b6f3764ec69ee8a937a63ade3fa4a8294331ae1b79abff322195b17e7ff0a

3.3.1 Future validation results

The final validation results of the platform and the pilot overall validation is on-going and will be reported in the final WP5 deliverable, D5.4 (Final Validation & Replication Guidelines). These results will also include information about the on-site pilot deployment as well as any updates on the pilot platform based on the feedback from the on-site deployment. Due to the major impact that COVID-19 had on all segments of the Food Supply Chain, we had to postpone the on-site validation until the health safety protocols would allow it. Therefore, we are scheduling the on-site validation within the last six months of SOFIE.

4. Decentralized Energy Data Exchange Pilot

4.1 Pilot overview

The Decentralized Energy Data Exchange (DEDE) pilot key driver is enabling the data owner (i.e. the entity/person who legally owns smart meter data) to:

- choose who gets access to the data,
- enable data transfer supported with business logic,
- receive proof of the parties accessing/using the data,
- receive guarantees that activities comply with GDPR and high security requirements.

The starting point for the decision-making regarding access rights is fixed on the data owner side, but the smart meter data storage and data processing can be different. The pilot is focusing in three different data access points:

- data from the National data hub (Estfeed platform),
- data from a regional database, energy subsystem (wind farm network),
- data from the single metering point from the household (zero-energy building)

The core idea of the pilot is to provide a proof-of-concept for secure data exchange and agreements to data access rights between smart meter data and infrastructure owners and energy service providers (intermediaries, distributors, brokers). The pilot will develop and use the capabilities of the SOFIE federated platform and Energy grid adapters to deliver the required functionality to stakeholders.

A general overview of the pilot can be seen here:

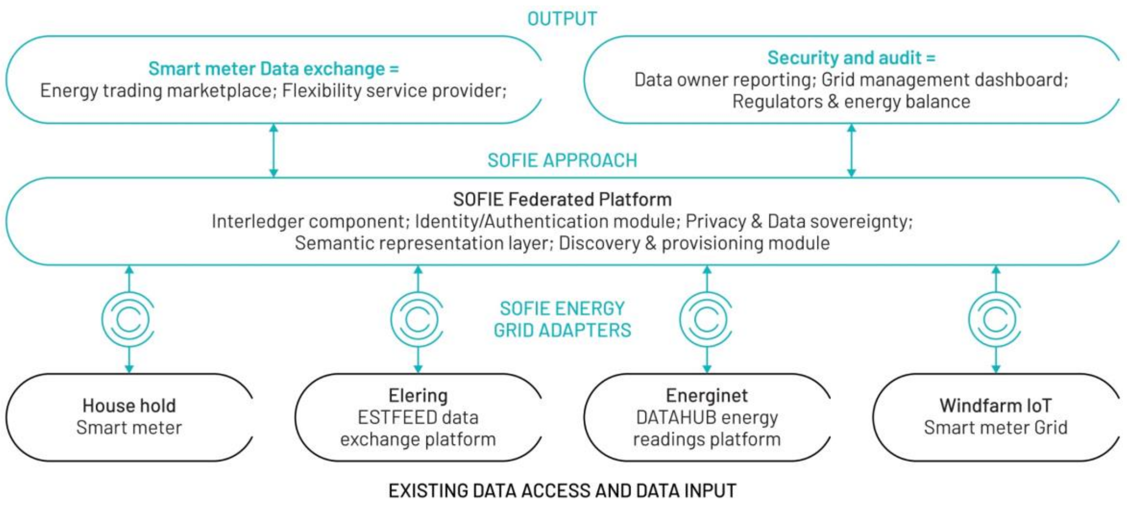

OUTPUT
Smart meter Data exchange =
Energy trading marketplace; Flexibility service provider;
Security and audit =
Data owner reporting; Grid management dashboard; Regulators & energy balance
SOFIE APPROACH
SOFIE Federated Platform
Interledger component; Identity/Authentication module; Privacy & Data sovereignty; Semantic representation layer; Discovery & provisioning module
SOFIE ENERGY GRID ADAPTERS
House hold Smart meter
Elering ESTFEED data exchange platform
Energinet DATAHUB energy readings platform
Windfarm IoT Smart meter Grid
EXISTING DATA ACCESS AND DATA INPUT

Figure 7: Basic building block of the DEDE pilot

The basic concept and building blocks of the DEDE pilot is depicted in Figure 7. We build a network where SOFIE adapters are installed and the interaction between the energy sellers (on the top here) and consumers (behind the data sources) is enabled.

On the bottom we can see the data sources. The stand-alone smart meters, national datahubs and regional solutions already exist, but are not able to deliver data as needed. On the upper part we have energy flexibility service providers needing access to the data on the left and various entities on the right that require the reporting, auditing, and traceability functionality of data access control. In the center we have the SOFIE Energy grid adapters and SOFIE federated platform that are the main components developed during the project. Using the

Hyperledger Indy, Verifiable credentials, Decentralized Identifiers and Guardtime's KSI blockchain, we enable governance and traceability when accessing the data for data owners and service providers.

The adapters will enable users to connect to those data sources and deliver matchmaking, cryptographic proof, security of transactions and logs to all parties involved. From the service providers and datahub operator's perspective, the key driver to be onboarded is the reduction of integration costs, as most of this is solved with adapters and the SOFIE federated platform.

More detailed information about the architectural design of the DEDE pilot, selected scenarios/use cases, roles of the involved actors, as well as the added value to stakeholders that are involved can be found in detail in Deliverable 5.2 (Initial Platform Validation).

At this stage, the first integration of the DEDE pilot platform has been completed and the first round of end-to-end validation results have been performed to verify the platform's operation with respect to the defined requirements. Overall, the main achievements of the DEDE pilot during the period M18-M30 and the activities which are scheduled for the last period of the project are summarized in Table 1.

4.2 Pilot platform architecture and services

4.2.1 Introduction

The DEDE platform connects energy data providers with energy data consumers in a secure, open, and decentralized way. Both the data providers and the data consumers connect to the platform through their own instance of the *Federation Adapter* (FA). The FA is a common software component for both data providers and data consumers and needs no extension or customization. The FA of a data consumer connects directly to the FA of a data provider to exchange messages according to the FA communication protocol. Messages between the two FAs are transported securely over a mutually authenticated TLS connection, using Hyperledger Indy-based decentralized identifiers and verifiable credentials to establish trust. This document describes the architecture of the platform and the FA component that enables it.

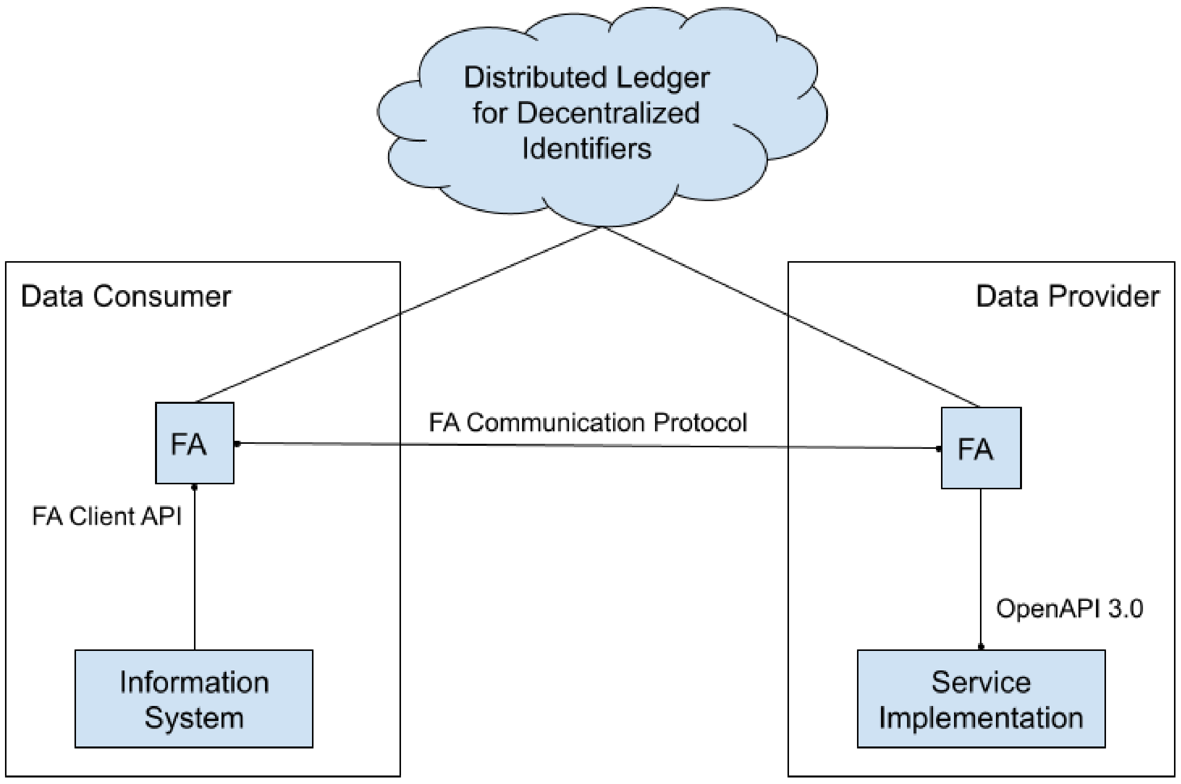


Figure 8: Overview of the DEDE platform

4.2.2 Responsibilities of the FA

The FA is the key component in enabling the DEDE platform. Its main function is to ensure interoperability and to secure the communication with other entities on the DEDE platform. It acts as a forward proxy for the data consumer and as a reverse proxy for the data provider, but it can also perform both roles at the same time, enabling entities that are both data consumers and data providers. The FA takes care of the security aspects of the integration and lets the data provider concentrate on implementing services and the data consumer on using these services. The only requirement for a data provider is to describe its services in OpenAPI 3.0 format.

Each entity on the DEDE platform is identified by a *Decentralized Identifier* (DID). It is a new type of globally unique identifier, that is self-administered and whose ownership can be cryptographically verified. DIDs form the base layer for *Verifiable Credentials* (VC) that are used to make authorization decisions in the FA. Every DID is associated with a public key and the mapping is published on a Hyperledger Indy instance - a distributed ledger built for this purpose. The private key that gives control over the DID is stored in a wallet managed by the FA. This makes it possible for the FA to sign every message sent out from that FA. Use of DIDs and VCs makes the DEDE platform independent of DNS names and Web PKIs. Although there are many different methods to define DIDs and basic CRUD operations to manage them (https://w3c-ccg.github.io/did-method-registry/), the DEDE platform only supports the Sovrin[1] method implemented by Hyperledger Indy.

The FA is also a natural place to construct the audit log of the messages exchanged between entities. A data provider will have a log of signed request messages that can be used to prove which data consumer has asked for which data. Likewise, a data consumer will have a log of

[1] https://sovrin.org/

signed response messages to prove which data provider gave out which data. The FA will take care of securing this audit log with KSI.

4.2.3 The FA Communication Protocol

4.2.3.1 Choosing the Source and Target DID

On the DEDE platform, a data consumer initiates the connection to a data provider. Both entities are represented by a DID, generated for them in their FA. Each FA can manage one or more DIDs to represent one or more entities. The first step for the data consumer is to choose the source (its own) DID for the connection and the target (data provider) DID. Providing the source DID is optional, as the FA only has a single DID generated by default that will be used as the source. Providing the target DID is required. The FA then resolves the endpoint of the target DID from the ledger, by reading the published attribute *proxy-endpoint* of the target DID.

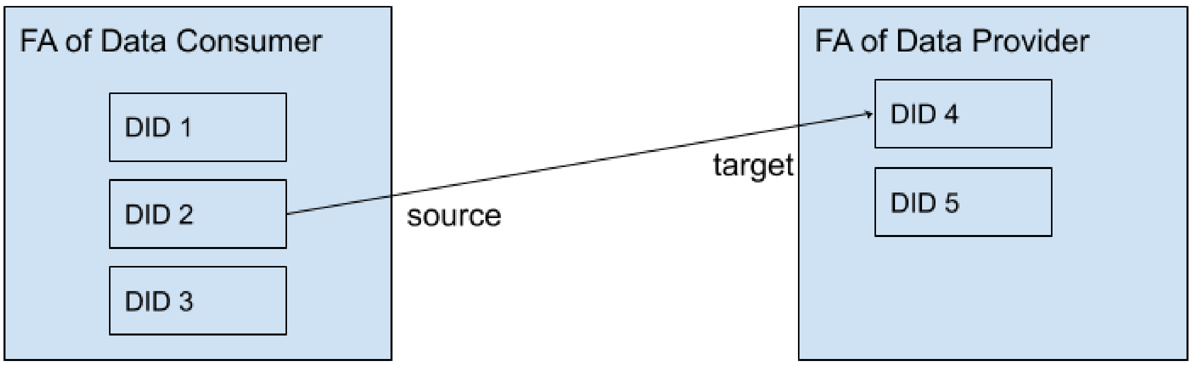


Figure 9: A connection between two DIDs controlled by different FAs

4.2.3.2 Proving the Control of a DID

The first step in establishing trust between a data consumer and a data provider is proving the control of the DID that is used for the connection. For this purpose, both sides generate a self-signed X.509 certificate with the Subject Common Name (CN) field containing the DID and publish its SHA-256 hash as an attribute to the ledger. This certificate is then used to establish a mutually authenticated TLS connection. Verification of the certificate presented in a TLS handshake contains the following steps for both sides:

1. Read the DID from the Common Name field of the certificate
2. Fetch **certificate-hash** attribute for the DID from the ledger
3. Calculate the SHA-256 hash of the certificate
4. Compare the calculated hash with the published hash
5. If the hashes match, the other party must be in control of the DID. If the hashes do not match, the connection should be dropped

Essentially, the public key used in the X.509 certificate is an alternative authentication key for the DID. Theoretically, it should be possible to issue an X.509 certificate for the native Ed25519 public key generated for each DID in Hyperledger Indy and use that for TLS connections as well. But the support for the Ed25519 signature system is currently lacking in TLS implementations. The FA initiating the TLS connection must specify the target DID in the Server Name Indication (SNI) extension of the TLS protocol. This allows the receiving end to present the correct certificate if it happens to have more than one. Only TLS 1.3 is used between the FAs.

4.2.3.3 Proving the Identity of a DID Owner

Once both parties have proved the control of the DID, they are using for the connection, it is possible to start consuming services that do not require further authentication. But, if the data consumer sends a request to a service that requires proving his identity, the data provider FA sends a proof request to the data consumer FA, requesting proof of the attributes configured for the service. For example, a service called *getConsumptionData* can be configured to require proof of the *nationalId* attribute in a verifiable credential issued by any of the issuers trusted by the data provider. The data provider can either configure a set of allowed *nationalId* values for the service in the FA, or it can do the authorization decision later in the service implementation, for which the FA will pass on the proved attributes.

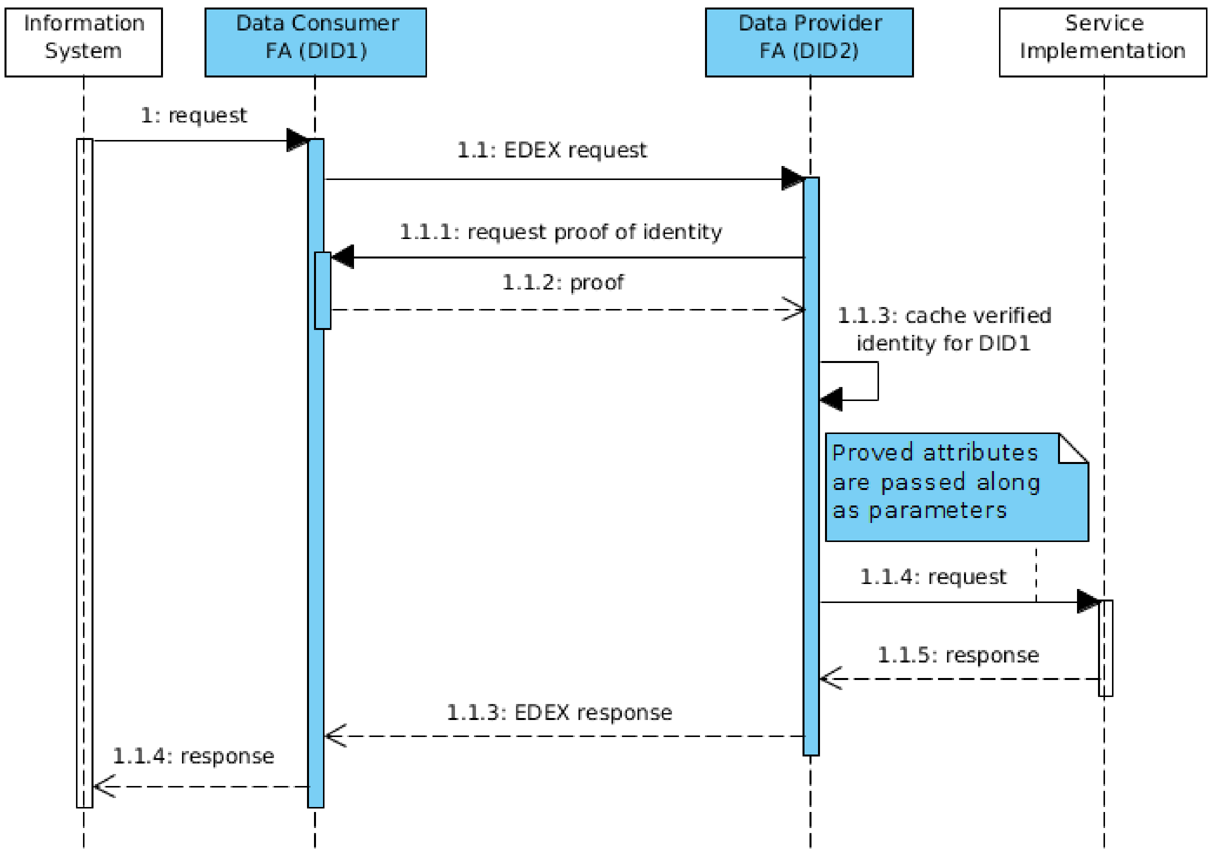


Figure 10: Request to a service that requires proved attributes

The proof presented by the data consumer must contain proof of non-revocation of the verifiable credential that contains the proved attribute. It is up to the service provider policy to decide the frequency of asking this proof again to verify non-revocation.

4.2.3.4 Proving Delegated Authorization

If an entity is authorized to consume some service, it is also possible for that entity to delegate that access right to other entities.

As the first step, the authorized entity (delegator) must prove its identity to the data provider. This is required to establish trust between the delegator and the data provider and contains the same interactions described in the previous section. If the data provider trusts and allows access to an entity, it can also trust the delegations made by that entity.

As the second step, the delegator entity issues a credential to the delegated entity. The credential must follow the standard *Edex-Delegation* credential schema. This credential can be revoked by the delegator at any time.

As the last step, the delegated entity sends a request to the data provider, specifying the DID of the delegator entity as a header parameter. This is the flag for the data provider FA, saying that instead of asking for proof of configured attributes, it needs to ask for proof of delegation. If the data provider allows access to the delegator DID and the request sender can prove that it has been delegated by that entity, then the data provider FA can safely proceed with the request. Figure 11 shows this interaction between the FAs in three different roles.

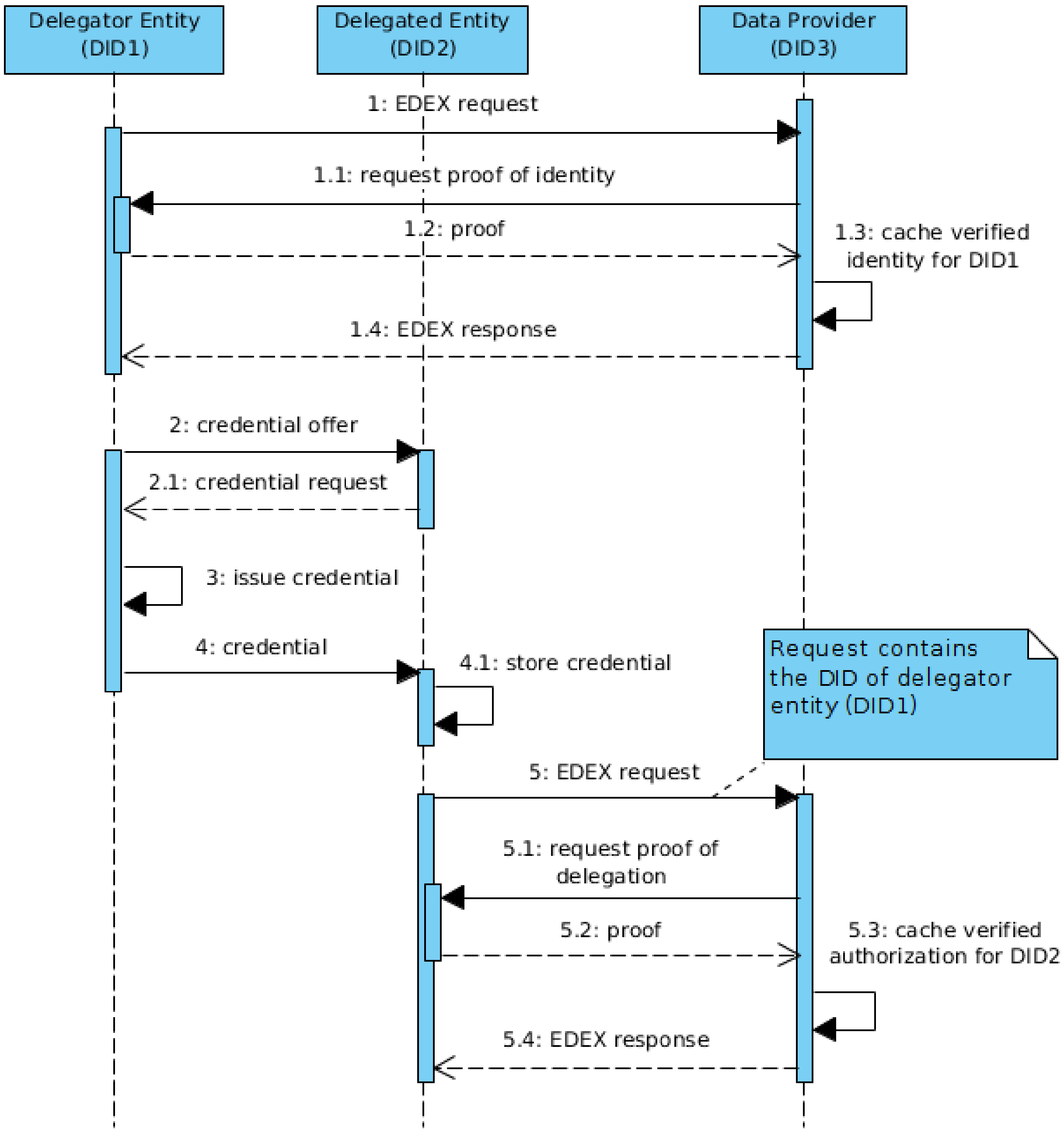


Figure 11: Request to a service with delegated access rights

4.2.4 FA Internals and the Use of SOFIE Components

The two main responsibilities of the FA is to proxy messages and to manage the identity of the represented entity. The internal structure of the FA mirrors this with two loosely coupled services: *proxy* and *ssi-agent* (Self-Sovereign Identity agent). Both components have public and private interfaces, for external and internal use accordingly.

