



## D2.3 Pledger Overall Architecture

Document Identification			
<b>Status</b>	Final	<b>Due Date</b>	30/11/2020
<b>Version</b>	1.0	<b>Submission Date</b>	30/11/2020

<b>Related WP</b>	WP2	<b>Document Reference</b>	D2.3
<b>Related Deliverable(s)</b>	D2.1 “Pledger Detailed Use Cases” D2.2 “Pledger Requirements Analysis”	<b>Dissemination Level (*)</b>	PU
<b>Lead Participant</b>	ICCS	<b>Lead Author</b>	Orfevs Voutyras (ICCS)
<b>Contributors</b>	ATOS, ENG, ICCS, INTRA, INNOV, FILL, i2CAT, HOLO, IMI	<b>Reviewers</b>	Ioannis Sarris (INTRA)
			Ana Juan (ATOS)

Keywords:
Architecture, Functional View, Functional Components, System, Subsystem, System Use Cases, UML diagrams, Pledger Core System

### Disclaimer

This document is issued within the frame and for the purpose of the PLEDGER project. This project has received funding from the European Union’s Horizon2020 Framework Programme under Grant Agreement No. 871536. The opinions expressed and arguments employed herein do not necessarily reflect the official views of the European Commission.

The dissemination of this document reflects only the author’s view and the European Commission is not responsible for any use that may be made of the information it contains. **This deliverable is subject to final acceptance by the European Commission.**

This document and its content are the property of the PLEDGER Consortium. The content of all or parts of this document can be used and distributed, provided that the PLEDGER project and the document are properly referenced.

Each PLEDGER Partner may use this document in conformity with the PLEDGER Consortium Grant Agreement provisions.

(\*) Dissemination level: **PU**: Public

## Document Information

List of Contributors			
Name	Partner	Name	Partner
Ana Juan	ATOS	August Betzler	i2CAT
Roi Sucasas Font	ATOS	Leonardo Ochoa-Aday	i2CAT
Antonio Castillo	ATOS	Antonis Litke	ICCS
Francesco Iadanza	ENG	Orfefs Voutyras	ICCS
Gabriele Giammatteo	ENG	Gonzalo Cabezas	IMI
Stefan Murauer	FILL	Nikolaos Kapsoulis	INNOV
Verena Stanzl	FILL	Olga Segou	INTRA
Alexander Werlberger	HOLO	Ioannis Sarris	INTRA
Carina Pamminger	HOLO		

Document History			
Version	Date	Change editors	Changes
0.1	01/09/2020	Orfefs Voutyras (ICCS)	First version of the deliverable
0.2	04/09/2020	Orfefs Voutyras (ICCS)	Chapters 2.1 and 2.2 finalised
0.3	11/09/2020	Orfefs Voutyras (ICCS)	Chapter 3 finalised
0.4	19/10/2020	ENG (all members)	Chapter 4.3.1 finalised
0.5	19/10/2020	ATOS & ENG (all members)	Chapter 4.3.2 finalised
0.6	20/10/2020	ENG (all members)	Chapter 4.3.3 finalised
0.7	21/10/2020	ATOS (all members)	Chapter 4.3.4 finalised
0.8	21/10/2020	INNOV (all members)	Chapter 4.3.5 finalised
0.9	22/10/2020	INTRA (all members)	Chapter 4.3.6 finalised
0.10	23/10/2020	INTRA (all members)	Chapter 4.3.7 finalised
0.11	27/10/2020	Orfefs Voutyras (ICCS)	Chapter 4.1 and 4.2 finalised
0.12	30/10/2020	FILL, HOLO, IMI, i2CAT	Chapters 4.4.1, 4.4.2, and 4.4.3, finalised
0.13	05/11/2020	Orfefs Voutyras (ICCS)	Chapter 5 finalised
0.14	12/11/2020	Orfefs Voutyras (ICCS)	Ex. Summary, Chapters 1 and 6 finalised
0.15	13/11/2020	Orfefs Voutyras (ICCS)	Version ready for internal review
0.16	20/11/2020	Ioannis Sarris (INTRA)	Internal review 1
0.17	20/11/2020	Ana Juan Ferrer (ATOS)	Internal review 2
1.0	27/11/2020	Orfefs Voutyras (ICCS)	Final version to be submitted

Quality Control		
Role	Who (Partner short name)	Approval Date
Deliverable leader	Orfefs Voutyras (ICCS)	27/11/2020
Quality manager	Cesar Mediavilla (ATOS)	30/11/2020
Project Coordinator	Ana Juan (ATOS)	30/11/2020

<b>Document name:</b>	D2.3 Pledger Overall Architecture	<b>Page:</b>	2 of 40
<b>Reference:</b>	D2.3 <b>Dissemination:</b> PU	<b>Version:</b>	1.0 <b>Status:</b> Final

# Table of Contents

---

Document Information .....	2
Table of Contents .....	3
List of Tables.....	5
List of Figures .....	6
List of Acronyms.....	7
Executive Summary .....	8
1 Introduction .....	9
1.1 Purpose of the document.....	9
1.2 Relation to other project work.....	9
1.3 Structure of the document .....	10
2 Overall Methodology.....	11
2.1 Steps, Activites, and Exercises.....	11
2.2 Means of results presentation.....	13
2.3 Glossary .....	13
3 Pledger Behavioural View.....	15
3.1 Introduction.....	15
3.2 Core System Use Case diagrams.....	16
3.2.1 SaaS manager & SaaS DSS.....	16
3.2.2 IaaS manager .....	17
3.2.3 IaaS evaluation .....	18
3.2.4 IaaS/ SaaS providers interactions .....	19
4 Pledger Structural View .....	21
4.1 Introduction.....	21
4.2 Core System Overview .....	21
4.3 Core Subsystems & Functional Components.....	24
4.3.1 Configuration subsystem.....	24
4.3.2 Orchestration subsystem.....	25
4.3.3 Benchmarking subsystem.....	27
4.3.4 SLAs subsystem .....	28
4.3.5 Blockchain subsystem .....	30
4.3.6 Big Data Platform subsystem.....	31
4.3.7 Security subsystem .....	33
4.4 Project Use Cases Subsystems .....	35
4.4.1 UC1 subsystem – Mixed reality applications on the edge.....	35
4.4.2 UC2 subsystem – Edge Infrastructure for safety of VRUs .....	36

<b>Document name:</b>	D2.3 Pledger Overall Architecture	<b>Page:</b>	3 of 40
<b>Reference:</b>	D2.3	<b>Dissemination:</b>	PU
	<b>Version:</b>	1.0	<b>Status:</b> Final

4.4.3 UC3 subsystem – Data mining on the edge.....	37
5 Development Process .....	38
6 Conclusions & Next Steps.....	40

<b>Document name:</b>	D2.3 Pledger Overall Architecture			<b>Page:</b>	4 of 40		
<b>Reference:</b>	D2.3	<b>Dissemination:</b>	PU	<b>Version:</b>	1.0	<b>Status:</b>	Final

## List of Tables

---

<i>Table 1: Exercises related to D2.3</i>	12
<i>Table 2: Glossary</i>	14
<i>Table 3: SaaS manager &amp; DSS System Use Cases</i>	16
<i>Table 4: IaaS manager System Use Cases</i>	18
<i>Table 5: IaaS evaluation System Use Cases</i>	19
<i>Table 6: IaaS/ SaaS providers interactions System Use Cases</i>	20
<i>Table 7: Configuration subsystem Components</i>	24
<i>Table 8: Orchestration subsystem Components</i>	26
<i>Table 9: Benchmarking subsystem Components</i>	28
<i>Table 10: SLA subsystem Components</i>	29
<i>Table 11: Blockchain subsystem Components</i>	31
<i>Table 12: Big Data Platform subsystem Components</i>	32
<i>Table 13: Security subsystem Components</i>	34
<i>Table 14: Assets list</i>	38
<i>Table 15: Mapping of components to assets, partners, and tasks</i>	38

<b>Document name:</b>	D2.3 Pledger Overall Architecture	<b>Page:</b>	5 of 40	
<b>Reference:</b>	D2.3	<b>Dissemination:</b>	PU	
	<b>Version:</b>	1.0	<b>Status:</b>	Final

## List of Figures

---

<i>Figure 1: Relation of D2.3 to other Tasks and WPs</i>	10
<i>Figure 2: System Architecture Definition methodology</i>	11
<i>Figure 3: SaaS manager &amp; DSS System Use Cases</i>	16
<i>Figure 4: IaaS manager System Use Cases</i>	17
<i>Figure 5: IaaS evaluation System Use Cases</i>	18
<i>Figure 6: IaaS/SaaS providers' interactions System Use Cases</i>	19
<i>Figure 7: The Pledger Core Subsystems</i>	22
<i>Figure 8: The Pledger Core System</i>	23
<i>Figure 9: Configuration subsystem Component Diagram</i>	24
<i>Figure 10: Orchestration subsystem Component Diagram</i>	26
<i>Figure 11: Benchmarking subsystem Component Diagram</i>	27
<i>Figure 12: SLA subsystem Component Diagram</i>	29
<i>Figure 13: Blockchain subsystem Component Diagram</i>	30
<i>Figure 14: Big Data Platform subsystem Component Diagram</i>	32
<i>Figure 15: Security subsystem Component Diagram</i>	33
<i>Figure 16: UC1 subsystem Component Diagram</i>	35
<i>Figure 17: UC2 subsystem Component Diagram</i>	36
<i>Figure 18: UC3 subsystem Component Diagram</i>	37

<b>Document name:</b>	D2.3 Pledger Overall Architecture	<b>Page:</b>	6 of 40	
<b>Reference:</b>	D2.3	<b>Dissemination:</b>	PU	
	<b>Version:</b>	1.0	<b>Status:</b>	Final

## List of Acronyms

Abbreviation / Acronym	Description
5W1H	Who, What, When, Where, Why, and How
App	Application
AR	Augmented Reality
AS	Asset
BaaS	Blockchain-as-a-Service
CBD	Component-Based Development
CPU	Central Processing Unit
DApp	Decentralized Application
DB	Database
DLT	Distributed Ledger Technology
DSS	Decisions Support System
Dx.y	Deliverable number y belonging to WP x
EC	European Commission
FC	Functional Component
FG	Functional Group
FR	Functional Requirement
GPU	Graphics Processing Unit
IaaS	Infrastructure-as-a-Service
M	Month
MR	Mixed Reality
MVP	Minimum Viable Product
PaaS	Platform-as-a-Service
QoS	Quality of Service
SaaS	Software-as-a-Service
SLA	Service Level Agreement
SOA	Service-Oriented Architecture
SoW	Statement of Work
SUC	System Use Case
UC	Use Case (Project Use Case)
UML	Unified Modeling Language
URI	Uniform Resource Identifier
US	User Story
VR	Virtual Reality
VRU	Vulrebrable Road Users
WP	Work Package

## Executive Summary

---

This deliverable presents the initial Pledger architecture based on the Project Use Cases described in D2.1 “Pledger Detailed Use Cases” at M6 and the requirements presented in D2.2 “Pledger Requirements Analysis’ in M9. Several updated versions of the architecture are expected to be produced throughout the project lifecycle, while the final one will be reported in an updated version of this document in M24.

The scope of the current deliverable comprehends:

- ▶ identification of the main functionalities, functional components and services of Pledger;
- ▶ identification of the main functional subsystems originating from the said functionalities, functional components and services;
- ▶ extraction of the functional view of the Pledger architecture;
- ▶ extraction of several other architectural views and perspectives of Pledger;
- ▶ mapping the several components and subsystems to specific project activities and Tasks.

The main goal of this report is to produce practical architectural views that will support the development activities in WP3 “Performance, QoS and orchestration mechanisms” and WP4 “Trust, Smart Contracts and Decision Support mechanisms” as well as the corresponding integration activities in WP5 “Integration, Pilots and Overall Evaluation”.

In this version of the deliverable, in total, 43 core components, 7 core subsystems and 3 functional groups have been identified, and their descriptions and interrelations have also been provided. Moreover, all these components and subsystems have been mapped to specific Tasks of the project, potential Assets (as solutions), and partners, while their connection to the Requirements Analysis and Use Cases description has also been identified. Figure 8 summarises most of the important results of the deliverable.

<b>Document name:</b>	D2.3 Pledger Overall Architecture	<b>Page:</b>	8 of 40				
<b>Reference:</b>	D2.3	<b>Dissemination:</b>	PU	<b>Version:</b>	1.0	<b>Status:</b>	Final

# 1 Introduction

---

## 1.1 Purpose of the document

---

This document is the main outcome of the T2.3 “Pledger Overall Architecture”. It provides the full specification and architectural layout of the Pledger system and includes the description of the Pledger functionalities and UML diagrams. The main goal behind the activities towards the completion of this deliverable is to deliver a new architectural paradigm for edge computing providers and adopters. This covers Result 1 and 1<sup>st</sup> measure of success of the project (“Pledger architecture blueprint”). An updated version of the document will be provided on M24.

The main scope of the document is to enable the consortium to agree on a common vocabulary, framework and methodology to rely on for specifying the Pledger system. That way, all partners adopt the same language and have for instance all components described in the same manner, utilising the same viewpoints. This helps in the designing process and in reaching a common understanding for the project’s needs and expected results, as well as the means to achieve them.

This approach ensures the continuous alignment with technical and stakeholders’ requirements, conformance with state-of-the-art technologies, and proper definition of the full appropriate functionality of Pledger. It also ensures interoperability between different systems that have to be integrated, allows for openness of the interfaces and data formats and exchange approaches, and ensures a high level of security, so that the system is trustworthy by the stakeholders and users.

This deliverable contains all the Architectural Views required for the coordination of the partners and also presents extensively the process through which those views were extracted. As such, the primary audience of this document consists of the members of the consortium that participate in the design and development of the components and modules of the Pledger pilots as well as of the Pledger core system per se. Additionally, the document is of wider interest to stakeholders that are active in the domains of Cloud and Edge Computing, including researchers participating and contributing to H2020 projects under the aforementioned topics, especially to the ones that are expected to use or extend the results of Pledger in the future.

