PANTHEON

Community-Based Smart City Digital Twin Platform for Optimised DRM operations and Enhanced Community Disaster Resilience

D3.7

Overall Architecture and High-level Functionalities

The project has received funding from the European Union's Horizon Europe programme under Grant Agreement N°101074008.

DOCUMENT INFO

Deliverable Number	D3.7
Work Package Number and Title	3 Requirements, Participatory Design Process and Pilot Use-Cases Specifications
Lead Beneficiary	THL
Due date of deliverable	28/02/2024
Deliverable type[1]	R
Dissemination level[2]	PU
Author(s)	Mike Karamousadakis (THL); George Stergiopoulos (THL); Lampis Papakostas (THL); Iacob Crucianu (SIMAVI); Otilia Bularca (SIMAVI); Jean-Philippe Crepin (M3SB); Fanis Fakoukakis (FINT); Cristina Barrado (UPC); Thanos Kyritsis (INTEROPT); Jim Sharples (ENAC); Marc Bonazountas;
Internal reviewer(s)	Benjamin Schuster (JOAFG), Ana-Maria Dumitrescu (SIMAVI)
Version - Status	1.3

TASK ABSTRACT

This deliverable presents the analysis of the technical specifications and the design of the high-level reference architecture of PANTHEON. The deliverable focuses on the specification of the PANTHEON platform architecture, showing how the platform components are connected to each other. Each component of the architecture is described in detail, covering Data Aggregators, Data pre-processing and curation, Data Analysis, ML models, Concept Models, Simulation models, Decision Support Systems, What-if analysis, UAV swarming schemes and resource controllers and User Interfaces. In addition, the deliverable includes sequence diagrams showing how the platform components interact with each other to perform the core functions of the platform and to support the PANTHEON usage scenarios. Finaly additional details related to integration, deployment and verification of the platform are provided.

[1] **Please indicate the type of the deliverable using one of the following codes**:
R = Document, report
DEM = Demonstrator, pilot, prototype, plan designs
DEC = Websites, patents filing, press & media actions, videos
DATA = data sets, microdata
DMP = Data Management Plan
ETHICS: Deliverables related to ethics issues.
OTHER: Software, technical diagram, algorithms, models, etc.

[2] **Please indicate the dissemination level using one of the following codes**:
PU = Public
SEN = Sensitive

REVIEW HISTORY

Version	**Date**	**Modifications**	**Editor(s)**
1.0	26/04/2024	First draft	Mike Karamousadakis, George Stergiopoulos, Lampis Papakostas **(THL)**; Iacob Crucianu, Otilia Bularca **(SIMAVI**); Jean-Philippe Crepin **(M3SB)**; Fanis Fakoukakis **(FINT)**; Cristina Barrado **(UPC)**; Thanos Kyritsis **(INTEROPT)**; Jim Sharples **(ENAC)**; Marc Bonazountas **(EPSILON)**
1.1	09/05/2024	Internal Review	Benjamin Schuster (JOAFG)
1.2	15/05/2024	Internal Review	Ana-Maria Dumitrescu (SIMAVI)
1.3	20/06/2024	Final version	Mike Karamousadakis (THL)

DISCLAIMER

Funded by the European Union. Views and opinions expressed are, however, those of the author(s) only and do not necessarily reflect those of the European Union or European Commission. Neither the European Union nor the granting authority can be held responsible for them.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Description
AI	Artificial Intelligence
API	Application Programming Interface
CBDRM	Community-based Disaster Risk Management
FR	Functional Requirement
GIS	Geographic Information System
IT	Information Technology
IUC	Intended Use and Classification
DAG	Directed Acyclic Graphs
JSF	Java Server Faces
JMS	Java Messaging System
ML	Machine Learning
NFR	Non-functional Requirement(s)
SCDT	Smart City Digital Twin
UAV	Unmanned Aerial Vehicle

EXECUTIVE SUMMARY

Background	Goals
To enable the development of the PANTHEON technical components, the system specifications and the high-level reference architecture have been defined and described in detail. The work also constitutes a basis for consolidating, testing, evaluating, and refining the results by various work tasks throughout the project.	The aim of this Deliverable is to provide the system specifications and the design of the high-level reference architecture of PANTHEON, so that the development of individual components can take place in WP4, WP5 and WP6 along with the integration of the platform in WP7. Specifically, the work presented builds upon the work done in D2.4, D3.1, D3.2, D3.3, D3.4 and D3.6, by further specifying (i) the necessary system requirements to fulfil the user requirements described in D3.2 and D3.6 and (ii) the technical components that participate in the architecture along with the sequence diagrams of their interconnection after analysis on (a) the existing technological landscape (D2.4 & D3.1) (b) the conceptual model of the SCDT (D3.3), (c) the data delivery schemes in D3.4 and (d) the usage scenarios and use cases as documented in D3.6.

Approach and course of action

Utilise work achieved in previous work packages and previous tasks:

- WP2 PANTHEON Approach for Building Disaster-Resilient Communities
- WP3 PANTHEON Requirements, Participatory Design Process and Pilot Use-Cases Specifications

A requirements engineering approach was followed, including three workshops with the relevant technical partners for further defining the requirements of the PANTHEON platform along with its technical components and their interconnections.

Findings and results

Herewith the results of the approach are presented. For the platform, two views were created; (i) the layered view and (ii) the functional view. The system's functional structure, including the key functional elements, their responsibilities, the interfaces they expose, and the interactions between them have been documented. From the description of the components, we created the operation flows for the core procedures (user authorisation and creation) as well as the PANTHEON usage scenarios. Finally, we created a preliminary version of the integration, deployment and verification plans to be further elaborated in future WPs.

Impact	Planned dissemination and exploitation
High-level reference architecture, its components and their interactions have been defined in detail. Preliminary versions of operational flows, integration, deployment, and verification plans have been created. These components set the groundwork for the technical development of the PANTHEON platform in WP4, WP5, WP6 and WP7.	Public.

1. INTRODUCTION

This deliverable builds upon the work done in D2.4, D3.1, D3.2, D3.3, D3.4 and D3.6, by further specifying:

- the necessary system requirements to fulfil the user requirements described in D3.2 and D3.6,
- the technical components that participate in the architecture along with the sequence diagrams of their interconnection after analysis on:
    - the existing technological landscape (D2.4 & D3.1),
    - the conceptual model of the SCDT (D3.3),
    - the data delivery schemes in D3.4 and
    - the usage scenarios and use cases as documented in D3.6.

The main objective of this deliverable is to derive specific system requirements that the PANTHEON system should meet along with the internal structure of the PANTHEON platform up to component level, along with the component interconnections and dependencies. The deliverable presents preliminary work regarding the integration and deployment infrastructure to be employed, as well as its verification process. The defined architecture and technical specifications will be used as a basis for the implementation of the PANTHEON platform in WP4, WP5 and WP6. Thus, this deliverable connects the previous work done in WP2 and WP3 with the implementation phase, and therefore facilitates the transition from the design to the implementation phase.

1.1 DELIVERABLE STRUCTURE

- In Chapter 2 we describe the system requirements defined by the technical partners of the consortium, given the definition of the usage scenarios, use cases and user requirements in previous deliverables.
- In Chapter 3 we introduce the platform architecture overview, providing two views; (i) the layered view which helps encapsulate and decouple the PANTHEON platform by grouping the distinct segments using their functional role and (ii) the functional view which defines the architectural elements that deliver the functions of the system being described and documents the system's functional structure, including the key functional elements, their responsibilities, the interfaces they expose, and the interactions between them.
- In Chapter 4 we further elaborate on the platform components, describing in each case the expected inputs and outputs, the algorithms they will work with, the infrastructure they need to run and give an internal architecture schematic, if relevant.
- In Chapter 5 we describe the operation flows, how the system components will interact with each other to cover the usage scenarios defined.
- In Chapter 6 we provide several guidelines towards platform integration planning and its application.
- In Chapter 7 we provide several preliminary considerations for the deployment of the PANTHEON platform.
- In Chapter 8 we provide a preliminary platform verification plan, including objectives of the verification process, planification of the verification and relevant envisioned activities.
- In Chapter 9 we conclude with the work done and provide a preliminary insight on how the results from this deliverable will be useful for the future tasks and the overall project.

2. ADDITIONAL USER REQUIREMENTS AND SYSTEM REQUIREMENTS

This section starts with several additional user requirements that have been created after additional input provided to ENAC and then proceeds with the system requirements of the PANTHEON platform, based on the user requirements of D3.6 and section 2.1.

2.1 ADDITIONAL USER REQUIREMENTS

While most of the user requirements have been outlined in D3.6, ENAC in collaboration with JOAFG, HPOL and other first responder organisations elicited some additional design recommendations after a series of interviews. These requirements are outlined in Table 1.

Table 1: Catalogue of additional design recommendations for PANTHEON as elicited by ENAC.

ID	Title	Description
Application A: Planning and early warning according to simulations		
UR-A-07	Live extent of disaster and its damage	The PANTHEON system can visualise the live extent of the disaster (e.g. the border lines of fires real-time updated on map or the live extent of the damage caused by an earthquake)
UR-A-08	Foresight of the disaster	The PANTHEON system can provide foresight of the disaster (e.g. direction, speed of the wildfire or ranking of damaged areas in earthquake)
Application B: Training and exercises		
UR-B-14	Live extent of disaster and its damage	The PANTHEON system can visualise the live extent of the disaster (e.g. the border lines of fires real-time updated on map or the live extent of the damage caused by an earthquake)
UR-B-15	Friend or foe	The PANTHEON system can visualise friend and enemy detection live on maps. Friend=waterbombing aircraft, firefighting troops, drones
UR-B-16	Identification of population	The PANTHEON system can provide identification of population in crisis
UR-B-17	Identification of hazardous substances	The PANTHEON system can provide identification of hazardous substances (based on signs as well as due to measurement tools)
UR-B-18	Identification of changes to buildings	The PANTHEON system can provide identification of changes to the building structure
UR-B-19	Roads' situation	The PANTHEON system can provide identification of roadblocks; Identification of free routes

Prior to designing PANTHEON's system architecture, it is essential to identify in detail the functional and non-functional system requirements. Functional requirements define the basic system behavior. They are the features focusing on the user requirements as outlined in the previous section and D3.6, that allow the system to function as it was intended. Put another way, if the functional requirements are not met, the system will not work (although the importance field as described below highlights the difference between

must-have requirements and nice-to-have functional requirements). Non-functional requirements specify the proper way for the system to achieve its intended behavior. They are features and general characteristics that affect and optimize the user experience and system performance. Even if they are not met, the system will still perform its basic purpose. The main architecture that is described based on the User Requirements as set in D3.6, can be split into two main technical categories, i.e. Frontend – FRND and Backend - BKND. Thus, the functional requirements are broken down into these two main categories, with the name presenting the following scheme: FR-<TYPE>-<ID>, with ID being an incremental number starting from 1. The functional requirements have an additional importance field, which is described in Table 2.

Table 2 The importance field of the functional requirements.

Importance	Description
Mandatory	A requirement that is defined as necessary for the completion of the project, as described either through the definition of the work or from needs that arise from other requirements.
Optional	A requirement that can provide an improvement to the project (for example something that improves the user experience) but does not affect the successful use of the product. This can be not defined within the DOW and might either be completed before or after the end of the project.

2.2 FUNCTIONAL REQUIREMENTS

2.2.1 FRONTEND FUNCTIONAL REQUIREMENTS

For the Frontend the following functional requirements are identified:

ID	FR-FRND-01
Title	Allow PANTHEON users not registered to request registration
Status	Active
Requirement	The PANTHEON User AA (Authentication and Authorization) system should provide the possibility for guests to request registration in the system
Rationale	The access to each module is allowed for specific users having the right roles
Acceptance criteria	Users able to request registration
Reference	Based on minimum user requirements for accessing the PANTHEON platform
Importance	Mandatory

ID	FR-FRND-02
Title	Allow PANTHEON users to log in to modules
Status	Active
Requirement	The PANTHEON User AA (Authentication and Authorization) system should provide the possibility for users to log in a specific module
Rationale	User should have the possibility to log in to the necessary module
Acceptance criteria	Users logged in
Reference	Based on minimum user requirements for accessing the PANTHEON platform
Importance	Mandatory

ID	FR-FRND-03
Title	PANTHEON User Authentication and Authorization
Status	Active
Requirement	The PANTHEON platform should call AA mechanism for each user
Rationale	User access is permitted for registered users based on roles
Acceptance criteria	User login before start the application
Reference	Based on minimum user requirements for accessing the PANTHEON platform
Importance	Mandatory

ID	FR-FRND-04
Title	Allow PANTHEON users to log out
Status	Active
Requirement	The PANTHEON User AA (Authentication and Authorization) system should provide the possibility for users to log in from a specific module
Rationale	User should have the possibility to log out from the module
Acceptance criteria	Users logged out
Reference	Based on minimum user requirements for accessing the PANTHEON platform
Importance	Mandatory

ID	FR-FRND-05
Title	Allow PANTHEON users to request reset password
Status	Active
Requirement	The PANTHEON User AA (Authentication and Authorization) system should provide the possibility for users to request a password reset
Rationale	Users should have the possibility to request a password reset
Acceptance criteria	PANTHEON Users able to request password reset
Reference	Based on minimum user requirements for using the PANTHEON platform
Importance	Mandatory

ID	FR-FRND-06
Title	Define roles at the project level
Status	Active
Requirement	The PANTHEON User AA (Authentication and Authorization) system should provide the possibility to create roles at the project (REALM) level
Rationale	There are necessary roles at the project level
Acceptance criteria	Administrator, client role, guest, manager role created
Reference	**UR-GEN-05, UR-B-03**
Importance	Mandatory

ID	FR-FRND-07
Title	Visualisation of 2D maps and stakeholders' assets
Status	Active
Requirement	The PANTHEON UI should present to the user a graphical representation in 2D format depicting the designated scenarios along with relevant stakeholders' assets.