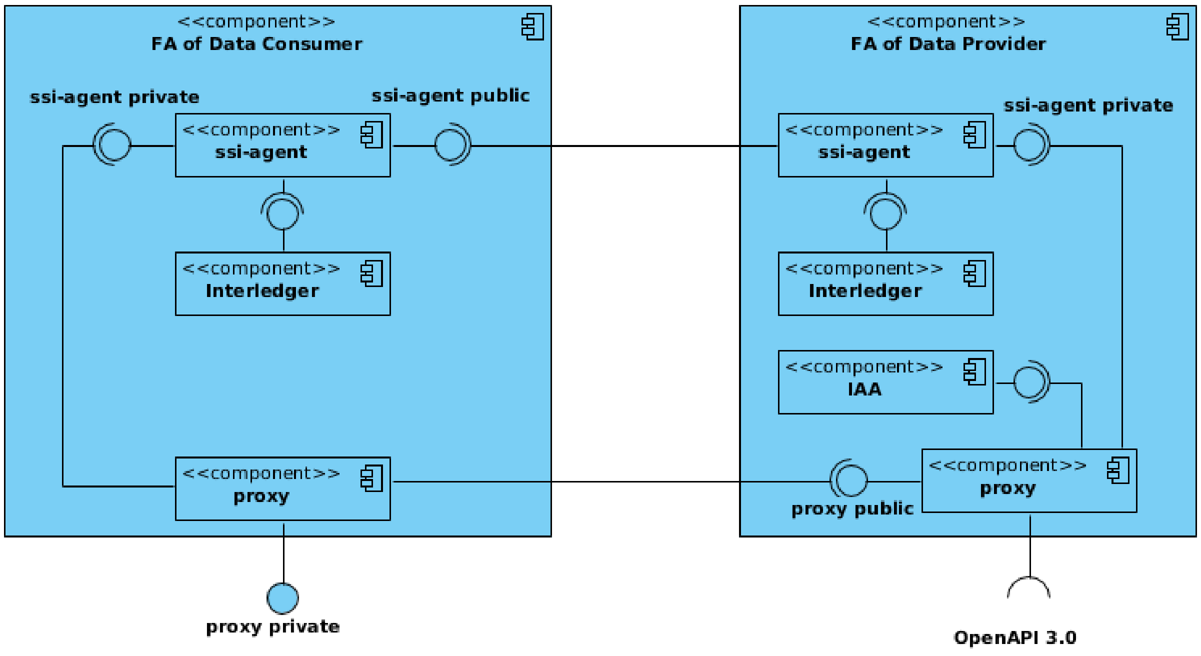


Figure 12: Internal structure of the FA

The information system of the data consumer first sends a request to the proxy private interface. The proxy uses the *ssi-agent* private interface to resolve the endpoint of the target DID and to sign the request with the source DID. Then, it initiates a secure connection to the public interface of the data provider (target DID) proxy. Both sides use the *ssi-agent* private interface to retrieve the hash of the currently valid certificate to verify the authenticity of the connection. Once the connection is set up, the data provider proxy will use the *ssi-agent* private interface to verify request signature. If the request is for a service that requires further authorization, the data provider proxy will also use the *ssi-agent* private interface to get the proved values of the attributes required for the authorization decision. If the data provider *ssi-agent* receives such a request, it will send a proof request to the public interface of the data consumer *ssi-agent*. Once the data provider proxy has values for all the proved attributes, it can forward the request to the service implementation that is described using the OpenAPI 3.0 specification. The signing of the response and the verification of the response message signature is analogous to the processing of the request.

An alternative for the data consumer to proving its credentials on demand, is to send them together with the request, as a JSON Web Token issued by the SOFIE *Privacy and Data Sovereignty* (PDS) component. In this setup, it is the responsibility of the data consumer information system to acquire the token accepted by the data provider and include it in the request header. If the data provider proxy receives such a request, it will use the SOFE *Identification, Authentication, and Authorization* (IAA) component to verify the token and get the trusted values for the attributes that are required for the authorization decision.

The ssi-agent on both sides uses the SOFIE Interledger component to periodically record the state of the Hyperledger Indy instance with KSI.

4.2.5 Proxy Private API

The most important API that the FA provides is the private API of the proxy service that data consumers use to send requests to data providers. Although the goal for the proxy is to be as transparent as possible and make it feel like the data provider services are invoked directly, there are still some aspects that the data consumer needs to be explicit about. In addition to the request payload, which is constructed according to the target service description, data consumer must provide the following DEDE platform-specific parameters in the request:

Table 3: Proxy Private API

Parameter	Type	Required
Target DID	Path parameter (*targetDid*)	yes
Source DID	Header parameter (*X-EDEX-SourceDid*)	no (only if the FA controls more than one DID)
Authorization Token	Header parameter (*Authorization*)	no (only if PDS based tokens are used for authorization)

The API defines just one endpoint: ***/proxy/{targetDid}/*****

Currently, HTTP GET and POST methods are supported. Everything in the path after the *targetDid* parameter is used as is to execute the service on the data provider side. That includes the query parameters. Also, the request body and headers are proxied as is.

The response looks exactly as returned by the service implementation, except for an additional header (*X-EDEX-Error)* in case an error happens in the FA of either side. This helps to distinguish service implementation errors from the DEDE platform errors.

There is also a metaservice implemented by the FA at ***/proxy/{targetDid}/services***. It can be used by data consumers to discover the services that different data providers offer. The service description follows the OpenAPI 3.0 specification.

4.2.6 Pilot Deployment View

The pilot environment consists of two data providers and a single data consumer that supports multiple data owners.

- *Smart Meter* is a data provider that serves data for a single metering point, owned by an individual who has sole rights to the data served by it.
- *Estfeed* is an aggregate data provider that serves data for many metering points, each owned by a different individual. Since Estfeed has a different service description and message format than the one expected by the DEDE platform, it needs an additional converter that converts between the two.
- *Data Owner Portal* is a data consumer that can represent many data owners. It controls the DID for each data owner that it represents. In this setup the data owner portal is also a trusted party for the data providers who can issue verifiable credentials or bearer tokens to prove data owner identities. These roles do not have to belong to a single entity. Each data owner could control its own DID in a mobile app version of the FA and receive verifiable credentials or tokens from external sources.

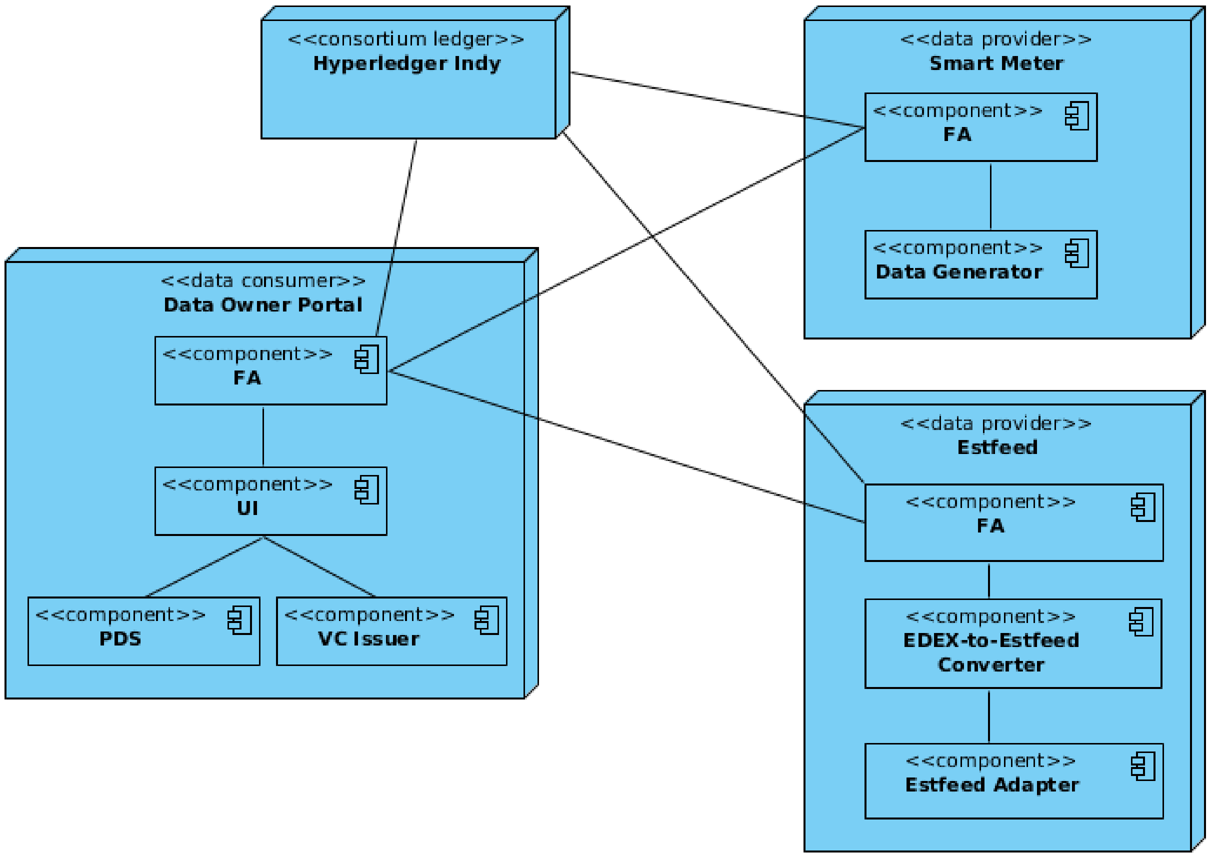


Figure 13*: Deployment view of the DED platform pilot environment*

4.3 Validation results

The validation approach can be roughly divided into three sections.

- Firstly, the business case and processes validation with end-users considering the technical details that are already fixed based on the previous analyze and work in SOFIE.
- Secondly, the pilot's solution components validation with full functionality in the test environment. This also includes the validation of the technical approach that we have taken in the pilot.
- Thirdly, the on-site testing with end users.

Since the validation process inside the industry has been impacted by the COVID-19 situation we have mainly focused on the First and Second section of the validation, leaving the on-site testing to be conducted in the last 6 months in the SOFIE project.

4.3.1 Validation approaches

The Business case and processes validation was ongoing until the end of February 2020 and started again after the COVID-19 outbreak eased in June 2020. The methodology to do validation was mainly using one to one telcos, participation in workshops and conducting questionnaires to end-users. Key stakeholders that were targeted were selected based on the Dissemination and Exploitation plan and covered the full spectrum from National data hub operators (Elering, Energinet, PSE), Service providers (Spotty Energy, Enoco), Integrators (AKKA) and multiple smart home groups.

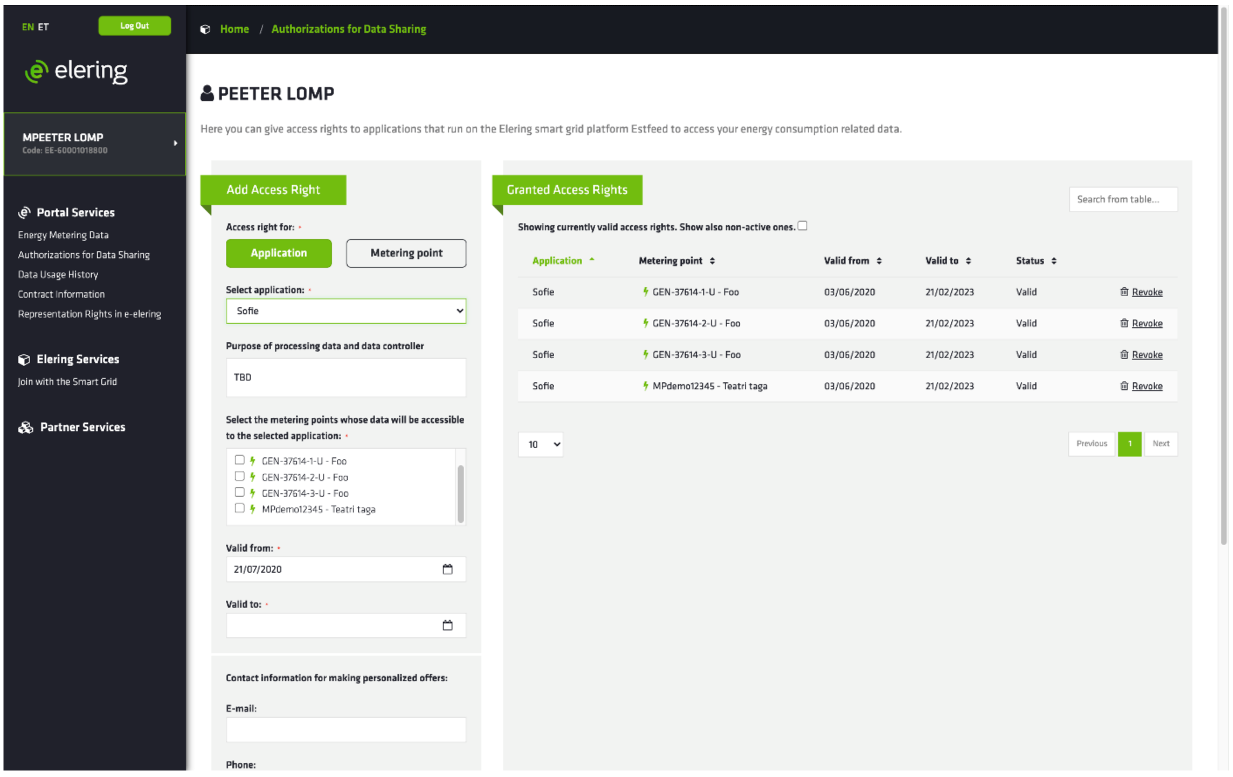
The main outcome of this validation was that there is a definite demand for the SOFIE DEDE platform and a high interest to investigate the cost benefit of using the platform from each stakeholders' premises as well as agreeing the preliminary business model for operation. More details can be found in the SOFIE deliverable *link: D6.8 Interim Report on Communication, Dissemination and Exploitation*

From the pilot technical concept perspective (presenting the SOFIE federation architecture, framework and interledger approach), we have validated that a decentralized and auditable approach to energy data exchange is feasible with the use of Hyperledger Indy and KSI technologies. Each integration is independent from all others yet enabling secure data exchange with every other entity that has joined before.

Integration of different energy data sources and data consumers is a key element for the data exchange pilot. So far, we have validated onboarding and integration steps with a national energy data hub in Estonia (Estfeed). With other national data hubs there has been preparation work, but no additional integrations are available yet. Additionally, we have prepared and validated integration with single data sources (could be an individual smart meter, small network etc.) in a simulated fashion.

The flexibility for integrating existing systems allows to create additional value for existing connections by multiplying the potential data exchange partners. Here is the description of integration steps for the Estonian Energy data hub called Estfeed.

- The SOFIE platform had to be onboarded at Estfeed by signing an agreement with Elering AS as the service provider for Estfeed platform.
- User logs in to the portal: https://portal.research.estfeed.ee/#/. Users can see data consumption and manage authorizations for data sharing. SOFIE has been onboarded as a trusted service provider to whom users can grant access to its data.



PEETER LOMP

Here you can give access rights to applications that run on the Elering smart grid platform Estfeed to access your energy consumption related data.

Add Access Right

Access right for: * [✓ Application] Metering point

Select application: * Sofie

Purpose of processing data and data controller: TBD

Select the metering points whose data will be accessible to the selected application: *
- ☐ GEN-37614-1-U - Foo
- ☐ GEN-37614-2-U - Foo
- ☐ GEN-37614-3-U - Foo
- ☐ MPdemo12345 - Teatri taga

Valid from: * 21/07/2020

Valid to: *

Contact information for making personalized offers:

E-mail:

Phone:

Granted Access Rights

Search from table...

Showing currently valid access rights. Show also non-active ones. ☐

Application	Metering point	Valid from	Valid to	Status	
Sofie	GEN-37614-1-U - Foo	03/06/2020	21/02/2023	Valid	Revoke
Sofie	GEN-37614-2-U - Foo	03/06/2020	21/02/2023	Valid	Revoke
Sofie	GEN-37614-3-U - Foo	03/06/2020	21/02/2023	Valid	Revoke
Sofie	MPdemo12345 - Teatri taga	03/06/2020	21/02/2023	Valid	Revoke

10 Previous 1 Next

Figure 14: Access rights delegation for SOFIE platform in Estfeed user portal

- Estfeed provides system components that were installed and configured. The Estfeed adapter is the web application that exchanges messages between the client application and the decentralized network, where different data sources provide data services. The pilot had to build a separate application that provides services in OpenAPI version 3 format. Here is the list of services as a screenshot from Swagger UI:

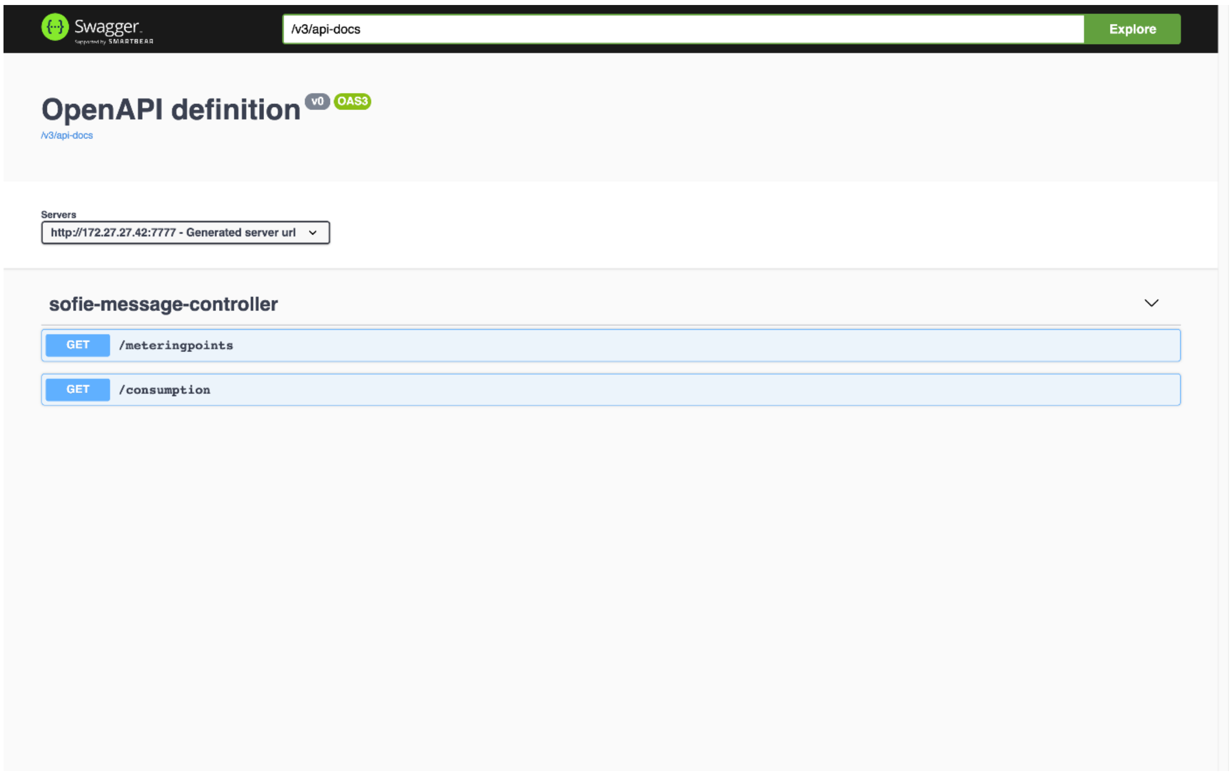


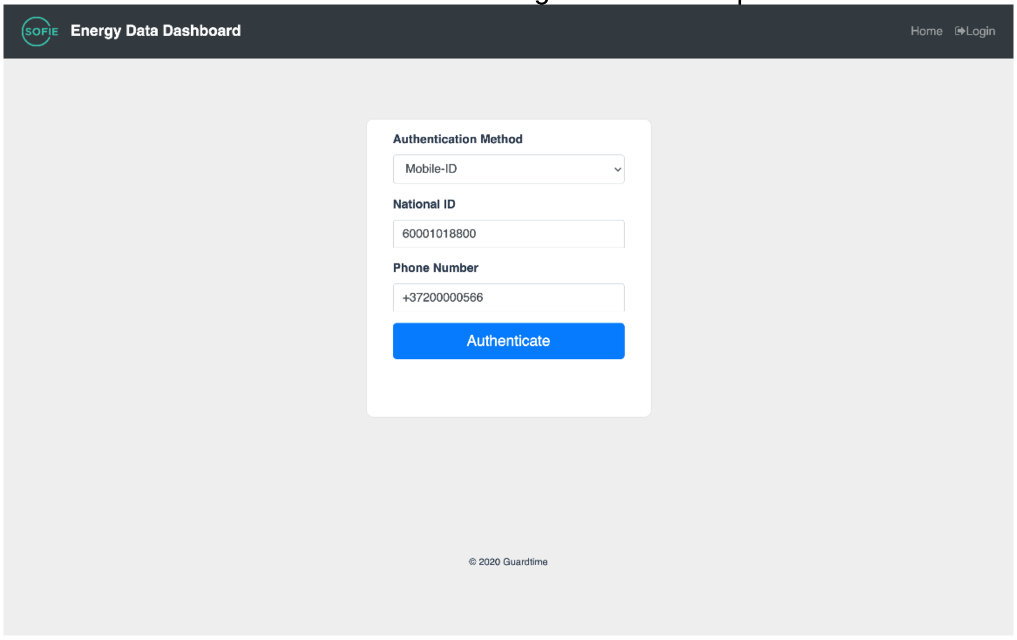
Figure 15: SOFIE Estfeed adapter service list

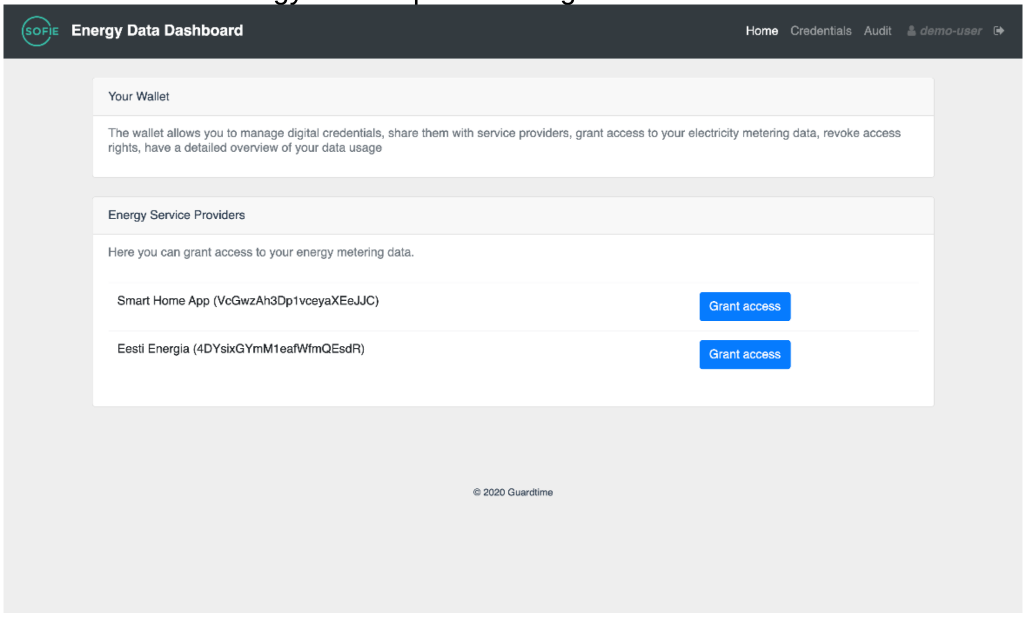
- The SOFIE adapter proxies the requests to the estfeed adapter and shows the energy data in common format. Here is the sample payload in JSON format:

```
[
 {
   "person":"PNOEE-38502136521",
   "usagePoint":"38Z121212123-U",
   "from":1567332000000,
   "to":1571616000000,
   "readings":[
     {
       "value":123456.789,
       "from":1567332000000,
       "to":1567335600000,
       "flowDirection":"o",
       "commodity":"electricity",
       "measurementKind":"energy",
       "unit":"kWh"
     },
     {
       "value":123456.789,
       "from":1567335600000,
       "to":1567339200000,
       "flowDirection":"o",
       "commodity":"electricity",
       "measurementKind":"energy",
       "unit":"kWh"
     },
     {
       "value":123456.789,
       "from":1567339200000,
       "to":1567342800000,
       "flowDirection":"o",
       "commodity":"electricity",
       "measurementKind":"energy",
       "unit":"kWh"
     }
   ]
 }
]
```

4.3.2 Validation results

The following pilot test cases defined in D5.1 have been validated.