## 1.2 Relation to other project work

---

Figure 1 presents the relation of this deliverable to other Tasks and WPs. As it can be seen, Task 2.3 (the main outcome of which is this deliverable), receives input directly from Task 2.2 “Requirements Analysis” and, as an extension, from Task 2.1 “Use cases detailed description”, through the corresponding deliverables (through D2.2 “Pledger Detailed Use Cases” and D2.1 “Pledger Requirements Analysis”). More specifically, these deliverables provide in a holistic way an overview of the use cases description along with particular details on their implementation within the pilots. To do so, D2.1 provides a full analysis of the use cases covered in the Pledger project. D2.2 focuses on the requirements analysis from potential users of the Pledger platform, including. Artefacts used directly from those deliverables are the Use Cases overview, the identified User Stories and Stakeholders, the Relevant Technologies for the Project, and the Requirements. This input has been used to extract the **behavioural view** of Pledger (see Chapter 3) and recognize *what the interactions between the Pledger system and users are*.

It should also be noted that, although WPs 3 and 4 have not published any official results yet in the form of deliverables, they have both provided valuable input to this deliverable. By filling in templates and working documents provided through WP2 activities, the technical descriptions of required functionalities, and available and potential technical solutions have been acquired. This input has been used to extract the **structural view** of Pledger (see Chapter 4) and recognize *what the functionalities the Pledger system should and will offer are*.

<b>Document name:</b>	D2.3 Pledger Overall Architecture	<b>Page:</b>	9 of 40
<b>Reference:</b>	D2.3	<b>Dissemination:</b>	PU
	<b>Version:</b>	1.0	<b>Status:</b> Final

On the other hand, in their turn, the results of this deliverable will be used within WPs 3 and 4 which will be focusing on the development and adoption/ adaptation of tools upon which the implementation of the Pledger subsystems will be based on. Naturally, the same results will also be used not only for the development activities, but also for the integration ones (undertaken within Task 5.2 “Integration & Demonstration setup”).

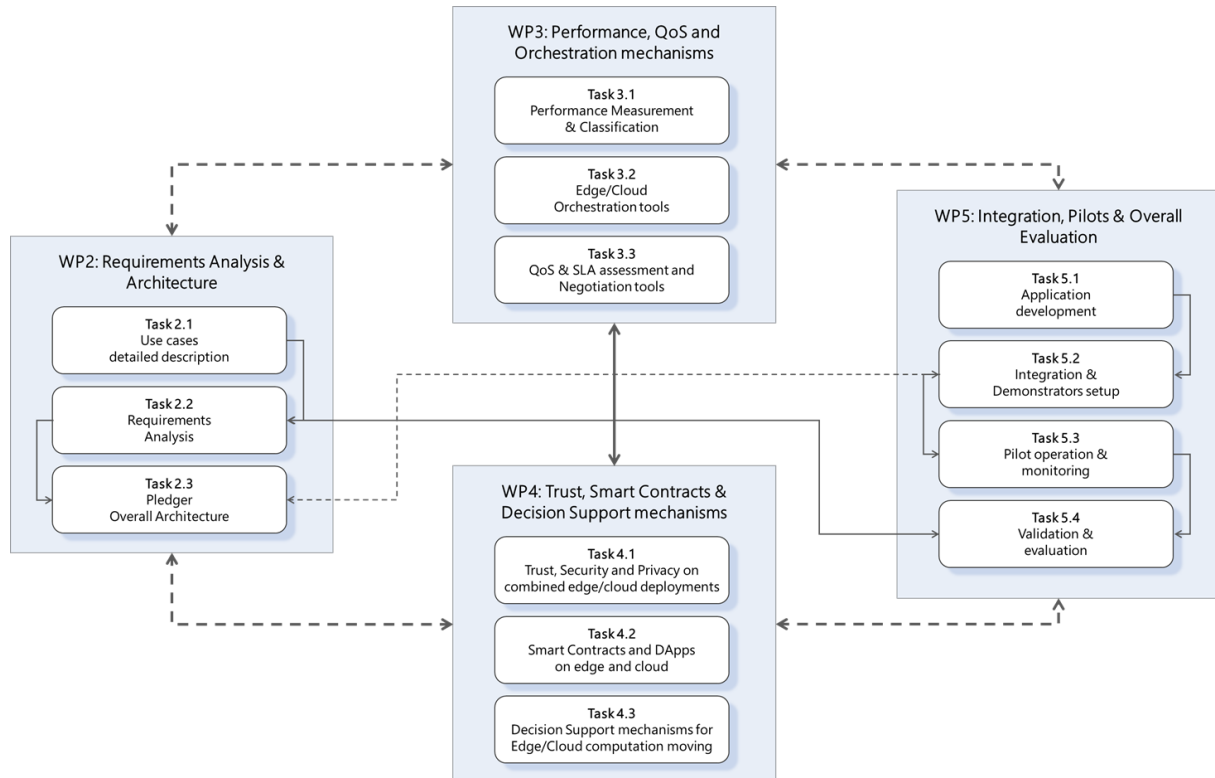


Figure 1: Relation of D2.3 to other Tasks and WPs

### 1.3 Structure of the document

This document is structured in the following major chapters:

- ▶ **Chapter 2** presents the overall methodology followed, on the one hand for the extraction of the main results of this document, and on the other hand for their presentation.
- ▶ **Chapter 3** presents the behavioural view of the Pledger system
- ▶ **Chapter 4** introduces the main functional subsystems and components of Pledger. It presents the structural view of the Pledger system, the main outcome of the deliverable, in the form of UML diagrams. This input is necessary for future activities within the project, especially in the case of integration work in WP5 “Integration, Pilots and Overall Evaluation”. More details are expected to be provided on this topic through future deliverables of WP3 “Performance, QoS and orchestration mechanisms” and WP4 “Trust, Smart Contracts and Decision Support mechanisms”.
- ▶ **Chapter 5** aggregates useful information related to the development of the Pledger components (e.g. mapping of functional components to specific assets, tasks, and partners). More details are expected to be provided on this topic through future WP3 and WP4 deliverables.
- ▶ Finally, **Chapter 6** concludes the deliverable by summarising its main results and identifying future steps to be followed for the next version of the document.

<b>Document name:</b>	D2.3 Pledger Overall Architecture	<b>Page:</b>	10 of 40
<b>Reference:</b>	D2.3	<b>Dissemination:</b>	PU
	<b>Version:</b>	1.0	<b>Status:</b> Final

## 2 Overall Methodology

### 2.1 Steps, Activities, and Exercises

The following figure gives an overview of the overall **System Architecture Definition methodology** devised and followed in order to complete Task 2.3 and provide valuable results for other Tasks.

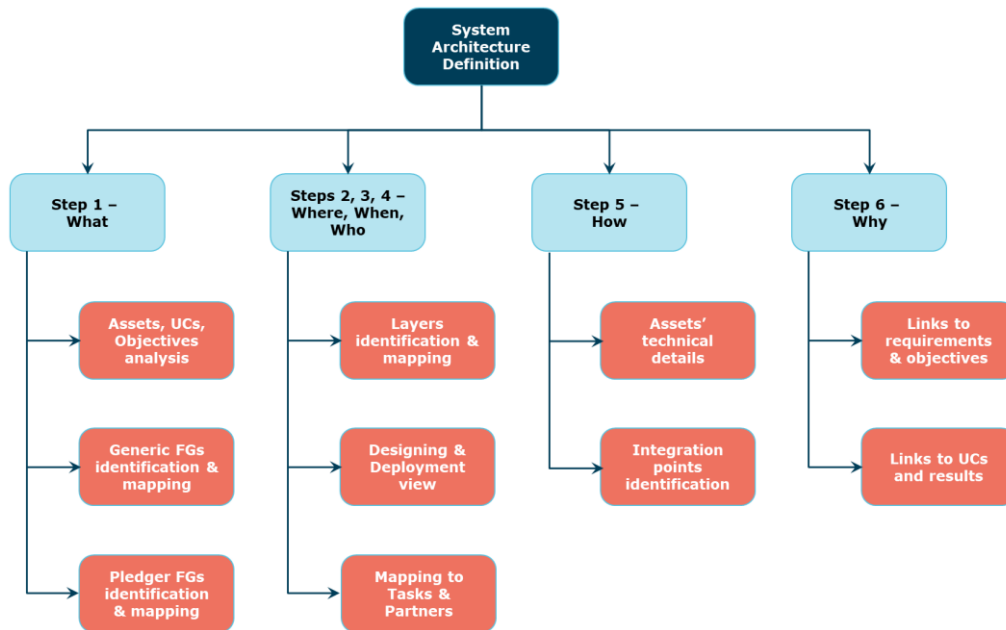


Figure 2: System Architecture Definition methodology

The methodology is based on the **5W1H approach**. The 5W1H (Five Ws and How, 5W1H, or Six Ws) are questions whose answers are considered basic in information gathering or problem solving. They are often mentioned in journalism, research and police investigations. According to the principle of 5W1H, a report can only be considered complete if it answers these questions starting with an interrogative word: Who, What, When, Where, Why, and How.

The “problem” to be solved in the case of this task is the definition of the Pledger System Architecture. The five Ws and 1 H are asked for each and every identified component, and thus the corresponding steps also take place, having on their centre each component (and in some cases, specific groups of components and subsystems). The aggregation of the answers to those questions provide the final results for the definition of the Pledger system as a whole. Some details about each step are presented below:

- ▶ **Step 1 – Answering to WHAT the components and the subsystems can do:** This step includes all the activities required to define the functionalities that the Pledger components provide. Such activities include the study of the SotA, identification of the assets available by the partners of the consortium, description of the said assets, identification of their functionalities, categorisation of assets based on those functionalities, and identification of groups of assets with similar functionalities (resulting to the identification of Functional Groups and subsystems).
- ▶ **Step 2 – Answering to WHERE the components “run”:** In the scope of Pledger, we transform this question to “*WHERE the different artefacts, services and components are hosted?*” (e.g. VMs, Public/ Private Cloud, on the edge, etc.). This step includes all the activities required to define such a layered system and mapping the components to it. A deployment view (see Chapter 2.2) will be one of the main results of the next version of the deliverable that will answer this question.

<b>Document name:</b>	D2.3 Pledger Overall Architecture	<b>Page:</b>	11 of 40
<b>Reference:</b>	D2.3	<b>Dissemination:</b>	PU
	<b>Version:</b>	1.0	<b>Status:</b> Final

- ▶ **Step 3 – Answering to WHEN the components “run”:** This step is pretty straightforward compared to the previous ones and focuses on identifying at what phases of the project (and the lifetime of the corresponding system) each component is expected to be used or be active. Thus, the only activity of this step is the categorisation of the components based on the said phases.
- ▶ **Step 4 – Answering to WHO is responsible for a component:** This step is focused on assigning each component to a specific partner of the consortium and to a specific technical Task under which the said component will be developed and be integrated into the overall system.
- ▶ **Step 5 – Answering to HOW the components and the subsystems work:** This step includes an analysis on a components-level and on a system-level. In the case of the components-level analysis, how each component actually works is identified. Such technical details will be provided in the deliverables of WP3 “Performance, QoS and orchestration mechanisms” and WP4 “Trust, Smart Contracts and Decision Support mechanisms”. In the case of the system-level analysis, how the components are linked with each other is identified. In other words, in this step, the several subsystems and functional flows make their appearance. The identification of the links between the components are the main subject of Task 5.2 “Integration & Demonstration setup” (in the deliverables of which these links will be also mentioned as integration points).
- ▶ **Step 6 – Answering to WHY the components must do what they do:** This step is the main one that links the technical solution presented through the results of the previous steps with the actual scope of the project. It includes all those activities that link the requirements of the use cases and the overall scope and objectives of the project to the specific solution presented in the architecture.

The order of the steps and activities described above is not indicative of their chronological order. Most of the activities and the progress towards the completion of these steps have been and will be taking place in parallel during the whole lifetime of the project. Moreover, even though the activities are presented under different steps, most of them are highly interlinked with each other, being dependent and providing continuous feedback to each other. Table 1 presents an overview of the several activities and exercises implemented under WP2 in order to provide the first version of the System Architecture:

Table 1: Exercises related to D2.3

Exercise	Answers	Reported at
Creation of common Glossary		Table 2
Identification of System Use Cases (SUCs)	What	Table 3 to Table 6
Mapping of SUCs to USs and Requirements	Why	Table 3 to Table 6
Extraction of SUCs diagrams	How	Figure 3 to Figure 6
Identification of core components and subsystems	What	Table 7 to Table 13
Mapping of SUCs to core components and subsystems	Why	Table 7 to Table 13
Extraction of interrelations between core components	How	Figure 8 and Figure 9 to Figure 15
Creation of UC-specific subsystems component diagrams	What/ How	Figure 16 to Figure 18
Assets identification	How	Table 14
Mapping of core components/ subsystems to assets	How	Table 15
Mapping of core components/ subsystems to Tasks/ Partners	Who	Table 15

<b>Document name:</b>	D2.3 Pledger Overall Architecture	<b>Page:</b>	12 of 40	
<b>Reference:</b>	D2.3	<b>Dissemination:</b>	PU	
	<b>Version:</b>	1.0	<b>Status:</b>	Final

## 2.2 Means of results presentation

In order to present in a visual manner some of the outcomes of this deliverable, the Unified Modelling Language (UML)<sup>1</sup> is used. UML is a general-purpose, developmental, modelling language in the field of software engineering that is intended to provide a standard way to visualize the design of a system.

UML specification defines two major kinds of UML diagram:

- ▶ **Behaviour diagrams:** Behaviour diagrams show the dynamic behaviour of the objects in a system, which can be described as a series of changes to the system over time.
- ▶ **Structure diagrams:** Structure diagrams show the static structure of the system and its parts on different abstraction and implementation levels, and how these parts are related to each other. The elements in a structure diagram represent the meaningful concepts of a system, and may include abstract, real world and implementation concepts. Structure diagrams are not utilizing time related concepts and do not show the details of dynamic behaviour.

In Chapters 3 and 4, the following types of diagrams are used:

- ▶ **Use Case diagrams:** Use Case diagrams are Behaviour diagrams that describes a set of actions (use cases) that some system or systems (subject) should or can perform in collaboration with one or more external users of the system (actors) to provide some observable and valuable results to the actors or other stakeholders of the system(s). They are used in Chapter 3.
- ▶ **Component diagrams:** Component diagrams are Structure diagrams that show components and dependencies between them. This type of diagrams is used for Component-Based Development (CBD), to describe systems with Service-Oriented Architecture (SOA). They are used in Chapter 4.