Rationale	Users should be able to access a graphical user interface for the defined scenarios in D3.6
Acceptance criteria	A graphical representation in 2D format is depicted for every PANTHEON scenario.
Reference	**UR-GEN-01, UR-GEN-02, UR-GEN-04, UR-GEN-05, UR-GEN-07, UR-GEN-10, UR-A-04, UR-A-05, UR-A-06, UR-B-01, UR-B-04, UR-B-08, UR-B-11, UR-B-12, UR-B-13**
Importance	Mandatory

ID	**FR-FRND-08**
Title	Live visualisation of disaster
Status	Active
Requirement	The PANTHEON system should help live visualisation of disaster
Rationale	The PANTHEON system should aid the users in understanding the designated scenarios
Acceptance criteria	Disaster and borderlines are updated in real-time on maps the user can access
Reference	**UR-A-07, UR-A-08, UR-B-14**
Importance	Optional

ID	**FR-FRND-09**
Title	Live visualisation of assets damage
Status	Active
Requirement	The PANTHEON system should help live visualisation of the extent of the assets damage
Rationale	The PANTHEON system should aid the users in understanding the designated scenarios
Acceptance criteria	A visual, geolocated picture of the damaged zone is updated real-time on interfaces the user can access
Reference	**UR-A-07, UR-A-08, UR-B-14**
Importance	Optional

ID	**FR-FRND-10**
Title	Live identification of friends and foes
Status	Active
Requirement	The PANTHEON UI should have live identification of friends and foes
Rationale	The PANTHEON system should aid the users in understanding the designated scenarios
Acceptance criteria	Friends and foes are visible and updated real-time on interfaces the user can access
Reference	UR-B-15
Importance	Optional

ID	**FR-FRND-11**
Title	Identification of population in crisis
Status	Active
Requirement	The PANTHEON UI should offer identification of population in crisis

Rationale	The PANTHEON system should aid the users in understanding the designated scenarios
Acceptance criteria	The PANTHEON system can provide identification of population in crisis
Reference	UR-B-16
Importance	Optional

ID	**FR-FRND-12**
Title	Identification of hazardous substances
Status	Active
Requirement	The PANTHEON system should help Identification of hazardous substances (based on signs as well as due to measurement tools)
Rationale	The PANTHEON system should aid the users in understanding the designated scenarios
Acceptance criteria	hazardous substances (based on signs as well as due to measurement tools);are visible and updated real-time on interfaces the user can access
Reference	UR-B-17
Importance	Optional

ID	**FR-FRND-13**
Title	Live Identification of changes to the building structure
Status	Active
Requirement	The PANTHEON system should help live Identification of changes to the building structure
Rationale	The PANTHEON system should aid the users in understanding the designated scenarios
Acceptance criteria	Changes to the building structure can be visualised real-time on interfaces the user can access
Reference	UR-B-18
Importance	Optional

ID	**FR-FRND-14**
Title	Live visualisation of road blocks and free routes
Status	Active
Requirement	The PANTHEON system should help live visualisation of roadblocks and free routes
Rationale	The PANTHEON system should aid the users in understanding the designated scenarios
Acceptance criteria	Roadblocks and free routes can be visualised real time on interfaces the user can access
Reference	UR-B-19
Importance	Optional

ID	**FR-FRND-15**
Title	Visualization Tools
Status	Active
Requirement	The PANTHEON system architecture should include visualization tools for presenting data/what-if analysis results to enhance user understanding.

Rationale	Based on the optimal requirements to operate the service.
Acceptance criteria	The presence of the above-mentioned ability.
Reference	**UR-GEN-04, UR-GEN-06, UR-GEN-09, UR-GEN-10, UR-A-03, UR-B-02, UR-B-05, UR-B-07,**
Importance	Mandatory

ID	**FR-FRND-16**
Title	SCDT User Engagement
Status	**Active**
Requirement	The PANTHEON UI should maintain a high level of user engagement for the SCDT platform, by providing interactive and user-friendly interfaces for simulation customization and data visualization.
Rationale	High user engagement ensures that the platform is effectively used by city planners, emergency responders, and other stakeholders for disaster preparedness and response.
Acceptance Criteria	User satisfaction scores should average 4 out of 5, with at least four different and discrete executions by each user.
Reference	**UR-GEN-07, UR-GEN-09, UR-GEN-10, UR-B-04, UR-B-13**
Importance	Optional

ID	**FR-FRND-17**
Title	Parameter Configuration and Variation
Status	Active
Requirement	The PANTHEON UI should enable their users to configure and adjust input parameters, allow easy exploration of different scenarios, and systematically or randomly vary input parameters within defined scenarios to simulate different conditions.
Rationale	Based on the minimum requirements to operate the services.
Acceptance criteria	The presence of the above-mentioned ability.
Reference	**UR-GEN-04, UR-GEN-07, UR-GEN-10, UR-B-04, UR-B-13**
Importance	Mandatory

2.2.2 BACKEND FUNCTIONAL REQUIREMENTS

For the Backend (including the SCDT), the following functional requirements have been identified:

ID	**FR-BKND-01**
Title	Aggregation of appropriate data types and formats
Status	Active
Requirement	The PANTHEON platform should be able to gather/aggregate all the required data from its designated input sources, including data from satellites, IoT weather stations, infrastructure, traffic, UAV, and community sources, in the appropriate types and formats.
Rationale	Based on the minimum requirements to operate the service and the PANTHEON platform.
Acceptance criteria	Proper aggregation and pre-processing of all the required data.

Reference	**UR-GEN-02, UR-GEN-03, UR-GEN-04, UR-GEN-08, UR-A-05, UR-B-01, UR-B-04, UR-B-09, UR-B-10, UR-B-11**
Importance	Mandatory

ID	**FR-BKND-02**
Title	Curated Data Availability, Input, and Integration
Status	Active
Requirement	The backend services of PANTHEON should be able to accept and integrate data from its designated input sources, including aggregated, pre-processed, and curated data originating from PANTHEON inputs such as satellite, IoT weather stations, infrastructure, traffic, UAV, and community sources.
Rationale	Ensuring curated data is readily available facilitates accurate simulation and analysis of disaster impacts, aiding in predictive analytics and decision support.
Acceptance criteria	Data sources are accessible with a minimum uptime of 99.5%, and data freshness is maintained, with updates processed within 10 minutes of receipt.
Reference	**UR-GEN-02, UR-GEN-04, UR-GEN-08, UR-A-05, UR-B-01, UR-B-04, UR-B-09, UR-B-10, UR-B-11**
Importance	Mandatory

ID	**FR-BKND-03**
Title	Data Analysis Execution, Data Export/Integration
Status	Active
Requirement	The PANTHEON platform should execute data analysis based on input data and generate relevant output data for the SCDT.
Rationale	Based on the minimum requirements to operate both services.
Acceptance criteria	The presence of the above-mentioned ability, along with the ability to identify trends, differences, and optimal strategies, as well as how changes in input parameters impact the outcomes of simulation models.
Reference	**UR-GEN-02, UR-GEN-04, UR-GEN-08, UR-A-05, UR-B-01, UR-B-04, UR-B-09, UR-B-10, UR-B-11**
Importance	Mandatory

ID	**FR-BKND-04**
Title	What-If Analysis Execution, Data Export/Integration, Risk Analysis
Status	Active
Requirement	The PANTHEON platform should execute what-if analysis based on input data and generate relevant Decision Recommendations. It should also enable defining What-If scenarios and specifying changes to input parameters or conditions for each scenario.
Rationale	Based on the minimum requirements to operate both services.
Acceptance criteria	The presence of the above-mentioned ability, along with how changes in input parameters impact the outcomes of decision models, as well as support the assessment of risks associated with different scenarios, considering uncertainties and variations in input parameters.
Reference	**UR-GEN-02, UR-GEN-04, UR-GEN-08, UR-A-05, UR-B-01, UR-B-04, UR-B-06, UR-B-09, UR-B-10, UR-B-11**

Importance	Mandatory

ID	**FR-BKND-05**
Title	Decision support for CBDRM
Status	Active
Requirement	The PANTHEON platform should offer decision support for CBDRM operators, providing the proper suggestions and recommendations to the appropriate operators/stakeholders. These recommendations should be also displayed in the UI.
Rationale	
Acceptance criteria	Tangible recommendations according to the input data and the disaster scenarios
Reference	**UR-GEN-04, UR-GEN-06, UR-A-03, UR-B-05, UR-B-07**
Importance	Mandatory

ID	**FR-BKND-06**
Title	SCDT Spatial Resolution (Transient)
Status	**Active**
Requirement	The SCDT should ensure a spatial numerical resolution of 100 km to accurately simulate transportation networks under transient conditions.
Rationale	Achieving a fine-grained spatial resolution is critical for detailed analysis and accurate forecasting of transportation network behaviour in response to varying conditions.
Acceptance Criteria	Spatial resolution for transient simulations should not exceed 100 km under any operational scenario.
Reference	**UR-GEN-02, UR-GEN-07, UR-GEN-10, UR-B-01, UR-B-04, UR-B-13**
Importance	Optional

ID	**FR-BKND-07**
Title	SCDT Computing Time (Transient)
Status	**Active**
Requirement	PANTHEON SCDT computing time per physical time should be less than 1 to enable real-time simulation capabilities.
Rationale	Real-time computing capability is essential for dynamic response and decision-making in critical situations affecting the transportation network.
Acceptance Criteria	System demonstrates computing times less than the physical time, ensuring real-time processing and simulation.
Reference	**UR-GEN-02, UR-GEN-07, UR-GEN-10, UR-B-01, UR-B-04, UR-B-13**
Importance	Optional

ID	**FR-BKND-08**
Title	SCDT Domain Size (Transient)
Status	**Active**

Requirement	The PANTHEON SCDT domain should encompass a total road length of 100 km to cover the extensive transportation network infrastructure.
Rationale	**A comprehensive domain size is necessary to model the entire transportation network, facilitating accurate predictions and analyses across the network.**
Acceptance Criteria	System accurately simulates and analyzes the transportation network within a domain size of 100 km.
Reference	**UR-GEN-02, UR-GEN-07, UR-GEN-10, UR-B-01, UR-B-04, UR-B-13**
Importance	Optional

ID	**FR-BKND-09**
Title	SCDT Precision
Status	**Active**
Requirement	The PANTHEON SCDT must achieve high precision in its predictive analytics, accurately forecasting the impacts of potential disaster scenarios on the city's infrastructure.
Rationale	Precision in predictive analytics is crucial for effective disaster preparedness and mitigation strategies, reducing potential damages.
Acceptance Criteria	Predictive forecasts should have a precision rate of 85% or higher, as verified by post-event assessments.
Reference	**UR-GEN-02, UR-GEN-07, UR-GEN-10, UR-B-01, UR-B-04, UR-B-13**
Importance	Optional

ID	**FR-BKND-10**
Title	SCDT Accuracy
Status	**Active**
Requirement	Simulations conducted by the PANTHEON SCDT must accurately reflect real-world conditions and disaster scenarios to provide valid predictive analytics.
Rationale	**The effectiveness of the SCDT in planning and response hinges on the accuracy of its simulations, impacting the reliability of its predictions and recommendations.**
Acceptance Criteria	Simulated outcomes must align with historical data and expert assessments with an accuracy rate of 90% or higher.
Reference	**UR-GEN-02, UR-GEN-07, UR-GEN-10, UR-A-01, UR-B-01, UR-B-04, UR-B-13**
Importance	Mandatory

ID	**FR-BKND-11**
Title	UAVs Availability
Status	Active
Requirement	The number of available UAVs should be greater than a defined number (approx 3 ~10), fully functional regarding electromechanical and software components.
Rationale	Based on the minimum requirement to operate this service.
Acceptance criteria	Proper condition of available UAVs and the presence/format of input data.