Test ID	**EDE_TC01**
Test environment	Federation adapter has been installed and configured on both sides of the data exchange - customer portal and data hub. Data owner has been authenticated in customer portal and a credential proving his identity has been issued to him.
Validation flow	Validation results are provided in graphical and textual form. Users must authenticate before accessing the customer portal.  Energy Data Dashboard Home Login Authentication Method Mobile-ID National ID 60001018800 Phone Number +37200000566 Authenticate © 2020 Guardtime

	User can select energy service provider to grant access to its data: 

Energy Data Dashboard

Your Wallet

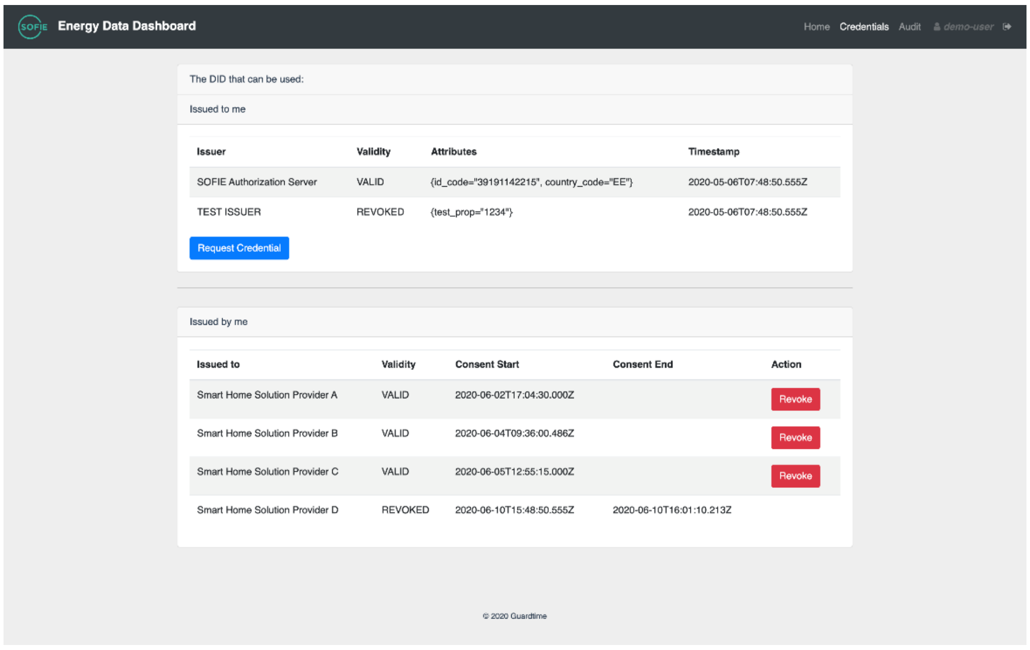
The wallet allows you to manage digital credentials, share them with service providers, grant access to your electricity metering data, revoke access rights, have a detailed overview of your data usage

Energy Service Providers

Here you can grant access to your energy metering data.

Smart Home App (VcGwzAh3Dp1vceyaXEeJJC)	Grant access
Eesti Energia (4DYsixGYmM1eafWfmQEsdR)	Grant access

© 2020 Guardtime

Test ID	**EDE_TC02**
Test environment	The Federation adapter has been installed and configured by a third party. Third party is known to the customer portal and offered as a target for delegation.
Validation flow	Validation results are provided in graphical and textual form. Logged in users in the customer portal can request access credentials. The list of issued credentials is displayed. Additionally, the user has an overview of credentials issued to her/him and can revoke them. 

Energy Data Dashboard

The DID that can be used:

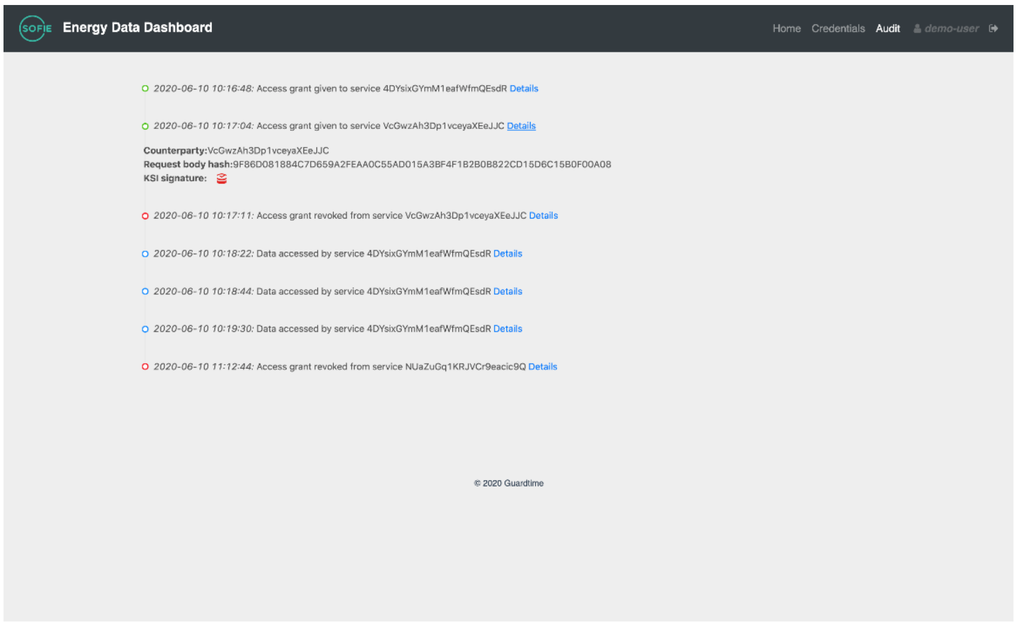
Issued to me

Issuer	Validity	Attributes	Timestamp
SOFIE Authorization Server	VALID	{id_code="39191142215", country_code="EE"}	2020-05-06T07:48:50.555Z
TEST ISSUER	REVOKED	{test_prop="1234"}	2020-05-06T07:48:50.555Z

Issued by me

Issued to	Validity	Consent Start	Consent End	Action
Smart Home Solution Provider A	VALID	2020-06-02T17:04:30.000Z		Revoke
Smart Home Solution Provider B	VALID	2020-06-04T09:36:00.486Z		Revoke
Smart Home Solution Provider C	VALID	2020-06-05T12:55:15.000Z		Revoke
Smart Home Solution Provider D	REVOKED	2020-06-10T15:48:50.555Z	2020-06-10T16:01:10.213Z	

© 2020 Guardtime

Test ID	**EDE_TC03**
Test environment	Both data owner and delegated third party have performed requests to data hub. The audit log is not empty at the data hub.
Validation flow	Every interaction in the system produces a signed audit log. For users, a human-friendly presentation of the audit log is displayed.  Energy Data Dashboard Home Credentials Audit demo-user *2020-06-10 10:16:48:* Access grant given to service 4DYsixGYmM1eafWfmQEsdR Details *2020-06-10 10:17:04:* Access grant given to service VcGwzAh3Dp1vceyaXEeJJC Details **Counterparty:**VcGwzAh3Dp1vceyaXEeJJC **Request body hash:**9F86D081884C7D659A2FEAA0C55AD015A3BF4F1B2B0B822CD15D6C15B0F00A08 **KSI signature:** *2020-06-10 10:17:11:* Access grant revoked from service VcGwzAh3Dp1vceyaXEeJJC Details *2020-06-10 10:18:22:* Data accessed by service 4DYsixGYmM1eafWfmQEsdR Details *2020-06-10 10:18:44:* Data accessed by service 4DYsixGYmM1eafWfmQEsdR Details *2020-06-10 10:19:30:* Data accessed by service 4DYsixGYmM1eafWfmQEsdR Details *2020-06-10 11:12:44:* Access grant revoked from service NUaZuGq1KRJVCr9eacic9Q Details © 2020 Guardtime

4.3.3 Future validation results

Technical validation tests are on-going and different performance metrics will be collected. To validate the integration process to alternative existing data hubs, an additional data hub integration process will be described.

5. Decentralized Energy Flexibility Marketplace Pilot

5.1 Pilot overview

The objective of the Decentralized Energy Flexibility Marketplace (DEFM) pilot is to demonstrate the use of the SOFIE architecture and its components to support the implementation of a decentralized energy flexibility marketplace in the context of an energy district/electricity grid with a high penetration of distributed generation from renewable energy sources. The pilot is designed to address the needs of the main actors: DSO operators, requesting flexibility to balance the grid, and EV fleet managers, interested in Demand Response campaign benefits. The actors are supported by:

i. The creation of flexibility requests on the marketplace

ii. The proposal of flexibility offers, in response to the requests

iii. Requests and offers matching

iv. Verification and payment settlement

The pilot's architectural design, main actors, scenarios and use cases, and the added value provided by SOFIE to the pilot BP can be found in detail in Deliverables 5.1 (Baseline System and Measurements) and Deliverable 5.2 (Initial Platform Validation.

At this stage, the first integration of the pilot platform has been completed and is being demonstrated at the pilot site (to be reported in D5.4). The first validation tests have been performed, in order to verify the platform's results with respect to the defined objectives. More details about the platform's implementation are discussed in Section 5.2, while the validation results are discussed in Section 5.3.

5.2 Pilot platform architecture and services

This section describes the updated pilot platform architecture (Figure 16) used for the initial validation. The pilot's architecture mixes pilot-specific components with the following SOFIE platform components:

- Federation Adapters
- Decentralized Marketplace
- Interledger
- Semantic Representation

The pilot-specific components include the IoT platforms, the operators' dashboards, and the necessary backend services and APIs.

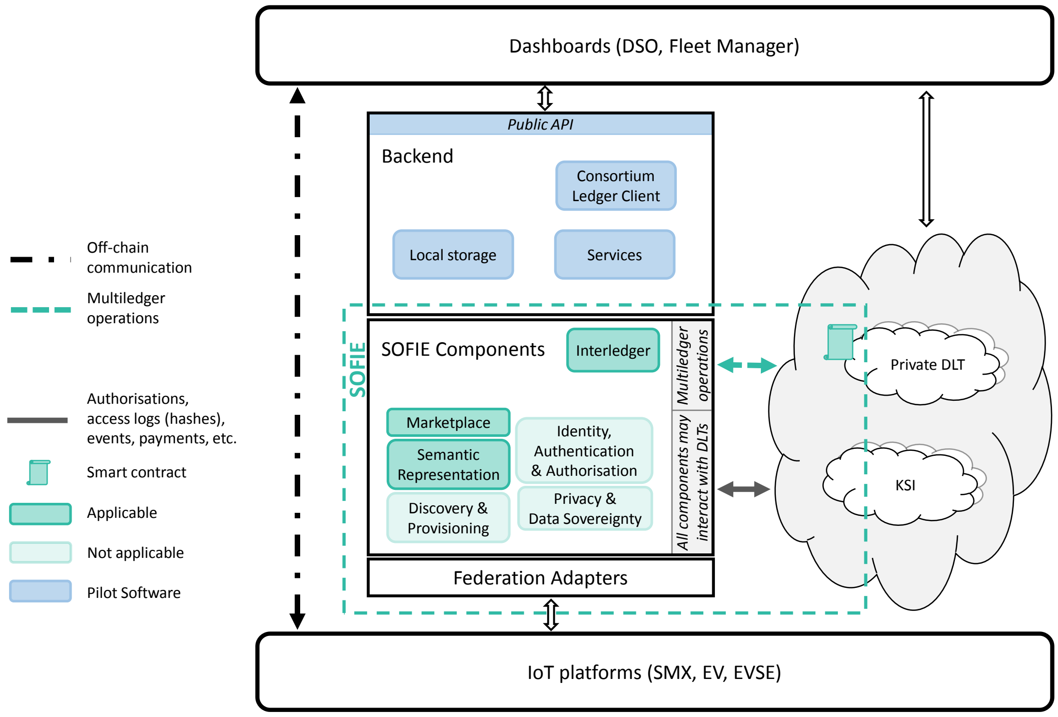


Figure 16: High-level architecture of the DEFM pilot

Figure 17 shows the interaction among the different actors and subsystems, with the steps needed to create a flexibility market request, participate with an offer to a market request, select the winning offer and, after the delivery phase, verify the fulfilment of the request and make the payment.

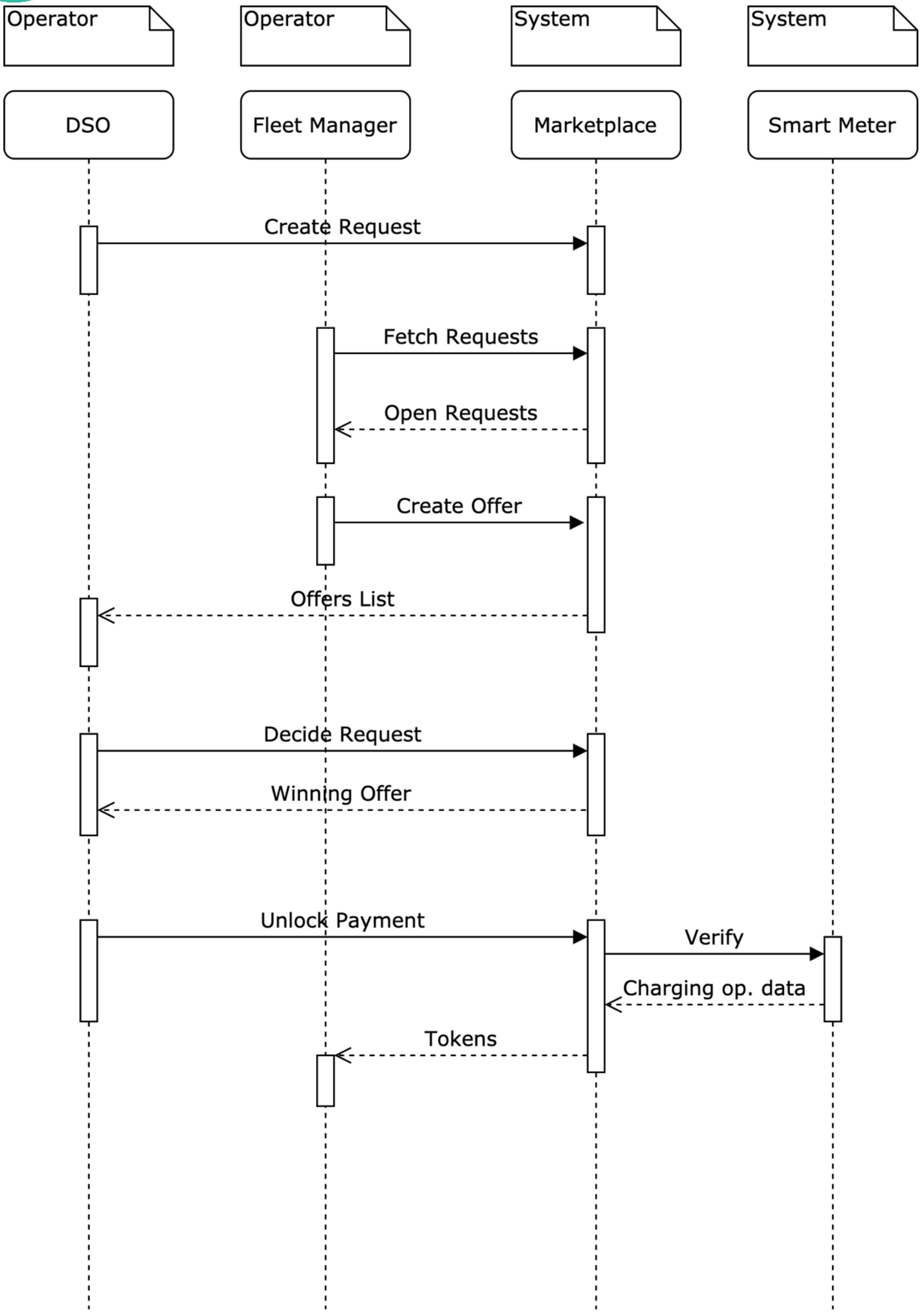


Figure 17: Message flow in the DEFM pilot

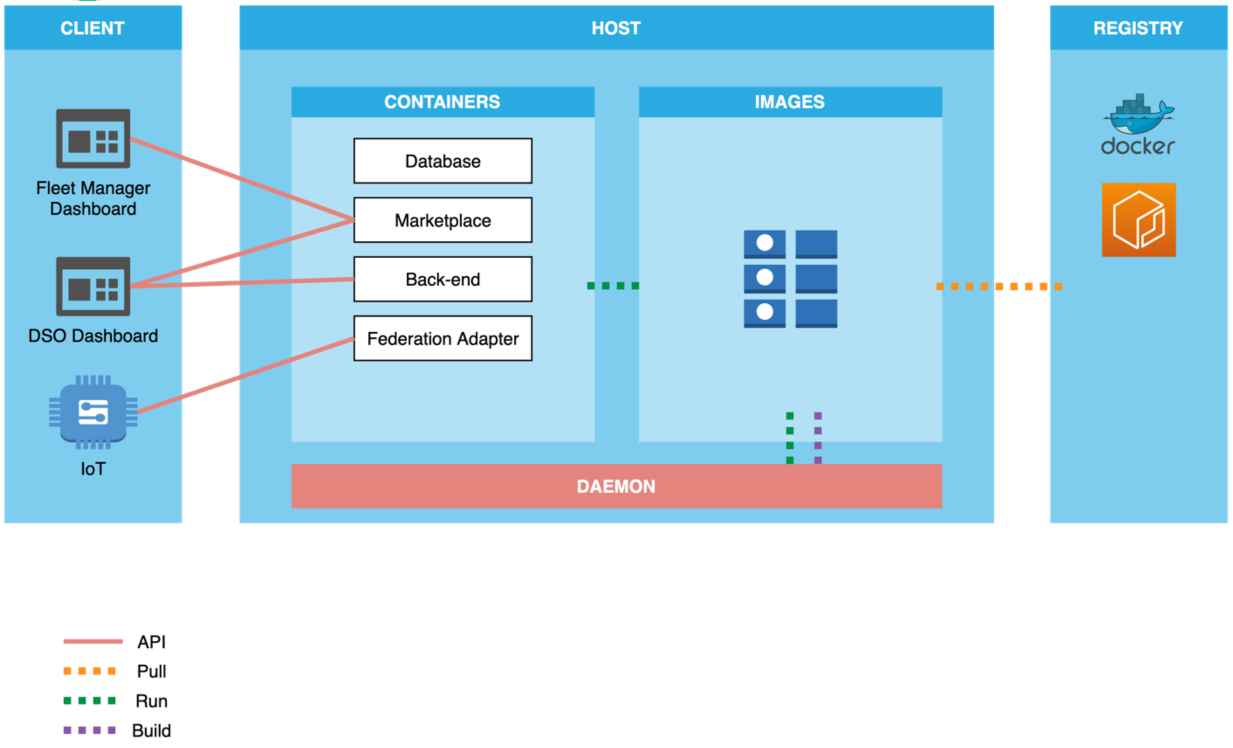


Figure 18: DEFM platform deployment view

Finally, Figure 18, presents the deployment view of the pilot platform. The actual deployment of the platform is based on container images. A container image can be considered as a software unit that packages together the code with all its dependencies (runtime, system tools, libraries, and settings) to form a standalone executable package that can be executed in different environments.

The host system operates a Docker instance as the runtime environment for the container images. The source code of the different modules contains all the instructions needed to build the related container images. Every time a new feature is added, a new image is built and is pushed on a container registry.

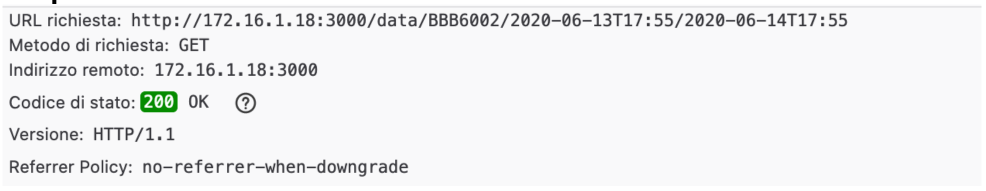
The host system utilizes Docker to fetch the updated images and execute them. Each (micro) service is managed then as a separated container which exposes its own set of APIs to interact with the clients (human operators through web interfaces, or IoT systems in an automated way).

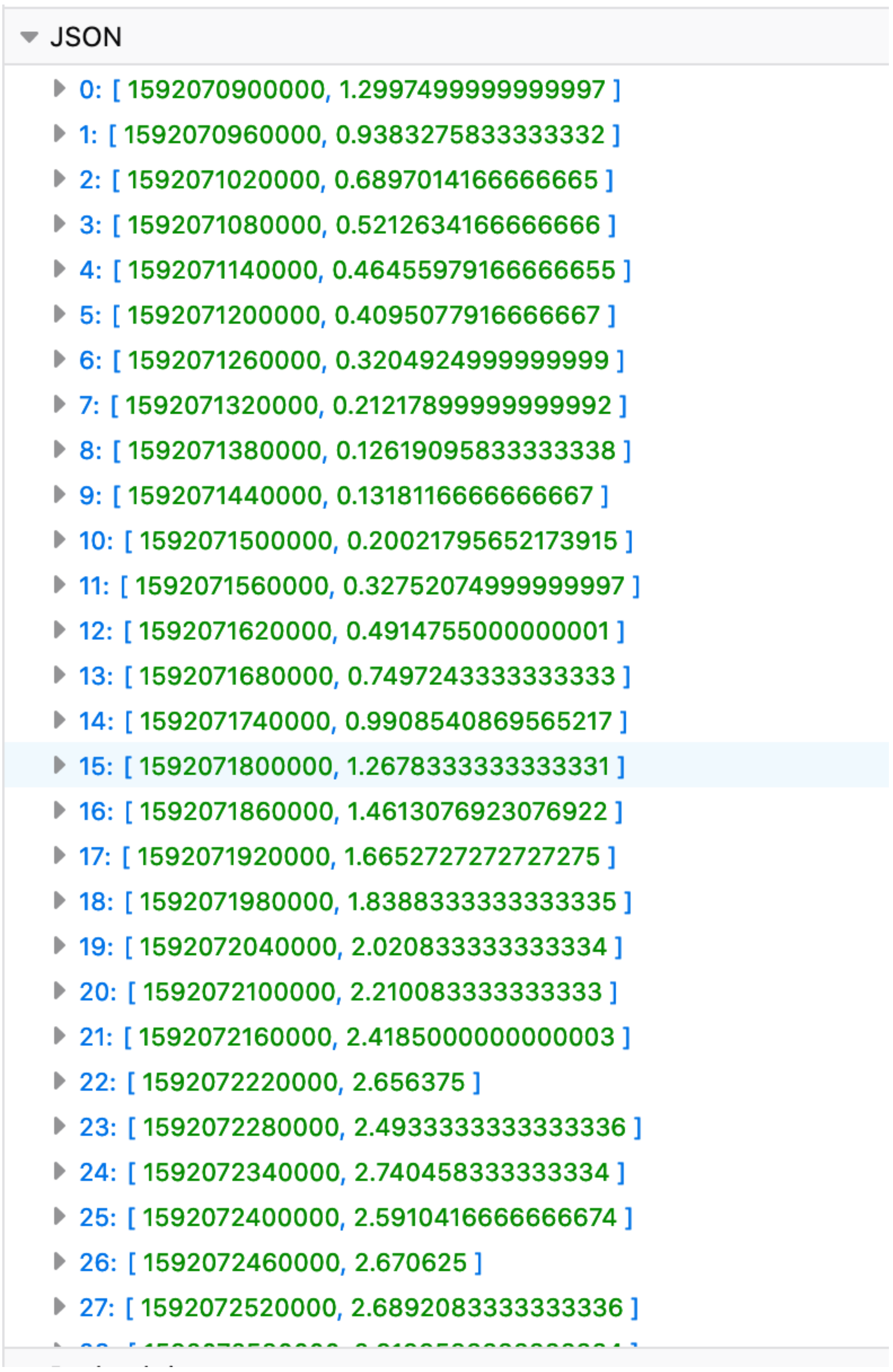
5.3 Validation results

The pilot test cases defined in deliverable D5.1 involve Metering & data collection and the Decentralized marketplace management. Below, the results of the platform tests are presented (Table 4, Table 5, Table 6).