As the project progresses and the development and integration activities reach a sufficient level, the following types of diagrams will also be used (and be reported in the next version of the deliverable):

- ▶ **Sequence diagrams:** Sequence diagrams are Behaviour diagrams that show interactions between the elements of a system, by focusing on the message interchange between lifelines (objects).
- ▶ **Deployment diagrams:** Deployment diagrams are Structure diagrams that show the architecture of the system as deployment (distribution) of software artefacts to deployment targets. Components are deployed to nodes indirectly through artefacts. Specification level deployment diagrams show some overview of deployment of artefacts to deployment targets, without referencing specific instances of artefacts or nodes. Instance level deployment diagrams show deployment of instances of artefacts to specific instances of deployment targets.

Aside from the above types of diagrams, throughout the document, several tables are provided which aggregate the outcomes of the deliverable. The nature of those tables is briefly presented in Table 1.

## 2.3 Glossary

One of the main scopes of the document is to enable the consortium to agree on a common vocabulary. That way, all partners adopt the same language and have for instance all components described in the same manner, utilising the same viewpoints. This helps in the designing process and in reaching a common understanding for the project's needs and expected results, and the means to achieve them.

As the project evolves, the understanding of the project's calls and potential also evolve. For this reason, for example, earlier deliverables like D2.1 "Pledger Detailed Use Cases" and D2.2 "Pledger Requirements Analysis" may in some cases present the use of different terms than the ones to be used in this document. This is not an indication of conflict between the vocabulary being used in the several tasks and deliverables, but rather an indication of the aforementioned evolution.

<sup>1</sup> <https://www.uml.org/>

<b>Document name:</b>	D2.3 Pledger Overall Architecture	<b>Page:</b>	13 of 40
<b>Reference:</b>	D2.3	<b>Dissemination:</b>	PU
	<b>Version:</b>	1.0	<b>Status:</b> Final

Below we present a short list of some of the terms that are being defined in a strict manner for the sake of proper communication between all partners. Although some of these terms may change slightly or be replaced in the upcoming months, the Glossary of the project shall reach an appropriate “final” form as soon as possible, preferably before the delivery of the first deliverables of the technical Tasks in WPs 4 and 5.

Table 2: Glossary

Term	Definition
Requirement	A functional or non-functional need that a particular design, product or process aims to satisfy.
User Story	Identifies how an actor wants to interact with the system.
System Use Case	Use Cases as identified by UML. Identifies how an actor actually interacts with the system.
Project Use Case	Use Cases 1, 2, and 3 as described in D2.1 “Pledger Detailed Use Cases”
Actor	An entity that interacts with the Pledger system.
Stakeholder	An entity that is related the system. Superset of actors and any entities for the development and exploitation of the system.
IaaS provider	Actor offering edge/ cloud infrastructure as a service.
SaaS provider	Actor offering Software-as-a-Service (Application providers).
System integrator	Combination of IaaS provider and SaaS provider.
End User	A SaaS consumer. The end users of the project use cases.
Core System User	Actor using the core subsystems of Pledger (an IaaS provider or a SaaS provider).
Pledger User	A Core System User or an End User.
(Functional) Component	A module/component of the system offering a specific functionality.
Asset	A tool already available by one of the partners of the consortium for the materialisation of functional components or subsystems.
(Functional) Subsystem	A group of interrelated functional components.
(Pledger) Core System	The Pledger core system is considered the system that will be the “minimum viable product” (MVP) of Pledger, the system that will be freely available to users and other projects and initiatives. It is the exploitable part of the project and it contains all the main features that will deem Pledger successful as a project.
(Pledger) Use Cases subsystems	The Pledger Use Case subsystems are considered the subsystems that are developed for the pilots of the project. Use Case subsystems are considered the subsystems that are used only for the sake of completing successfully the objectives of the pilots per se (and not of the project as a whole).
Pledger System	The total system that Pledger will deliver as a project. It is the combination of the Pledger Core System with the Project Use Cases subsystems.

## 3 Pledger Behavioural View

### 3.1 Introduction

Following the UML methodology, the first step towards the definition of the behavioural view of the Pledger System is the identification of **System Use Cases (SUCs)**, which should not be confused with the Project Use Cases described in D2.1 “Pledger Detailed Use Cases”.

A System Use Case describes a single action that the system (subject) should or can perform in collaboration with one or more external users of the system (actors) to provide some observable and valuable result to the actors or other stakeholders of the system. In other words, each System Use Case represents a unit of useful functionality that subjects provide to actors.

The main actors identified within the context of Pledger are:

- ▶ **IaaS providers:** Actors offering edge/ cloud infrastructure as a service.
- ▶ **SaaS providers:** Actors offering Software-as-a-Service. In that sense, they can be considered Application providers. Since these actors are foreseen to use the services provided by the IaaS providers, they can also be referred to as IaaS consumers.
- ▶ **System integrators:** Actors that can be considered both IaaS providers and SaaS Providers.
- ▶ **End users:** Actors using the services provided by SaaS providers. End users can also be referred to as SaaS consumers.

The identification of System Use Cases can be seen as an intermediary between the definition of User Stories (D2.2 “Pledger Requirements Analysis”) identifying what the system is required/ supposed to do, and the definition of system functionalities (see Chapter 4) identifying what the system can and will do. As a result, they are linked to both the external requirements (requirements related to the needs of Pledger users) and the internal ones (requirements related to the constraints and needs identified by the Pledger consortium as necessary for the development of the system).

To this direction, Tables 3-6 present the Use Cases, provide their codes, and link them to specific Actors, User Stories, Requirements, and system functionalities (Functional Subsystems presented in Chapter 4).

The second step towards the definition of the behavioural view of the Pledger System is the identification of the interrelations between the System Use Cases. These interrelations are provided through Use Case diagrams in Figure 3, Figure 4, Figure 5, and Figure 6, following the conventions below:

- ▶ The subject (sometimes called a system boundary) is presented by a rectangle with subject's name, associated keywords and stereotypes in the top left corner. Use cases applicable to the subject are located inside the rectangle and actors - outside of the system boundary.
- ▶ An association between an actor and a use case indicates that the actor and the use case somehow interact or communicate with each other.
- ▶ The “include” relationship is a directed relationship between two use cases which is used to show that behaviour of the included use case (the addition) is inserted into the behaviour of the including (the base) use case. The “include” relationship between use cases is shown by a dashed arrow with an open arrowhead from the including (**base**) use case to the included (**common part**) use case. The arrow is labelled with the keyword «include».
- ▶ The “extend” relationship is a directed relationship that specifies how and when the behaviour defined in usually supplementary (optional) extending use case can be inserted into the behaviour defined in the extended use case. Extended use case is meaningful on its own, it is independent of the extending use case. Extending use case typically defines optional behaviour that is not necessarily meaningful by itself. The “extend” relationship is shown as a dashed line with an open arrowhead directed from the extending use case to the extended (base) use case. The arrow is labelled with the keyword «extend».

<b>Document name:</b>	D2.3 Pledger Overall Architecture	<b>Page:</b>	15 of 40
<b>Reference:</b>	D2.3	<b>Dissemination:</b>	PU
	<b>Version:</b>	1.0	<b>Status:</b> Final

Similar to the User Stories and Requirements in D2.2, the System Use Cases have been split into different groups. This not only makes their study easier, but also makes it possible to link the SUCs to the groups identified in the Requirements and the User Stories, and to translate them to groups of components (specific subsystems). The next section focuses on the Core System Use Cases, as the Project UC-specific Use Cases have already been presented to some extent in D2.1 “Pledger Detailed Use Cases” and D2.2 “Pledger Requirements Analysis”.

## 3.2 Core System Use Case diagrams

### 3.2.1 SaaS manager & SaaS DSS

Figure 3 presents all the System Use Cases that depict actions which the system should perform in collaboration with the SaaS providers, so as to enable the deployment and management of the corresponding SaaS (App) functionalities.

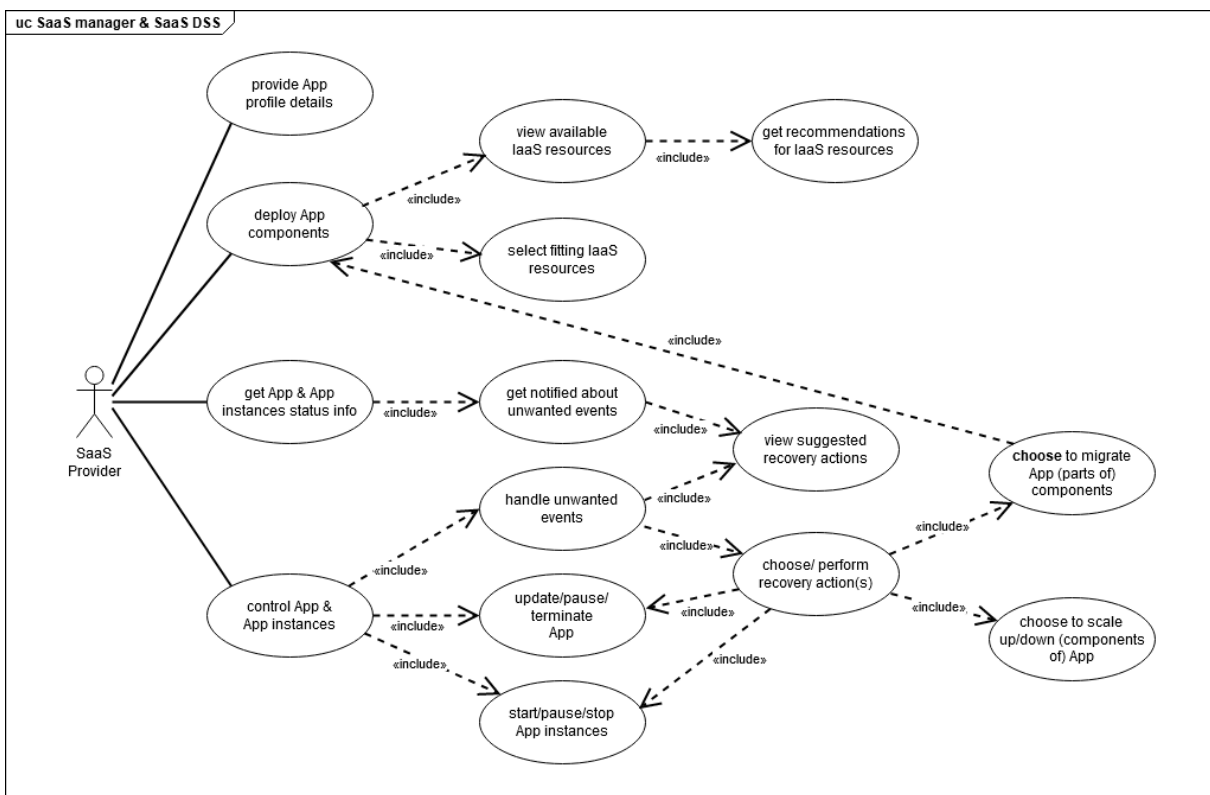


Figure 3: SaaS manager & DSS System Use Cases

Table 3 presents how these System Use Cases are related to specific Actors, User Stories, Requirements, and system functionalities (Functional Subsystems).

Table 3: SaaS manager & DSS System Use Cases

ID	System Use Case	Actor	Related USs	Related Requirements	Related Subsystem
SUC.01	provide App profile details	SaaS		FR.62, FR.02	Configuration
SUC.02	deploy App components	SaaS	US.28, US.29, US.30, US.33	FR.25, FR.29	Orchestration
SUC.03	view available IaaS resources	SaaS	US.28		Orchestration

<b>Document name:</b>	D2.3 Pledger Overall Architecture	<b>Page:</b>	16 of 40
<b>Reference:</b>	D2.3 <b>Dissemination:</b> PU	<b>Version:</b>	1.0 <b>Status:</b> Final

SUC.04	get recommendations for IaaS resources	SaaS			Orchestration
SUC.05	select fitting IaaS resources	SaaS		FR.28	Orchestration
SUC.06	get App & App instances status info	SaaS	US.24, US.41, US.43, US.48	FR.01, FR.06, FR.12	Orchestration
SUC.07	control App & App instances	SaaS		FR.07, FR.08, FR.11	Orchestration
SUC.08	get notified about unwanted events	SaaS	US.41	FR.64, FR.27, FR.13	Orchestration
SUC.09	handle unwanted events	SaaS	US.33, US.49, US.50	FR.60, FR.74, FR.09	Orchestration
SUC.10	view suggested recovery actions	SaaS	US.50, US.51	FR.75, FR.76, FR.77, FR.78, FR.79, FR.09	Orchestration
SUC.11	choose/ perform recovery action(s)	SaaS	US.50, US.51	FR.75, FR.76, FR.77	Orchestration
SUC.12	update/ pause/ terminate App	SaaS	US.26	FR.03, FR.05, FR.13	Orchestration
SUC.13	start/ pause/ stop App instances	SaaS		FR.13	Orchestration
SUC.14	choose to migrate (parts of) components of App	SaaS		FR.18, FR.10	Orchestration
SUC.15	choose to scale up/ down (components of) App	SaaS		FR.04	Orchestration

### 3.2.2 IaaS manager

Figure 4 presents all the System Use Cases that depict actions which the system should perform in collaboration with the IaaS providers, so as to enable the management and evaluation of the corresponding IaaS functionalities.

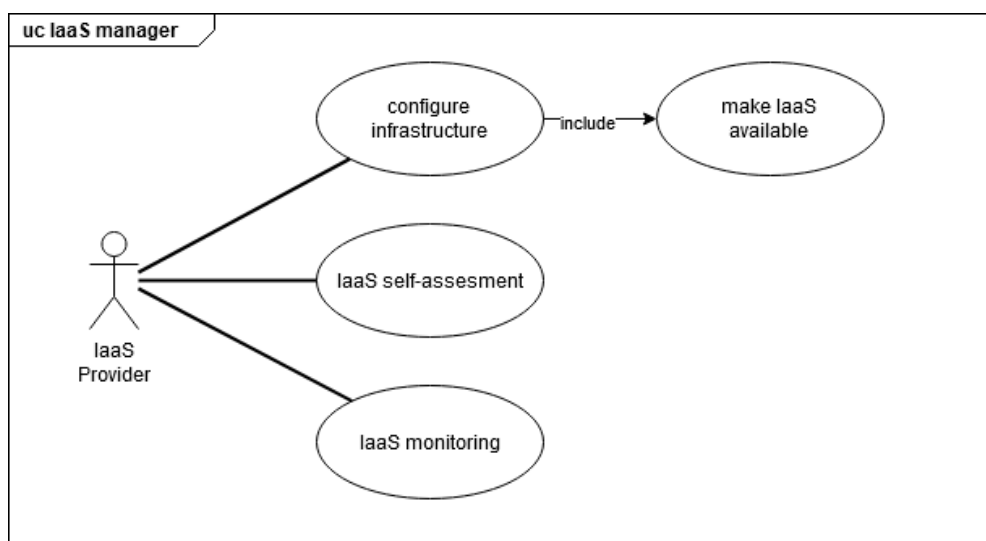


Figure 4: IaaS manager System Use Cases

<b>Document name:</b>	D2.3 Pledger Overall Architecture	<b>Page:</b>	17 of 40
<b>Reference:</b>	D2.3	<b>Dissemination:</b>	PU
	<b>Version:</b>	1.0	<b>Status:</b>
			Final

The following presents how these System Use Cases are related to specific Actors, User Stories, Requirements, and system functionalities (Functional Subsystems).