Reference	**UR-GEN-07, UR-B-04**
Importance	Mandatory

ID	**FR-BKND-12**
Title	Conceptual Models
Status	Active
Requirement	The PANTHEON platform should offer a high-level overview of how its components function in the defined usage and disaster scenarios
Rationale	The Conceptual Model is ideal for showing the high-level view of a tool. In PANTHEON it will help during the development process and when finished will help end-users understanding and training.
Acceptance criteria	Charts have been developed describing the PANTHEON Conceptual Model
Reference	**UR-GEN-10**
Importance	Mandatory

ID	**FR-BKND-13**
Title	Message Broker
Status	Active
Requirement	The PANTHEON platform should have a message broker for asynchronous data exchange
Rationale	The PANTHEON platform should have a publish/subscribe mechanism for asynchronous data exchange
Acceptance criteria	A message placed on a specific topic
Reference	**UR-GEN-02, UR-GEN-04, UR-GEN-08, UR-A-05, UR-B-01, UR-B-04, UR-B-09, UR-B-10, UR-B-11**
Importance	Mandatory

2.3 NON-FUNTIONAL REQUIREMENTS

The main architecture that is described based on the User Requirements as set in D3.6, can be split into two main technical categories, i.e. Frontend – FRND and Backend - BKND. Additionally, there exist general non-functional requirements that apply to both the frontend and backend of the PANTHEON platform – GEN. Thus, the non-functional requirements are broken down into these three main categories, with the name presenting the following scheme: NFR-<TYPE>-<ID>, with ID being an incremental number starting from 1. Additionally, Non-functional Requirements may have different functional scope and are classified into:

- TECH - Impacts technology
- USBY - Impacts system usability
- INFO - Relates to information that has to be relayed.

2.3.1 GENERAL NON-FUNCTIONAL REQUIREMENTS

First, some general non-functional requirements for the PANTHEON platform have been identified, which apply both to the frontend and the backend:

ID	**NFR-GEN-01**
Title	Platform independence
Status	Active

Requirement	The PANTHEON platform should run on platforms like Linux, Windows or MacOS
Rationale	Deployment anywhere
Acceptance criteria	Deployment on Linux, Windows, MacOS
Reference	FREQ_S03-01
REQ Type	TECH

ID	**NFR-GEN-02**
Title	Horizontal scalability
Status	Active
Requirement	The PANTHEON platform should allow horizontal scalability by deployment in distributed environments
Rationale	Deployment anywhere
Acceptance criteria	Deployment on Linux, Windows, MacOS, in containers
Reference	FREQ_S03-01
REQ Type	TECH

ID	**NFR-GEN-03**
Title	Synchronous communication
Status	Active
Requirement	The PANTHEON platform should accommodate synchronous data exchange via REST APIs
Rationale	Asynchronous data exchange
Acceptance criteria	Call REST APIs
Reference	FREQ_S03-01
REQ Type	TECH

ID	**NFR-GEN-04**
Title	Instrumentation
Status	Active
Requirement	The PANTHEON platform should accommodate instrumentation, by storing logs on the system functionality
Rationale	Debug the system at runtime
Acceptance criteria	Log system
Reference	FREQ_S03-01
REQ Type	TECH

2.3.2 FRONTEND NON-FUNCTIONAL REQUIREMENTS

The following non-functional requirements have been identified for the front-end:

ID	**NFR-FRND-01**
Title	Define project scope (REALM)
Status	Active
Requirement	The PANTHEON User AA (Authentication and Authorization) system should provide the possibility to define the project scope (REALM)

Rationale	All the authorization and authentications should be placed in the same context
Acceptance criteria	Realm created
Reference	FR-FRND-02, FR-FRND-03
REQ Type	USBY

ID	**NFR-FRND-02**
Title	Define clients for each component
Status	Active
Requirement	The PANTHEON User AA (Authentication and Authorization) system should provide the possibility to define the subdomain for each module (client)
Rationale	All the authorization and authentications for a module should be placed in the same context
Acceptance criteria	Client created
Reference	FR-FRND-02, FR-FRND-03
REQ Type	USBY

ID	**NFR-FRND-03**
Title	Define roles at the module level
Status	Active
Requirement	The PANTHEON User AA (Authentication and Authorization) system should provide the possibility to create roles at the module (Client) level
Rationale	There are necessary roles at the module level
Acceptance criteria	Roles specific for each module created
Reference	FR-FRND-02, FR-FRND-03, FR-FRND-06
REQ Type	USBY

ID	**NFR-FRND-04**
Title	Define the possibility of using or not using second-factor authentication
Status	Active
Requirement	The PANTHEON User AA (Authentication and Authorization) system should provide the possibility to enable the usage of second-factor authentication and to select the way of usage (email, SMS, other)
Rationale	Second-factor authentication is required in some pilots
Acceptance criteria	Enable/Disable second-factor authentication and selection of second-factor usage (email, SMS, etc)
Reference	FR-FRND-02, FR-FRND-03
REQ Type	USBY

ID	**NFR-FRND-05**
Title	Add users to the client
Status	Active
Requirement	The PANTHEON User AA (Authentication and Authorization) system should provide the possibility to add users for each module (client)

Rationale	Access to each module is allowed for specific users
Acceptance criteria	Users created for each module
Reference	FR-FRND-02, FR-FRND-03
REQ Type	USBY

ID	**NFR-FRND-06**
Title	Associate roles to users
Status	Active
Requirement	The PANTHEON User AA (Authentication and Authorization) system should provide the possibility to associate one or more roles to each user
Rationale	The access to each module is allowed for specific users having the right roles
Acceptance criteria	Users created for each module and having roles associated
Reference	FR-FRND-02, FR-FRND-03
REQ Type	USBY

ID	**NFR-FRND-07**
Title	PANTHEON User AA Unused sessions automatically closed after a while
Status	Active
Requirement	The PANTHEON User AA (Authentication and Authorization) system should provide the possibility to automatically close a session if it has not been used for some time
Rationale	Unused sessions should be closed to avoid security issues
Acceptance criteria	The session closed after the setup period
Reference	FR-FRND-04
REQ Type	TECH

ID	**NFR-FRND-08**
Title	PANTHEON User AA SSO (Single Signed On)
Status	Active
Requirement	The PANTHEON User AA (Authentication and Authorization) system should provide the possibility to bypass the login in a module if the user is already logged in in another module
Rationale	Reduce the time consumed with unnecessary logs in
Acceptance criteria	The user automatically directs to module functionality
Reference	FR-FRND-02
REQ Type	USBY

ID	**NFR-FRND-09**
Title	PANTHEON User AA Audit
Status	Active
Requirement	The PANTHEON User AA (Authentication and Authorization) system should provide the possibility to audit user log-in and logout (including efforts without success)

Rationale	Have a history of access to the system
Acceptance criteria	History of connections available
Reference	FR-FRND-02, FR-FRND-03, FR-FRND-04
REQ Type	USBY

ID	**NFR-FRND-10**
Title	PANTHEON User AA access to API using bearer token
Status	Active
Requirement	The PANTHEON User AA (Authentication and Authorization) system should provide the possibility to allow client APIs to access the system only using bearer token authorization
Rationale	Have a secured API
Acceptance criteria	Connection to API guided by access TOKEN (JWT)
Reference	FR-FRND-02
REQ Type	TECH

ID	**NFR-FRND-11**
Title	PANTHEON User AA Import/Export capabilities
Status	Active
Requirement	The PANTHEON User AA (Authentication and Authorization) system should provide the possibility to export data with all the settings in JSON format, and import from JSON format
Rationale	Backup and recovery and the possibility to replicate in different deployments
Acceptance criteria	Export in JSON. Import a JSON.
Reference	FR-FRND-03
REQ Type	INFO

ID	**NFR-FRND-12**
Title	Administrator possibility to close a session
Status	Active
Requirement	The PANTHEON User AA (Authentication and Authorization) system should provide the possibility for an administrator to select an active session and close it.
Rationale	Orphan sessions close and suspicious sessions close.
Acceptance criteria	The session closed by the administrator
Reference	FR-FRND-02, FR-FRND-03, FR-FRND-04
REQ Type	USBY

ID	**NFR-FRND-13**
Title	Dashboard page
Status	Active
Requirement	There should be a dashboard page where the summary or current session statistics are displayed
Rationale	Inform users about the status of the system

Acceptance criteria	The presence of a dashboard page
Reference	FR-FRND-02, FR-FRND-03
REQ Type	USBY

ID	**NFR-FRND-14**
Title	Administration page
Status	Active
Requirement	There should be an administration page where the user can configure all the referred modules
Rationale	Administer the platform
Acceptance criteria	The presence of an administration page
Reference	FR-FRND-02, FR-FRND-03
REQ Type	USBY

ID	**NFR-FRND-15**
Title	Notifications page
Status	Active
Requirement	There should be a notification page where the user can see notifications from all the referred modules
Rationale	Have the notifications
Acceptance criteria	The presence of a notification page
Reference	FR-FRND-02, FR-FRND-03
REQ Type	USBY

ID	**NFR-FRND-16**
Title	Monitoring page
Status	Active
Requirement	There should be a monitoring page where the user can see pieces of information from all the referred modules
Rationale	Have the notifications
Acceptance criteria	The presence of a monitoring page
Reference	FR-FRND-02, FR-FRND-03
REQ Type	USBY

ID	**NFR-FRND-17**
Title	Global menu
Status	Active
Requirement	There should be a menu displayed all the time, indifferent of the active page
Rationale	Offer the possibility to command the system
Acceptance criteria	The presence of a monitoring page
Reference	FR-FRND-02, FR-FRND-03
REQ Type	USBY

ID	NFR-FRND-18
Title	Compatibility with Oauth2
Status	Active
Requirement	The PANTHEON User AA (Authentication and Authorization) system should implement features compatible with Oauth2 specifications
Rationale	Compatibility with other systems
Acceptance criteria	TBD
Reference	FR-FRND-02, FR-FRND-03
REQ Type	TECH

ID	NFR-FRND-19
Title	Implementation in different languages
Status	Active
Requirement	The User AA (Authentication and Authorization) system should be accessible in EU languages
Rationale	Ease of use
Acceptance criteria	Check different languages
Reference	FR-FRND-01, FR-FRND-02, FR-FRND-03
REQ Type	USBY

ID	NFR-FRND-20
Title	Internationalization
Status	Active
Requirement	The PANTHEON platform should accommodate internationalization, by allowing the usage of different languages in UI, added at runtime
Rationale	User interface in languages
Acceptance criteria	Internationalization
Reference	FR-FRND-02, FR-FRND-03
REQ Type	USBY

ID	NFR-FRND-21
Title	Include UI from modules
Status	Active
Requirement	The PANTHEON platform should accommodate the web UI from all modules
Rationale	User interface displayed for all modules
Acceptance criteria	Placement of different iframes in a central platform
Reference	FR-FRND-03
REQ Type	USBY

ID	NFR-FRND-22
Title	Different UI for different roles
Status	Active
Requirement	The PANTHEON platform should display different pages and buttons depending on the roles of the logged user

Rationale	The user interface displayed depending on roles
Acceptance criteria	Placement of different iframes in a central platform
Reference	FR-FRND-03
REQ Type	USBY

ID	**NFR-FRND-23**
Title	Web application
Status	Active
Requirement	The PANTHEON platform should be available in a web browser on a desktop, laptop, or mobile application
Rationale	User access from anywhere
Acceptance criteria	Access from most used browsers (Chrome, Firefox, Edge, Safari)
Reference	FR-FRND-01, FR-FRND-02, FR-FRND-03, FR-FRND-06
REQ Type	TECH

ID	**NFR-FRND-24**
Title	UI Performance
Status	Active
Requirement	The PANTHEON platform UI should display pages in less than 2 seconds
Rationale	The PANTHEON platform should have a UI of high performance.
Acceptance criteria	2 seconds Content Download
Reference	FR-FRND-01, FR-FRND-02, FR-FRND-03
REQ Type	TECH

ID	**NFR-FRND-25**
Title	What-if-analysis and decision support usability
Status	Active
Requirement	The what-if-analysis and decision support of the PANTHEON platform should have an intuitive user interface.
Rationale	Based on the minimum requirements to operate the service.
Acceptance criteria	Stakeholder satisfaction surveys, average time to complete tasks, and stakeholder training requirements.
Reference	FR-FRND-15
REQ Type	USBY

2.3.3 BACKEND NON-FUNCTIONAL REQUIREMENTS

Similarly for the Backend, the following non-functional requirements have been identified:

ID	**NFR-BKND-01**
Title	Data availability & structure
Status	Active
Requirement	The PANTHEON platform should be able to receive Satellite, UAV Traffic, Community and IoT-based weather data in a structured format, to utilise them in the simulations.