Table 4: DEFM test case 1

Test ID	DEFM_TC01
Feature(s) under test	Metering & data collection
Pass criteria	historical and real-time data provided by the smart meters are properly retrieved

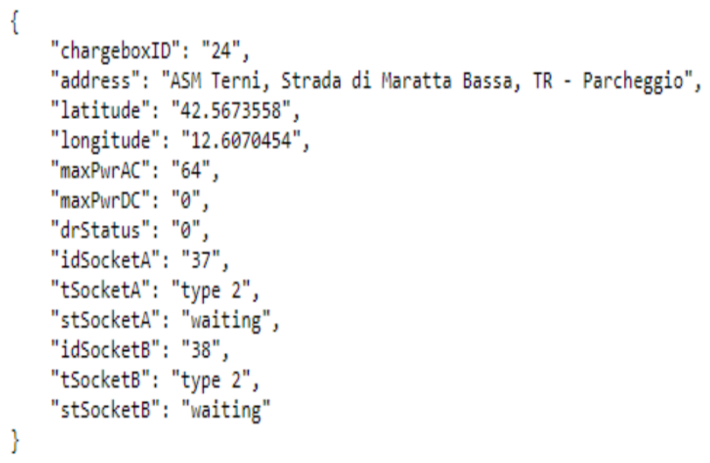
Methodology	API request invocation
Result	**Request**:  URL richiesta: `http://172.16.1.18:3000/data/BBB6002/2020-06-13T17:55/2020-06-14T17:55` Metodo di richiesta: GET Indirizzo remoto: `172.16.1.18:3000` Codice di stato: 200 OK Versione: HTTP/1.1 Referrer Policy: no-referrer-when-downgrade **Response** (truncated):

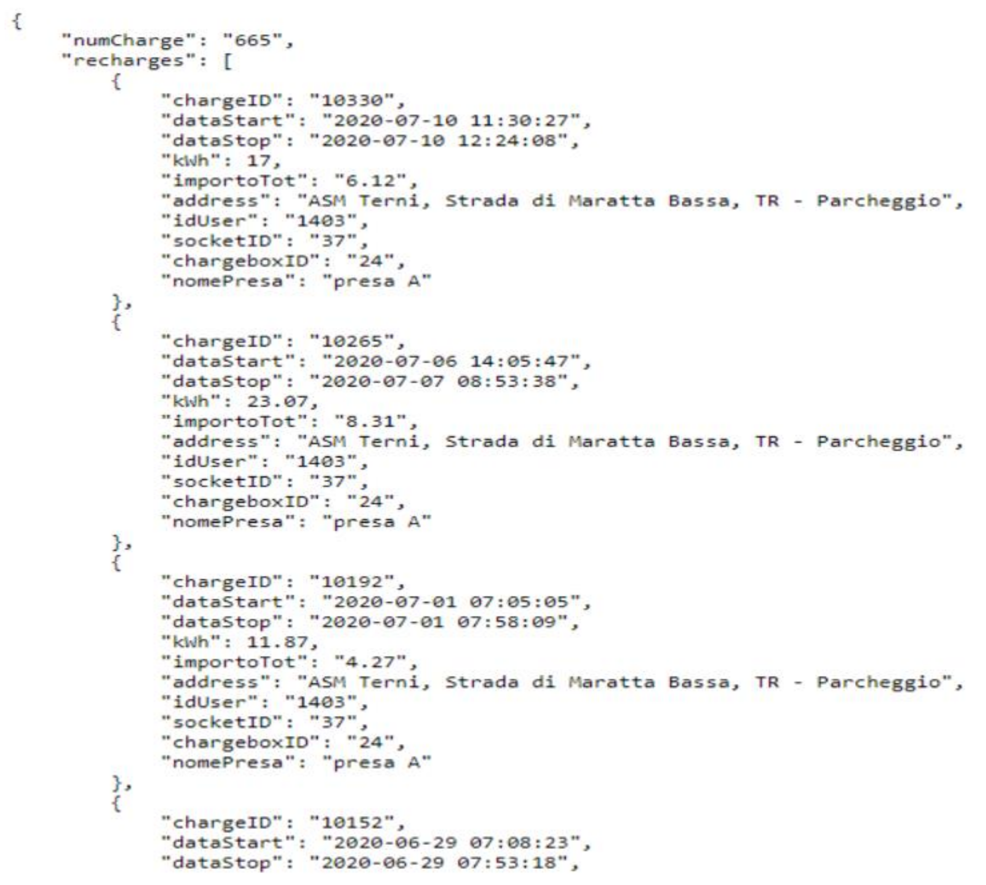


```
▼ JSON
  ▶ 0: [ 1592070900000, 1.2997499999999997 ]
  ▶ 1: [ 1592070960000, 0.9383275833333332 ]
  ▶ 2: [ 1592071020000, 0.6897014166666665 ]
  ▶ 3: [ 1592071080000, 0.5212634166666666 ]
  ▶ 4: [ 1592071140000, 0.46455979166666655 ]
  ▶ 5: [ 1592071200000, 0.4095077916666667 ]
  ▶ 6: [ 1592071260000, 0.3204924999999999 ]
  ▶ 7: [ 1592071320000, 0.21217899999999992 ]
  ▶ 8: [ 1592071380000, 0.12619095833333338 ]
  ▶ 9: [ 1592071440000, 0.131811666666667 ]
  ▶ 10: [ 1592071500000, 0.20021795652173915 ]
  ▶ 11: [ 1592071560000, 0.32752074999999997 ]
  ▶ 12: [ 1592071620000, 0.4914755000000001 ]
  ▶ 13: [ 1592071680000, 0.7497243333333333 ]
  ▶ 14: [ 1592071740000, 0.9908540869565217 ]
  ▶ 15: [ 1592071800000, 1.2678333333333331 ]
  ▶ 16: [ 1592071860000, 1.4613076923076922 ]
  ▶ 17: [ 1592071920000, 1.6652727272727275 ]
  ▶ 18: [ 1592071980000, 1.8388333333333335 ]
  ▶ 19: [ 1592072040000, 2.020833333333334 ]
  ▶ 20: [ 1592072100000, 2.210083333333333 ]
  ▶ 21: [ 1592072160000, 2.4185000000000003 ]
  ▶ 22: [ 1592072220000, 2.656375 ]
  ▶ 23: [ 1592072280000, 2.4933333333333336 ]
  ▶ 24: [ 1592072340000, 2.740458333333334 ]
  ▶ 25: [ 1592072400000, 2.5910416666666674 ]
  ▶ 26: [ 1592072460000, 2.670625 ]
  ▶ 27: [ 1592072520000, 2.6892083333333336 ]
```

Table 5: DEFM test case 2

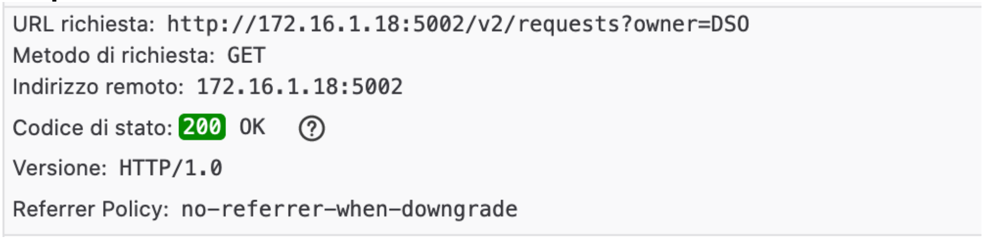
Test ID	DEFM_TC02
Feature(s) under test	Metering & data collection
Pass criteria	Real-time charging data provided by the EVSE is properly retrieved
Methodology	API request invocation

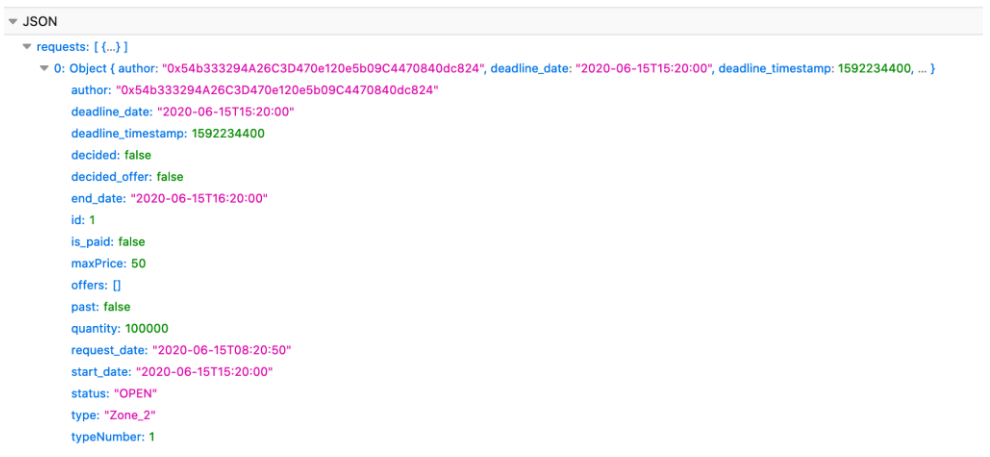
Result	**Request #1** (charging station info & real-time status):
	
	https://panel.spot-link.it/public/api/chargeboxes/{"chargeboxID":"24"}
	
	Method: GET
	
	Response #1:
	
	
	`{`
	`    "chargeboxID": "24",`
	`    "address": "ASM Terni, Strada di Maratta Bassa, TR - Parcheggio",`
	`    "latitude": "42.5673558",`
	`    "longitude": "12.6070454",`
	`    "maxPwrAC": "64",`
	`    "maxPwrDC": "0",`
	`    "drStatus": "0",`
	`    "idSocketA": "37",`
	`    "tSocketA": "type 2",`
	`    "stSocketA": "waiting",`
	`    "idSocketB": "38",`
	`    "tSocketB": "type 2",`
	`    "stSocketB": "waiting"`
	`}`
	
	Request #2 (charging sessions data):
	
	https://panel.spot-link.it/public/api/historyCharges/{"chargeboxID":"24"}
	
	Method: GET
	
	Response #2 (truncated):

```
{
    "numCharge": "665",
    "recharges": [
        {
            "chargeID": "10330",
            "dataStart": "2020-07-10 11:30:27",
            "dataStop": "2020-07-10 12:24:08",
            "kWh": 17,
            "importoTot": "6.12",
            "address": "ASM Terni, Strada di Maratta Bassa, TR - Parcheggio",
            "idUser": "1403",
            "socketID": "37",
            "chargeboxID": "24",
            "nomePresa": "presa A"
        },
        {
            "chargeID": "10265",
            "dataStart": "2020-07-06 14:05:47",
            "dataStop": "2020-07-07 08:53:38",
            "kWh": 23.07,
            "importoTot": "8.31",
            "address": "ASM Terni, Strada di Maratta Bassa, TR - Parcheggio",
            "idUser": "1403",
            "socketID": "37",
            "chargeboxID": "24",
            "nomePresa": "presa A"
        },
        {
            "chargeID": "10192",
            "dataStart": "2020-07-01 07:05:05",
            "dataStop": "2020-07-01 07:58:09",
            "kWh": 11.87,
            "importoTot": "4.27",
            "address": "ASM Terni, Strada di Maratta Bassa, TR - Parcheggio",
            "idUser": "1403",
            "socketID": "37",
            "chargeboxID": "24",
            "nomePresa": "presa A"
        },
        {
            "chargeID": "10152",
            "dataStart": "2020-06-29 07:08:23",
            "dataStop": "2020-06-29 07:53:18",
```

Table 6: DEFM test case 3

Test ID	DEFM_TC03
Feature(s) under test	Decentralized marketplace management
Pass criteria	All the marketplace functionalities are working as expected, tokens are transferred after a successful transaction
Methodology	API request invocation
Result	**Request**:  URL richiesta: http://172.16.1.18:5002/v2/requests?owner=DSO Metodo di richiesta: GET Indirizzo remoto: 172.16.1.18:5002 Codice di stato: 200 OK Versione: HTTP/1.0 Referrer Policy: no-referrer-when-downgrade **Response**:



```
▾ JSON
  ▾ requests: [ {…} ]
    ▾ 0: Object { author: "0x54b333294A26C3D470e120e5b09C4470840dc824", deadline_date: "2020-06-15T15:20:00", deadline_timestamp: 1592234400, … }
        author: "0x54b333294A26C3D470e120e5b09C4470840dc824"
        deadline_date: "2020-06-15T15:20:00"
        deadline_timestamp: 1592234400
        decided: false
        decided_offer: false
        end_date: "2020-06-15T16:20:00"
        id: 1
        is_paid: false
        maxPrice: 50
        offers: []
        past: false
        quantity: 100000
        request_date: "2020-06-15T08:20:50"
        start_date: "2020-06-15T15:20:00"
        status: "OPEN"
        type: "Zone_2"
        typeNumber: 1
```

The use cases and scenarios of the DEFM pilot have been presented in D5.1 “Baseline System and Measurements”; they describe a situation in which the DSO is facing a problem due to a condition of high penetration of renewable energy into the electricity grid. For this reason, the DSO requires flexibility through a marketplace that enables aggregators to make offers and to sign smart contracts for the provision of the service. At this phase of the project, several *Demand Response* (DR) campaigns were carried out and more than 5300 kWh were consumed (Figure 19) at the time and the place of the request through the charging of electric vehicles at the charging stations located at the electricity grid node where flexibility was needed. The flexibility provided not only guaranteed the stability of the electricity grid but also brought economic and environmental benefits; indeed, using renewable energy, as outlined by the Italian Institute for Environmental Protection and Research (ISPRA), it is possible to avoid emitting 491 g CO_2/kWh (Figure 21), an average amount of CO_2 associated to the energy mix that is purchased by energy retailers. Furthermore, if we consider that the average cost of energy purchased by the energy retailers is 0.26 €/kWh, as specified by Italian Energy Market Manager (GME), and the average cost of energy sold to the energy retailers by prosumers is 0.13 €/kWh, taking advantage of the renewable energy surplus to charge the electric vehicles, the economic saving is 0.13 €/kWh (Figure 20).

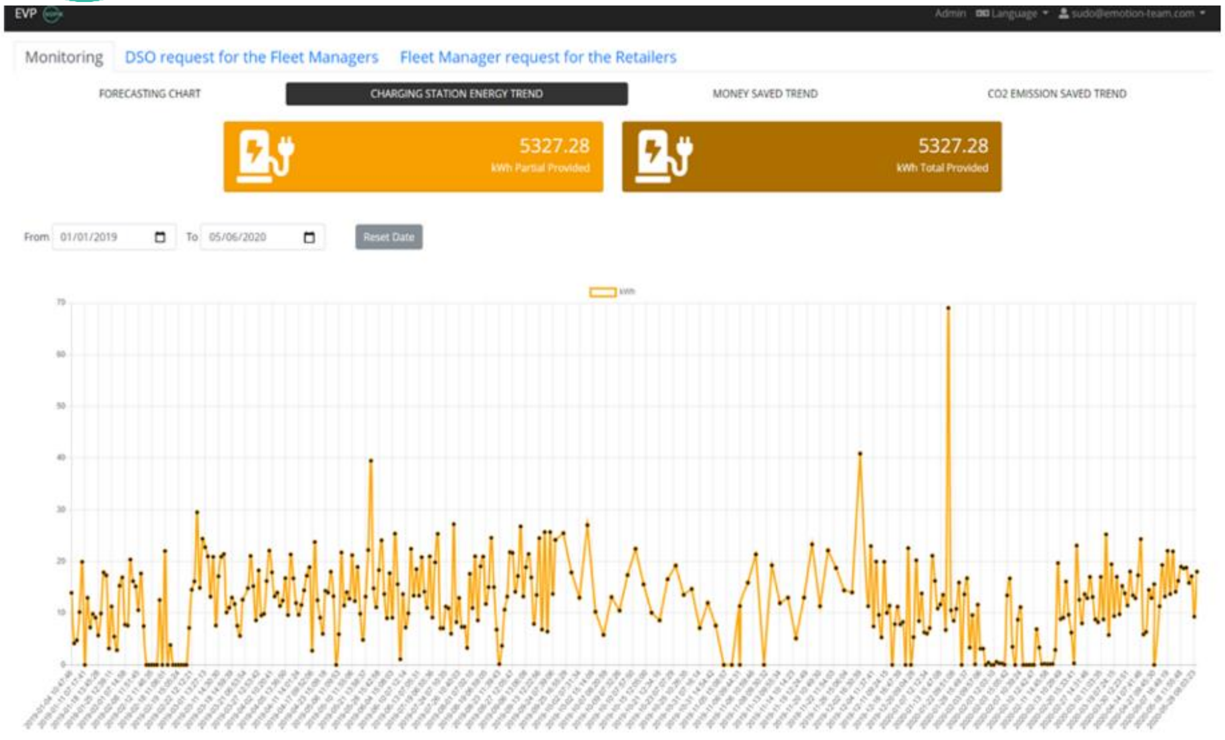


Figure 19: DEFM Pilot Charging Stations Energy Consumption Trend - 1,5 years

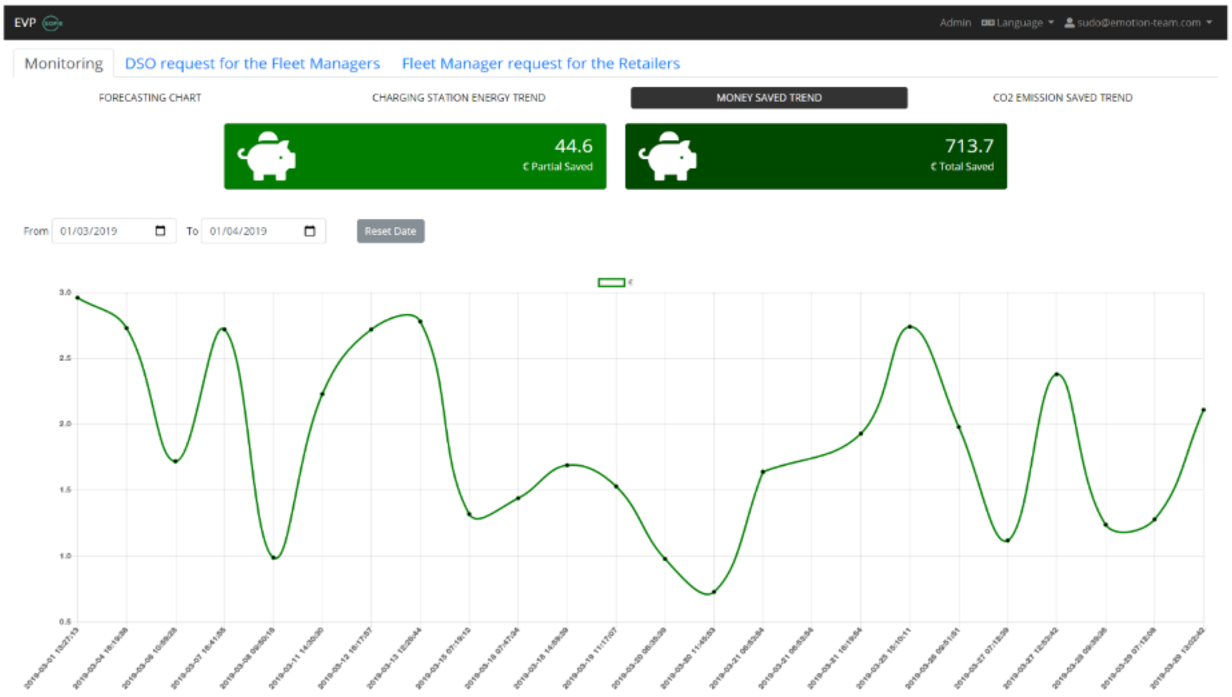


Figure 20: DEFM Pilot Charging Stations Money Saved Trend - 1 month

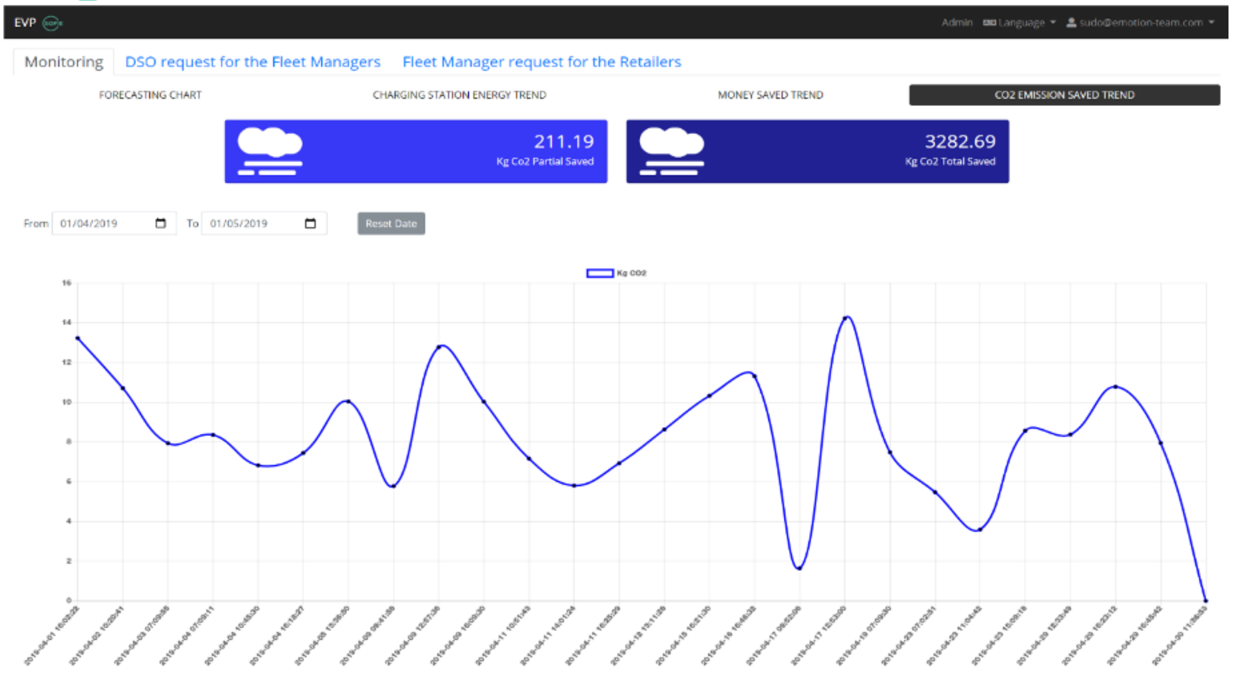


Figure 21: DEFM Pilot Charging Stations CO2 Saved Trend - 1 month

In order to evaluate the benefits coming from the flexibility marketplace based on the SOFIE architecture, the analysis of 1-year (2019) of consumption in the ASM headquarters was carried out. The data were collected and elaborated through a MATLAB script and they were referred to timestamps of 10 minutes.

Initially, the scenario where flexibility requests are not present was considered (ex-ante scenario). The local power generation comes from PV plants, whereas the consumption is due to the facilities and the EV charging sessions of a Renault ZOE equipped with a 52-kWh battery. In 2019, 402 charging sessions were recorded, with a total energy consumption of 5.1 MWh. The 22-kW nominal power of the charging station was assumed constant in the analysis, identifying the charging timestamps through the variation of the *State Of Charge* (SOC) recorded by the vehicle. Energy flows and indicators are reported in Table 7, which also shows the number of *Reverse Power Flow* (RPF) events (i.e., a 10 min timeslot in which energy balance is negative) as well as self-consumption (i.e., the ratio between the energy directly consumed by the loads and the produced energy) and self-sufficiency rate (i.e., the ratio between the energy directly consumed by the loads and the total energy used by the customer).

Table 7: Ex-ante Scenario

KPI	**Value**
Consumed energy	646,38 MWh
Produced energy	320,46 MWh

Injected energy	108,57 MWh
Charging station energy	5,14 MWh
Self-consumed energy	211,89 MWh
Self-consumption rate	66,12 %
Self-sufficiency rate	32,78 %
RPF events	10419

The second scenario is characterized by a different scheduling of the charging sessions according to some constraints, that are listed below:

- Charging sessions are shifted and scheduled only for reducing injected power flows (i.e., maximizing green energy consumption)
- The charging session would not be shifted if the new scheduling reduced the consumed energy up to an RPF greater than that reduced.
- The charging session is never divided in more sessions, but it is entirely postponed.
- Charging sessions are not rescheduled if the EV would not be at the EVSE premises for the new selected timeslot.