Table 4: IaaS manager System Use Cases

ID	System Use Case	Actor	Related USs	Related Requirements	Related Subsystem
SUC.16	configure infrastructure	IaaS	US.11, US.12, US.13, US.14	FR.19	Configuration
SUC.17	make IaaS available	IaaS	US.11, US.14		Configuration
SUC.18	IaaS self-assessment	IaaS	US.42, US.44, US.45, US.46, US.47	FR.26	Benchmarking
SUC.19	IaaS monitoring	IaaS	US.41, US.43, US.48	FR.64, FR.72, FR.73, FR.01, FR.06, FR.12	Orchestration

### 3.2.3 IaaS evaluation

Figure 5 presents all the System Use Cases that depict actions which the system should perform in collaboration with the SaaS providers, so as to enable the evaluation of IaaS functionalities.

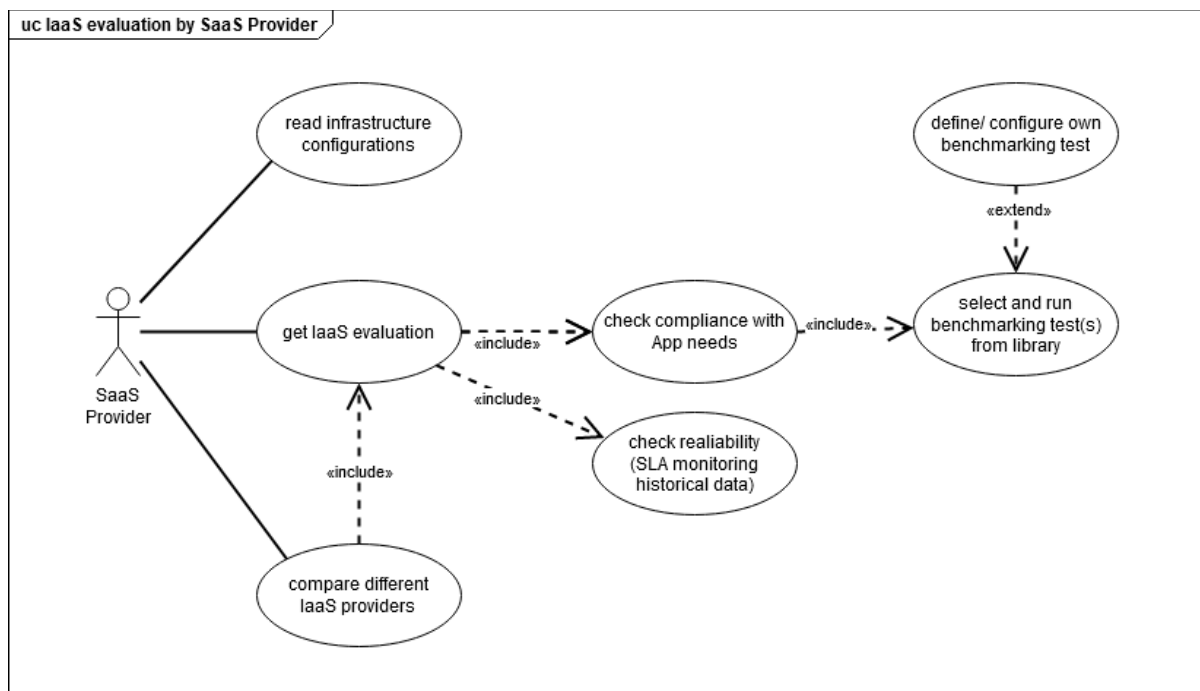


Figure 5: IaaS evaluation System Use Cases

The following presents how these System Use Cases are related to specific Actors, User Stories, Requirements, and system functionalities (Functional Subsystems).

Table 5: IaaS evaluation System Use Cases

ID	System Use Case	Actor	Related USs	Related Requirements	Related Subsystem
SUC.20	read infrastructure configurations	SaaS			Configuration
SUC.21	get IaaS evaluation	SaaS		FR.61, FR.68, FR.69, FR.70	Orchestration
SUC.22	compare different IaaS providers	SaaS		FR.61, FR.68, FR.69, FR.70, FR.63, FR.72, FR.26	Orchestration
SUC.23	check compliance (of IaaS) with App needs	SaaS	US.42, US.44, US.45, US.46, US.47, FR.02	FR.61, FR.67, FR.68, FR.69, FR.70, FR.26	Benchmarking
SUC.24	check reliability (of IaaS)	SaaS			SLAs/ Security
SUC.25	select and run benchmarking test(s) from library	SaaS	US.42, US.44, US.45, US.47	FR.61, FR.67, FR.65	Benchmarking
SUC.26	define/ configure own benchmarking test	SaaS	US.42, US.44, US.45, US.47	FR.61, FR.67, FR.71	Benchmarking

### 3.2.4 IaaS/ SaaS providers interactions

Figure 6 depicts the actions which the system should perform in collaboration with both the IaaS and the SaaS providers, so as to enable the interactions between these two types of actors.

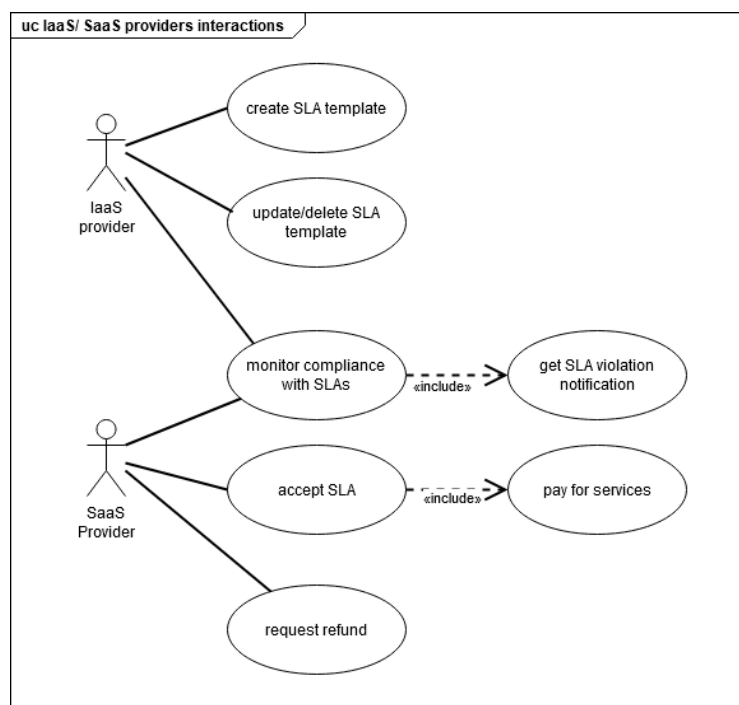


Figure 6: IaaS/ SaaS providers' interactions System Use Cases

The following table presents how these System Use Cases are related to specific Actors, User Stories, Requirements, and system functionalities (Functional Subsystems).

Table 6: IaaS/ SaaS providers interactions System Use Cases

ID	System Use Case	Actor	Related USs	Related Requirements	Related Subsystem
SUC.27	create SLA template	IaaS	US.35	FR.45, FR.55	SLAs/ Blockchain
SUC.28	update/ delete SLA template	IaaS		FR.44, FR.47	SLAs/ Blockchain
SUC.29	monitor compliance with SLAs	IaaS	US.49	FR.46, FR.51	SLAs/ Blockchain
SUC.30	get SLA violation notification	IaaS	US.43, US.49	FR.51, FR.27	SLAs/ Blockchain
SUC.31	accept SLA	SaaS	US.35	FR.49, FR.48, FR.50, FR.52, FR.53, FR.54, FR.55	SLAs/ Blockchain
SUC.32	pay for services	SaaS	US.36	FR.52, FR.53, FR.54, FR.57, FR.15	Blockchain
SUC.33	request refund	SaaS		FR.52, FR.53, FR.54, FR.57	Blockchain

## 4 Pledger Structural View

---

### 4.1 Introduction

---

This chapter introduces the main functionalities and functional components of Pledger and provides useful information related to the development of the Pledger system. More details are expected to be provided on this topic through future deliverables of WP3 “Performance, QoS and orchestration mechanisms” and WP4 “Trust, Smart Contracts and Decision Support mechanisms”. Each module will be specified as a component-based system with clearly defined internal and external interfaces. Application Programming Interfaces (APIs), data schemas and desired functionalities by the various modules will also be detailed.

Various constraints from the technological, scientific and market domain are taken under consideration, as captured in the requirements analysis (T2.2) and the various specifications for the Project Use Cases (T2.1). All technical partners collaborate and provide input regarding the expected architecture to meet the objectives set out in the project, especially with regards to novel edge/cloud computing contexts. Every partner focuses on the individual models that they are going to be responsible for during the implementation phase of WPs 3, 4 and 5, but also monitor the progress of the rest of the components, thus ensuring aligned development and seamless integration of the system components.

Following yet again the UML methodology, the Pledger Structural view is provided by the creation of Component diagrams.

### 4.2 Core System Overview

---

Considering the different objectives of Pledger, two main types of subsystems are identified:

- ▶ **Core subsystems:** The subsystems that belong to the Pledger core system. The Pledger core system is considered the system that will be the “minimum viable product” (MVP) of Pledger, the system that will be freely available to users and other projects and initiatives. It is the exploitable part of the project and it contains all the main features that will deem Pledger successful as a project. This chapter and the chapter 4.3 focus on these subsystems.
- ▶ **Use Cases subsystems:** The Pledger Use Case subsystems are considered the subsystems that are developed for the pilots of the project. Use Case subsystems are considered the subsystems that are used only for the sake of completing successfully the objectives of the pilots per se (and not of the project as a whole). These subsystems (and the corresponding components) will not be freely available or exploitable at the end of the project. Characteristic examples are the specific devices used in the pilots. These subsystems are linked to the UC-specific requirements presented in D2.2 “Pledger Requirements Analysis”. The specific subsystems are separately presented in chapter 4.4.

The core subsystems that have been identified are the following:

- 1 **Configuration subsystem:** Subsystem responsible for providing and storing the infrastructure and application configuration details.
- 2 **Orchestration subsystem:** Subsystem responsible for managing the orchestration of containerized applications (actions related to the deployment of applications, infrastructure scale up/down, etc.).
- 3 **Benchmarking subsystem:** Subsystem responsible for providing performance data of configured infrastructures to better characterise them and to optimise the orchestration with suggestions on application performance.
- 4 **SLAs subsystem:** Subsystem responsible for creating, managing and evaluating the SLAs associated to the applications running on the Pledger Cloud and Edge environment.

<b>Document name:</b>	D2.3 Pledger Overall Architecture	<b>Page:</b>	21 of 40
<b>Reference:</b>	D2.3	<b>Dissemination:</b>	PU
	<b>Version:</b>	1.0	<b>Status:</b> Final

- 5 **Blockchain subsystem:** Subsystem responsible for providing the Pledger DLT and the corresponding middleware (set of tools offering features such as smart contracts and cryptocurrencies).
- 6 **Big Data Platform subsystem:** Subsystem responsible for exposing a high-performance distributed streaming platform for interconnecting, storing, transforming.
- 7 **Security subsystem:** Subsystem responsible for enhancing the security of the overall system as a whole.

For convenience, these subsystems are split in three different functional groups:

- 1 **Management subsystems:** Subsystems that are focused in the management of IaaS and SaaS.
- 2 **Evaluation subsystems:** Subsystems that are focused in the evaluation of IaaS.
- 3 **Support subsystems:** Supporting subsystems related to communications and security.

The following figure gives the subsystems view of the Pledger Core System. The view is not hierarchical and the interrelations between the subsystems that are provided are the main ones.

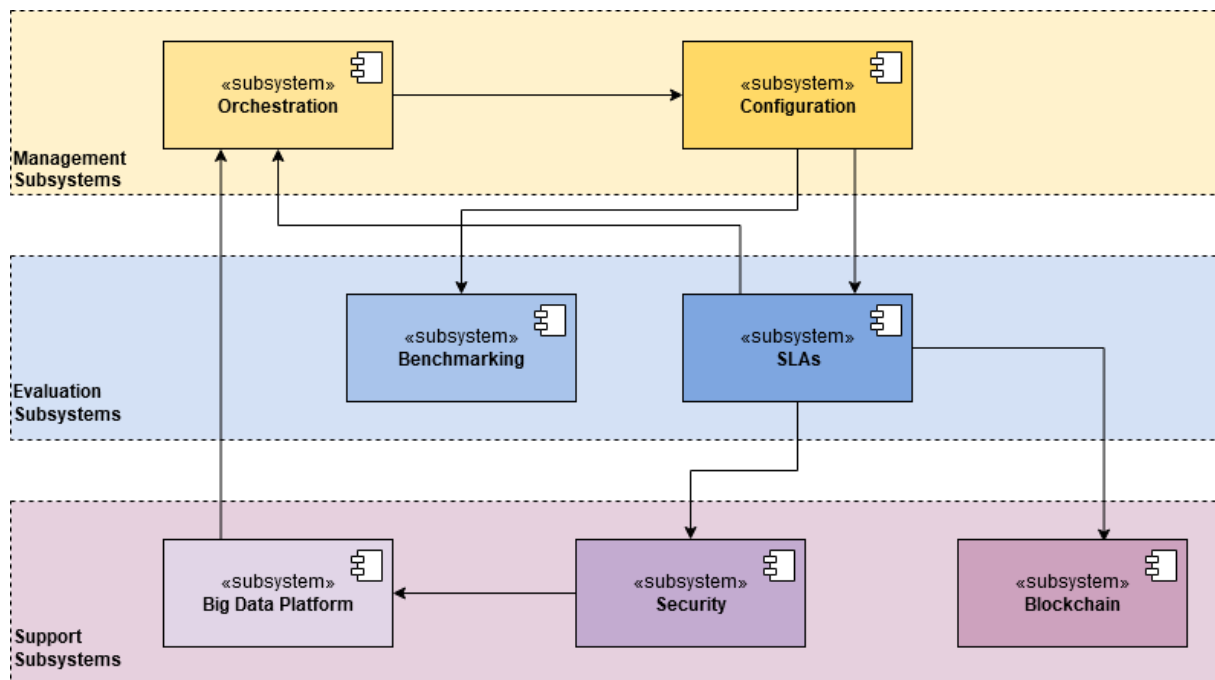


Figure 7: The Pledger Core Subsystems

<b>Document name:</b>	D2.3 Pledger Overall Architecture	<b>Page:</b>	22 of 40
<b>Reference:</b>	D2.3	<b>Dissemination:</b>	PU
	<b>Version:</b>	1.0	<b>Status:</b> Final

The following figure gives the overall view of the Pledger Core System in the form of a Component diagram. The diagram depicts all the different core components of the system as well as the associations between them (mostly input/output relations), with some of the associations being omitted to make the diagram easier to read. It also depicts the different subsystems as well as the different subsystem groups.