Rationale	
Acceptance criteria	Successful integration of at least three data sources into the PANTHEON platform.
Reference	FR-BKND-01
REQ Type	TECH

ID	**NFR-BKND-02**
Title	Data aggregation and pre-processing
Status	Active
Requirement	The data pre-processing and curation service should be able to aggregate all the required data in terms of volume, type and format, by means of an API, offered by each data provider.
Rationale	
Acceptance criteria	Successful communication between the pre-processing and curation service and the data providers/sources.
Reference	FR-BKND-02, FR-BKND-03
REQ Type	TECH

ID	**NFR-BKND-03**
Title	SCDT Responsiveness
Status	**Active**
Requirement	The SCDT platform should process inputs and deliver simulation results within a defined time frame to ensure timely decision support.
Rationale	Quick response times are critical in disaster management scenarios, where delays can exacerbate the situation.
Acceptance Criteria	The system processes requests and delivers results within 240 seconds for 95% of transactions.
Reference	FR-BKND-07
REQ Type	TECH

ID	**NFR-BKND-04**
Title	SCDT Scalability
Status	**Active**
Requirement	The PANTHEON SCDT must efficiently handle increasing amounts of data and user requests without degradation in performance.
Rationale	As the smart city infrastructure and data volume grow, the system must scale to maintain high levels of service and performance.
Acceptance Criteria	The system demonstrates linear performance degradation at most, with a network nodes and data increase of up to 100%.
Reference	FR-BKND-07
REQ Type	TECH

ID	**NFR-BKND-05**
Title	SCDT Data Processing Efficiency

Status	**Active**
Requirement	The PANTHEON SCDT must efficiently process large volumes of data to update simulations and analytics in near-real-time.
Rationale	Efficiency in data processing ensures that the system can keep up with the continuous influx of data from various sources, providing timely insights.
Acceptance Criteria	The system updates simulations and analytics within 5 minutes of receiving new data, for 95% of updates.
Reference	FR-BKND-06, FR-BKND-07, FR-BKND-08, FR-BKND-09, FR-BKND-10
REQ Type	TECH

ID	**NFR-BKND-06**
Title	SCDT Availability
Status	**Active**
Requirement	The PANTHEON SCDT must maintain high availability to ensure continuous access for users, especially during critical disaster response operations.
Rationale	High system availability is essential for supporting ongoing disaster management efforts and for ensuring that the platform is reliable in emergency situations.
Acceptance Criteria	The system APIs boast an uptime of 99.9%, with planned downtime communicated at least 48 hours in advance.
Reference	FR-BKND-06, FR-BKND-07, FR-BKND-08, FR-BKND-09, FR-BKND-10
REQ Type	TECH

5. ID	**NFR-BKND-07**
Title	SCDT Cybersecurity Standards Compliance
Status	**Active**
Requirement	The PANTHEON SCDT must adhere to industry-standard cybersecurity practices to protect sensitive data and infrastructure information.
Rationale	Ensuring the security of the SCDT and its data is paramount to maintaining trust and integrity, especially when dealing with critical infrastructure and personal data.
Acceptance Criteria	The SCDT must pass typical vulnerability assessment audits with no critical vulnerabilities found in software that is functional, adhering to standards such as ISO/IEC 27001.
Reference	FR-BKND-06, FR-BKND-07, FR-BKND-08, FR-BKND-09, FR-BKND-10
REQ Type	TECH

ID	**NFR-BKND-08**
Title	SCDT Input transportation Grid Topology
Status	**Active**
Requirement	The PANTHEON SCDT must accurately capture and utilize transportation grid topology for simulations and analysis.
Rationale	Precise transportation grid topology input is foundational for accurate modelling and simulation of transportation network dynamics.
Acceptance Criteria	Transportation grid topology is accurately represented and utilized in all simulations, with updates processed within 24 hours of receipt.

Reference	FR-BKND-06, FR-BKND-07, FR-BKND-08, FR-BKND-09, FR-BKND-10
REQ Type	TECH

ID	**NFR-BKND-09**
Title	SCDT Input transportation Grid State
Status	**Active**
Requirement	The PANTHEON SCDT should effectively process and reflect the current state of the transportation grid in simulations.
Rationale	An accurate representation of the transportation grid's current state is crucial for reliable simulations and operational planning.
Acceptance Criteria	transportation grid state data is accurately integrated and reflected in simulations, with a data freshness guarantee of 99.5% uptime.
Reference	FR-BKND-06, FR-BKND-07, FR-BKND-08, FR-BKND-09, FR-BKND-10
REQ Type	TECH

ID	**NFR-BKND-10**
Title	Output Standards Compliance
Status	**Active**
Requirement	The PANTHEON SCDT must comply with industry standards for output formats to ensure interoperability and usability.
Rationale	Adherence to standard output formats facilitates data exchange, analysis, and decision-making across different systems and stakeholders.
Acceptance Criteria	All system outputs comply with industry-standard formats, ensuring >95% compatibility and interoperability with external API and cross-API communication.
Reference	FR-BKND-06, FR-BKND-07, FR-BKND-08, FR-BKND-09, FR-BKND-10
REQ Type	**INFO**

ID	**NFR-BKND-11**
Title	Publishing information on a specific topic
Status	Active
Requirement	The message broker should accept input messages sent to the specified topic,
Rationale	The Message broker should have a publish/subscribe mechanism for asynchronous data exchange
Acceptance criteria	A message placed on a specific topic
Reference	FR-BKND-13
REQ Type	TECH

ID	**NFR-BKND-12**
Title	Subscribing for information on the specific topic
Status	Active
Requirement	The message broker should accept input messages sent to the specified topic,

Rationale	The Message broker should have a publish/subscribe mechanism for asynchronous data exchange
Acceptance criteria	A read from a specific topic
Reference	FR-BKND-13
REQ Type	TECH

ID	**NFR-BKND-13**
Title	Reading the last message placed on a topic
Status	Active
Requirement	The message broker should offer the possibility to access the last message placed on a topic
Rationale	The Message broker should have a publish/subscribe mechanism for asynchronous data exchange
Acceptance criteria	The last message read from a specific topic
Reference	FR-BKND-13
REQ Type	TECH

ID	**NFR-BKND-14**
Title	Reading the last messages placed on a topic in a period
Status	Active
Requirement	The message broker should offer the possibility to access the last message placed on a topic in a specific period
Rationale	The Message broker should have a publish/subscribe mechanism for asynchronous data exchange
Acceptance criteria	The last message read from a specific topic
Reference	FR-BKND-13
REQ Type	TECH

ID	**NFR-BKND-15**
Title	Reading all messages available on a topic.
Status	Active
Requirement	The message broker should offer the possibility to access all messages placed on a topic
Rationale	The Message broker should have a publish/subscribe mechanism for asynchronous data exchange
Acceptance criteria	All messages read from a specific topic
Reference	FR-BKND-13
REQ Type	TECH

ID	**NFR-BKND-16**
Title	Purging all messages available on a topic.
Status	Active

Requirement	The message broker should offer the possibility to purge (delete) all messages placed on a topic
Rationale	The Message broker should have a publish/subscribe mechanism for asynchronous data exchange
Acceptance criteria	No messages present on a specific topic
Reference	FR-BKND-13
REQ Type	TECH

ID	**NFR-BKND-17**
Title	Message Broker Data Persistence
Status	Active
Requirement	The message broker should offer the possibility to persist data for a selected period in a topic
Rationale	Reread the data
Acceptance criteria	Reread the topic
Reference	FR-BKND-13
REQ Type	TECH

ID	**NFR-BKND-18**
Title	Distribution of data to groups
Status	Active
Requirement	The message broker should offer the possibility to access data based on groups
Rationale	Fine-tune the destination of messages
Acceptance criteria	Different access for different groups
Reference	FR-BKND-13
REQ Type	TECH

ID	**NFR-BKND-19**
Title	Message Broker HTTPS public access
Status	Active
Requirement	Access to the message broker should be at a public address, using HTTPS
Rationale	Secured access in a public place
Acceptance criteria	Access via HTTPS
Reference	FR-BKND-13
REQ Type	TECH

ID	**NFR-BKND-20**
Title	Message Format JSON
Status	Active

Requirement	The messages should be placed in JSON format. Binary data will be transformed using base64.
Rationale	Uniform data exchange
Acceptance criteria	JSON publish and subscribe
Reference	FR-BKND-13
REQ Type	INFO

ID	**NFR-BKND-21**
Title	Easy implementation for publishers and subscribers in Java and Python
Status	Active
Requirement	The publish and subscribe mechanism should be easily implemented in Java or Python (libraries already in place)
Rationale	Easy to use in all modules
Acceptance criteria	Libraries already present (open source)
Reference	FR-BKND-13
REQ Type	TECH

ID	**NFR-BKND-22**
Title	Publish and subscribe mechanism maturity
Status	Active
Requirement	The publish and subscribe mechanism should be matured with proven functionalities, availability, and reliability
Rationale	A proven system, with high reliability.
Acceptance criteria	The system is already used with a solid community of users.
Reference	FR-BKND-13
REQ Type	TECH

ID	**NFR-BKND-23**
Title	Services Compatibility and Interoperability
Status	Active
Requirement	The backend services of the PANTHEON architecture should ensure compatibility with their dependent services to enhance interoperability.
Rationale	Based on the minimum requirements to operate the service.
Acceptance criteria	Successful integration of the services, compatibility with various data sources, and adherence to industry standards.
Reference	FR-BKND-01, FR-BKND-02, FR-BKND-03, FR-BKND-04, FR-BKND-05, FR-BKND-06, FR-BKND-07, FR-BKND-08, FR-BKND-09, FR-BKND-10, FR-BKND-11, FR-BKND-12, FR-BKND-13
REQ Type	USBY

ID	**NFR-BKND-24**
Title	Services Security, Data Privacy, Auditability, Compliance

Status	Active
Requirement	The services of the PANTHEON architecture should implement robust security measures to protect sensitive simulation and analysis data, ensure that access is restricted to authorized users, ensure compliance with data privacy regulations, adopt measures to safeguard the privacy of individuals and organizations involved in simulations, implement logging and auditing mechanisms to track user actions and changes to simulation parameters, ensure traceability, ensure compliance with relevant industry standards, legal requirements, and data protection regulations.
Rationale	Based on the optimal requirements to operate the service, as well as the industry recommendations for operational IT services.
Acceptance criteria	Compliance with relevant security standards and regulations, incident response time, and data encryption effectiveness, adherence to data protection laws, the effectiveness of anonymization techniques, effectiveness of audit logs in tracking changes and identifying potential issues.
Reference	FR-BKND-01, FR-BKND-02, FR-BKND-03, FR-BKND-04, FR-BKND-05, FR-BKND-06, FR-BKND-07, FR-BKND-08, FR-BKND-09, FR-BKND-10, FR-BKND-11, FR-BKND-12, FR-BKND-13
REQ Type	INFO

ID	**NFR-BKND-25**
Title	Services Scalability and Performance
Status	Active
Requirement	The services of the PANTHEON architecture should be capable of handling a large volume of simulation and scenario data and conducting analyses efficiently. Additionally, the services should support scaling horizontally or vertically to accommodate growing datasets and increasing usage demand.
Rationale	Based on the optimal requirements to operate the service, as well as industry-standard recommendations.
Acceptance criteria	Response time for analysis and throughput under increasing data loads, ability to handle a specified increase in the number of simulations, usage, or data points.
Reference	FR-BKND-01, FR-BKND-02, FR-BKND-03, FR-BKND-04, FR-BKND-05, FR-BKND-06, FR-BKND-07, FR-BKND-08, FR-BKND-09, FR-BKND-10, FR-BKND-11, FR-BKND-12, FR-BKND-13
REQ Type	TECH

ID	**NFR-BKND-26**
Title	Services Documentation
Status	Active
Requirement	The services of the PANTHEON architecture should provide comprehensive documentation for users, including guides on how to use the services effectively, interpret results, and understand the limitations of the analyses.
Rationale	Based on the optimal requirements to operate the service, as well as industry-standard recommendations.
Acceptance criteria	The presence of documentation material, stakeholder satisfaction with supplied material and resources.