The constraints introduce an important improvement in the realism of the model. In this scenario the new charging session scheduling is accepted by the end users and the exchanged power between the ASM headquarter and the distribution grid in the ex-ante and in the first scenario is reported in Figure 22. According to these assumptions, 190 sessions would be shifted, reducing the injected energy of 1.9 MWh.

An example of the charging session rescheduling is reported in Figure 23Figure 22, in which the charging session on the 4th of April reduces the Reverse Power Flow (RPF), shaving the peak. It is worth noting that the energy consumed by the EV in case of RPF corresponds to an increase of the part of renewable energy consumed by the headquarters. This part should be injected into the grid in the absence of EV absorption whilst the charging session rescheduling promotes this effect. It can be evaluated as the *Self-Sufficiency for EV* (SSEV) which corresponds to the share of the PV production drawn by the EV charging sessions and, in the simulated scenario, an amount of energy 2.73 times higher than the first one was calculated.

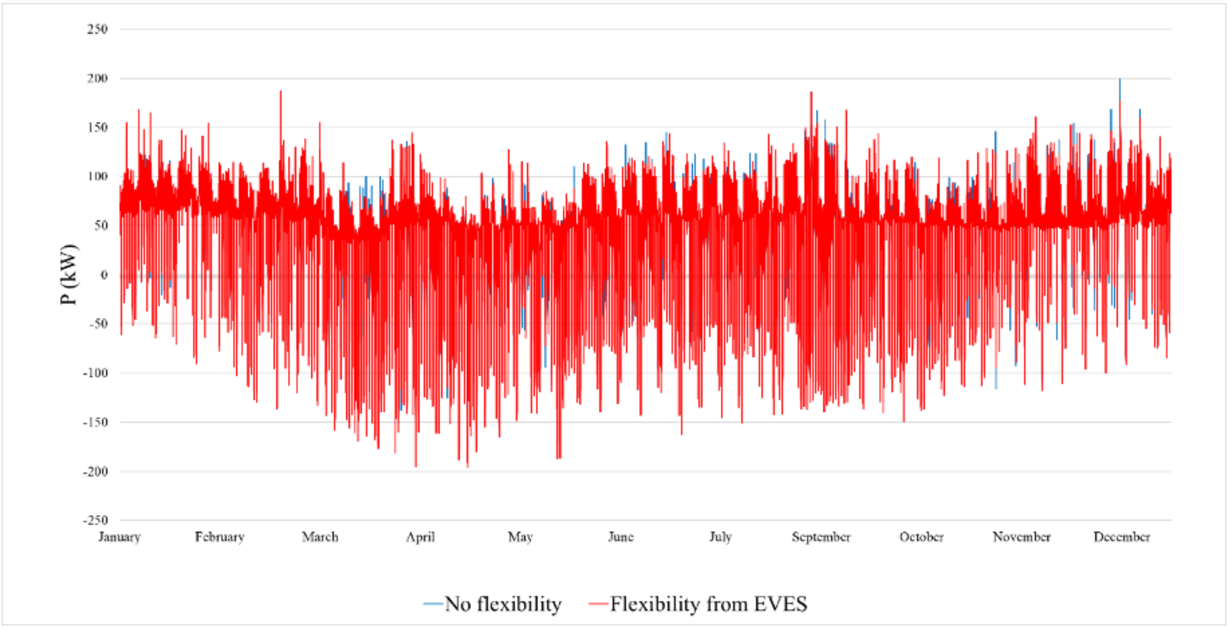


Figure 22: Exchanged power between the ASM headquarter and the grid in the ex-ante and first scenario

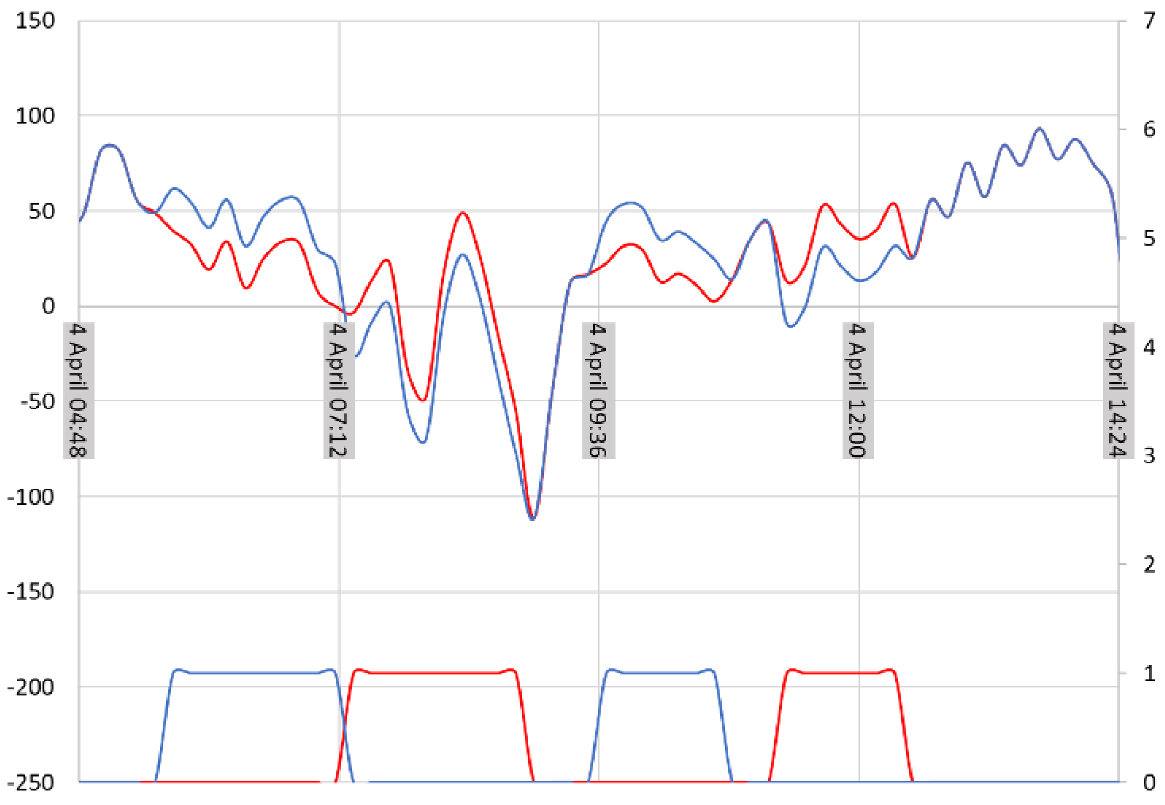


Figure 23: Exchanged power between the ASM headquarter and the grid on 4th April in the ex-ante and first scenarios

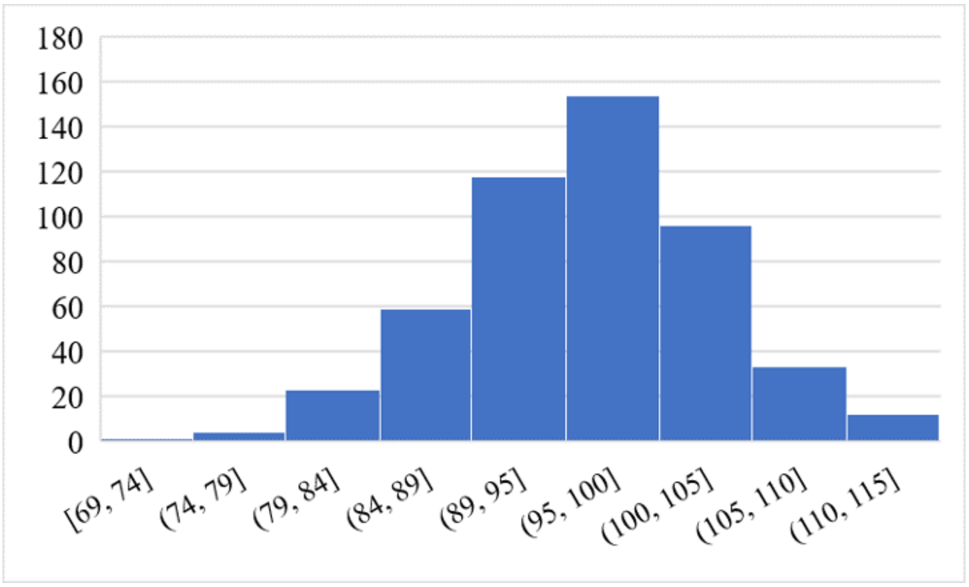


Figure 24: Distribution of number of EV charging sessions rescheduled

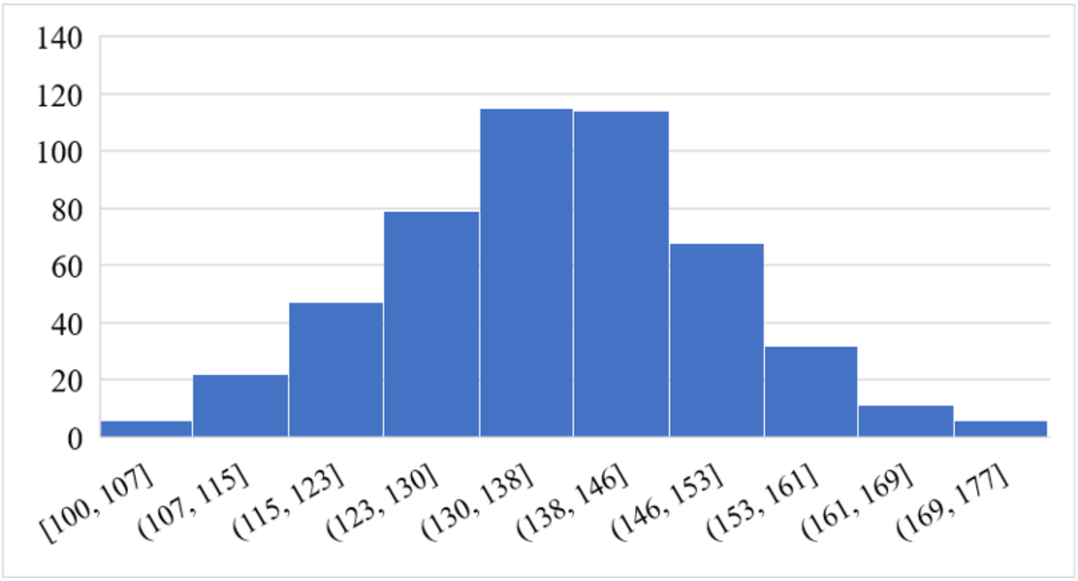


Figure 25: Renewable consumed energy rate increase [%]

The second scenario takes into account the possibility of the charging session rescheduling to be rejected, though the limitations mentioned above do not prevent the event. A 50% of probability that the displacement is accepted is considered in order to take into account the EV end user’s decision. This scenario is repeated 500 times to obtain a distribution of the main interesting index of the data analysis.

In particular, Figure 24 shows the distribution of the number of rescheduled charging sessions. The range between 95 and 100 displacements is the most present in the analysis, corresponding to the chosen 50% of probability. Figure 25 depicts the distribution of the renewable consumed energy rate increase as a percentage. The outcomes of the simulation are also summarized in Table 8, highlighting the increased self-sufficiency of the EVSE needs if an EV scheduler was applied.

According to the results, it can be argued that the flexibility from EV will have great potential if a proper session scheduling is applied. The next section presents the enabling technologies that have been implemented in the pilot site to carry out these scenarios, as well as some technical achievements.

Table 8: Comparison among scenarios

	Not scheduled charging sessions	Scheduled charging session	SSEV (MWh)	Energy from the grid for EVSE (MWh)
Ex ante	401	0	0.7	4.4
First Scenario	211	190	2.6	2.5
Second Scenario	[286, 332]	[69, 115]	[1.4, 1.8]	[3.3, 3.7]

With respect to the KPIs, a group of business goals were defined in D5.1, as well as some values to be achieved by the validation of the SOFIE project. Based on validation results, the KPI values are as follows:

- KPI_DEFM_6, RPF reduction Amount of RPF is on average 13.7 kWh/ day, applicable only if a charging session happen very close to the expected value reported in D5.1 (i.e., 15kWh/day).
- KPI_DEFM_7, Power losses reduction, considering that about 3 MWh could be consumed when it is locally produced, a beneficial effect is the reduction of that power in the Medium Voltage network and therefore power losses would be reduced up to 75% (i.e., a quarter of losses are produced in the LV part of the grid).
- KPI_DEFM_8, Voltage under the limits Voltage waveforms, simulation results show that maximum and minimum voltages are 0.98 p.u. and 1.06 p.u., respectively.
- KPI_DEFM_9, Green energy consumption, the increased share of consumption from green energy producers has been measured as the reduction of RPF and therefore the increased share of consumption drawn from green energy producers (i.e., about 13.7 kWh/day as in KPI _DEFM_6 RPF).
- KPI_DEFM_10, EV fleet manager metrics Involvement in DR campaign provide advantageous energy price for EV Fleet Manager, due to DSO benefits and Retailers auction Monetary savings Measure the money saved involving EV fleet in DR Campaigns: energy cost in DR campaign vs energy cost in non-DR campaign money saved: 0,13 €/kWh.

5.3.1 Future validation results

In D5.4, Final Validation & Replication Guidelines, the final results of the activities carried out during the whole project on the Terni pilot site and the outcome of the implementation of the cross-pilot testing will be represented, described and analysed.

6. Mixed Reality Mobile Gaming Pilot

6.1 Pilot overview

The focus of the Mixed Reality Mobile Gaming (MRMG) pilot is to explore how DLTs can be used to provide new gaming features for players, as well as to validate the potential of location-based IoT gaming use cases. The pilot seeks to overcome the known technical issues of DLTs with respect to scale, in order to cost-effectively support millions of active users per day.

As our earliest use case, we prototyped a game that enables players to collect and trade in-game content, swap or trade with other players (e.g., characters, weapons, equipment, parts), leveraging DLTs to provide player ownership of the asset, transparency and consistency of asset attributes and transactions. Attributes, or the “DNA” of the in-game assets were published on the blockchain.

As our second use case, we developed a Scavenger Hunt game prototype in order to explore location-based IoT gaming. In the game, the player starts a hunt, which takes them on a journey of predetermined real-world locations. At each location, a Bluetooth Low Energy (BLE) beacon is deployed, either indoors or outdoors. When the mobile game client detects the beacon, it means that the player has arrived at the correct location, and they receive a task in the form of a question. By observing their real-world surroundings, the player can answer the question and receive the clue on where the next correct location is. At the end of a hunt, the player receives rewards that can bring in-game advantages in the next hunts.

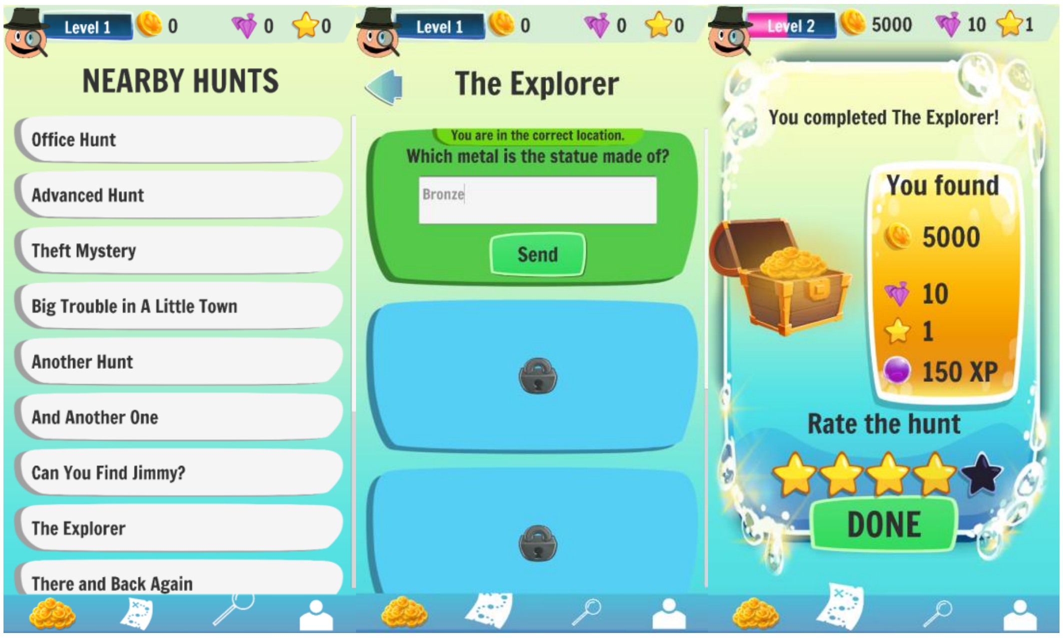


Figure 26: Scavenger Hunt game prototype. Starting, playing, and ending a hunt on a mobile client.

As additional rewards, the player receives items that are stored on a distributed ledger as non-fungible tokens. To browse and manage these items, a companion application was created - Blockmoji. In this mobile application, the player can see which items they own, and equip or unequip them on their virtual avatar. Shared items between Scavenger Hunt and Blockmoji demonstrate that it is possible to share the same items between multiple games, where it is up to the game designers on how to interpret the attributes of the player’s Blockmoji items and which in-game benefits they would bring. In our Scavenger Hunt game prototype, the Blockmoji do not bring in-game benefits, but instead, the game acts as a source of these items.

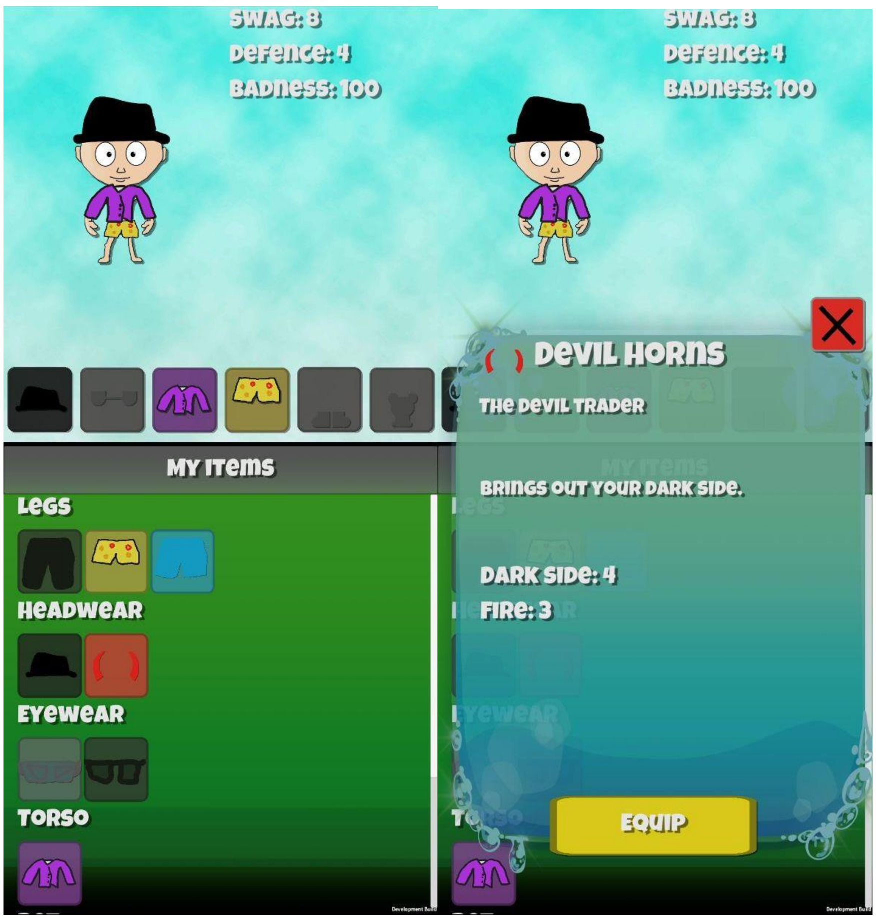


Figure 27: Viewing and equipping items in Blockmoji.

In addition to the use cases, we are now working with Aalto on integrating Interledger and Marketplace components into our pilot. In addition, the Discovery and Provisioning component can be used to discover IoT beacons and add them to the database for a location-based game, such as for our Scavenger Hunt prototype.

Before the end of the project, we plan to continue integrating the above-mentioned SOFIE components into our pilot, and validate our pilot use cases as described by the requirements in the validation matrix in D4.4 (page 49), as well as below in section 6.3. Moreover, we plan to develop an additional use case relating to a decentralized mobile advertisement profile. This use case would serve as a reference implementation of the IAA component. Furthermore, we plan to complete the playtesting of the Scavenger Hunt prototype, if the COVID-19 situation will allow it (tests need to be organized in the office spaces). Regardless, we will validate all requirements that are listed in the validation matrix in 2020, as these do not require physical presence in the office. The technical performance of our pilot has been measured and analysed, as seen in Section 6.3. In addition, several requirements have also been validated, also as seen in Section 6.3

Since D5.2, we have replaced multiple validation requirements of our pilot in order to better reflect which functionalities we expect from our use cases, as well as to better align with our planned mobile ads use case. The new requirements have IDs MRMG9.1-4, as can be seen in the updated validation matrix.

6.2 Pilot platform architecture and services

Our pilot employs a hybrid server-blockchain architecture in order to utilize the benefits of both worlds: the speed of a traditional game server and the transparency, traceability and a sense of true ownership of virtual assets brought by distributed ledger technology.

6.2.1 High-level architecture

In this hybrid architecture, most of the game logic runs on the game server. The server is written in Python using the Flask framework and deployed on AWS Lambda. For the mobile client to communicate with the server, and for creating new hunts, a REST API is used. The server is also connected to the DynamoDB database. The Web App, that is the AWS Console, can be used by the game designer to manage the game's hunts and players.

An additional Node.js Fabric SDK server is used, which runs on an AWS EC2 instance. This reduces the coupling between the game and the blockchain network. The Hyperledger Fabric network is deployed on AWS Managed Blockchain and exposes specific functions to the Node.js server for the game server to access. The Fabric consortium includes two organizations, with one peer node each. The network also includes an orderer node, two certificate authorities, and one channel to log all transactions.

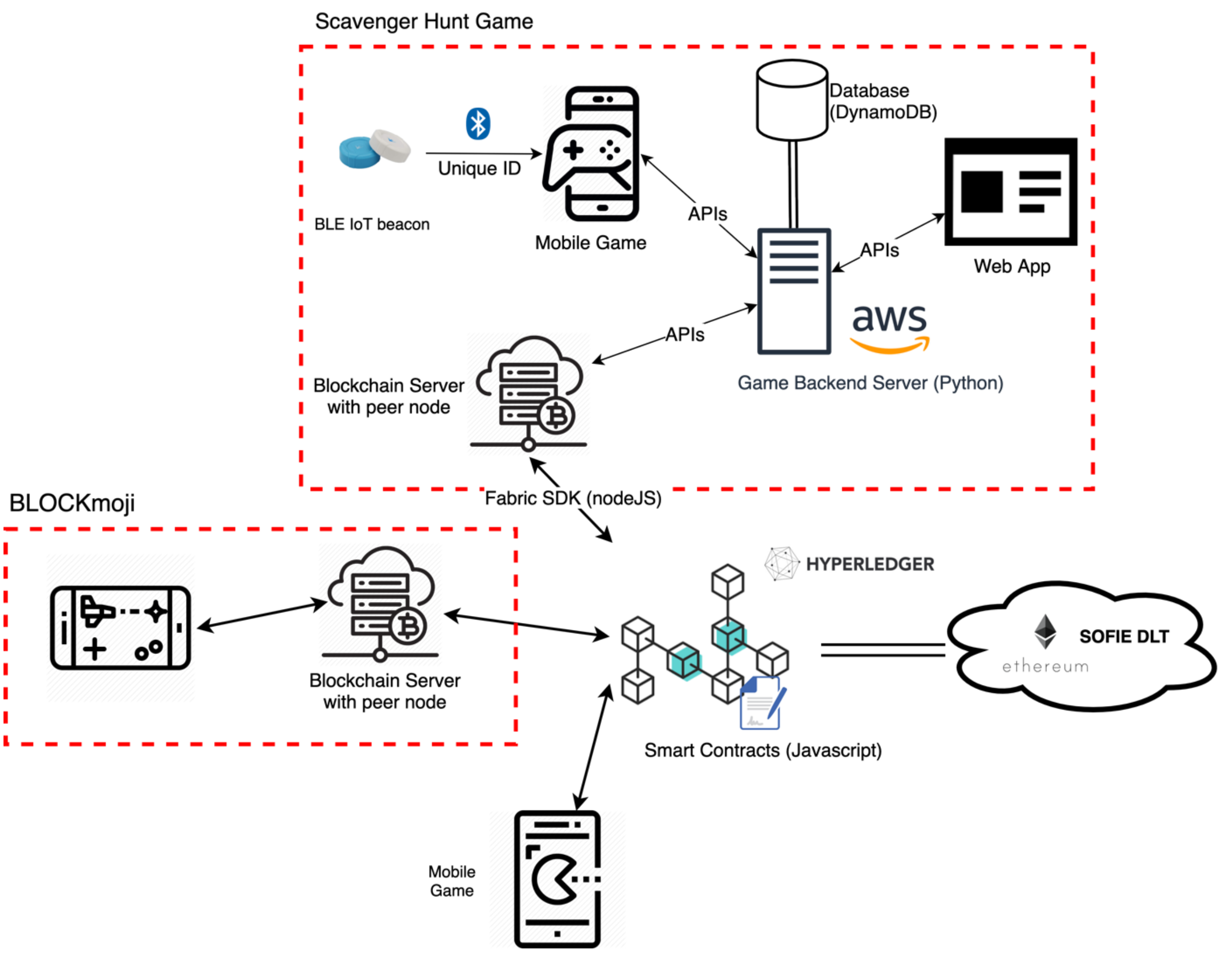


Figure 28: The high-level architecture of our pilot.