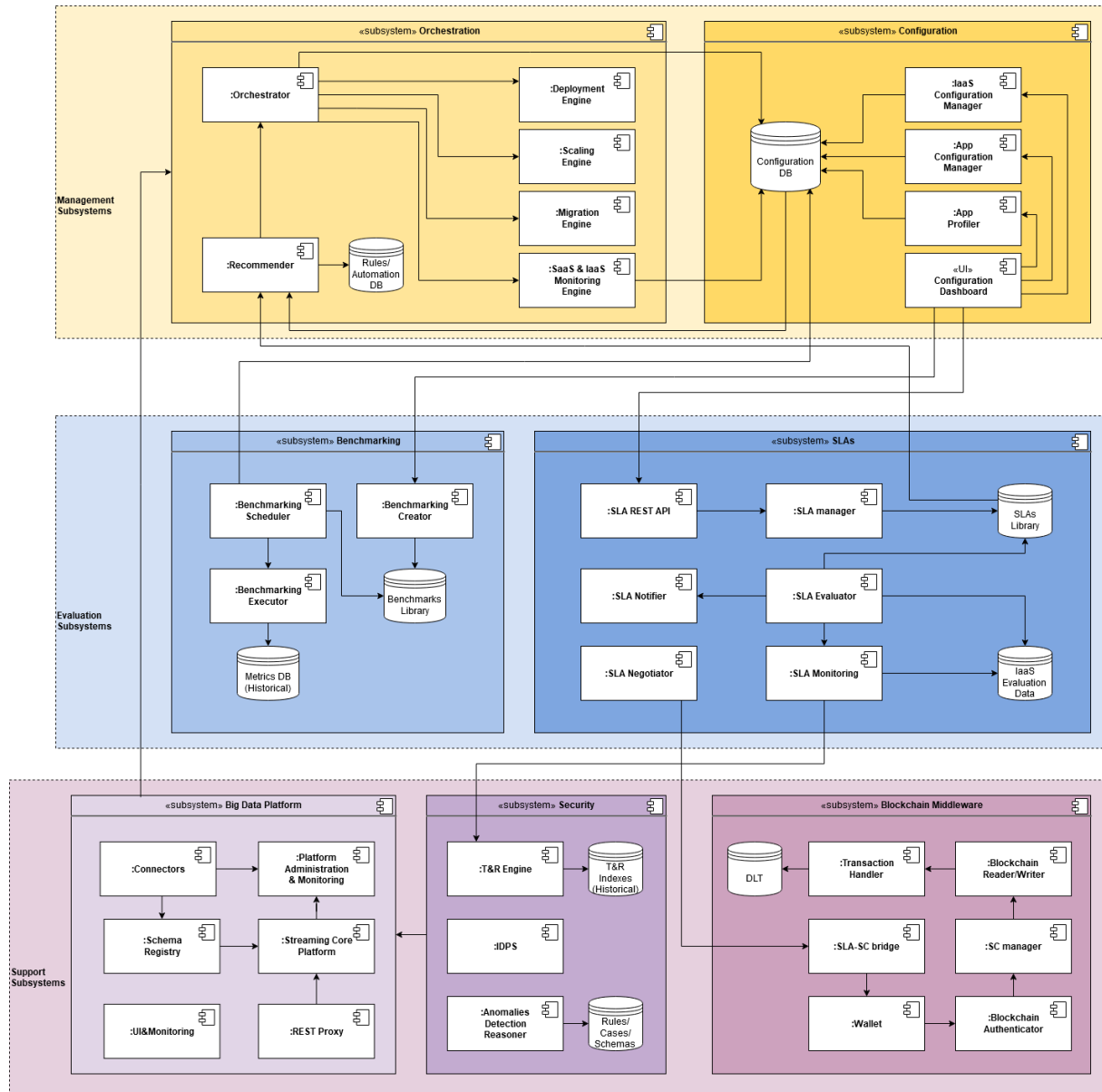


Figure 8: The Pledger Core System

From the above, it can be considered that Pledger will act as a PaaS.

In the following chapter, the different subsystems and components will be presented in more detail. A code will be assigned to each functional component following the pattern FC.X.Y, where X is the number of the corresponding subsystem.

<b>Document name:</b>	D2.3 Pledger Overall Architecture	<b>Page:</b>	23 of 40
<b>Reference:</b>	D2.3	<b>Dissemination:</b>	PU
<b>Version:</b>	1.0	<b>Status:</b>	Final

## 4.3 Core Subsystems & Functional Components

### 4.3.1 Configuration subsystem

The main responsibility of the Configuration subsystem is to store the infrastructure and application configuration that is managed either manually by the IaaS/ SaaS providers or by tools that support automatic discovery features (e.g. the App Profiler). This subsystem is related to the System Use Case diagrams presented in Chapters 3.2.1 and 3.2.2.

The main information stored is: users' configuration, infrastructure configuration, app configuration, multitenancy and authorizations for users to access specific infrastructures within resource limitations (e.g. resource quotas).

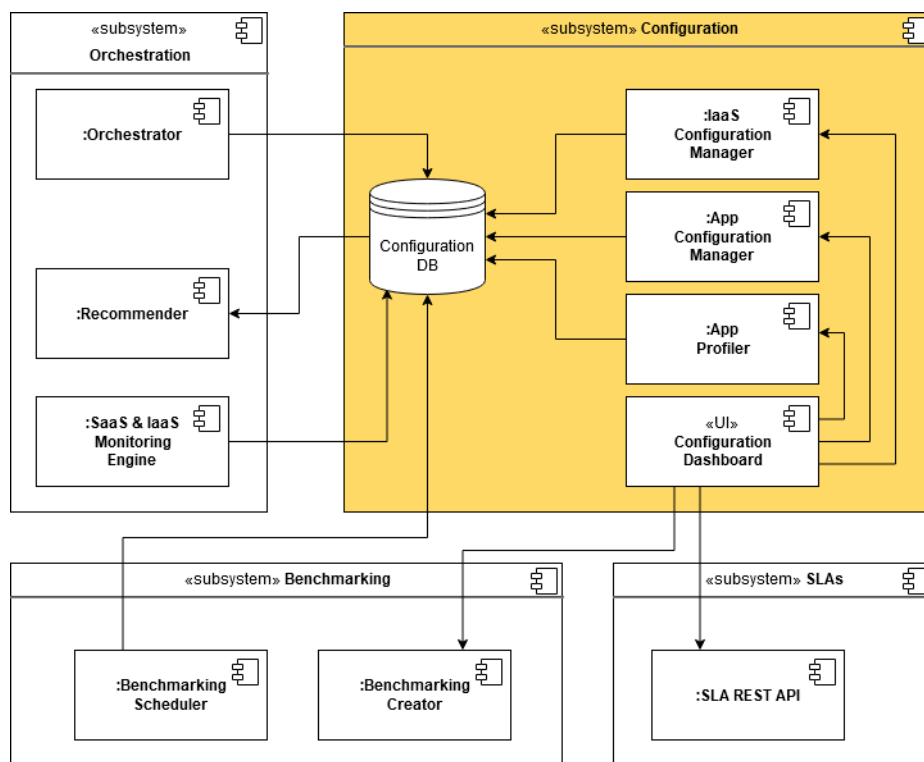


Figure 9: Configuration subsystem Component Diagram

More specifically, the subsystem comprises the following main components:

Table 7: Configuration subsystem Components

ID	Component	Functionality
FC.1.1	<b>IaaS Configuration Manager</b>	The IaaS Configuration Manager is responsible for the management of infrastructure configuration, spanning from the credentials to the main topology and properties of the infrastructure (such as the servers URI, the master/worker properties for Kubernetes, GPU type, CPU model, etc.) that could impact scheduling decisions, as well as resource limits configured from the IaaS providers.
FC.1.2	<b>App Configuration Manager</b>	The App Configuration Manager is responsible for the management of the app configuration, which includes generic information used to match SaaS providers' preferences expressed in their profiles, along with those specific related to QoS and SLAs. QoS keys are listed for each

<b>Document name:</b>	D2.3 Pledger Overall Architecture	<b>Page:</b>	24 of 40
<b>Reference:</b>	D2.3	<b>Dissemination:</b>	PU
	<b>Version:</b>	1.0	<b>Status:</b> Final

		application and SLA values/ thresholds are stored to allow the SLA Manager to check their violations and prioritize them.
FC.1.3	<b>App Profiler</b>	The App Profiler component is responsible for the manual or automatic profiling of applications in order to provide additional information about apps in the form of key-value set that can be used during the decision-taking phase before scheduling. The SaaS provider can add application-specific properties (e.g. GPU intensive, CPU intensive).
FC.1.4	<b>Configuration DB</b>	The Configuration DB is responsible for storing the aforementioned configuration information and sharing it through specific API to the other Pledger subsystems, such as the Recommender in the Orchestration subsystem, the Benchmarking and the SLA creators' components.
FC.1.5	<b>Configuration Dashboard</b>	The Configuration Dashboard is the UI provided to IaaS and SaaS users to allow the proper configuration of the aforementioned data and also includes reports to allow the SaaS users to have a detailed view of the infrastructures and apps status and the recommendations for the app orchestration.

#### 4.3.2 Orchestration subsystem

The Orchestration subsystem is the link between the Decision Support System (DSS) and the underlying infrastructure, providing an abstraction layer on top of infrastructure management tools of choice to the DSS. This way, it allows to decouple the DSS from the technical implementation details and facilitates interoperability with diverse technical solution and makes it possible to transparently update and change low level infrastructure management tools. The DSS (“Recommender” in the diagrams) will implement the intelligence of decision-making and the Orchestrator will be the executor arm of DSS decisions (implementing on demand the necessary actions).

This subsystem will manage orchestration of containerized applications (i.e. relying on Kubernetes implementation) to handle its management. The actions foreseen for this subsystem are related to the deployment of applications, infrastructure scale up/down and migration as well as operations in relation to the operational life-cycle of applications (start, stop, update and get current status of the application).

In addition, this subsystem also considers the management of complete clusters at different infrastructure levels (cloud, edge, on premise) as well as the deployment of cluster orchestration software (for single-node cluster) in the edge.

The Orchestration subsystem will be in charge of selecting the best nodes of a cluster (best effort) to run an application based on the SaaS provider’s preferences or the profile of the application (e.g. app requires GPU to run) or the recommendations added by DSS at runtime. The Orchestrator will also receive input from a Monitoring Engine to collect different metrics of the infrastructure in a time series database. With these functionalities we can settle the base for a QoS control system.

<b>Document name:</b>	D2.3 Pledger Overall Architecture	<b>Page:</b>	25 of 40
<b>Reference:</b>	D2.3	<b>Dissemination:</b>	PU
	<b>Version:</b>	1.0	<b>Status:</b> Final

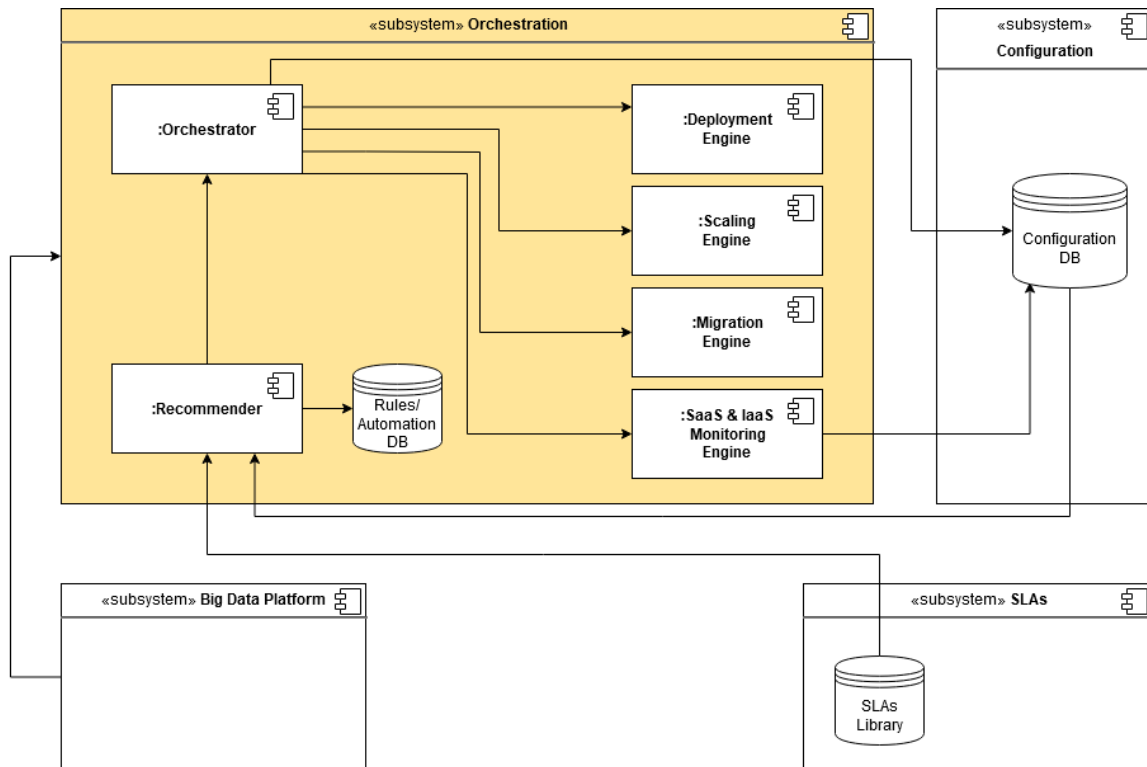


Figure 10: Orchestration subsystem Component Diagram

More specifically, the subsystem comprises the following main components:

Table 8: Orchestration subsystem Components

ID	Component	Functionality
FC.2.1	<b>Orchestrator</b>	This component is in charge of managing the information related to runtime environment specifications (profiles), such resources preferred/ required (like RAM, vCPU, etc.) for the applications, and of executing basic operations like start, stop, update or get status of applications in a cluster or infrastructure.
FC.2.2	<b>Deployment Engine</b>	This component is in charge of physically deploying applications in a cluster or a different infrastructure and undeploying (uninstalling) applications too. The “deployment” could be a first deployment, a redeployment due to runtime specification changes or an update of the version of the application.
FC.2.3	<b>Scaling Engine</b>	This component implements a horizontal scaling (scale out/ in) of the application creating replicas of the same application (or subcomponents of the application in case of microservices, Y axis in scale cube <sup>2</sup> ) or decreasing replicas. It also implements a vertical scaling (scale up) by calling the Migration Engine component to move the app to a node of the cluster with more resources (CPU, RAM, etc.) or with less resources (scale down).