Reference	FR-BKND-01, FR-BKND-02, FR-BKND-03, FR-BKND-04, FR-BKND-05, FR-BKND-06, FR-BKND-07, FR-BKND-08, FR-BKND-09, FR-BKND-10, FR-BKND-11, FR-BKND-12, FR-BKND-13
REQ Type	INFO

ID	**NFR-BKND-27**
Title	Services Reliability
Status	Active
Requirement	The services of the PANTHEON architecture should be highly reliable, with minimal downtime and robust error handling to prevent data loss or corruption.
Rationale	Based on the optimal requirements to operate the service, as well as industry-standard recommendations.
Acceptance criteria	Mean Time Between Failures (MTBF), Mean Time to Recovery (MTTR), and overall system uptime.
Reference	FR-BKND-01, FR-BKND-02, FR-BKND-03, FR-BKND-04, FR-BKND-05, FR-BKND-06, FR-BKND-07, FR-BKND-08, FR-BKND-09, FR-BKND-10, FR-BKND-11, FR-BKND-12, FR-BKND-13
REQ Type	TECH

ID	**NFR-BKND-28**
Title	Services Maintainability
Status	Active
Requirement	The services of the PANTHEON architecture should be designed with maintainability in mind, with clear documentation, modular code, and easy debugging capabilities.
Rationale	Based on the optimal requirements to operate the service, as well as industry-standard recommendations.
Acceptance criteria	Mean Time to Repair (MTTR) for bugs or issues, frequency of updates or patches.
Reference	FR-BKND-01, FR-BKND-02, FR-BKND-03, FR-BKND-04, FR-BKND-05, FR-BKND-06, FR-BKND-07, FR-BKND-08, FR-BKND-09, FR-BKND-10, FR-BKND-11, FR-BKND-12, FR-BKND-13
REQ Type	TECH

ID	**NFR-BKND-29**
Title	Services Resilience
Status	Active
Requirement	The services of the PANTHEON architecture should be resilient to failures or disruptions, with mechanisms in place for data recovery and system continuity.
Rationale	Based on the optimal requirements to operate the service, as well as industry-standard recommendations.
Acceptance criteria	Recovery time after a failure, effectiveness of backup and restore procedures.

Reference	FR-BKND-01, FR-BKND-02, FR-BKND-03, FR-BKND-04, FR-BKND-05, FR-BKND-06, FR-BKND-07, FR-BKND-08, FR-BKND-09, FR-BKND-10, FR-BKND-11, FR-BKND-12, FR-BKND-13
REQ Type	TECH

ID	**NFR-BKND-30**
Title	UAV Swarm algorithms functionality
Status	Active
Requirement	The swarm algorithm must provide the best answer possible to the functional requirements, provided that the input data is correctly updated to the algorithm. this will then generate a range of requirements such as frequency of the command update, frequency of the feedback data update, robustness to internal and external factors, robustness of the C2link etc.
Rationale	
Acceptance criteria	TBD
Reference	FR-BKND-11
REQ Type	TECH

ID	**NFR-BKND-31**
Title	Conceptual Models availability
Status	Active
Requirement	The Conceptual Models of the PANTHEON platform should be adequate to the descriptions of the scenarios given in D3.6 and use the methodology described in D3.3
Rationale	The PANTHEON platform should offer the conceptual models of its
Acceptance criteria	Survey to developers
Reference	FR-BKND-12
REQ Type	INFO

ID	**NFR-BKND-32**
Title	Decision support for CBDRM
Status	Active
Requirement	The PANTHEON decision support should provide useful and effective recommendations, by utilising the what-if analysis.
Rationale	PANTHEON should offer decision support to the users.
Acceptance criteria	Survey to stakeholders
Reference	FR-BKND-05
REQ Type	TECH/USBY

3. PLATFORM ARCHITECTURE OVERVIEW

3.1 LAYERED VIEW

This part presents a new view of the architecture, this time organized in five layers of major functionalities, based on the logical architecture of the whole system, and considers the placement and functionalities of its components. The proposed layers in this architecture are described below alongside their dependency's different other tasks.

3.1.1 L1- DATA AND PERSISTENCE LAYER

This layer is responsible for data storage and maintenance. The data will contain:

- Models (Classical, AI generated, Simulation based) previously developed in T4.2 and T4.3
- IoT Data (time series and data streams). Time series are registered at seconds intervals. Data steams refers to video, audio streaming.)
- Metadata
- Data received from other systems (Community, Satellite, Traffic, Infrastructure)
- Simulated data

The technologies supporting this layer are:

- InfluxDB[3] for time series of data
- PostgreSQL[4] for simulated data, well-structured data
- MongoDB[5] for big and unstructured data
- Spark[6] for data used in large-scale data analytics.

The layer is subject to activities of T7.4.

3.1.2 L2-MODELS LAYER

This layer is responsible for model management. Simulation and ML models are considered to be part of this layer. Model storage is in L1, but their management is considered in this layer. Management of models refers to:

- Model creation or importing.
- Adjusting parameters
- Full search mechanism
- Process matching (identify the right model for a specific task)
- Model presentation (visual presentation)

The technologies supporting this layer include custom-made modules in Java or Python (depending on the models used and the associated libraries)

[3] https://www.influxdata.com/
[4] https://www.postgresql.org/
[5] https://www.mongodb.com/
[6] https://spark.apache.org/

3.1.3 L3-BACKEND SERVICES LAYER

This layer hosts:

- The backend part of all functional modules (developed in WP4)
- The synchronous communication mechanism based on REST APIs (developed in T7.2)
- The asynchronous communication mechanism based on message exchange (developed in T7.2)
- The choreography mechanisms are based on an acknowledgment mechanism developed in Java for this project (developed in T7.2)
- The service orchestrator (developed in T7.2 and T7.3)
- The simulation procedures (developed in T4.4)
- The AI Based Engine (Part of T5.5)
- The Data Analytics engine (developed in T4.2)
- The Decision Support System (DSS) (Developed in T5.2 and T5.5)

The technologies supporting this layer include custom-made modules in Java or Python (depending on the modules developed and libraries used).

3.1.4 L4-ANALYSIS AND OPTIMIZATIONS

This layer is placed on top of L3 and is also a backend layer but is specialized in analysis and optimization of data provided at the level of Layer 3. It hosts:

- DSS for CBDRM (developed in T5.5)
- What if analysis to measure the impact of data uncertainties on performance indicators (developed in task T5.5)
- What if analysis to analyse trade-off relationships of conflicting performance indicators (developed in task T5.5)
- What if analysis for comparing alternative solutions (developed in task T5.5)

The technologies supporting this layer include custom-made modules in Java or Python (depending on the modules developed and libraries used)

3.1.5 L5-FRONTEND

This layer is placed on top of the architecture and is responsible for presenting data to end users. The presentation can be in a user interface or exposed as APIs.

The main modules hosted in the front end refer to:

- Visual Data Analysis (developed in T4.5)
- Data Presentation (developed in T4.5)
- System monitoring and administration (developed in T7.3)
- REST APIs (developed in T7.3)
- Asynchronous message exchange (developed in T7.2)

The technologies supporting this layer include custom-made modules in Java or Python (depending on the modules developed and libraries used), user interface in Angular[7] or JSF, API in Java, and Presentation with Grafana[8].

The layered view of the system architecture is represented in Figure 1.

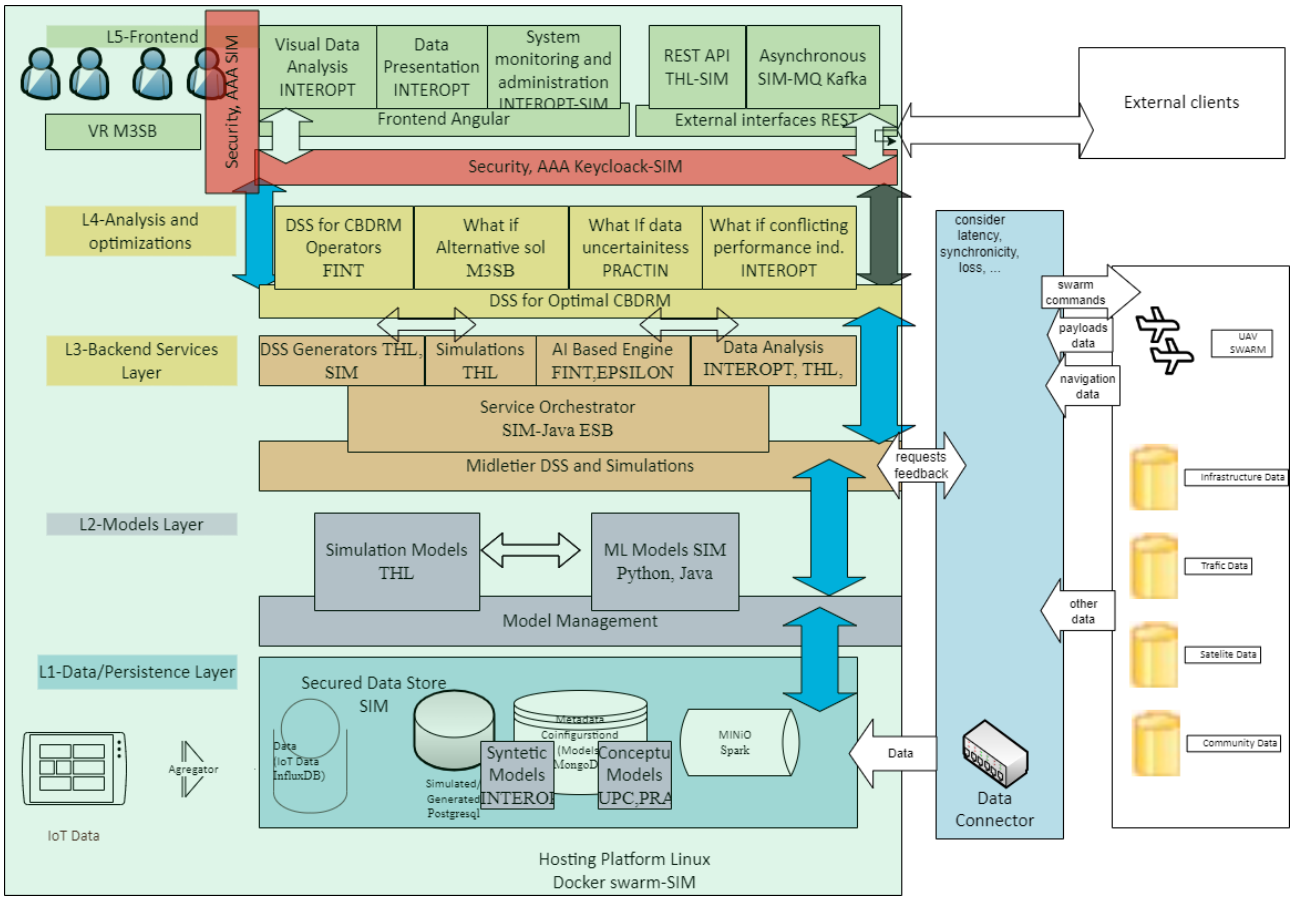


Figure 1 The Layered architecture of PANTHEON.

3.2 FUNCTIONAL VIEW

Based on the layered view as presented previously and for visualizing the interconnectivity and dependence of the different components in PANTHEON, an architecture diagram depicting the functional view of the architecture has been created. The functional view of the architecture is shown in Figure 2.

[7] Angular
[8] Grafana: The open observability platform | Grafana Labs

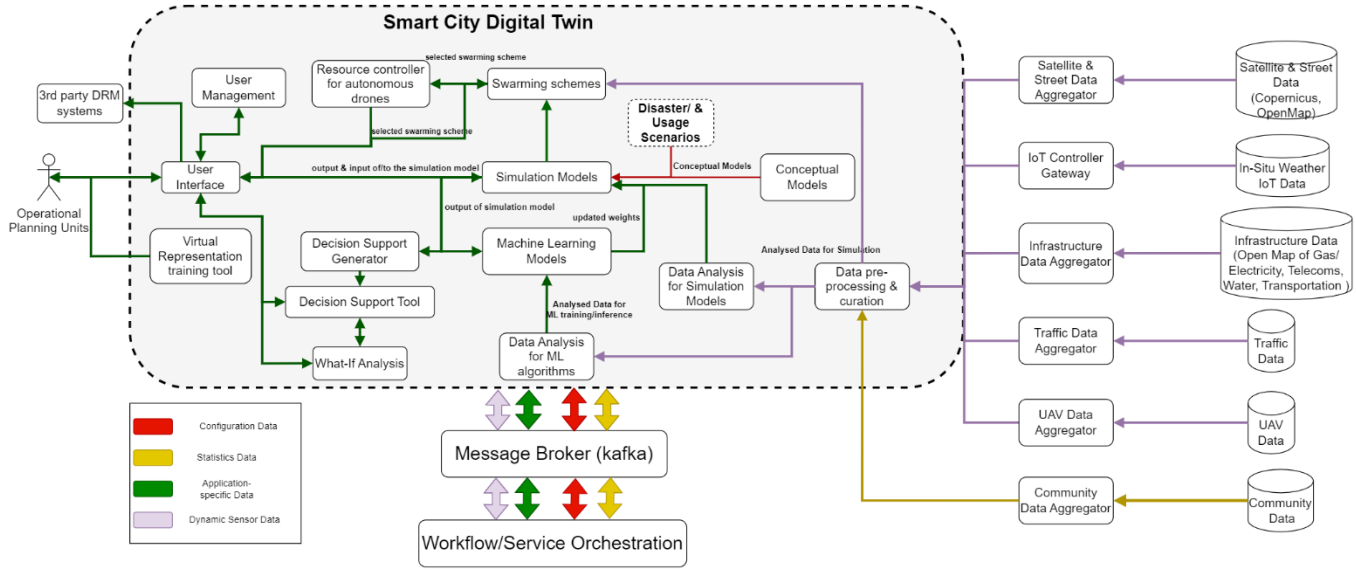


Figure 2 The functional view of the PANTHEON architecture.