Figure 28 demonstrates the high-level architecture of our pilot, showing how the Scavenger Hunt game prototype connects with our Hyperledger Fabric platform. As can be seen from the figure, additional games can be connected to the same network, sharing the virtual items between many games. That is exactly how Blockmoji and Scavenger Hunt use cases relate:

both use items that are stored on the managed Hyperledger Fabric blockchain. In order for the player to experience this interoperability, they have to use the same wallet address in both applications. In our prototypes, the player's device ID is automatically used to identify the player in both use cases.

The Hyperledger Fabric managed blockchain is used for fast transactions in order to support as many concurrent users as possible. For transparency and traceability during trading of the assets, a public Ethereum network will be used, marked as “Sofie DLT” in Figure 28. To enable this, the Marketplace and Interledger components will be utilized.

6.2.2 Flow of the game

As the player downloads and opens the Scavenger Hunt game application, they are automatically signed in based on their device ID. If they have not logged in before, a new account is created, along with a decentralized identity on the blockchain. From then on, the game flow is as follows, from the player's perspective:

1. The player sees available nearby hunts based on their coarse GPS location. The hunts are stored in a DynamoDB database and are retrieved through the Python game server.
2. The player sees the hunts' information, such as difficulty, user rating, and item rewards.
3. After starting a hunt, the game enables Bluetooth scanning on the mobile device, so as to discover IoT beacons. The player receives the clue to the first location from the game server.
4. As the player physically visits the correct location, the mobile device detects the BLE beacon in that location. The beacon advertises its ID, which the game compares to the ID of the beacon in that location, checking that the player is indeed in the correct location. The player receives the task for that location in the form of a text question. By observing their physical surroundings, the player answers the question correctly and receives the clue to the next location.
5. The user visits all locations by solving the clues and answering questions. The player can skip any task or receive relevant hints by consuming in-game tokens (stars and gems, respectively) that are received by completing previous hunts.
6. As the player completes the last task in a hunt, they receive the rewards. The rewards are moved from the hunt's escrow to the player's wallet address. This includes non-fungible Blockmoji item rewards.
7. After the player has received an item reward from completing a hunt in Scavenger Hunt, the item appears in the player's Blockmoji collection. There, the player can browse their items, and equip or unequip items from their avatar. Equipping and unequipping items are write functions on the Hyperledger Fabric ledger.

6.2.3 APIs description

The Scavenger Hunt and Blockmoji client communicate with the game server through a REST API. Numerous methods are utilized throughout the flow of Scavenger Hunt, such as getting the nearby hunts, clues to the next location in a hunt, question tasks, Blockmoji item information, and updating the backend to reflect the player's progress in a hunt. Outside of the game, the game designers can create new hunts and item rewards with POST methods. In Blockmoji, methods for getting and updating equipment are utilized.

Some of these API calls get forwarded to the Node.js server, which, in turn, forwards them to the Hyperledger Fabric network. When a game needs to get item information, a read call is forwarded to the ledger. When a player completes the hunt and receives rewards, the call is forwarded to the ledger to move the rewards from the hunt's escrow to the player's account.

Method	API endpoint	Description
POST	/api/players	Register the Player and create an account on the fabric network
GET	/api/player/<androidId>	Returns the details of the single player, specified by the id parameter
GET	/api/player/<androidId>/startedHunts	Returns the details of the started hunts by the player, specified by the id parameter
GET	/api/player/<androidId>/completedHunts	Returns the details of the completed hunts by the player, specified by the id parameter
PUT	/api/player/<androidId>/<huntID>/star	Players can use the stars for game related tasks, player ID and hunt id must be specified.
PUT	/api/player/<androidId>/<huntID>/<int:step_num>/answer	Checks for the answer of the clue, player id, hunt id and clue number must be specified.
GET	/api/player/<androidId>/<huntID>/star	Returns the details of the stars used by the player in the hunt, specified by the player and hunt id parameter
GET	/api/player/<androidId>/myassets	Returns the details of the assets owned by the player, specified by the player id parameter
PUT	/api/player/<androidId>/<huntID>/clues/<int:step_num>/hint	Use a hint for the clue, specified by the player id, hunt id and clue number parameter
POST	/api/hunts	Creates new hunt for the game.
GET	/api/hunt/<huntID>	Returns the details of the hunts, specified by the id parameter
PUT	/api/hunt/<huntID>/rate	Update the hunt rating using Id specified.
GET	/api/hunt/<huntID>/clues	Returns the details of all the clues in the hunt, specified by the hunt id parameter.
GET	/api/hunt/<huntID>/clues/<int:step_num>	Returns the details of the specific clues in the hunt, specified by the hunt id and clue number parameter.
GET	/api/hunt/<huntID>/clues/<int:step_num>/hint	Returns the details of the specific clues hint in the hunt, specified by the hunt id and clue number parameter.

GET	/api/hunt/<huntID>/task/<int:step_num>	Returns the details of the specific task in the hunt, specified by the hunt id and instance id
GET	/api/<androidID>/hunt/nearby	Returns the details of all hunts near the location of the player.
PUT	/api/<androidID>/<huntID>/start	Update the started hunts for the player.
GET	/api/asset/<assetId>	Returns the details of the specific assets, specified by id parameter.
GET	/useritems/<username>	Returns the details of the all the assets owned by the player, specified by id parameter.
PUT	/updateEquip	Update the equipped items of the player.
PUT	/updateOwn	Update the owned items of the player.
POST	/item	Creates new item for the game.
POST	/identity	Enrolls the player identity in the blockchain.

6.3 Validation results

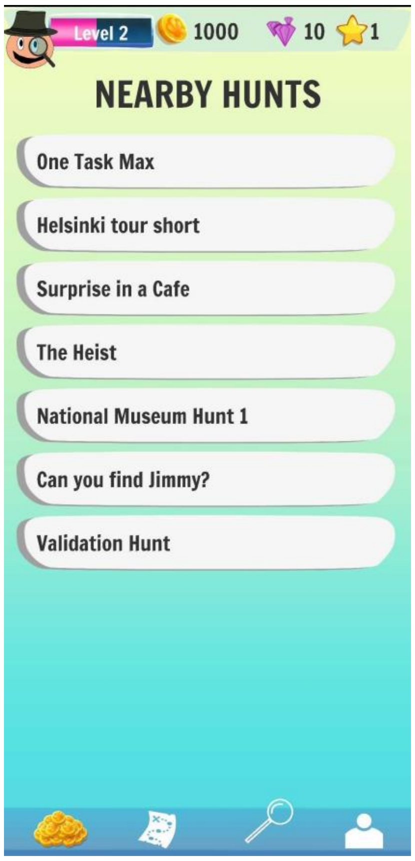
Due to the COVID-19 situation, physical engagement of end users in the form of internal playtesting has been interrupted. Therefore, we have instead shifted our focus to DLT and BLE beacon performance tests. In addition, there has been progress in validation results regarding the Scavenger Hunt and Blockmoji use cases.

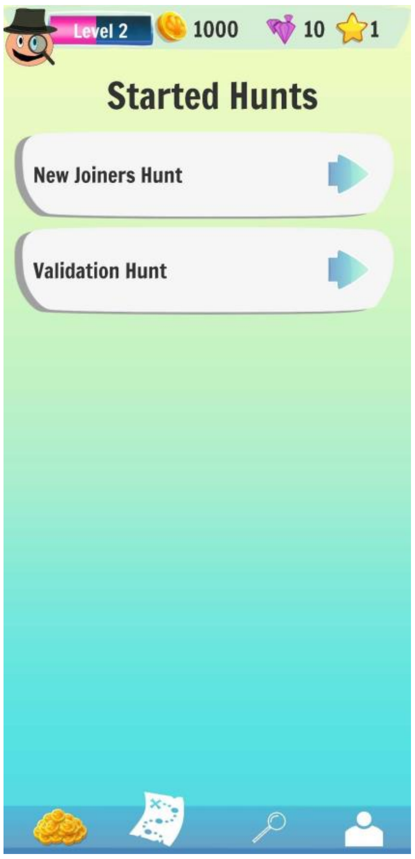
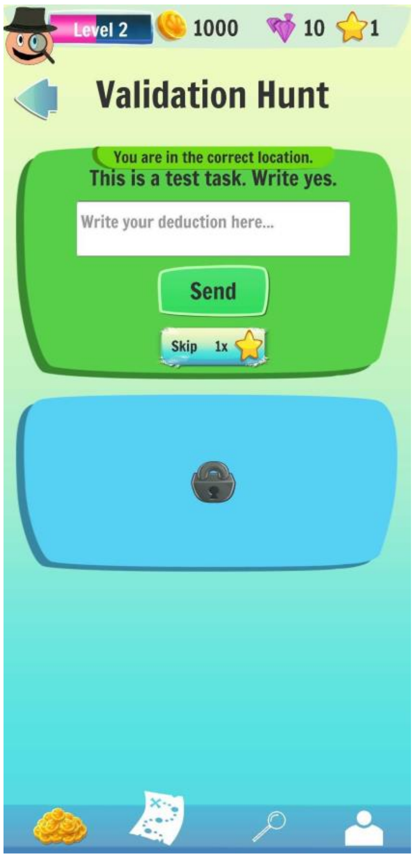
6.3.1 Validation Results

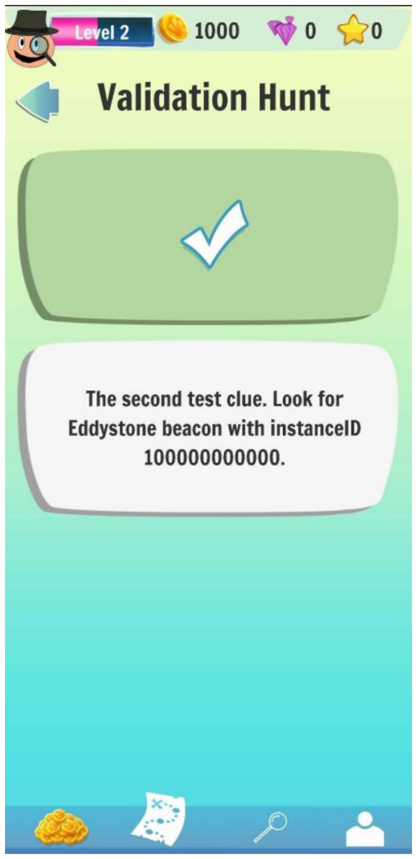
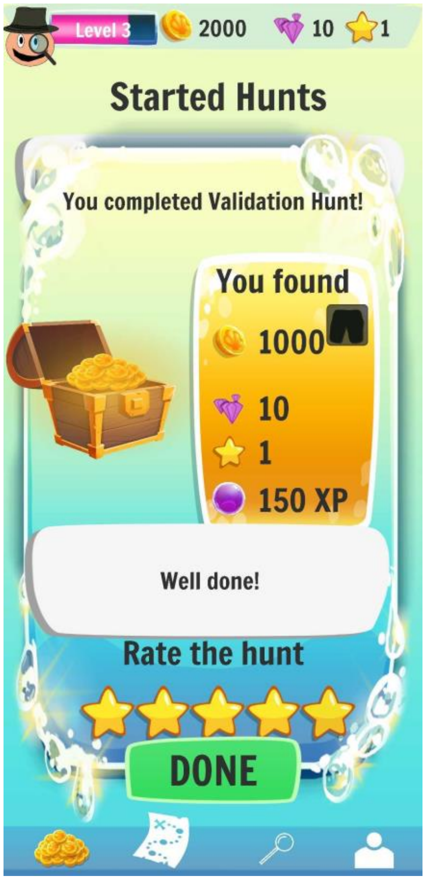
To this date, we have validated a number of our pilot requirements, as seen inen as screenshots in the table.

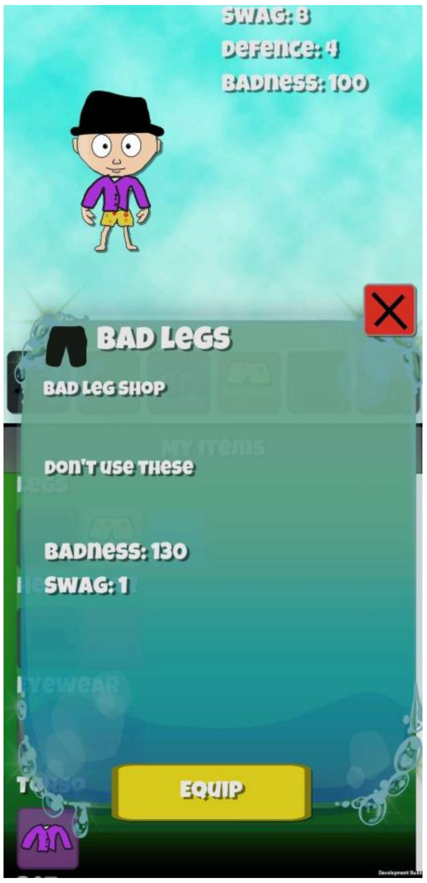
Table 9. Proof of requirement fulfilment are given as screenshots in the table.

Table 9: Requirement validation

Requirement ID	Requirement Description	Test Description	Screenshot
REQ_ MRMG1.1	Game challenges are accessible using the Android application	In the test, the user opens the Scavenger Hunt game application and enters the Nearby Challenges tab. The user should see a list of (uncompleted) challenges that start in GPS coordinates that are within a set radius from the user. The requirement is met if the nearby challenges that exist on the backend are indeed visible in the Nearby Challenges tab.	 Level 2 1000 10 1 NEARBY HUNTS One Task Max Helsinki tour short Surprise in a Cafe The Heist National Museum Hunt 1 Can you find Jimmy? Validation Hunt

REQ_ MRMG1.2	Players can join any nearby challenge from the game app.	The requirement is met if a challenge is added to the list of the player's current challenges, after the player presses the Start button in the client.	 Level 2 1000 10 1 Started Hunts New Joiners Hunt Validation Hunt
REQ_ MRMG1.5	Players should receive unique tasks when near the IoT beacons based on their challenge.	The requirement is met if a user standing next to a BLE beacon receives a task in the mobile application.	 Level 2 1000 10 1 Validation Hunt You are in the correct location. This is a test task. Write yes. Write your deduction here... Send Skip 1x

REQ_ MRMG1.6	Players should be able to skip any task and receive the location of the next IoT beacon using the In-App tokens.	If a player has Star items in-game, they can use one star to skip a task. The requirement is met if, when presented with a task and using a star, the current task auto-completes and the user receives the clue to the next beacon.	 Level 2 1000 0 0 Validation Hunt The second test clue. Look for Eddystone beacon with instanceID 100000000000.
REQ_ MRMG1.8	System should automatically calculate rewards after player has completed a challenge	After a player completes a challenge, the requirement is met if the player sees rewards in the client application.	 Level 3 2000 10 1 Started Hunts You completed Validation Hunt! You found 1000 10 1 150 XP Well done! Rate the hunt DONE

REQ_ MRMG7.1	Blockmoji item rewards be can offered to players through challenges	If a challenge offers a Blockmoji item reward, the player should see it in their mobile application reward screen after completing the challenge.	 SWAG: 8 Defence: 4 BADness: 100 BAD LeGS BAD LeG SHOP Don'T use These BADness: 130 SWAG: 1 EQUIP

6.3.2 Technical performance tests

In this section, we evaluate how the new technologies, such as blockchains and IoT, perform in the mobile gaming ecosystems. We describe experiments evaluating the proof of concept implementation of the game on an AWS managed Hyperledger Fabric network. We have performed multiple experiments measuring the time taken for the end-to-end process to execute a transaction, and the throughput of the Fabric network. Our Hyperledger Fabric test network consists of two organizations, each with one peering node. There is one channel where all the entities perform the transactions and one solo ordering node for the creation of the blocks. The chaincode was written in the Go programming language. We performed multiple experiments to test the performance of Fabric with "creating new data", "querying data" and "updating data" using the custom chaincode written for the games.

6.3.2.1 Response Time

For a quantitative system performance evaluation, various measurable metrics are required. The most common performance metric of any system is the response time required by the system to execute read and write requests. In our case, where the gaming system utilized a hybrid architecture of a centralized backend and a distributed ledger, the response time metric corresponds to the time that the system performs read or write transactions of the various game functions. We ran the experiment 50 times before taking an average and found that it takes on average 2.247s for a write request with a confidence interval of 0.011s and on average read request takes 0.026s with a confidence interval of 0.0007s.

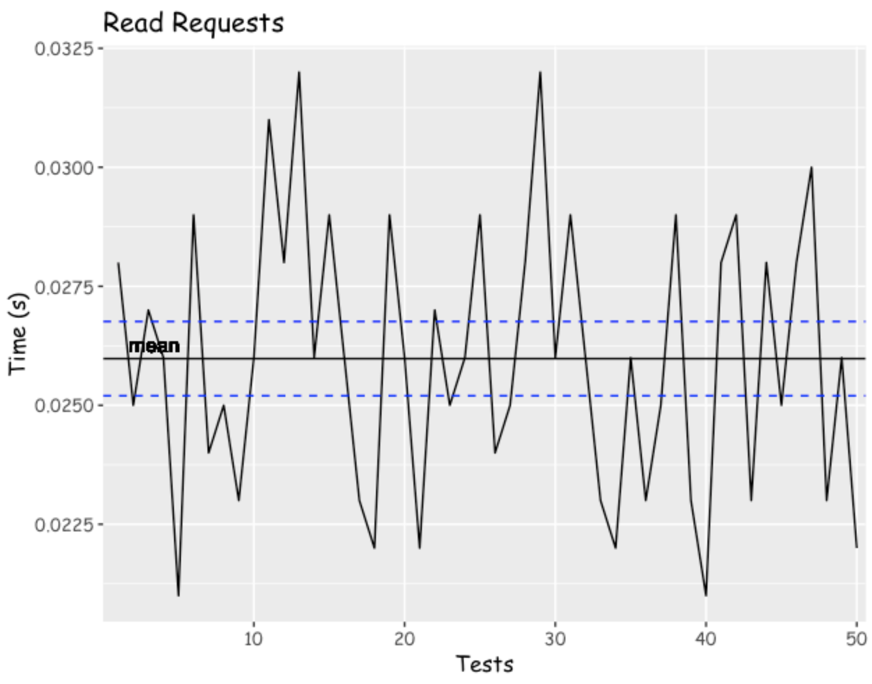


Figure 29: Response time for Read requests

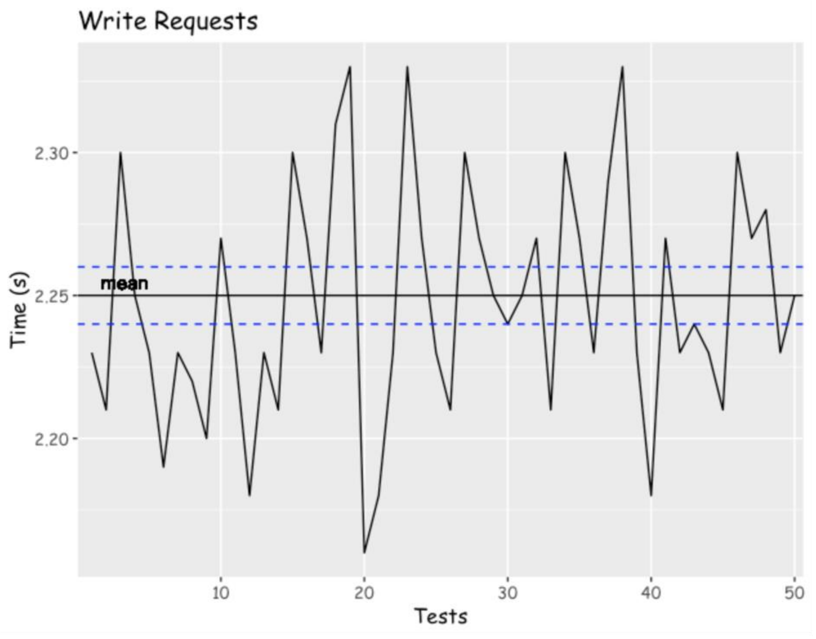


Figure 30: Response time for Write requests

The figures above illustrate the average read and write response times and the variation over 50 runs of the scenario. Blue dotted lines show the 95% confidence level for the mean. This delay is closely linked with the average time for block generation in the Fabric network, i.e. 2s. This shows that block generation has the highest impact on the writing requests.

6.3.2.2 Throughput

In order to determine the throughput of the proposed architecture, we used Hyperleder Caliper, a blockchain performance benchmark framework, which allowed us to test different blockchain solutions with custom use cases and get a set of performance test results.

Fixed Rate

In the first experiment, we measured the throughput of the architecture by submitting multiple transactions to the blockchain at the Fixed Rate. We ran the test with a fixed rate of 250 transactions per second (TPS) until the total number of transactions reached 10000. We performed individual tests for creating, querying, and updating the data.

With a fixed transaction arrival rate, the throughput for writing new data on the blockchain increased linearly as expected until it flattened out at around 177 TPS, the saturation point. When the arrival rate was close to or above the saturation point, the latency increased.

Table 10: Performance validation – fixed rate

Name	Send Rate	Total # of transactions	Failed transactions	Max Latency	Throughput (TPS)
Create	250 TPS	10000 transactions	0 %	-	177 TPS
Query	500 TPS	10000 transactions	0 %	-	351 TPS
Update	250 TPS	10000 transactions	0.24 %	-	191 TPS

As shown in Table 10 above, the send rate for querying the blockchain was set to 500 and it reached its saturation point at around 351 transactions per second with the latency increasing significantly around it. In the last test, throughput for updating the data on the blockchain came to be 191 TPS, which is mainly depended on for writing the new data on the blockchain.

Composite rate

In the second experiment, we ran the tests to determine the throughput of the architecture by submitting transactions at the Composite Rate. This was done to simulate a real-life scenario and benchmark the blockchain network. We performed these tests with a duration-based round, a total of 100 seconds. In this case, an initial 30 seconds normal workload is followed by a 30 seconds intensive workload, which is followed by 10 seconds of low workload and ending with another 30 seconds of normal workload. We performed individual tests for creating, querying, and updating the data.

Table 11: Performance validation – composite rate

Name	Total # of transactions	Failed transactions	Max Latency	Throughput (TPS)

Create	100 seconds	0 %	-	128 TPS
Query	100 seconds	0 %	-	307 TPS
Update	100 Seconds	0.42 %	-	135 TPS

6.3.2.3 Modeling Active Player Support

In order to support the game with a maximum number of active users without any performance degradation, we calculated the daily active number of users supported by our hybrid architecture and maximum number of concurrent users at a given time. We make the following assumptions without loss of generality:

- Ideal network conditions of the Player.
- Centralized backend is fully scalable depending on the requests.
- An average user queries 10 read transactions and 6 write transactions per session on blockchain.
- An average user plays for one hour per day.
- Throughput of the backend to be taken from composite rate tests i.e., Table 11

These transaction frequency numbers are consistent with early play tests that were performed for the game. This number of transactions translates to completing one hunt in an hour, on a daily average.