<sup>2</sup> <https://uniknow.github.io/AgileDev/site/0.1.10-SNAPSHOT/scale-cube.html>

FC.2.4	<b>Migration Engine</b>	This component implements the migration of applications already deployed, by moving the app to different infrastructure with more resources (e.g. edge to cloud, other node type with more resources like RAM, CPU) or with less resource (e.g. cloud to edge). The reasoning for this migration covers different needs like colocation for decreasing latency, lift & shift for more computer capacity, etc.
FC.2.5	<b>Monitoring Engine</b>	This component collects metrics of cluster infrastructure and stores these values in a time series database. It is also the query interface for metric values from other subsystems or components.
FC.2.6	<b>Recommender</b>	The Recommender component is responsible for the provisioning of suggestions to the SaaS providers related to the app orchestration. In particular, the suggestions focus on the most efficient allocation within the available infrastructures, taking into account the infrastructure and app properties from the configuration subsystem, the infrastructure and app status from the IaaS Monitoring component and the SLA Manager, and the user preferences stored again in the configuration subsystem.
FC.2.7	<b>Rules/ Automation DB</b>	The Rules/ Automation DB component stores the Recommender suggestions and includes rules to decline the decision-making process (with regard to the user preferences) and possibly provide (small) automations to allow edge nodes act autonomously, depending on the resources available, whenever a disconnection from the infrastructure occurs.

### 4.3.3 Benchmarking subsystem

The main responsibility of the Benchmarking subsystem is to provide performance data of configured infrastructures to better characterise them and to optimise the orchestration with suggestions on application performance.

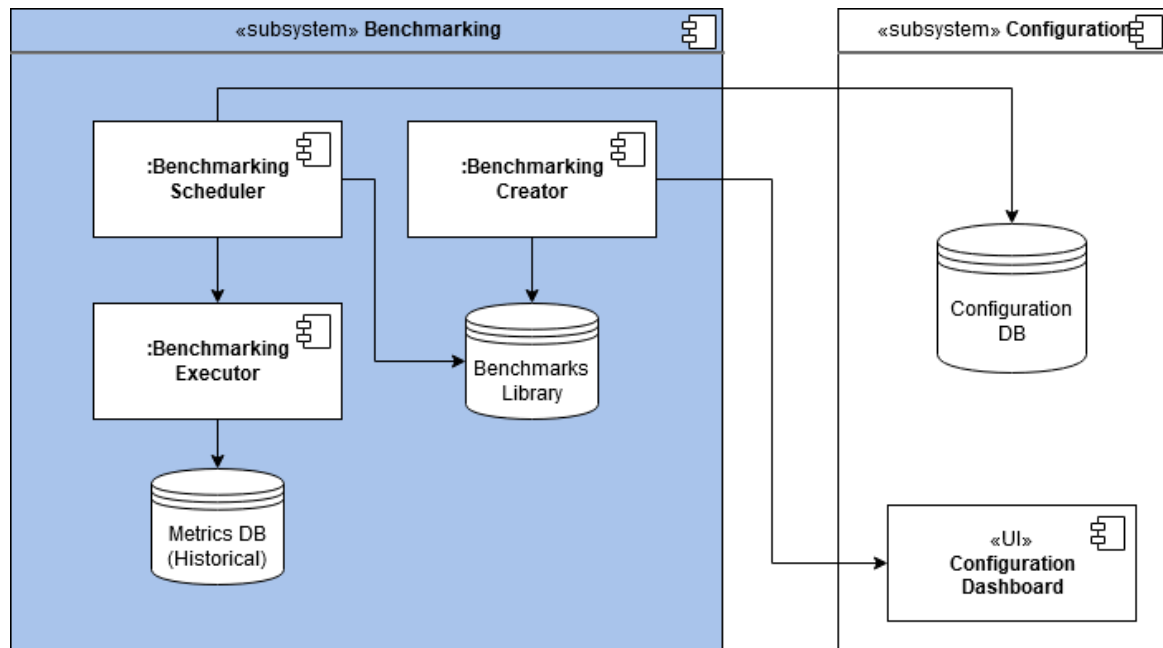


Figure 11: Benchmarking subsystem Component Diagram

<b>Document name:</b>	D2.3 Pledger Overall Architecture	<b>Page:</b>	27 of 40
<b>Reference:</b>	D2.3 <b>Dissemination:</b> PU	<b>Version:</b>	1.0 <b>Status:</b> Final

More specifically, the subsystem comprises the following main components:

Table 9: Benchmarking subsystem Components

ID	Component	Functionality
FC.3.1	<b>Benchmarking Creator</b>	The Benchmarking Creator is responsible for the creation of user-defined benchmarking tests to be included in the Benchmarking Library.
FC.3.2	<b>Benchmarking Scheduler</b>	The Benchmarking Scheduler is responsible for the scheduling of the executions of benchmarking tests on the infrastructures. Each schedule managed by this component executes a set of tests on a given infrastructure with a given frequency.
FC.3.3	<b>Benchmarking Executor</b>	The Benchmarking Executor is responsible for execute the tests. It connects to the target infrastructure, creates the needed resources with the required configuration, install and execute tests and collects the results.
FC.3.4	<b>Benchmarks Library</b>	The Benchmarking Library is a database of benchmark tests that include, for each test, the tools, the execution instructions and the result parsers. The library contains both generic tests available to all use cases, as well as tests defined by the user and used only for her specific use case.
FC.3.5	<b>Benchmarking Metrics DB</b>	The Benchmarking Metrics DB stores the benchmarking results of the executed tests. It provides an interface to query the results and perform mathematical operations over the results as well as returning the results in a timeseries format.

#### 4.3.4 SLAs subsystem

The SLAs subsystem is the subsystem responsible for creating, managing and evaluating the SLAs associated to the applications running on the Pledger Cloud and Edge environment. This subsystem relies on the information gathered by external monitoring tools, which are used to continuously evaluate the SLAs, and to notify other components about violations or other relevant information related to the QoS of these applications.

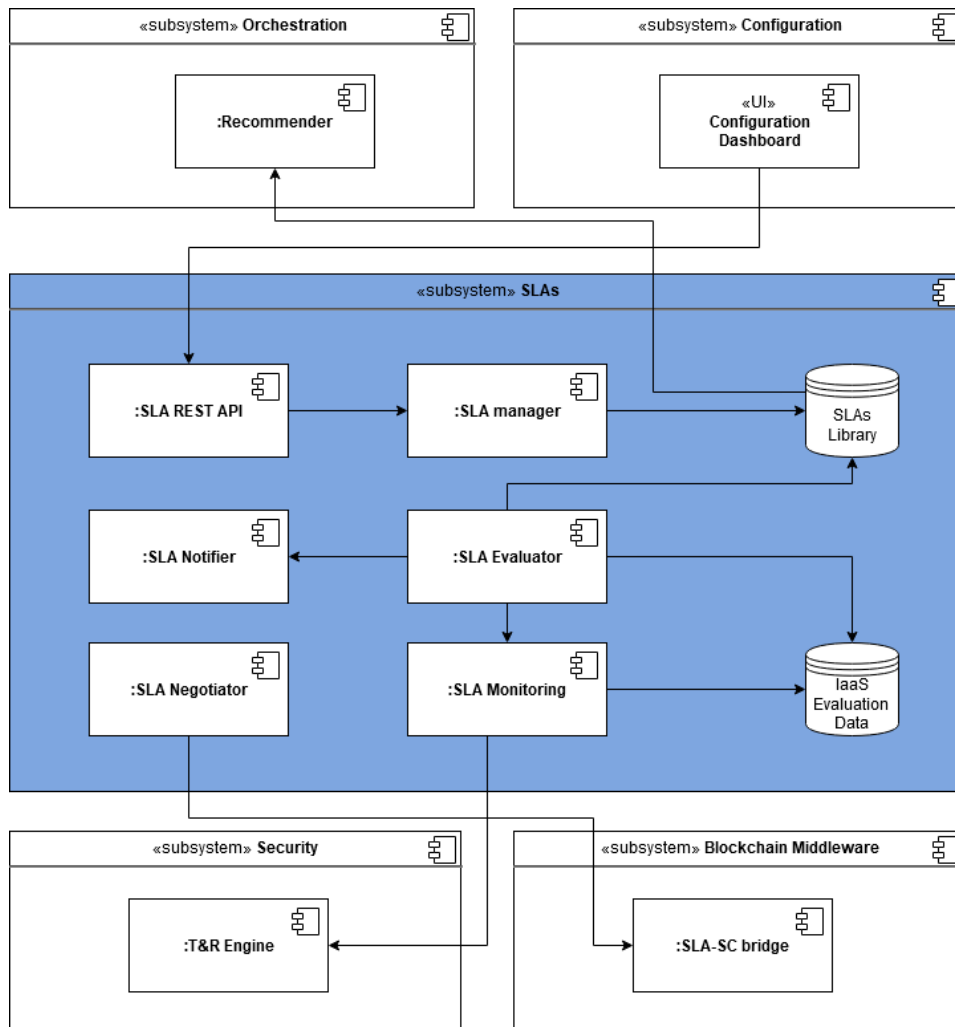


Figure 12: SLA subsystem Component Diagram

More specifically, the subsystem comprises the following main components:

Table 10: SLA subsystem Components

ID	Component	Functionality
FC.4.1	<b>SLA REST API</b>	The REST API module exposes all the methods needed to create and manage the SLAs, the SLA templates and the metrics associated to the applications running in the system. This module acts a facade and constitutes the entry point for other Pledger components and subsystems.
FC.4.2	<b>SLA Manager</b>	The SLA Manager component processes all the requests from the REST API. It offers two main features. On one hand, it is responsible for generating and storing the SLAs of the applications based on the QoS constraints defined by IaaS providers. On the other hand, this component handles the metrics used later by the Evaluator to check these SLAs.
FC.4.3	<b>SLAs library</b>	Both SLAs and associated metrics required for the SLA evaluation phase are stored in the SLAs library.
FC.4.4	<b>SLA Monitoring</b>	The SLA Monitoring component connects the SLA subsystem with a set of external monitoring tools by means of a set of plug-ins/ monitoring collectors

		(e.g. Prometheus) to get the metrics/ information about the applications and infrastructures behaviour.
FC.4.5	<b>SLA Evaluator</b>	The SLA Evaluator is the component responsible for the assessment of SLAs by checking that values obtained from monitoring metrics comply with the rules defined in the SLA template that correspond to a certain SLA stored in the system. First, it gets the metrics values from the monitor connectors (monitor module), and then it evaluates these metrics against the rules defined in the SLA applicable for a certain application or service.
FC.4.6	<b>IaaS Evaluation Data</b>	Both the monitoring and evaluator components collect the resultant information in the IaaS Evaluation DB.
FC.4.7	<b>SLA Notifier</b>	If the Evaluator detects a violation, it makes use of the SLA Notifier to spread this information to other Pledger components (e.g. the DSS). This component connects the SLA subsystem to other Pledger subsystems.
FC.4.8	<b>SLA Negotiator</b>	Through this component a SaaS reviews and accepts an SLA, ensuring there are no misunderstandings regarding the promised services and prices. In certain cases, modification of the available SLAs may be possible, facilitating real negotiation between IaaS providers and SLAs providers.

#### 4.3.5 Blockchain subsystem

Pledger will provide its own Blockchain/ Distributed Ledger, taking under consideration features such as openness, access, speed, security, use of consensus/ security mechanisms, etc. Depending on the implementation scenario chosen by the users (Blockchain as a Service – BaaS, vertical solutions, etc.) Pledger will also provide a set of tools offering features such as smart contracts and cryptocurrencies, thus enabling a wide range of applications under different business models.

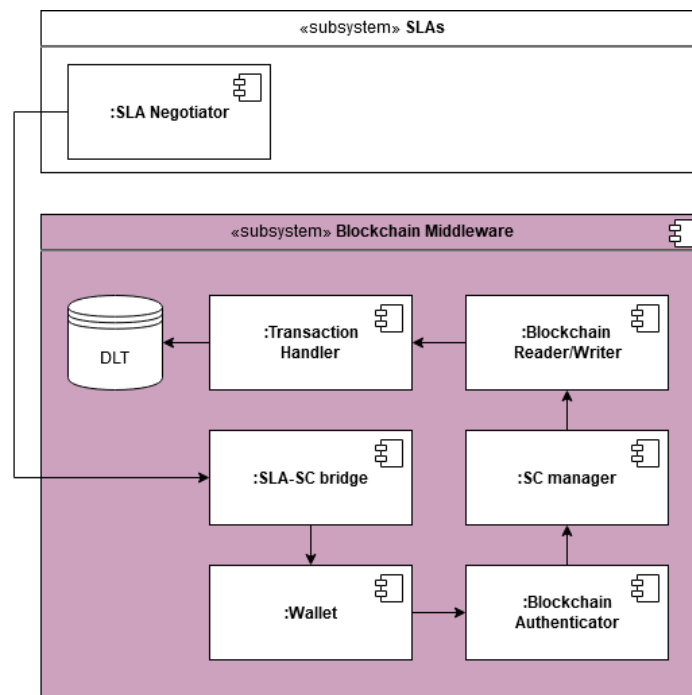


Figure 13: Blockchain subsystem Component Diagram

More specifically, the subsystem comprises the following main components:

**Table 11: Blockchain subsystem Components**

ID	Component	Functionality
FC.5.1	<b>Blockchain Authenticator</b>	This component grants access to specific data of the blockchain network, exploiting the corresponding credentials information of the components or users that are requesting the access, i.e. private keys and certificates.
FC.5.2	<b>Blockchain Reader/ Writer</b>	This component fetches the requested data from the blockchain and submits transactions on it. The module consists of various methods that describe the way and technique or query specification the information is read from the blockchain. Different data need different kinds of reading and fetching; hence the module is constituted from several sub-modules.
FC.5.3	<b>Wallet</b>	This component manages the credential information of a blockchain participant entity (either it is a component, administrator, end-user or other). The module is the main method with which an entity is present inside the blockchain network through its transaction activities and smart contract or DApps deployments.
FC.5.4	<b>SLASC</b>	The <b>SLA-SC bridge</b> links the SLA contractual terms with the blockchain system. The component describes the methods and functions that define the connection between SLAs and smart contracts in the project while it introduces the one-to-one representation of the SLA terms into smart contract specific contractual terms, being executable code that is triggered by external systems.
FC.5.5	<b>Smart Contract Manager</b>	This component is responsible for all the activities regarding smart contracts, from deployment to maintenance. The component is managing the various deployed smart contracts inside the blockchain network while its main goal is to holistically sustain the Pledger system of smart contracts inside the Pledger blockchain.
FC.5.6	<b>Transaction Handler</b>	This component administers and maintains the transaction flows of the blockchain network. The module subtly checks and confirms the transactions validations and endorsements on the background, while at the same time it offers an apparent role of a gatekeeper where errors or other malfunctions occur.
FC.5.7	<b>DLT</b>	Pledger's blockchain network.