Each component of the functional view of the architecture is further described in chapter 4. For each component, we provide a name, a brief description, the pilot and the scenario it will run, the module and submodule it has (if applicable), its inputs & outputs (if applicable) from/to the rest of the PANTHEON components, its algorithms (if applicable), its technologies (if applicable), its hardware requirements (if any) and a schematic for its internal architecture (if applicable).

3.3 INTEGRATION AND DEPLOYMENT INFRASTRUCTURE

3.3.1 PLATFORM HOSTING

The hardware infrastructure needed to implement the Pantheon platform, described in this document is mapped on a private cloud instance hosted by SIMAVI.

At the beginning of the deployment, an instance with the following characteristics is used:

- 16 CPU
- 32 GB RAM
- 500 GB SSD

A network with 1 GBs (Gigabits per second) GB is also accessible for communication lines.

The deployment in another environment, based on the proposed approach, mainly will clone the virtual machine created here, and place it in the target environment. In this way, the deployment effort is reduced.

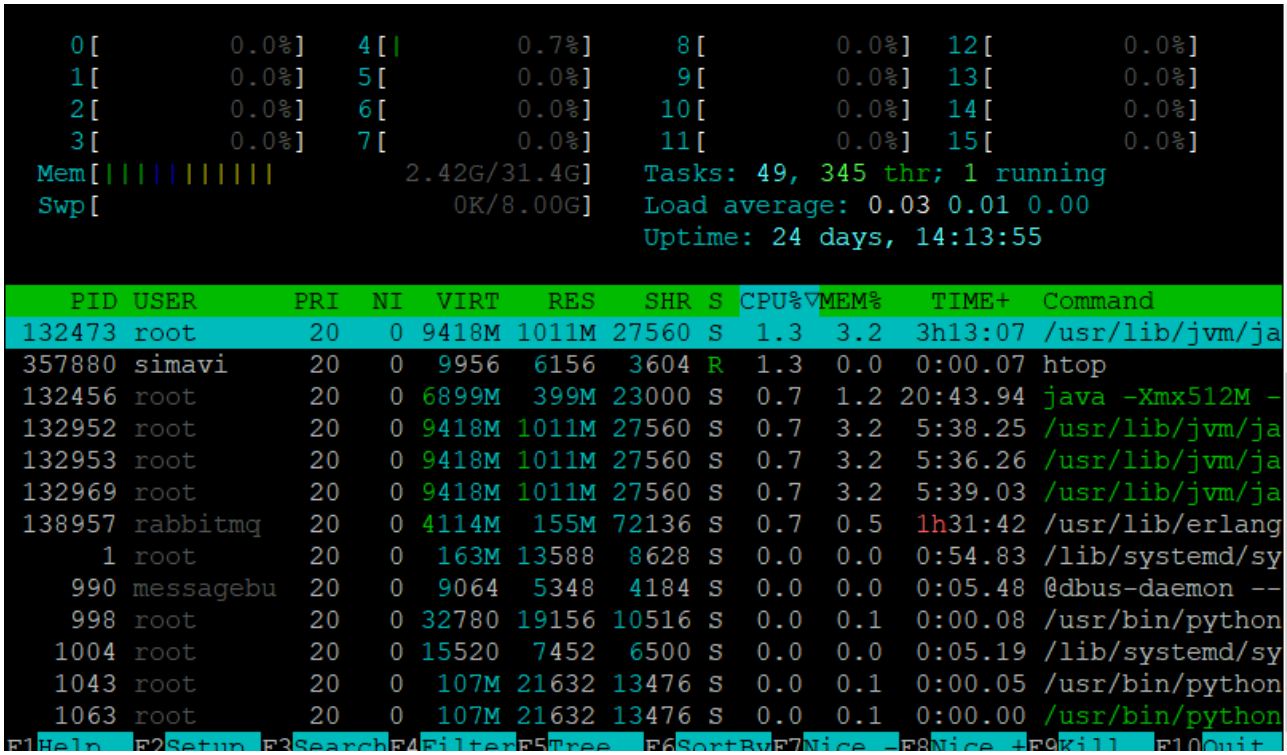


Figure 3. CPU allocated for the framework

3.3.2 VPN CONFIGURATION

The connection to the system will be enabled via VPN, managed by the FortiClient server.

Each user will have a client and will be able to use the client to connect to the system. This applies to all users of the system (developers, final users, system administrators).

To use FortiClient VPN the user must activate their account using the provided activation token. After activating the VPN account, it needs to be associated with a mobile device using FortiToken Mobile. Next, the relationship between the account and the mobile device is viable. After these settings, the user must install FortiClientSSLVPN then add "Remote Gateway" sslvpn.siveco.ro and click save (as shown in Figure 4).

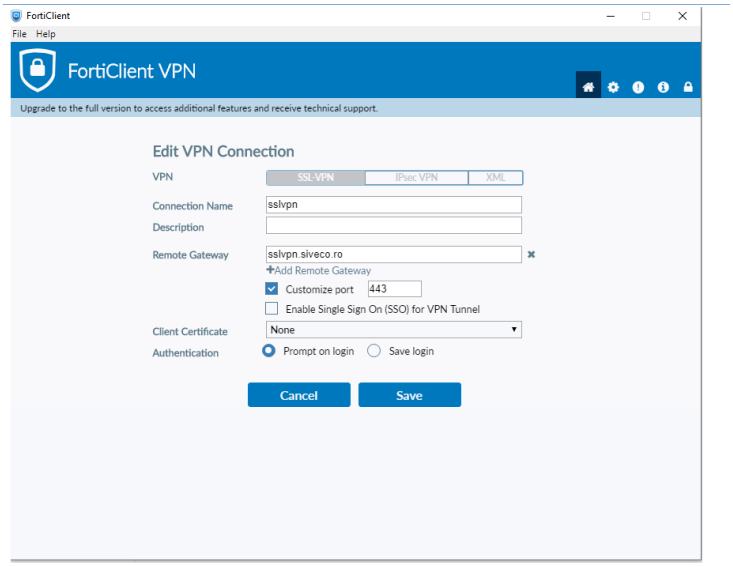


Figure 4 VPN connection.

Following this the user must select the VPN name, to introduce its credentials and the token listed on the mobile app Figure 5), then click ok and the connection should be established.

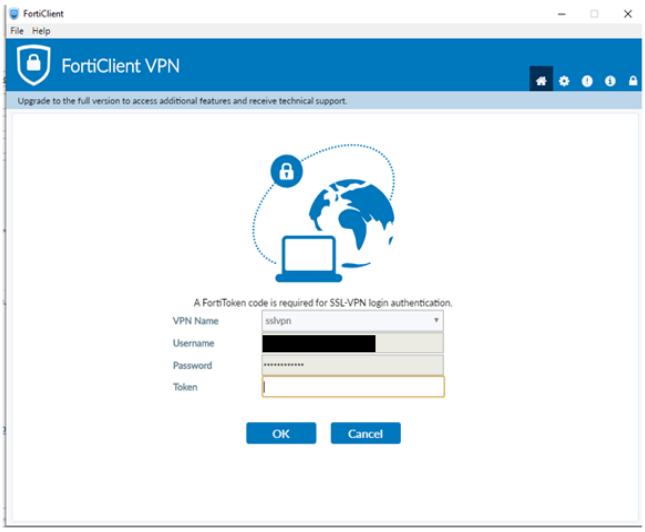


Figure 5 VPN access.

3.3.3 SOFTWARE INFRASTRUCTURE

The following elements are considered for the software infrastructure:

- Operation system,
- Docker engine,
- Java EE,
- Container repository.

The **operating system** is Ubuntu Linux 22 LTS. Ubuntu 22.04.1 allows one to use Ubuntu Terminal and run Ubuntu command line utilities including bash, ssh, git, apt, and many more.

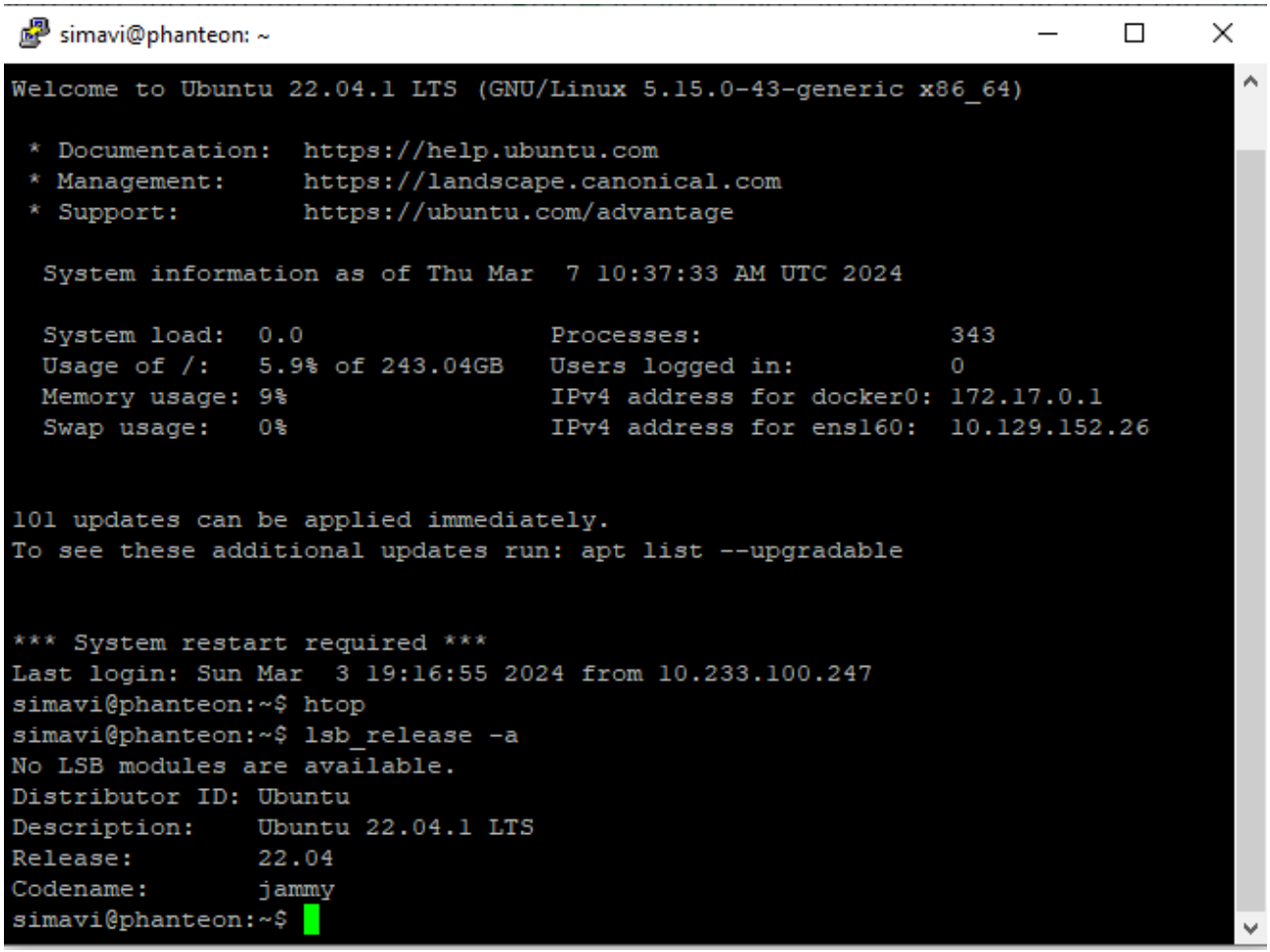


Figure 6 OS Linux version.