Maximum users supported / hour = (Maximum transactions / hour) / Average number of transactions per hour

	Throughput	**Maximum transaction / hour**	**Maximum users supported**
Read transactions	307 TPS	1,105,200	110,520
Write Transactions	128 TPS	460,800	76,800

From the above calculation, it can be seen that the major limiting factor are the write transactions on the blockchain. Using that, we calculated the maximum number of players that can play the game throughout the day without any delays.

Maximum players / day = 76,800 x 24 = 1,843,200

6.3.2.4 Beacon Detection Time

In addition, detection times for BLE beacons have been measured. We measured how long it takes for a mobile device to register a beacon in various room conditions. These results are summarized in Table 12. A mobile device of the model Huawei Nova 3 was the detecting device in the performance tests.

Table 12: Beacon performance results.

Case	Average time	Standard deviation	Sample size
1: Phone and beacon on the same table, with 5 other beacons nearby	6.1 s	6.8 s	31
2: Case 1 repeated at a later time of day	3.9 s	1.6 s	31
3: Lounge area. 5 m distance with beacon behind a concrete pillar	3.3 s	3.2 s	31
4: Open kitchen area. 10 m away from the beacon	10.4 s	10.5 s	31
5: Sofa area, 3 m away from the beacon	5.1 s	3.5 s	31
Total	**5.8 s**	**6.4 s**	**155**

These results show that BLE beacons are not necessarily appropriate for real-time location-based games. Detection delays are very noticeable to the players. If the player is clearly in the correct room and it takes several seconds to detect the beacon, the player might get confused on whether or not they are in the correct room, or whether their mobile device only now detected a beacon from the previous room. Indeed, localization may be ambiguous if players are walking fast. In addition, as can be seen from the results, the deviation from the average detection time is very wide. This can result in inconsistent performance results. Specifically, as Cases 1 and 2 show, detection results may in the same physical place may not be consistent over time. In addition to the beacon detection delay, the player also experiences delay that is due to the communication with the server. The Scavenger Hunt prototype is playable if the correct locations are far apart and not in very precise locations, but the aforementioned quirks have been noticed to negatively affect the player experience.

6.3.3 Future plans

For increased insight in the evaluation, we had planned a gameplay test, which is not part of any REQ_MRMG requirements listed. The idea is that several playtesters would play an in-office challenge and answer a feedback questionnaire after it. As mentioned before, this playtesting was interrupted during spring 2020, but we plan to resume these tests in the second part of 2020. The internal playtests should test the suitability of IoT beacons for general location-based gaming from the perspective of players, and gauge player engagement. The current feedback form includes the following questions:

- Points of frustration
- Points of enjoyment
- How noticeable are beacon delays to real players? Do beacons often position players in incorrect rooms?
- What did players think of text tasks, and whether they have ideas for other kinds of tasks
- How interesting indoor location-based games are

- Other application ideas. How can this infrastructure be used outside a Scavenger Hunt game?
- Open feedback

In addition, several new requirements have been added to the validation matrix to accommodate our new planned mobile ads use case. These requirements have IDs REQ_MRMG9.1-9.4 as described in D4.4. As we work on this new use case in H2 2020, these requirements will be fulfilled. The rest of the requirements that are not included in Table 9 shall also be validated for D5.4.

7. SMAUG

7.1 Purpose and use case

The *Secure Marketplace for Access to Ubiquitous Goods* (SMAUG), is a decentralised and open marketplace where smart locker owners can advertise their smart lockers for rent, and potential smart locker renters can place bids, to get the authorisation to use those smart lockers. Smart locker owners publish smart lockers on the marketplace by creating a request, i.e., a request for offers. The bids that smart locker renters place for those requests are called offers.

SMAUG is intended as a reference implementation, to show how all the different SOFIE components can be used together to develop a system that benefits from all the properties that the SOFIE framework provides. Furthermore, SMAUG is being developed by LMF as a WP3 leader, and this means that an important target for SMAUG is to provide high-quality feedback to SOFIE component developers about the set of features the components offer, their level of reusability and extensibility, and their quality, related to how easily they can be integrated into systems other than the four pilots under development. This is achieved by following a “learn by doing” approach and testing the components via direct integration into a system developed from scratch during the last year of the project. LMF will use SMAUG along with providing a CI/CD environment, as already presented in D3.3, Integration Plan, to ensure high standards of quality for all SOFIE components.

7.1.1 Use case

SMAUG showcases the potential unlocked by all the SOFIE components when combined into a single use case. Specifically, the SMAUG use case concerns the creation of a marketplace that is open, decentralised, and secure. The marketplace is *open* because it allows anyone to put lockers for rent, and to find available lockers for rent by interacting with the marketplace, where access to the smart locker can be purchased. The marketplace is *decentralised* because it is rooted in a blockchain, which provides availability (by removing single points of failure) and avoids the concentration of all the data about marketplace interactions and users into a single place, with all the risks that this entails. Being rooted in a blockchain, the marketplace also benefits from the *security* guarantees deriving from that. Specifically, a blockchain guarantees non-repudiation of the actions performed by the different entities which, together with immutability, provides a strong tool in case of dispute resolutions.

The goal of the marketplace is, like several existing marketplaces, to meet the supply and demand of smart lockers. Three different actors are involved in SMAUG:

- **marketplace owners (MPO)**: entities owning and managing one or more instances of a decentralised marketplace that enables interactions between the supply and demand of smart lockers. Each marketplace defines its own set of policies, such as what data is written on the blockchain, who can interact directly with the blockchain, and what operations require user authentication. Figure 31 illustrates this by showing that different marketplace instances can co-exist within the global marketplace space.
- **smart locker owners (SLO)**: entities willing to rent out their smart lockers and get compensated for the service offered.
- **smart locker renters (SLR)**: entities interacting with the marketplace to purchase access to a smart locker for a given time frame.

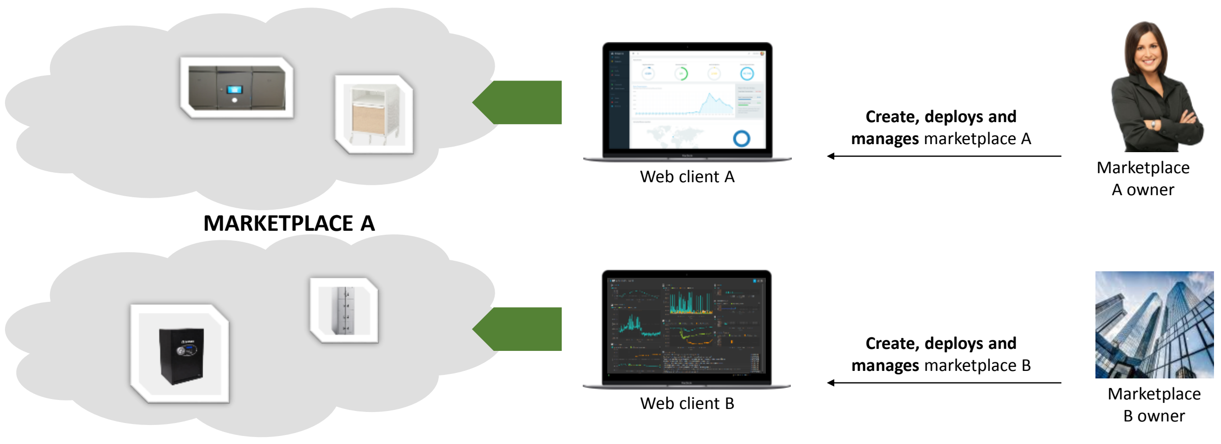

MARKETPLACE A
Web client A
Create, deploys and manages marketplace A
Marketplace A owner
Web client B
Create, deploys and manages marketplace B
Marketplace B owner

Figure 31: There may be multiple instances of marketplaces in the global ecosystem. Suppliers and renters of smart lockers can choose any of them depending on their preferences and the policies implemented by each marketplace.

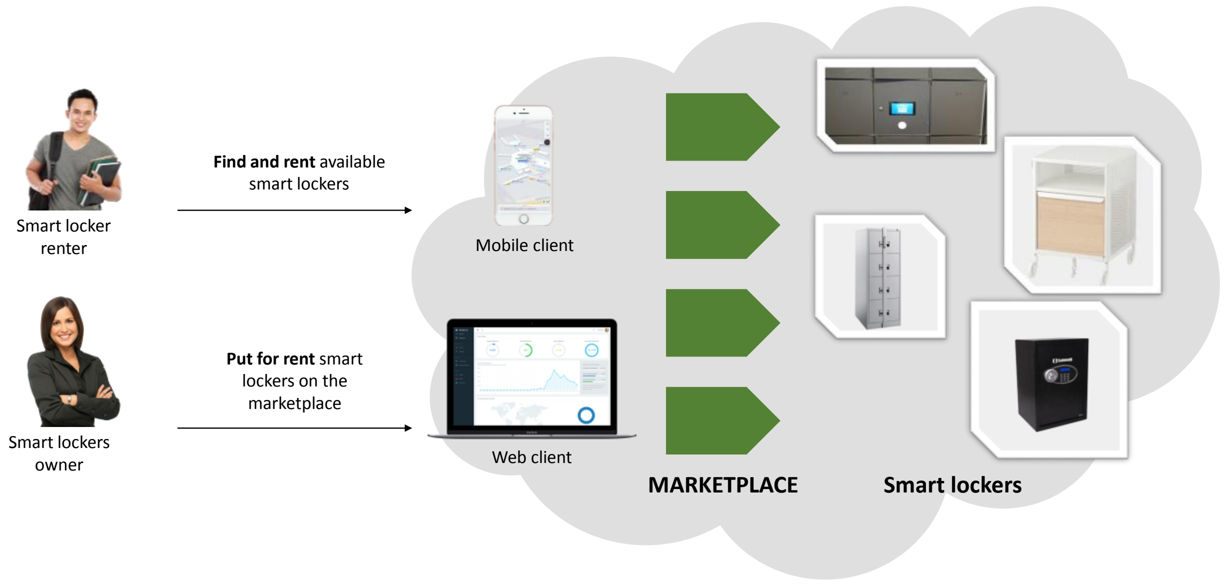


Figure 32: Smart locker owners can rent their smart lockers by publishing their availability on the SMAUG marketplace. On the other hand, potential smart locker renters can discover nearby available smart lockers and pay to purchase access.

7.1.1.1 Smart locker management

One of the two main features that SMAUG offers is to allow smart locker owners (SLO) to manage the smart lockers (SL) they own. Specifically, once registered with one marketplace provider, SLOs can monitor the status of their SLs in the marketplace, register new SLs, open new requests for registered SLs, and close and decide previously opened requests. While the monitoring of smart lockers and their status is an intuitive operation that deserves no in-depth description, the remaining three operations are explained in more detail in the following sections.

Smart locker registration

A precondition for an SLO to manage marketplace details and operations for a specific smart locker (SL) is to register the SL with the marketplace. Specifically, the SLO provides information such as the unique identifier of a smart locker and its physical properties (e.g. width, height). All the information specific to an SL must be compliant with the SOFIE semantic representation: this allows SLs belonging to different marketplaces to be understood by potential smart locker renters (SLR) without the need to use marketplace-specific client applications. Similarly, using a uniform way of representing smart lockers allows different clients to understand SL-related information regardless of the specific marketplace. At the moment, no check is performed to verify that the information provided by the SLO matches the real properties of the SL being registered.

Request creation

From the marketplace management interface, SLOs can manage marketplace-related information of the SLs previously registered. For instance, SLOs can create a new request for an SL. This request will enable, on the other side, potentially interested SLRs to place offers to access the SL. The details of these offers are specified below in the relevant section.

When creating a request, the SLO specifies some information related to the request, such as the starting time of the request (e.g., at 6:00 PM of tomorrow), the deadline to receive offers (e.g., offers can be made only within 15 minutes from when the request is published), the maximum duration of the rental period (e.g., with this request, an SL can be rented for **at most** 30 minutes), and the identifier of the SL being rented.

Furthermore, the marketplace allows SLOs to open either auction-only or auction+instant rent requests. The former only accept bids from competing SLRs, that can submit offers until the request expiration time, after which the SLO can choose one or more winning offers that will be granted access to the SL for the time indicated in the offer. Auction+instant rent requests allow SLOs to publish auction requests, as just explained, and also instant rent requests: these requests contain additional pricing information that the SLO provides when creating the request (e.g. 1€/minute of usage if total rental time is less than or equal to 5 minutes, 0.75€/minute of usage if total rental time is less than or equal to 15 minutes, etc.) and that give the possibility to SLRs to instantly purchase access to a smart locker (if the conditions in the offer match the required conditions specified in the request) without waiting for the expiration of a request and/or the decision by the SLO.

Request decision

The SLO that has previously opened a request for a SL can close and decide it at any time, either before or after the request expiration time. If an instant rent offer is presented that matches the requirements of the request, the request is automatically closed and decided without the intervention of the SLO.

When a request is decided, regardless of how the decision process took place, an access token to use the SL is generated and logged (encrypted) on the marketplace blockchain. The encryption allows only the authorised SLR, that paid to get that access token, to retrieve the original access token, and the fact that it is logged on the blockchain makes it possible to use the issued token for future dispute resolutions, in case something goes wrong along the chain of events that should normally lead to the authorised SLRs accessing the SL for the time purchased.

7.1.1.2 Smart locker discovery and access

Other than SL management, the other important feature that SMAUG offers is to discover nearby SLs, place offers for them to get access to an SL storing space. Both interactions are explained in more detail in the next sections.

Smart locker discovery

Although it is theoretically possible for SLRs to place offers for SLs they are not physically close to, the most common scenario will involve potential SLRs in need to access, in a relatively short time frame, an SL that is nearby. For instance, conference attendees might want to rent smart lockers that are located within the venue.

For this purpose, SLRs use a mobile device that can discover nearby SLs that are advertising their presence using Bluetooth Low Energy (BLE) communication. Since many devices use BLE to communicate today, the device will filter only the relevant signals that are advertised by SMAUG-compliant SLs. Since these SLs will all use the same semantic representation, a mobile client can parse the information advertised regardless of the specific marketplace that manages access to the SL.

The information that the client receives includes the information that the SLO provided, for that specific SL, when registering the SL with the specific marketplace. In addition to locker-specific information, the information also includes marketplace-specific information, such as the address of the smart contract to which the SLR should send the rent offer.

At this stage, the client then queries the marketplace to check if there are any open requests for the SL in question. This process takes place for each SL discovered and is transparent to the SLR, which will only see the smart lockers that are available for rent.

Smart locker purchase and access

Assuming a SLR has found some nearby lockers that are available for rent for the time he needs, an offer for that locker can be placed on the marketplace. As already described above in the section relative to request creation, a request can support either only auction offers or also instant rent offers. In the first case, the SLR will decide how much she is willing to pay, and for how long she requests access. The offer is then added to the list of offers for the SL and will be examined by the SLO against all the other offers placed for that specific request and that specific SL. If, on the other hand, the SLR needs to get instant access to the SL, and the request supports instant rent offers, then he can choose to pay the amount required as specified in the request, according to the total duration of the rent. For instant rent offers, the feedback loop is much shorter since the response is almost immediate: either the offer is accepted and the relative access token is issued (unless something goes wrong), or the offer is rejected because the request requirements are not met. Each offer contains additional information that allows the party issuing the access token to access an SL to encrypt it so that it can only be used by the offer creator (the only party authorised to make use of that access token).

Once the encrypted access token is retrieved and decrypted by the authorised SLR, it will then be presented to the SL to unlock access to its storage space. The SL verifies that the access token is valid and if so, grants access to the user.

7.2 Architecture

As previously presented, in a typical marketplace deployment three main entities are interacting with each other: the Marketplace Owner (MPO) manages the marketplace platform and enables SLOs and SLRs to interact via request and offer creations. The three main roles are also reflected in the architecture of the resulting system, as shown in Figure 33.

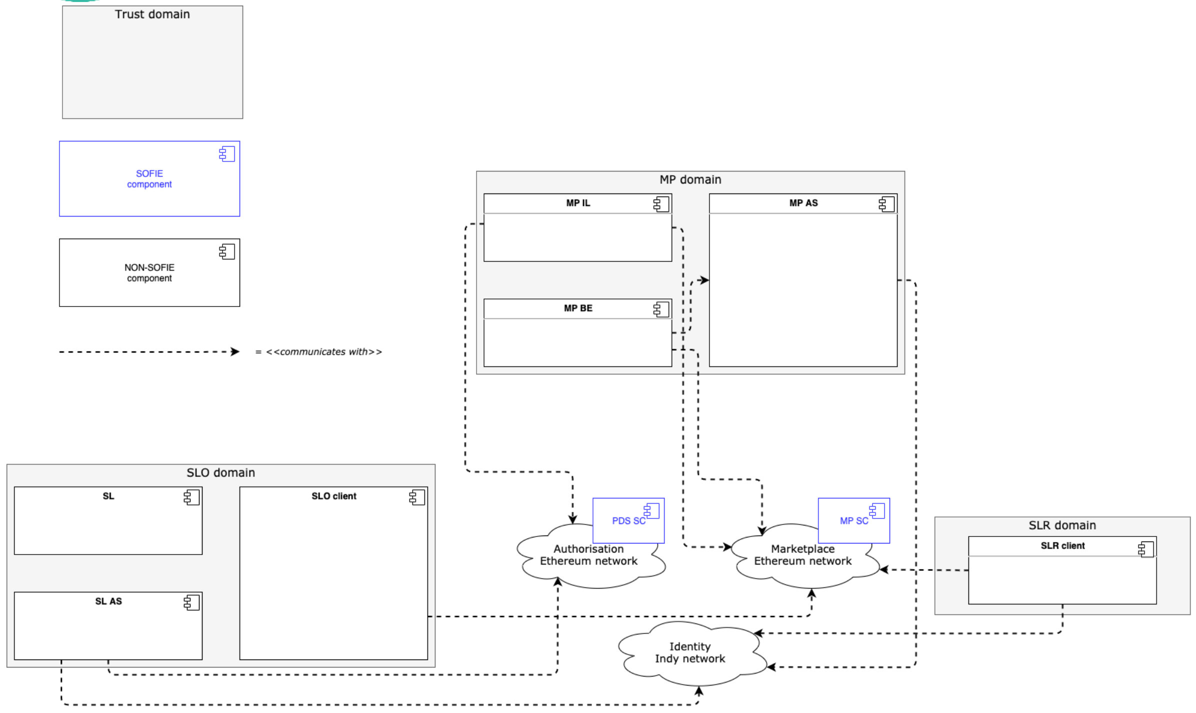


Figure 33: SMAUG architecture.

The system is composed of three trust domains, i.e., the system components that each party trusts and/or manages directly. At the left is the smart locker owner (SLO) trust domain. At the centre-top is the marketplace (MP) trust domain. At the right is the smart locker renter (SLR) domain. At the centre, the three clouds represent the three blockchains that the system relies upon to provide its services: an Ethereum blockchain to run the marketplace, an Ethereum blockchain to manage authorisation-related information, and an Hyperledger Indy blockchain to manage the identities of both SLRs and SLOs.

7.2.1 MP domain

The marketplace (MP) domain includes components that are directly run and managed by the MPO, or that are trusted by the MPO that relies on them to achieve some tasks. Specifically, the MP domain is composed of:

- **Backend (MP BE):** the backend is used by SLOs to manage their SLs. Specifically, through the backend, the SLOs can register new SLs and can manage their status (e.g., publishing requests on the marketplace or deciding the winning offers for a given request).
- **Authorisation Server (MP AS)**: the authorisation server manages access to the marketplace platform. The MP BE relies on the MP AS to authenticate users and grant them access to the platform. Authenticating users allow the MPO to track usage of the marketplace by its customers (e.g. how many requests SLOs have created, or how much money they have obtained from marketplace transactions).
- **Interledger (MP IL)**: the interledger bridges the communication between the marketplace blockchain and the authorisation blockchain. Specifically, the interledger will notify the authorisation blockchain whenever a request is closed, and the winning offers decided. Similarly, parties operating on the authorisation blockchain, after performing some actions in response to the event received from the marketplace blockchain, can interact with the interledger to send response data back to the marketplace blockchain.

7.2.2 SLO domain

The SLO domain includes components that are either owned and direct control of the SLO, or that are trusted by the SLO to provide the agreed services. Specifically, the SLO domain is composed of:

- **Smart Locker (SL)**: the physical resource being rented and purchased on the marketplace. They offer storage space service to authorised users for the duration they have purchased. The presence of an SL is advertised via BLE, such that interested potential renters (SLR) can discover them using a BLE-capable mobile phone and a compatible application. All the communication between the SL and the SLR's mobile device takes place via NFC technology. At the time when SLRs wish to use an SL, they need to prove their authorisation by presenting a valid attestation that the SL can verify and validate.
- **Web client (SLO client)**: runs on the browser of the SLO's device and allows the SLO to perform SL management operations. To access the management interface, SLOs must authenticate themselves and must be authorised to perform the required operation. Furthermore, the SLO client allows the SLO to directly interact with the marketplace blockchain to perform SL management operations.
- **Authorisation Server (SL AS):** this authorisation server manages access to one or more smart lockers (SL). It includes an agent listening on the authorisation blockchain for interledger events, and in response to those events logs an access token that the winning users of that specific request can use to access the smart locker they have purchased access for. The SL AS does not have to be directly managed by the SLOs (although nothing prevents them from doing so), but can also be used following an as-a-Service model, where the SLO delegates the management of one or more SLs to the SL AS.

7.2.3 SLR domain

The SLR domain includes only the mobile device that a potential SLR uses to discover nearby SLs, purchase access for a specific time frame, and interact with the SL to access its enclosing storage space. The **SLR client**, therefore, allows SLRs to discover nearby SMAUG-compliant SLs using BLE, interact with the Ethereum marketplace where access to the SL can be purchased, and interact with them using NFC.

7.3 SMAUG and SOFIE

As presented in the section introducing SMAUG, SMAUG places itself as a *reference implementation* with the goal, among others, to showcase how all the SOFIE framework components can be used together and how a system can be designed and developed to be SOFIE-compliant from its conception. This is the key characteristic that sets SMAUG apart from the other four pilots developed in the project. The four pilots have been designed and built as IoT platforms that were purposefully siloed and independent from each other, to prove how SOFIE opens up those borders and enables cross-platform communications and operations. On the other side, SMAUG has been conceived starting from the set of features that the SOFIE framework offers, so it could be said that SMAUG follows the SOFIE-by-design rules.

Even though not implemented in the scope of this project, the usage of SOFIE framework components makes it theoretically possible for SMAUG to interact with other SOFIE-compliant systems, e.g., the other four pilots developed in the project. For instance, in the context of the energy flexibility marketplace pilot developed by the Italian partners, electric vehicle users that need to leave their cars to charge for extended periods, but do not want to leave valuable objects unsupervised in such vehicles, might want to rent available smart lockers around for the estimated time that the car will take to fully charge. This is possible since SMAUG (and therefore

SOFIE)-compliant smart lockers will all follow the same discovery rules and data semantics, enabling different applications (e.g., the one provided by an energy retailer to its customers) to interact with them in the same standard way.

7.3.1 Usage of SOFIE components

Following is a description of how the different SOFIE components are used within SMAUG, and what benefits they bring.

7.3.1.1 Marketplace

The marketplace is a key component in SMAUG, without which no interaction can take place. SMAUG integrates and extends the functionalities provided in the marketplace component (particularly the set of Ethereum smart contracts) to fit SMAUG-specific needs. For instance, the SMAUG marketplace smart contract allows SLOs to specify a set of pricing rules that makes it possible for the smart contract to automatically and immediately select winning offers upon their presentation by SLRs, reducing to only a few seconds the time the SLRs making valid offers need to wait before having access to a smart locker, and also removing the need for the SLO to be online and manually decide which offers to select for a given request.

7.3.1.2 Interledger

SMAUG uses the Interledger component to complement the marketplace functionality since it allows to use a separate Ethereum blockchain for authorisation-related operations. Specifically, the Interledger bridges the communication from and to either ledger. In one case (from the marketplace to the authorisation ledger), Interledger propagates marketplace events on the authorisation blockchain, so that interested parties can perform actions accordingly e.g., issue a new access token. In the second case, the notified parties can trigger interledger events that will communicate data back to the marketplace blockchain, typically communicating that the access token for a specific offer has been logged, along with the access token. The marketplace can then take action, e.g., moving the money to the SLO's account, when it receives a notification that an access token for a specific offer has been issued. A high-level description of the flow is presented in Figure 34.