#### 4.3.6 Big Data Platform subsystem

The Big Data Platform subsystem is the subsystem that exposes a high-performance distributed streaming platform for interconnecting, storing, transforming. It can efficiently ingest and handle massive amounts of data into processing pipelines, for both real-time and batch processing.

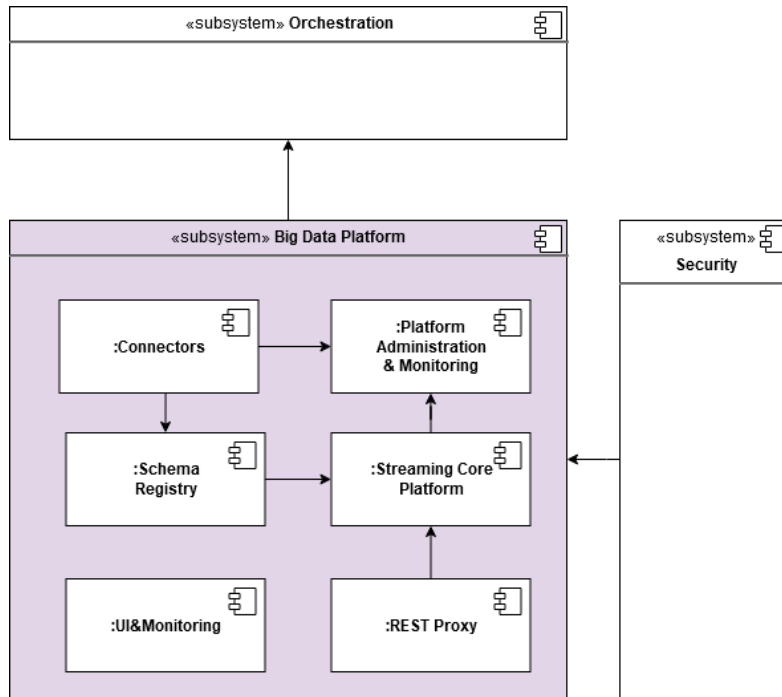


Figure 14: Big Data Platform subsystem Component Diagram

More specifically, the subsystem comprises the following main components:

Table 12: Big Data Platform subsystem Components

ID	Component	Functionality
FC.6.1	<b>Platform Administration</b>	This component utilises a centralized service for maintaining cluster configuration information and naming and for providing distributed synchronization among the subsystem components. The service sends changes of the topology to the Kafka cluster, so each node in the cluster knows when a new broker joined, a broker died, a topic was removed, or a topic was added.
FC.6.2	<b>Streaming Core Platform</b>	The component includes a cluster with three or more Brokers which provide a setup of adequate availability and replication for many scenarios. The Brokers themselves deal with the actual storage of the incoming data and provide the necessary functionality for producers and consumers to perform their work. Information is stored in streams of messages or records that belong to a particular category called a Topic.
FC.6.3	<b>Connectors</b>	This component facilitates the performant streaming of data between data systems in a scalable, reusable, and reliable fashion. A Connect cluster consists of a set of Worker processes that are containers that execute Connectors and Tasks. Workers automatically coordinate with each other to distribute work and provide scalability and fault tolerance.
FC.6.4	<b>REST Proxy</b>	The importance of the REST Proxy arises when there is a need to interact with an application implemented using technologies that are not supported already by the subsystem or in cases where an administrative interface is built to better manage and review the state of the Cluster. The REST Proxy provides a RESTful interface to a Kafka cluster, making it easy to produce

		and consume messages, view the state of the cluster, and perform administrative actions without using the native Kafka protocol or clients.
FC.6.5	<b>Schema Registry</b>	The Schema Registry aims to fulfil the role of maintaining and serving all the necessary Avro schemas information related to the data models of the communicated messages of the system, via a well-defined RESTful interface. Registered schemas are given unique IDs and the registry can maintain a versioned history of the schemas in question with the actual storage of this information performed in a dedicated Kafka topic.
FC.6.6	<b>UI &amp; Monitoring</b>	For enabling easy overview and access to the functionality offered by the several components of the Big Data Platform, an extensive set of GUI components is configured and is available for monitoring, administration, and configuration usage.

#### 4.3.7 Security subsystem

By using blockchains one can also mitigate several of the trust issues. However, a multi-layer approach is required to address security challenges at the edge. The Security subsystem includes components that enhance the security of the overall system as a whole. Of course, several other security features and mechanisms are included in other components of other subsystems, but this subsystem is specifically focused on the security aspects of Pledger.

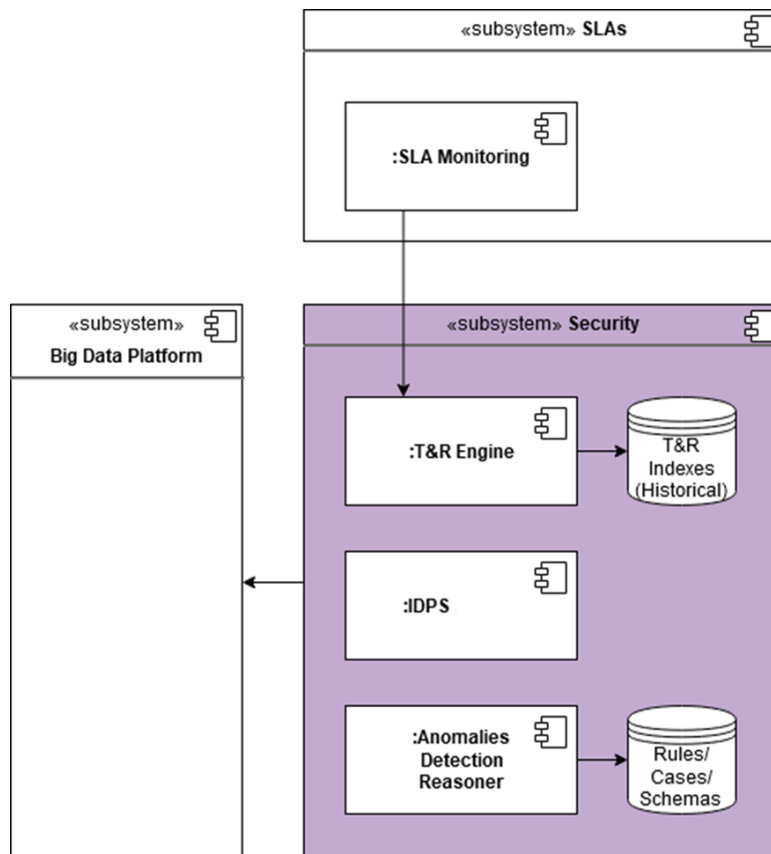


Figure 15: Security subsystem Component Diagram

More specifically, the subsystem comprises the following main components:

**Table 13: Security subsystem Components**

ID	Component	Functionality
FC.7.1	<b>IDPS</b>	IDPS stands for Intrusion Detection and Prevention System. The component is used as network threat detection engine with real time intrusion detection capabilities. Such a component, when placed in the Edge to Cloud gateway may inspect network traffic using different rules and detect possible threats.
FC.7.2	<b>Anomalies Detection Reasoner</b>	The Anomalies Detection Reasoner is responsible for the identification of rare items, events or observations which raise suspicions by differing significantly from the majority of the data. Typically, the anomalous items will translate to some kind of problem such as fraud or a structural defect. Several reasoning techniques will be considered for the implementation of this component, including Rules Based Reasoning (RBR), Case Based Reasoning (CBR), and ML algorithms.
FC.7.3	<b>Rules/ Cases/ Schemas</b>	This DB will include the identifiers of anomalies. Depending on the reasoning technique that will be selected, it may include Cases, Rules or other schemas.
FC.7.4	<b>T&amp;R Engine</b>	This component assigns Trust & Reputation indexes to the IaaS providers, based on their performance and reliability. That way, the ranking of the IaaS providers based on their trustworthiness is enabled.
FC.7.5	<b>T&amp;R Indexes</b>	In this DB the T&R stores the calculated Trust & Reputation indexes, thus making possible the historical trustworthiness evaluation of actors.

## 4.4 Project Use Cases Subsystems

### 4.4.1 UC1 subsystem – Mixed reality applications on the edge

The goal of UC1 is to enhance the capabilities of AR/MR/VR solutions by coupling them with edge computing technologies and the corresponding load allocation and optimisation tools, in order to provide high-level services in industrial environments.

More specifically, the industrial environment which will run the pilots of UC1 is the one provided in Use Case 3 (see D2.1 “Pledger Detailed Use Cases”). In particular, the UC will focus on three case studies integrating machine data into MR interfaces and optimising them via edge computing technology:

- ▶ Collaborating in Fast Prototyping
- ▶ Assisting in Manufacturing and Service & Utility procedures
- ▶ Enhancing Training & Interaction

Working on top of the input provided in Chapters 2.2.3 and 2.3.3 of D2.1, the following Component diagram is extracted for this specific Use Case.

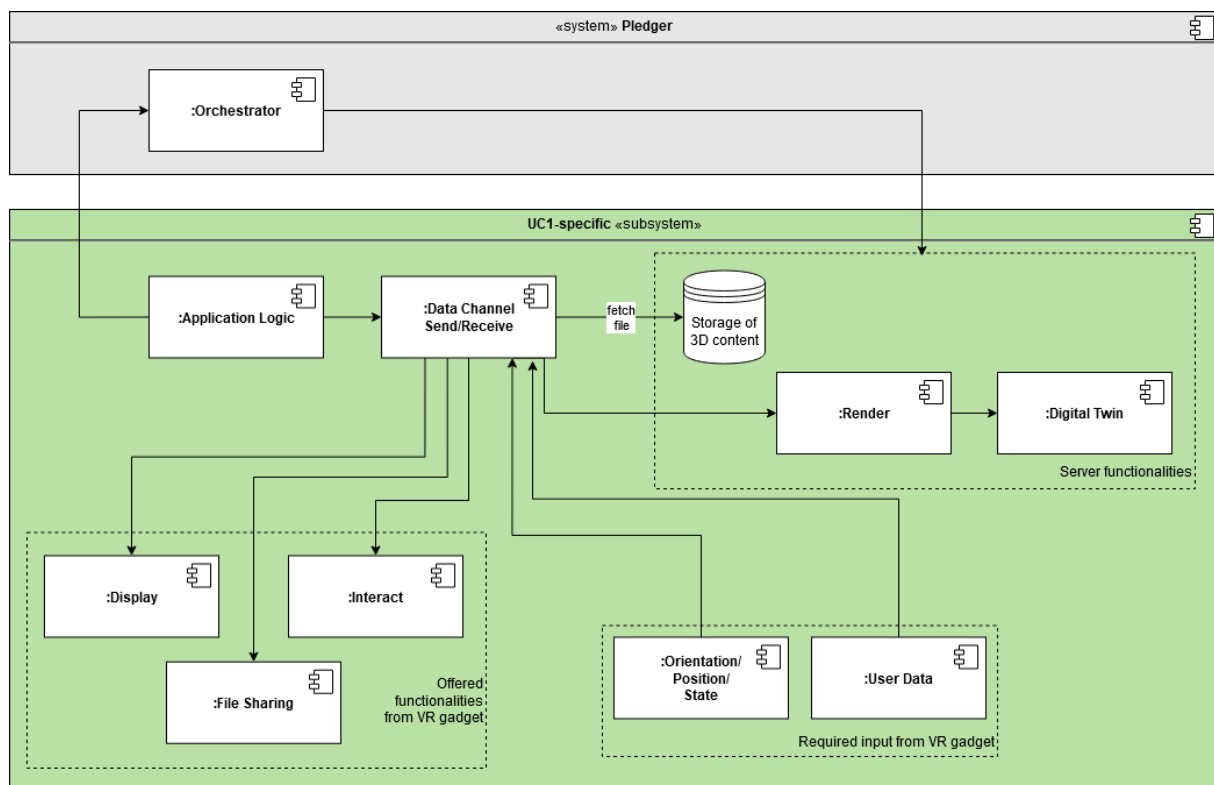


Figure 16: UC1 subsystem Component Diagram

<b>Document name:</b>	D2.3 Pledger Overall Architecture	<b>Page:</b>	35 of 40	
<b>Reference:</b>	D2.3	<b>Dissemination:</b>	PU	
	<b>Version:</b>	1.0	<b>Status:</b>	Final

#### 4.4.2 UC2 subsystem – Edge Infrastructure for safety of VRUs

The main goal of UC2 is to increase safety for VRUs using an edge compute-enabled infrastructure, as it is provided by the Pledger framework. In general terms, there are many ways to provide and to increase safety for VRUs in cities. Safety can come from having a well-defined regulatory framework that considers VRUs and their interactions with other vehicles and the infrastructure. Safety can also be implemented by building dedicated infrastructure elements and by adapting the city layout to respect the mobility of VRUs, such as dedicated bicycle lanes or pedestrian areas.