The **container** system is a Docker engine. Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications. By taking advantage of Docker's methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in the product. We are using Docker version 25.0.3. and docker compose v24.

```
simavi@phanteon:~$ docker version
Client: Docker Engine - Community
 Version:           25.0.3
 API version:       1.44
 Go version:        go1.21.6
 Git commit:        4debf41
 Built:             Tue Feb  6 21:13:09 2024
 OS/Arch:           linux/amd64
 Context:           default
permission denied while trying to connect to the Docker daemon socket at unix:/
//var/run/docker.sock: Get "http://%2Fvar%2Frun%2Fdocker.sock/v1.24/version": d
ial unix /var/run/docker.sock: connect: permission denied
simavi@phanteon:~$
```

```
simavi@phanteon:~$ sudo docker compose version
[sudo] password for simavi:
Docker Compose version v2.24.5
simavi@phanteon:~$
```

Figure 7 Docker engine and docker compose version.

Java is the preferred programming language for the development of the Pantheon platform. It reduces costs, shortens development timeframes, drives innovation, and improves application services. The Java version used is 17.

```
simavi@phanteon:~$ java -version
openjdk version "17.0.10" 2024-01-16
OpenJDK Runtime Environment (build 17.0.10+7-Ubuntu-122.04.1)
OpenJDK 64-Bit Server VM (build 17.0.10+7-Ubuntu-122.04.1, mixed mode, sharing)
simavi@phanteon:~$
```

Figure 8 Java version.

Python is the programming language used by several modules, especially those related to AI tools. Python version that is used here is the following:

```
simavi@phanteon:~$ python3 --version
Python 3.10.12
simavi@phanteon:~$
```

Figure 9 Python version.

Container repository. This component is used to store modules, such as Docker images. Next, the Docker images are taken and opened as Docker containers on the implementation platform.

The repository is a private one, with access for each partner, located at: [sivconnexions/pantheon general | Docker Hub]

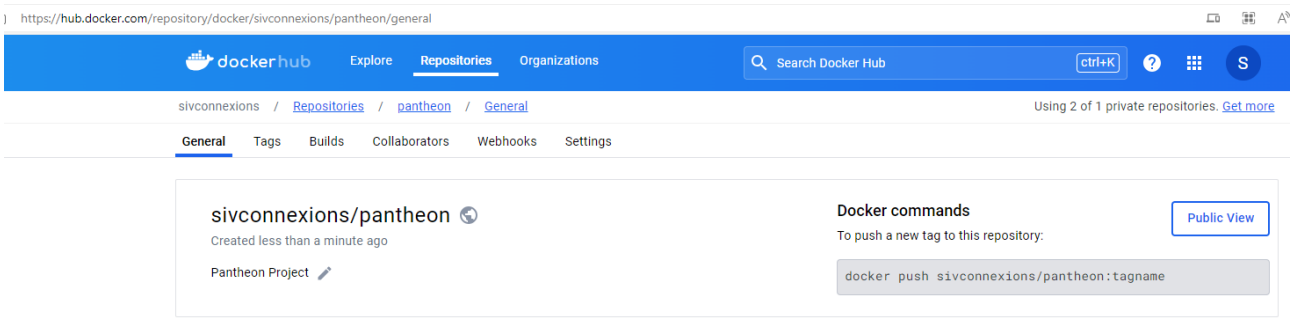


Figure 10 Container repository.

Message Broker. The message broker used is Kafka and is updated to its latest version. Apache Kafka is an open-source distributed event streaming platform used by thousands of companies for high-performance data pipelines, streaming analytics, data integration, and mission-critical applications. The version used is 3.6.1.

```
simavi@phanteon:/usr/local/kafka/bin$ sudo sh kafka-topics.sh --version
3.6.1
simavi@phanteon:/usr/local/kafka/bin$
```

Figure 11 Kafka Version.

3.4 COMPONENTS OVERVIEW

Component	Section	Relevant task(s)	Deliverable(s)
Satellite & Street Data Aggregator	4.1.1.1, 4.1.1.2, 4.1.1.3	T6.3	D6.2
Weather IoT Data Aggregator	4.1.2	T6.3	D6.2
UAV Data Aggregator	4.1.3	T6.3	D6.2
Traffic Data Aggregator	4.1.4	T4.3	D4.3
Community Data Aggregator	4.1.5	T4.1	D4.4
Data Pre-processing and Curation	4.2	T4.1	D4.4
Data Analysis for ML & ML models	4.3	T4.2, T4.3	D4.2, D4.3
Data Analysis for Simulation models	4.4	T4.2	D4.2
Conceptual Models	4.5	T4.2	D4.2
Simulation Models & Decision Support Generator	4.6	T4.4, T5.2	D4.3, D5.1
Decision Support & What-If Analysis	4.7	T5.5	D5.1
UAV Swarming Schemes	4.8	T6.1	D6.1
Resource Controller	4.9	T6.2	D6.1
User Interface	4.10	T4.5	D4.4
Virtual Representation Training Tool	4.10.1	T4.5	D4.4
Message Broker	4.11	T7.1	D7.1
Service & Workflow Orchestrator	4.12	T7.1	D7.1
User Management	4.13	T7.1	D7.1

4. PLATFORM COMPONENTS

4.1 DATA CONNECTORS & AGGREGATORS

4.1.1 SATELLITE & STREET DATA AGGREGATOR

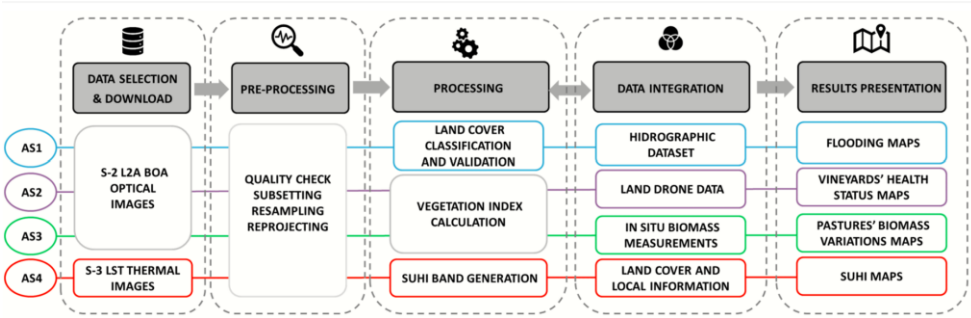
4.1.1.1 *Copernicus Data Aggregator*

Service Name	**Copernicus data services**				
Description	The European Copernicus program provides a comprehensive suite of Earth observation data and services. These services cater to various societal benefits including environmental protection, climate monitoring, natural disaster assessment, and others. Here's a breakdown of the types of data provided by the six (6) main Copernicus services and the common data formats they use. Services consider: 1. Copernicus Atmosphere Monitoring Service (CAMS): 2. Copernicus Marine Environment Monitoring Service (CMEMS): 3. Copernicus Land Monitoring Service (CLMS): 4. Copernicus Climate Change Service (C3S): 5. Copernicus Emergency Management Service (CEMS): 6. Copernicus Security Service:				
Pilots	All				
Scenario	All				
Module	**Copernicus data services**				
Submodule	Soil fuel maps for fire propagation (key input, next to all other)				
Input	**Input name**	**Input value type**	**Input data format**	**Description**	**From**
	Copernicus data	**Copernicus Atmosphere Monitoring Service (CAMS):** Data: Air, atmospheric composition, and climate forcing. **Copernicus Land Monitoring Service (CLMS):** Data: Land cover, vegetation, water cycle, and energy budget data. **Copernicus Climate Change Service (C3S):** Data: Climate indicators, climate predictions, and reanalysis datasets. **Copernicus Emergency Management Service (CEMS):** Data: for disaster response and recovery, including flood, fire, and drought monitoring.	Accessible via API: - NetCDF - GRIB - GEOTIFF - KML - ShapeFiles	**Access Methods**: Data Portals: Each service has its dedicated portal where data can be accessed, downloaded, or interacted with through APIs. Web Services: Many Copernicus services offer web services that allow for direct integration into software applications, including GIS tools. These services are structured to provide free, full, and open access to data, typically under a Creative Commons Attribution license, which fosters a wide range of uses from scientific research to commercial applications.	PATHEON Consortium Partners
Copernicus data Platforms					
https://www.coe.int/en/web/europarisks					
https://sim4nexus-space.eu/					
https://www.eea.europa.eu/highlights/natural-hazards-and-technological-accidents					

https://www.ipcc.ch/report/ar6/wg2/about/background
https://www.epc.eu/en/closed-projects/A-European-Approach-to-Risk-and-Threat-Management~10c6ec
https://drmkc.jrc.ec.europa.eu/knowledge/science-for-drm/science-for-disaster-risk-management-2017
https://drmkc.jrc.ec.europa.eu/knowledge/science-for-drm/science-for-disaster-risk-management-2020
http://rain-project.eu/about/deliverables/
https://recipe.ctfc.cat/results/
Rasor Project – Rasor Platform (rasor-project.eu)
FireHub (beyond-eocenter.eu)
FloodHUB Crowd Source Portal (beyond-eocenter.eu)
Presentazione standard di PowerPoint (noa.gr)
FloodHub (noa.gr)
About Copernicus-AI4Copernicus (ai4copernicus-project.eu)

Output	**Output name**	**Output value type***	**Output data format****	**Description**	**To*****
	Raster and vector data related to Copernicus services.	Multiple types of spatial data, to be used on-line or downloadable to be further processed	- NetCDF - GRIB - GEOTIFF - KML - ShapeFiles	These services are structured to provide free, full, and open access to data, typically under a Creative Commons Attribution license, which fosters a wide range of uses from scientific research to commercial applications.	**Modules** - Wildfires - Foods - Geoplatform Google Earth
Algorithm	The Copernicus Atmosphere Monitoring Service (CAMS) utilizes a variety of sophisticated algorithms to process data and provide information. These algorithms are integral to converting raw satellite observations and in-situ measurements into usable composition data and forecasts. Some of the key algorithms and processing techniques used in CAMS: • Data Assimilation Algorithms: 4D-Var (Four-Dimensional Variational Assimilation): This algorithm integrates observations with forecasts from a weather model to produce the most accurate analysis of the state of the atmosphere. • Ensemble Kalman Filter: Used in some applications to assimilate data over time, balancing the uncertainty in the model and the observations. • Atmospheric Models: ECHAM/MESSy Atmospheric Chemistry (EMAC) Model: A comprehensive model that includes detailed chemistry and microphysics processes. • Integrated Forecasting System (IFS): Developed by the European Centre for Medium-Range Weather Forecasts (ECMWF), this model combines meteorological forecasting with atmospheric composition aspects. • Emissions Databases and Processing: MACCity (Monitoring Atmospheric Composition and Climate CityZen): A global emission inventory used for historical data. • CAMS-GLOB-ANT: An anthropogenic emissions dataset used in global atmospheric composition models. • Aerosol and Trace Gas Retrievals: Sentinel Algorithms: Specific algorithms designed for data from Sentinel satellites, part of the Copernicus program, to detect and quantify aerosols and trace gases. • MODIS Data Processing Algorithms: Used for processing data from the MODIS instrument aboard NASA satellites, adapted for CAMS uses.				