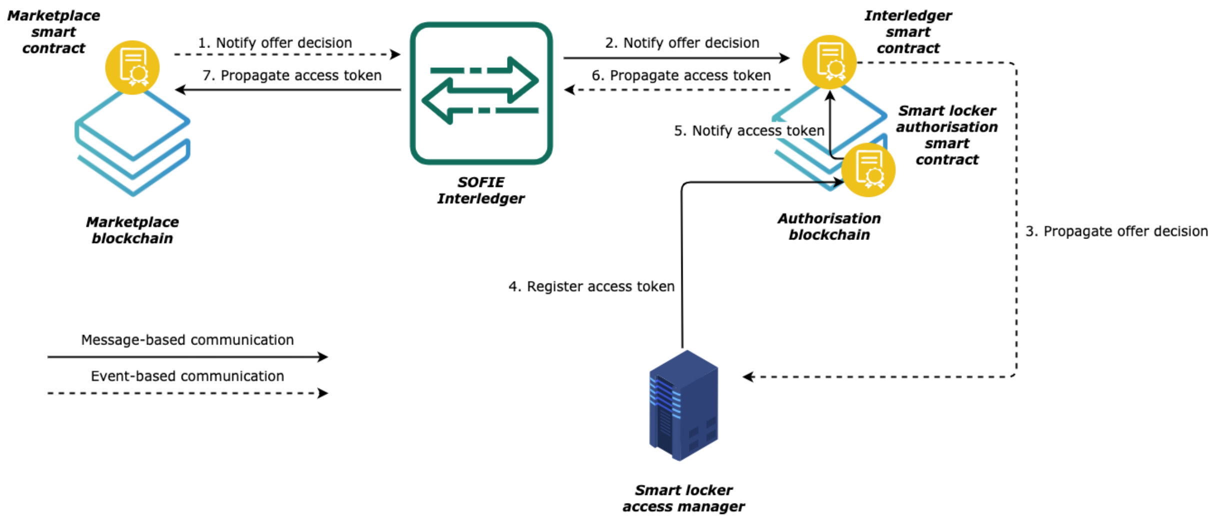

Marketplace smart contract
1. Notify offer decision
7. Propagate access token
2. Notify offer decision
6. Propagate access token
Interledger smart contract
5. Notify access token
Smart locker authorisation smart contract
Marketplace blockchain
SOFIE Interledger
Authorisation blockchain
3. Propagate offer decision
4. Register access token
Message-based communication
Event-based communication
Smart locker access manager

Figure 34: Interledger flow for a typical marketplace transaction

In a typical marketplace transaction, first, when an offer is decided, an event is emitted to start the Interledger procedure (step 1) which is captured by the Interledger agent. Then, the agent calls a smart contract on the authorisation blockchain (step 2) which, in turns, emits an event to notify potential listeners about the Interledger operation and its associated data (step 3). Interested listeners can then perform custom actions (step 4) and, if they need to propagate the result of the action back to the marketplace, interact with the Interledger smart contract (step 5), which then propagates the information (step 6), which is captured by the Interledger agent and forwarded to the marketplace smart contract (step 7).

7.3.1.3 Identity, Authentication and Authorisation (IAA)

The Identity, Authentication and Authorisation (IAA) component is used in several places for different reasons. In one case, it is used in the SL to verify the validity of the access token presented by potential users, or SLRs. In another case, IAA is also used in the marketplace platform to validate web access tokens (JSON Web Tokens, or JWT) that SLOs must present to access the marketplace web interface. Since IAA also provides the authorisation smart contract used to log smart locker access tokens, it is used in SMAUG and deployed on the authorisation blockchain.

7.3.1.4 Privacy and Data Sovereignty (PDS)

The Privacy and Data Sovereignty (PDS) component, similarly to IAA, is used in different parts of the system. It is used as the authorisation server for the marketplace platform: SLRs must interact with the authorisation server (hence with the PDS) and authenticate themselves using a decentralised identifier (DID) previously registered (or register one in the case of new users) to get a JWT that would grant them access to the marketplace web interface. Furthermore, PDS is also used to log the smart locker access tokens in response to Interledger events originated from the marketplace blockchain. Specifically, the PDS contains the logic of generating access tokens to access a smart locker and knows the address of the smart locker authorisation smart contract where those access tokens must be logged.

7.3.1.5 Provisioning and Discovery (P&D)

SMAUG utilises only the *discovery* functionality provided by the Provisioning and Discovery (P&D) component. SMAUG-compliant smart lockers will use this component to advertise their presence to nearby users using Bluetooth Low Energy (BLE) as the communication medium. Once the discovery of a smart locker takes place, the information, which is compliant with the SOFIE Semantic Representation format, is parsed by the smart locker renters who can then decide to proceed further and interact with the marketplace to purchase access to that smart locker.

7.3.1.6 Semantic Representation (SR)

The Semantic Representation (SR) is the key component that makes SMAUG open and interoperable with external systems. The data that each smart locker advertises include its physical properties (e.g., capacity, identification number) and information about the marketplace that potential renters will interact with. By using the semantic representation defined in the SOFIE framework, SMAUG smart lockers (like any other IoT system that uses the same semantic representation) are easily integrable into and can easily communicate with other systems that can parse and understand data semantically annotated following the SOFIE representation.

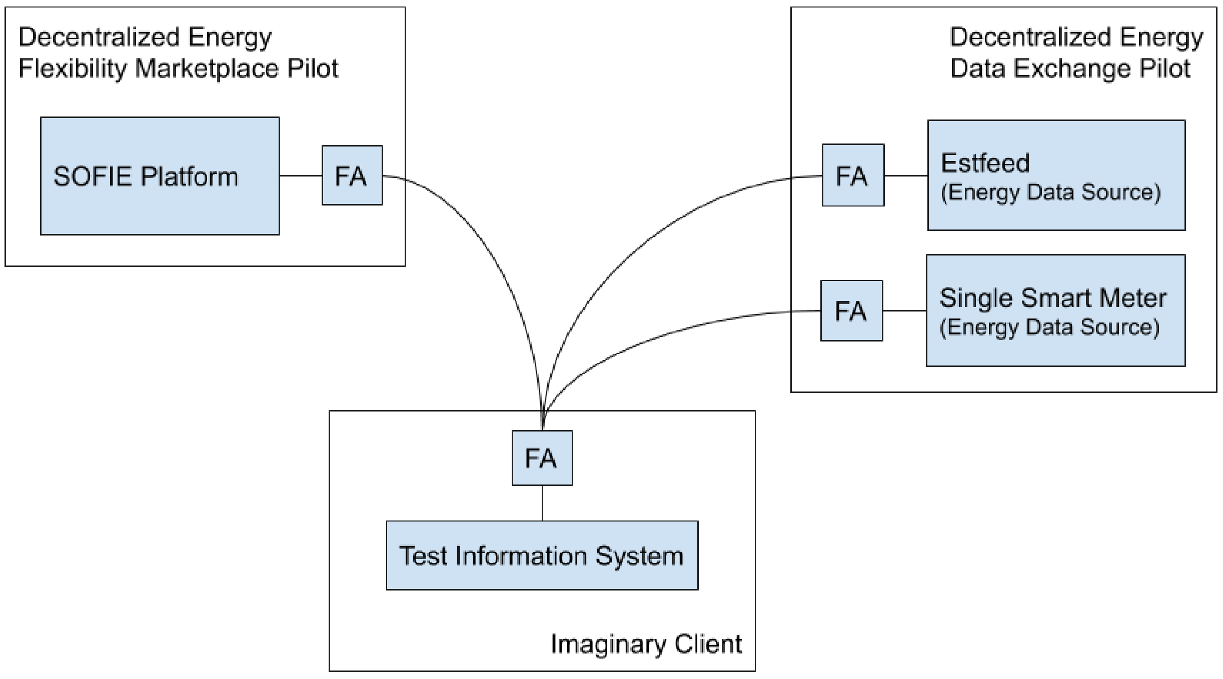
8. Cross pilot scenarios and testing plan

In this section, we briefly describe cross pilot scenarios that will be implemented in the following period. These cases will demonstrate and highlight the interoperability aspects between different pilot cases. More detailed description about the cross-pilot cases will follow in the next deliverable, D5.4 (Final Validation & Replication guidelines).

8.1 Cross pilot data exchange

In this scenario, we federate the Decentralized Energy Data Exchange (DEDE) pilot with the Decentralized Energy Flexibility Marketplace (DEFM) pilot and enable secure data exchange between them. We use the Federation Adapter (FA) developed for the DEDE pilot to achieve this. Although the main goal of the DEDE pilot is to liberate energy data, the technical solution is not limited to this single domain. The exchanged data can be anything, and the solution is thus suitable for a cross-pilot scenario. The architecture of the DEDE pilot and its Federation Adapter is described in Section 4.2.

No changes to the existing platforms of the federated pilots is necessary for this cross-pilot scenario. The only requirement for each pilot is to be able to describe the services that it offers in the OpenAPI 3.0 format. That is the only format currently supported by the FA. If the pilot does not already offer services that can be described in OpenAPI 3.0 format, it is possible to develop a converter on top of the existing platform services. Although all the federated pilots could easily consume services offered by other pilots, we deploy a separate client dedicated to the purpose of testing and evaluating this federation approach. This way, the cross-pilot testing does not force the pilots to implement functionality that does not align with their business goals. The setup is depicted in Figure 35.



Decentralized Energy Flexibility Marketplace Pilot
SOFIE Platform
FA
Decentralized Energy Data Exchange Pilot
FA
Estfeed (Energy Data Source)
FA
Single Smart Meter (Energy Data Source)
FA
Test Information System
Imaginary Client

Figure 35: The setup of cross-pilot testing using the FA from the DEDE pilot

The following services will be offered by the Decentralized Energy Flexibility Marketplace pilot:

- *getChargingStation* - returns charging station information given its ID

- *getChargingSession* - returns a list of historical charging sessions given a charging station ID or a detailed charging session data given a charging session ID
- *getElectricVehicle* - returns electric vehicle information given its ID

The following services will be offered by all the data sources in the Decentralized Energy Data Exchange pilot:

- getMeteringPoints - returns a list of metering points this data source has data for
- getConsumptionData - returns electricity consumption data given a metering point ID

Both pilots give access to all their services for the Test Information System.

8.1.1 Test cases

8.1.1.1 Latency overhead of the FA

We measure the latency overhead (in milliseconds) added by both the service provider FA and the service consumer FA of the request-response cycle. This metric is constant as the network grows.

8.1.1.2 Throughput of the FA

We measure the throughput (in requests per second) of both the service provider FA and the service consumer FA. This is important to estimate the load that a single FA can carry in a production environment.

8.1.1.3 Integration Effort and Comparison to Current Situation

We evaluate the integration effort, separately for the service provider and service consumer. We compare this to the current situation and possible alternative approaches.

8.2 Cross pilot reward exchange

In this scenario, we try to exchange the data between Mobile gaming and Decentralized Energy Flexibility Marketplace pilot to make a collaborative ecosystem where users from one pilot can earn reward by using the other pilot. The main goal of this cross-pilot scenario is to cultivate the growth of both pilots and users who could be engaged by gamification elements to fuel their motivation and make activities more interesting. Being able to trade, buy and sell goods for real value will further encourage users to engage more time to earn virtual items, certain that they will get a good return of investment.

In order to develop such a scenario, we will be leveraging SOFIE platform and its components. The smart contracts will be developed to enable this cross pilot. No major changes to the existing platforms of the pilots is necessary for the implementation of the possible scenarios. Both pilots together will be developing a platform where users' information is shared in a secure and anonymous way. In one probable scenario, Gaming pilot can create a specific challenge and multiple unique virtual items that can be given to the users of Energy pilot. On the other hand, in-game tokens can also be used on Energy marketplace for enabling trading. Another scenario of a user acquiring tokens on Energy marketplace can be used in-game for buying virtual items. Figure below illustrates the setup of the cross-pilot scenario.

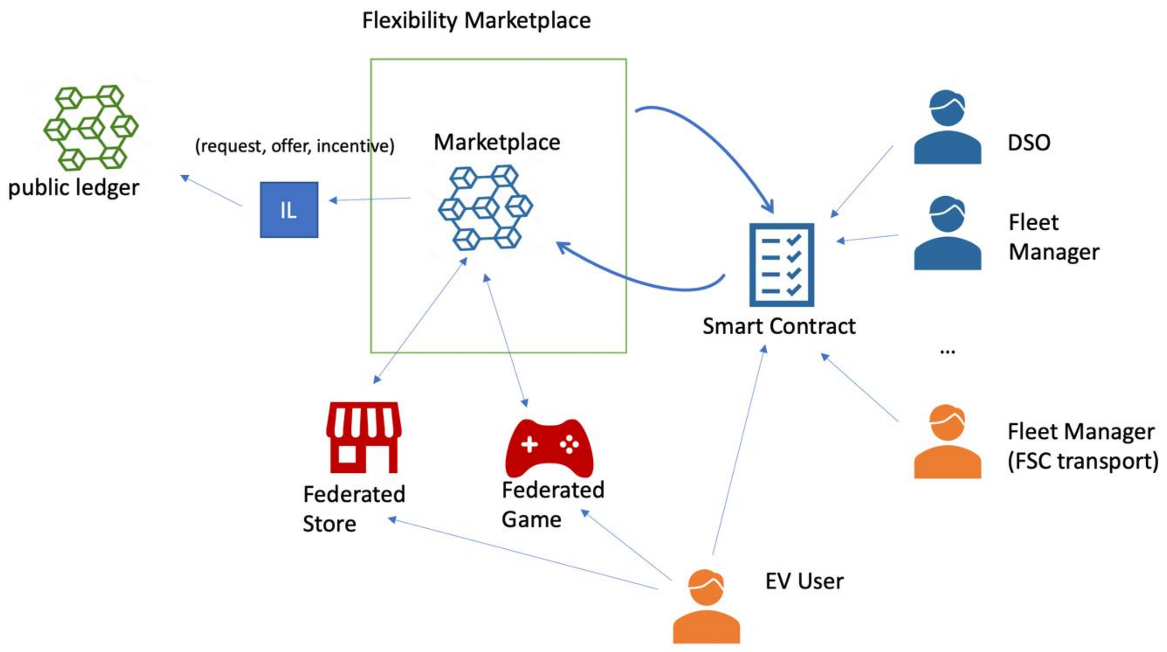


Figure 36: The setup of cross-pilot testing

The following services will be offered by the Mobile gaming pilot:

- getPlayerInfo: returns Player information given its ID (preserving anonymity)
- claimItem: transferring virtual items for the EV users.

The following services will be offered by the Decentralized Energy Flexibility Marketplace pilot:

- *getChargingSession* - returns a list of charging sessions of specific user given a charging station and user ID

8.2.1 Test cases

8.2.1.1 Validation

We will validate the cross-pilot scenario by running end-to-end process ensuring data transferred is correct and useful and all the requirements are fulfilled.

8.2.1.2 Latency

We will measure the latency overhead (in milliseconds) added by both pilots when the data is shared, and rewards are transferred between them.

9. Conclusions

WP5 aims at setting up the four pilots of the SOFIE project and validating its federation architecture in real operating conditions. This deliverable has presented the technical system implementation validation results, and the end-to-end (prototype integration) validation results of the pilot platforms as well as the next steps of each pilot in terms of validation. Also, SMAUG, a reference implementation of the SOFIE framework has been presented. In addition to the four pilots, a cross-pilot case has been described, which will be further described and presented in the final deliverable of WP5. The results of the validation of SOFIE components have also been included in the Annex of this deliverable. The results provided in this document will be used as a reference point for the last, overall pilot evaluation which will be reported near the end of the project.

10. References

[D2.4] T. Elo et al. “SOFIE Deliverable 2.4 - Federation Architecture, 2nd version”, June 2019. Available at: https://media.voog.com/0000/0042/0957/files/SOFIE_D2.4-Federation_Architecture_2nd_version_v1.10.pdf.

[D2.5] Y. Kortesniemi et al. “SOFIE Deliverable D2.5 – Federation Framework, 2nd version”, August 2019. Available at: https://media.voog.com/0000/0042/0957/files/SOFIE_D2.5-Federation_Framework%2C_2nd_version.pdf.

11. Appendix I: Validation of SOFIE components

ID	***Validation Process***		**Result**
Interledger			
RF01	*Requirement Description*	User interaction is not required for interledger operations.	OK
	Test approach	Functional test	
	Test Description	Event on one ledger automatically triggers the transfer of data/asset to another ledger	
	Test location	Interledger: tests/system/test_interledger_ethereum.py	
RF02	*Requirement Description*	There should be support for atomic interledger operations.	OK
	Test approach	Functional test	
	Test Description	Status of asset transfers is atomic, so that the asset can be accessible only in one ledger	
	Test location	Interledger: tests/system/test_interledger_ethereum, solidity/test/tokenTest (testing contract for GameToken)	
Identification, Authentication, and Authorization (IAA)			
RF03	*Requirement Description*	Resource owners must be able to delegate the authentication and authorisation tasks for their resources.	OK
	Test approach	Documentation	
	Test Description	The IAA can be configured to operate with any authorization server. Configuration examples will be provided	
	Test location	IAA's repository documentation, "Configuration" chapter	
RF04	*Requirement Description*	The IAA component must provide users the capability to revoke authorisations.	OK
	Test approach	Functional test	
	Test Description	A token is created, and it is logged in an ERC-721 smart contract. Then it is marked as revoked in the smart contract. IAA rejects the token.	
	Test location	IAA tests/test_erc721.py	

RF05	*Requirement Description*	The IAA component must allow individuals to control their personal information and digital identities (e.g. support self-sovereign identity technology).	OK
	Test approach	Functional test	
	Test Description	The test is configured with a valid DID and a valid VC. It interacts with indy_agent.py which generates a challenge. The test sends a report to the challenge.	
	Test location	IAA tests/test_indy_agent.py	
RF06	*Requirement Description*	The IAA component must support secure, tamper-proof, and verifiable logging of transactions and events.	OK
	Test approach	Functional test	
	Test Description	The test is configured with a valid token. It interacts with iaa_logger.py which records the token in a configured Ethereum smart contract. The test verifies the record.	
	Test location	IAA test/test_logging.py	
RF07	*Requirement Description*	The IAA component must support Role Based Access Control (RBAC).	OK
	Test approach	Documentation	
	Test Description	RBAC is implemented with the use of VCs. IAA can be used to verify a VC.	
	Test location	IAA's repository documentation, "Examples" chapter	
RF08	*Requirement Description*	Cryptographic algorithms used by SOFIE should be open-source, transparent, and as independent as possible of any particular architecture.	OK
	Test approach	Documentation	
	Test Description	IAA supports standardized cryptographic algorithms.	
	Test location	IAA's repository documentation, "Key technologies" chapter	
RF09	*Requirement Description*	SOFIE should support the execution of authorisation and authentication functionality on devices with constrained processing, storage, battery, and network connectivity.	OK
	Test approach	Functional test	

	Test Description	The test pre-configures IAA with the DID document of a DID. Then IAA authenticates this DID using only local information, and without needing network connectivity.	
	Test location	IAA test/test_indy_api.py	
Privacy & Data Sovereignty (PDS)			
RF10	*Requirement Description*	SOFIE must follow the data minimisation principle for personal data and only request or process what is necessary for the situation and purpose.	OK
	Test approach	Documentation	
	Test Description	PDS can be configured with a specific proof request	
	Test location	PDS's repository documentation, "Configuration" chapter	
RF11	*Requirement Description*	Processing of an individual's personal data is justified by a valid legal basis, e.g. a valid consent from the individual.	OK
	Test approach	Functional test	
	Test Description	The test is configured with a valid VC. The test invokes the VC verification, which generates a proof request. The test generates the proof and outputs the verification result.	
	Test location	PDS tests/test_indy_agent.py	
RF12	*Requirement Description*	Consent to process personal data must be revocable at any time.	OK
	Test approach	Documentation	
	Test Description	The documentation described how to set an expiration time on a VC	
	Test location	PDS's repository documentation, "Examples" chapter	
RF13	*Requirement Description*	SOFIE must allow organisations and actors to manage (create, update, delete) their own data privacy policies.	OK
	Test approach	Documentation	
	Test Description	PDS can be configured with arbitrary VC schemas.	
	Test location	PDS's repository documentation, "Configuration" chapter	

RF14	*Requirement Description*	SOFIE should support user privacy even when aggregate statistics are made public (e.g. using differential privacy mechanisms).	TBD
	Test approach	Documentation	
	Test Description	PDS can be configured to apply RAPPOR local differential privacy mechanism.	
	Test location	TBD	
Semantic Representation			
RF15	*Requirement Description*	SOFIE must define an IoT things description model based on well-known standards (e.g. W3C standards).	OK
	Test approach	Functional test	
	Test Description	The test shows that only objects conforming to the component schema (W3C standards) are validated.	
	Test location	Semantic Representation: tests/test_api.py -> test_api_validate()	
RF16	*Requirement Description*	SOFIE must implement standardised metadata and data representation formats and support various data modalities.	OK
	Test approach	Documentation	
	Test Description	The component uses JSON objects.	
	Test location	Semantic representation's repository documentation, "Main decision" chapter	
RF17	*Requirement Description*	The semantic representation model of the system must be open and extensible by third parties (e.g. support the extension of the existing knowledge base and associations by extracting supplementary triples from RDF documents).	OK
	Test approach	Functional test	
	Test Description	The test shows how is possible to add a schema and subsequently add a schema extension. A message then is validated against both the extended schema and the schema extension.	
	Test location	Semantic Representation: tests/test_api.py -> test_api_extended_validation()	

RF18	*Requirement Description*	SOFIE must provide service discovery and resources selection processes based on multiple criteria over the features, associations, and interaction patterns of integrated resources.	TBD
	Test approach	TBD	
	Test Description	TBD	
	Test location	TBD	
RF19	*Requirement Description*	SOFIE should support the semantic update and enhancement of resources' descriptions and associations in a dynamic way.	OK
	Test approach	Functional tests	
	Test Description	The test shows that a schema can be updated and enhanced with improved semantics.	
	Test location	Semantic Representation: tests/test_api.py -> test_api_update_schema()	
Marketplace			
RF20	*Requirement Description*	The marketplace must log the configuration of all trading actions (including offers, bids, parameters of resources, transactions etc.).	OK
	Test approach	Functional test	
	Test Description	The test sets up an auction, accepts bids, and decides which offer wins - and verifies all the related information is stored on the ledger.	
	Test location	Marketplace: solidity/test/flowermarketplace	
RF21	*Requirement Description*	The marketplace must provide actors the capability to post/claim offers and sell/negotiate/exchange/buy resources and digital objects.	OK
	Test approach	Unit tests	
	Test Description	The test sets up an auction, accepts bids, and decides which offer wins (and verifies all the related information is stored on the ledger).	
	Test location	Marketplace: solidity/test/flowermarketplace	

RF22	*Requirement Description*	The marketplace must support transparent trading of resources, i.e. the bids/offers matching process and the payments must be transparent.	OK
	Test approach	Functional test	
	Test Description	The test sets up an auction, accepts bids, and decides which offer wins - and verifies all the related information is stored on the ledger.	
	Test location	Marketplace: solidity/test/flowermarketplace	
RF23	*Requirement Description*	The marketplace must provide evidence once trades have been completed and resources have been properly delivered to the buyers.	TBD
	Test approach	Functional test	
	Test Description	The transaction determining the winning bid is logged on the distributed ledger. Evidence of the delivery of resources must also be logged on the distributed ledger by the winner and seller, after which the evidence can be verified.	
	Test location	TBD	
RF24	*Requirement Description*	The marketplace should allow integration of payment technologies.	OK
	Test approach	Documentation	
	Test Description	The marketplace component provides interfaces for integrating payment solutions and an example from the Energy Flexibility pilot provided by Engineering integrates the ERC20 tokens payment in the energy marketplace.	
	Test location	Marketplace: solidity/vendors/ENG/EnergyMarketPlace.sol	