Safety can also come from using state of the art ICT technologies to implement a “smart” infrastructure: using sensors to gather data about the position and trajectory of VRUs and other mobile elements (cars, public transport, etc.), algorithms to process the gathered information and notification systems to alert VRUs if situations of risk are detected. This latter approach to increase security is the one that is investigated in this UC.

Working on top of the input provided in Chapters 3.2.3 and 3.3.3 of D2.1 “Pledger Detailed Use Cases”, the following Component diagram is extracted for this specific Use Case.

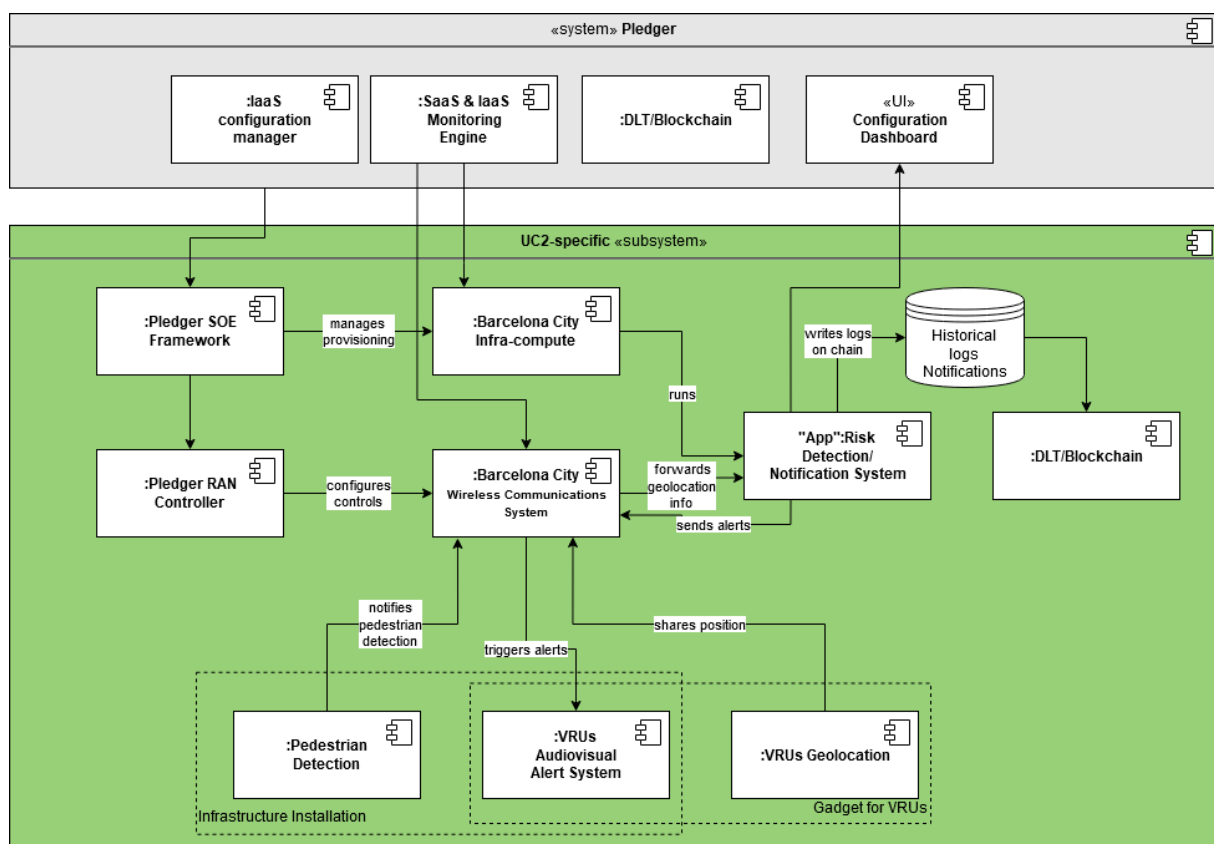


Figure 17: UC2 subsystem Component Diagram

<b>Document name:</b>	D2.3 Pledger Overall Architecture	<b>Page:</b>	36 of 40
<b>Reference:</b>	D2.3	<b>Dissemination:</b>	PU
	<b>Version:</b>	1.0	<b>Status:</b> Final

#### 4.4.3 UC3 subsystem – Data mining on the edge

UC3 will pick up the needs of machine tools operators, especially workers and maintenance engineers and the developed assets will be tested and demonstrated in the UC. The subsystem is most likely to assist operators, guide maintenance and service tasks and prepare data for algorithms designed for machine learning.

Working on top of the input provided in Chapters 4.2.3 and 4.3.3 of D2.1 “Pledger Detailed Use Cases”, the following Component diagram is extracted for this specific Use Case.

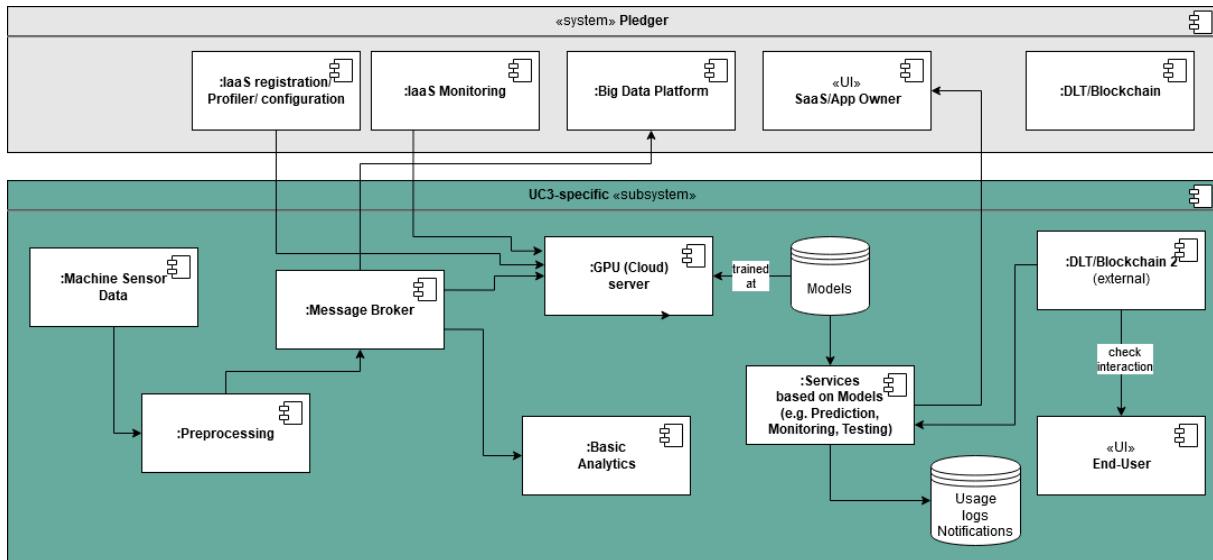


Figure 18: UC3 subsystem Component Diagram

<b>Document name:</b>	D2.3 Pledger Overall Architecture	<b>Page:</b>	37 of 40
<b>Reference:</b>	D2.3	<b>Dissemination:</b>	PU
	<b>Version:</b>	1.0	<b>Status:</b> Final

## 5 Development Process

The following table presents some of the basic tools and platforms that have been identified as appropriate solutions for the materialisation of the Pledger Core Subsystems. These assets form the backbone of most of the core components/ subsystems, but of course, several adaptations are needed, especially in the context of extending the assets capabilities towards scenarios on the edge.

Table 14: Assets list

ID	Asset	Partner	Description
AS.1	LifeCycle Manager	ATOS	An Edge-To-Cloud Resource Orchestration tool that provides life-cycle management for composed services described as collections of containers and relying in container orchestration tools such as Docker Swarm and Kubernetes.
AS.2	Benchmarking Suite	ENG	A benchmarking process orchestrator that wraps several third-party benchmark tools and schedules, executes and analyses their results. It supports different target infrastructures (e.g. Cloud, Kubernetes) and provide programmatic APIs to query results.
AS.3	ATOS SLA Framework	ATOS	The SLA management component provides a mechanism that permits the definition and enforcement of QoS parameters for services to be executed in Edge and across Fog and Cloud environments.
AS.4	3ALib & QoET	ICCS	An SLA monitoring library with extension to analytics queries for identification of key metrics, improvement of front-ends and multi-user functionality, improved packaging and deployment process.
AS.5	Hyperledger Blockchain & Middleware	INNOV	Hyperledger is an open source collaborative effort created to advance cross-industry blockchain technologies. It is a global collaboration, hosted by The Linux Foundation, including leaders in finance, banking, IoT, supply chains, and Technology.
AS.6	Streamhandler	INTRA	A high-performance distributed streaming platform for interconnecting, storing, transforming and processing data as well as training, testing and executing machine learning and artificial intelligence algorithms.

The following table summarises the links between the Pledger components, the used assets, and the partners and tasks responsible for the development, extension and adaptation of the corresponding tools.

Table 15: Mapping of components to assets, partners, and tasks

ID	Component	Task	Asset	Partner
FC.1.1	IaaS Configuration Manager	Task 3.1		ENG
FC.1.2	App Configuration Manager	Task 3.1		ENG
FC.1.3	App Profiler	Task 3.1		ENG & ICCS
FC.1.4	Configuration DB	Task 3.1		ENG
FC.1.5	Configuration Dashboard	Task 3.1		ENG
FC.2.1	Orchestrator	Task 3.2	AS.1	ATOS

<b>Document name:</b>	D2.3 Pledger Overall Architecture	<b>Page:</b>	38 of 40
<b>Reference:</b>	D2.3 <b>Dissemination:</b> PU	<b>Version:</b>	1.0 <b>Status:</b> Final

FC.2.2	Deployment Engine	Task 3.2	AS.1	ATOS
FC.2.3	Scaling Engine	Task 3.2	AS.1	ATOS
FC.2.4	Migration Engine	Task 3.2	AS.1	ATOS
FC.2.5	Monitoring Engine	Task 3.2	AS.1	ATOS
FC.2.6	Recommender	Task 4.3		ENG
FC.2.7	Rules/ Automation DB	Task 4.3		ENG
FC.3.1	Benchmarking Creator	Task 3.1	AS.2	ENG
FC.3.2	Benchmarking Scheduler	Task 3.1	AS.2	ENG
FC.3.3	Benchmarking Executor	Task 3.1	AS.2	ENG
FC.3.4	Benchmarks Library	Task 3.1	AS.2	ENG
FC.3.5	Benchmarking Metrics DB	Task 3.1	AS.2	ENG
FC.4.1	SLA REST API	Task 3.3	AS.3	ATOS
FC.4.2	SLA Manager	Task 3.3	AS.3	ATOS
FC.4.3	SLAs library	Task 3.3	AS.3 & AS.4	ATOS & ICCS
FC.4.4	SLA Monitoring	Task 3.3	AS.3 & AS.4	ATOS & ICCS
FC.4.5	SLA Evaluator	Task 3.3	AS.3 & AS.4	ATOS & ICCS
FC.4.6	IaaS Evaluation Data	Task 3.3	AS.3 & AS.4	ATOS & ICCS
FC.4.7	SLA Notifier	Task 3.3	AS.3 & AS.4	ATOS & ICCS
FC.4.8	SLA Negotiator	Task 3.3	AS.3 & AS.4	ATOS & ICCS
FC.5.1	Blockchain Authenticator	Task 4.2	AS.5	INNOV
FC.5.2	Blockchain Reader/ Writer	Task 4.2	AS.5	INNOV
FC.5.3	Wallet	Task 4.2	AS.5	INNOV
FC.5.4	SLASC	Task 4.2		INNOV & ICCS
FC.5.5	Smart Contract Manager	Task 4.2	AS.5	INNOV
FC.5.6	Transaction Handler	Task 4.2	AS.5	INNOV
FC.5.7	DLT	Task 4.2	AS.5	INNOV
FC.6.1	Platform Administration	Task 4.1	AS.6	INTRA
FC.6.2	Streaming Core Platform	Task 4.1	AS.6	INTRA
FC.6.3	Connectors	Task 4.1	AS.6	INTRA
FC.6.4	REST Proxy	Task 4.1	AS.6	INTRA
FC.6.5	Schema Registry	Task 4.1	AS.6	INTRA
FC.6.6	UI & Monitoring	Task 4.1	AS.6	INTRA
FC.7.1	IDPS	Task 4.1		INTRA
FC.7.2	Anomalies Detection Reasoner	Task 4.1		All
FC.7.3	Rules/ Cases/ Schemas	Task 4.1		All
FC.7.4	T&R Engine	Task 4.1		ICCS
FC.7.5	T&R Indexes	Task 4.1		ICCS

## 6 Conclusions & Next Steps

---

In this document, the first version of the Pledger behavioural and structural views have been successfully provided. The main outcome of the deliverable is the Pledger Core System view provided in Figure 8. In total, 43 core components, 7 core subsystems and 3 functional groups have been identified, and their descriptions and interrelations have also been provided. Moreover, all these components and subsystems have been mapped to specific Tasks of the project, potential Assets (as solutions), and partners, while their connection to the Requirements Analysis and Use Cases description has also been identified.

The above results will be directly used not only for the coordination of the development activities in WPs 3 and 4, and the integration activities in WP5 “Integration, Pilots and Overall Evaluation”, but also for the creation of the corresponding deliverables. More specifically, the deliverables of WP3 “Performance, QoS and orchestration mechanisms” and WP4 “Trust, Smart Contracts and Decision Support mechanisms” (one for each Task) will follow Table 15 and will split the description of components following the subsystems structure provided in Chapter 4. Similarly, D5.2 “Pledger integrated demonstrator” will use Figure 8 as the baseline for the development of the integrated Pledger system.

To sum up, this deliverable provides the full specification and architectural layout of the Pledger system and includes the description of the Pledger functionalities and UML diagrams. Activities related to the refinement of the Pledger Architecture will continue throughout the second year of the project. The final version of the deliverable will include:

- ▶ an updated view and description of the core and UC-specific components (new components may have to be introduced, following the maturity of the project, such as privacy-enhancing components).
- ▶ deployment models and diagrams providing the specifications for the deployment view of the architecture.
- ▶ sequence diagrams providing even more details regarding the interrelations between components.
- ▶ a common data model pretraining most of the data and information flows of the system.

<b>Document name:</b>	D2.3 Pledger Overall Architecture			<b>Page:</b>	40 of 40
<b>Reference:</b>	D2.3	<b>Dissemination:</b>	PU	<b>Version:</b>	1.0
				<b>Status:</b>	Final