	• Satellite Data Processing: GRASP (Generalized Retrieval of Atmosphere and Surface Properties): A versatile algorithm used for retrieving properties of aerosols and clouds from satellite observations. • Near-Real-Time and Reanalysis Data Processing: Copernicus Atmosphere Data Store Retrieval and Interpolation: Algorithms to facilitate user access to atmospheric data through the CAMS data store, enabling interpolation and customization of data outputs.
Technologies	Processing Copernicus data can be done using several software tools, ranging from general-purpose scientific data processing applications to specialized remote sensing and GIS software. Each of tools of below offer different capabilities, so the choice of software often depends on the specific needs of the project, such as the type of analysis, the format of the data, or the user's familiarity with the software. • **GOOGLE EARTH**: that can accommodate output from 2 GIS and many software technologies reported below via a KML/API. • **QGIS**: An open-source Geographic Information System (GIS) that is very versatile for spatial data analysis. It supports raster and vector data formats commonly used in Copernicus data, such as GeoTIFF and Shapefile. • **ArcGIS**: A comprehensive GIS platform by ESRI used for creating, managing, analyzing, and mapping all kinds of geospatial data. It supports extensive data formats and processing capabilities suitable for Copernicus data. • **ERDAS IMAGINE**: A remote sensing application which provides tools for analyzing satellite and aerial imagery. It's particularly useful for working with large datasets and performing complex transformations and analyses. • **GRASS GIS**: An open-source GIS software which provides powerful tools for spatial modeling, visualization, and analysis of raster and vector data, including time series. • **PANOPLY**: A data viewer developed by NASA's Goddard Institute for Space Studies. It reads netCDF, HDF, GRIB, and other types of data. Panoply is particularly useful for visualizing data slices, plotting geographic maps, and exporting images. • **PYTHON**: A language with numerous libraries for data analysis and scientific computing. Libraries like NumPy, pandas, Matplotlib, and especially GDAL for geospatial data, are useful for scripting custom processing workflows for Copernicus data. • **R-Language:** A language favored for statistical computing and graphics, with packages like raster, rgdal, and sp for spatial data analysis, which can be used to handle and analyze Copernicus data. • **SNAP** (Sentinel Application Platform): A common tool specifically designed for the processing of Sentinel satellite data, part of the Copernicus program. It offers tools for processing and analyzing radar and optical data from Sentinel-1, Sentinel-2, and Sentinel-3. • **ENVI + IDL**: A software solution used for processing and analyzing geospatial imagery. It is equipped with advanced tools for image processing and is compatible with various data formats used in Copernicus data. • **BEAM**: A tool primarily developed for processing and analysing ocean color data but has been extended to support the processing of other types of remote sensing data, including data from Copernicus satellites. **Google Earth** can be a powerful tool for **PANTHEON** for visualizing geographic data and can provide a compelling visual context for the data provided by Copernicus services and

	PANTHEON simulations (e.g., Wildfire propagation), making it easier to understand and communicate the information contained within the data. **Specifically**: Data from the Copernicus program can be displayed on **Google Earth**, provided it's in a compatible format. Google Earth primarily uses KML (Keyhole Markup Language) and KMZ (a compressed version of KML) formats for overlaying geographic data on its Earth images. The display of Copernicus data on Google Earth is achieved via the steps: 1. Conversion to KML/KMZ: If the Copernicus data is in a different format, such as GeoTIFF or NetCDF, it will need to be converted to KML or KMZ. Tools like GDAL (Geospatial Data Abstraction Library) can be used to convert raster data to KML. For vector data, GIS software like QGIS or ArcGIS can perform the conversion. 2. Direct Use of KML/KMZ Files: If the Copernicus service provides data directly in KML/KMZ format, you can simply download these files and open them directly in Google Earth. 3. Creating Overlays: For visual data like satellite imagery, creating an image overlay in Google Earth might be necessary. This involves converting your data into an image file, and then using the image as an overlay within Google Earth. The image file must be georeferenced accurately to align correctly on the map. 4. Using Plugins and Extensions: Some specialized plugins and tools can bridge the gap between complex GIS data formats and Google Earth's visualization capabilities. These might be useful for more advanced data types or detailed analytical tasks. 5. Scripting: For automation or handling large datasets, you can use scripting in languages such as Python. Libraries like simplekml allow you to create KML files from various data sources programmatically, which can then be loaded into Google Earth.
Hardware Requirements	• PC station • 1 terra hard disk on PC • 64 K memory • 8-12 terra external disk
Schematic	https://www.copernicus.eu/en/open-architecture-and-spatial-data-infrastructure-risk-management https://www.mdpi.com/2220-9964/11/2/121  The schematized workflows of the four ASs highlighting the Sentinel data used, the processing goals, the data integration, and the result representation.

4.1.1.2 *Street Data Aggregator*

Service Name	Street Network Service
Description	Digital street networks (such as the one provided by OpenStreetMap, OSM), are essential in the simulations for management of wildfires, fires, earthquakes, and floods

	Optional in PANTHEON). These networks offer critical infrastructure data that can significantly enhance the effectiveness of emergency response and disaster management efforts. They provide essential geographical frameworks that enhance the simulations (modelling) situational awareness, operational coordination, and strategic planning necessary for managing wildfires, fires, and floods effectively. For example, key applications in which digital street networks are used in this context are: • **Emergency Response and Evacuation**: (1) Route Planning: Digital street networks enable responders to plan the most effective routes for reaching affected areas quickly. They can also identify alternative routes when primary paths are blocked or hazardous; (2) Evacuation: For evacuations, having an up-to-date digital map helps in designing multiple evacuation routes to prevent congestion and ensure that people can leave dangerous areas safely and efficiently. • **Resource Allocation and Logistics**: (1) Resource Distribution: Managers can use street networks to plan the logistics of distributing resources such as water, food, medical supplies, and firefighting equipment. This is crucial in ensuring timely and equitable distribution, especially in extensive or rapidly changing situations; (2) Locating Critical Infrastructure: Knowing the location of hospitals, fire stations, and other critical infrastructure helps in planning how resources are allocated and how assistance is rendered during an emergency. • **Risk Assessment and Mitigation**: (1) Flood and Fire Risk Mapping: Integrating digital street maps with geographic information systems (GIS) and remote sensing data allows for detailed risk assessments. Planners can visualize which streets, neighbourhoods, and infrastructures are most at risk from floods and earthquakes or fires; (2) Preventive Measures: By analyzing the street network and landscape, authorities can identify critical areas needing preventive measures such as firebreaks, improved drainage systems, or barriers. • **Public Information and Communication**: (1) Real-Time Updates: During wildfires and floods and earthquakes, real-time updates using digital maps can be provided to the public about the spread of hazards, evacuation routes, and the locations of shelters or aid stations; (2) Community Engagement: Digital maps allow for community contributions, which can help update and verify the data in real-time, enhancing the accuracy and relevance of the information provided during a crisis. • **Recovery and Reconstruction**: (1) Damage Assessment: After a disaster, street networks overlaid with post-event imagery help assess the extent of damage and prioritize where recovery efforts should be concentrated; (2) Planning Reconstruction: Detailed street maps can assist in planning the reconstruction of destroyed infrastructure, ensuring that rebuilding efforts are optimized for future risk reduction. • **Integration with Other Data Sources**: (1) Enviroinfo: Hydrological and Meteorological Data, e,g., for floods and earthquakes, integrating street maps with river network data and rainfall or storm surge forecasts can aid in predicting flood extents and impacts; (2) Thermal and Satellite Imagery: e.g., For wildfires, integrating street maps with heat maps or fire spread models can help in monitoring the fire's progress and strategizing containment efforts.
Databases	There are several prominent databases and sources for geographic and street network data, each serving different needs and sectors. Each of the databases offers unique features and data coverage, catering to different user needs ranging from commercial and industrial applications to public service and academic research. For PANTHEON notable street databases are:

<table>
<tr><td></td><td colspan="5">

Bing Maps: Developed by Microsoft, Bing Maps offers road maps, aerial images, and street-side photographs along with various tools for route planning and location queries.
ESRI's ArcGIS Online: ArcGIS Online is a collaborative web GIS that allows users to use, create, and share maps, scenes, apps, layers, analytics, and data. The ArcGIS platform is extensively used in professional GIS applications, including urban planning and environmental management.
Google Maps: Google's proprietary map service provides detailed street maps, aerial imagery, and street views. It is widely used for commercial applications, navigation, and location-based services (EPSILON has been contributor to the Google Mapping System for GR and CY)
HERE Technologies: HERE (ex NAVTEC Technologies, of The Netherlands) offers location data and mapping services that are extensively used in automotive navigation systems, mobile applications, and enterprise solutions. Their data includes detailed road networks, points of interest, and traffic data.
Mapbox: Mapbox is a service provider for custom online maps for websites and applications. They provide extensive tools for creating custom stylized maps based on vector data and offer extensive API support for developers.
National Geographic Information Infrastructure (NSDIs): Various countries have their own geographic databases as part of their national geographic information infrastructure, providing authoritative data for government, commercial, and public use.
OpenStreetMap (OSM): A unique and versatile platform that has many distinguishing features which contribute to its popularity and widespread use across various domains, from navigation to research and humanitarian efforts. It features: (1) An Open Data Model, (2) it is Community-Driven; (3) It covers a Global Community: (4) It features Extensive and Detailed Data with rich attributes, with Customizability and Flexibility, etc.
TomTom: TomTom (ex TeleAtlas, in Greece mobile mapping system by EPSILON)[9] is another key player in the navigation sector providing maps and traffic data for automotive, enterprise, and IoT (Internet of Things) applications. Their data is also used for real-time navigation and route planning for Apple, Inc (Mapping system)

<u>Proposal</u>: For PANTHEON the proposition is, in case of:

QGIS Platform selection, then use of OpenStreetMap
Google Earth Platform Selection, the use of Google Street Network
A FIREmodel platform selection, then use of OpenStreetMap

</td></tr>
<tr><td>Pilots</td><td colspan="5">Wildfire (GR), Earthquakes (GR), Fire (AT), Heatwave (AT)</td></tr>
<tr><td>Scenario</td><td colspan="5">Emergency entails during to location of disaster</td></tr>
<tr><td>Module</td><td colspan="5">Digital Network Twin (wit attributes)</td></tr>
<tr><td>Submodule</td><td colspan="5"></td></tr>
<tr><td rowspan="2">Output</td><td>Output Name</td><td>Output value type</td><td>Output data format</td><td>Description</td><td>To</td></tr>
<tr><td></td><td>Road network</td><td>OpenStreetMap (OSM):
.<u>osm</u>: This is the primary data format used by OpenStreetMap. It is an XML</td><td>From the manuals</td><td>Data Aggregation & Curation</td></tr>
</table>

[9] http://www.epsilon.gr/geoinformatics/overview

			format that describes the map data. GPX: Used for uploading GPS track data to the OSM database. Shapefile: Often used for importing geographic data from other sources, though it requires conversion tools to import into OSM's native format. **Google Maps** uses KML (Keyhole Markup Language) and KMZ (zipped KML) files for adding user-generated data on the maps. GeoJSON and CSV: These formats are commonly supported for importing data through various Google Maps APIs.		

4.1.1.3 *Critical Infrastructure Location Data Aggregator*

Service Name	**Critical Infrastructures Service**
Description	When conducting emergency and risk analyses for Emergency Plans in a region, particularly focusing on natural disasters like wildfires, floods, earthquakes, and in-city fires, it's essential to consider various critical infrastructures. These infrastructures are pivotal as they support daily life, economic stability, and disaster response capabilities. Here are key critical infrastructures to consider: • **Energy Systems**: Electricity generation, transmission, and distribution facilities. Fuel supply chains, including storage and distribution points for gas and oil. Renewable energy sources, where applicable. • **Water and Sanitation**: Water treatment plants and distribution systems. Sewage and stormwater management systems. Flood control infrastructures such as dams, levees, and pumps. • **Telecommunications**: Phone and internet service infrastructure, including cell towers and data centers. Radio and television broadcasting facilities to disseminate information during emergencies. • **Transportation**: Roads, bridges, tunnels, and railway systems crucial for evacuation and emergency response. Ports and airports, which are vital for delivering emergency supplies and personnel. Public transport systems and their operational facilities.

	• **Health Care**: Hospitals, clinics, and other medical facilities. Pharmaceutical supply chains, including storage and distribution of critical medications. Emergency medical services and mobile health care units. • **Food and Agriculture**: Food production, processing facilities, and distribution networks. Agricultural infrastructure, including irrigation systems and equipment necessary for food security. • **Government Services**: Buildings and facilities that house key government functions, including emergency management centers. Data storage and processing facilities for government records and communication. • **Emergency Services**: Fire stations, police stations, and emergency operations centers. Infrastructure for disaster response equipment and personnel. • **Financial Services**: Banks and financial institutions critical for economic stability and recovery. Payment processing infrastructure and cash distribution points. • **Education Facilities**: Schools and universities, which can serve as emergency shelters or operational centers during disasters. • **Community Support Services**: Shelters and community centers that can provide refuge and aid during and after disasters. Non-profit organizations that play a role in disaster response and recovery.
Data/Information on GIS / Geoplatform	For each critical infrastructure identified when preparing for natural disasters such as wildfires, floods, earthquakes, and in-city fires, it's crucial to collect specific data, and store data on GIS/GeoPlatform, to enhance resilience and preparedness. Here's a detailed breakdown of the types of data needed for each infrastructure category as follows for: **Energy Systems** • Locations and specifications of power generation and distribution facilities. • Capacity, usage rates, and vulnerability assessments of grids and substations. • Fuel reserves and logistics for fuel distribution, including alternative fuel sources. **Water and Sanitation** • Maps and operational details of water sources, treatment plants, and distribution networks. • Capacities and backup systems for water and sewage treatment facilities. • Risk assessments for flooding or seismic activities impacting these systems. **Telecommunications** • Inventory and status of communication towers, cables, and switching centers. • Redundancy and failover capabilities of the network. • Coverage maps and areas susceptible to service disruption. **Transportation** • Maps and structural assessments of roads, bridges, tunnels, and railways. • Status and operational capabilities of ports and airports. • Evacuation route efficiency and alternatives. **Health Care:** • Capacities and capabilities of hospitals and clinics, including bed counts and specialization. • Inventory of medical supplies, pharmaceuticals, and critical equipment. • Emergency operation plans for medical facilities. **Food and Agriculture:**

<table>
<tr><td></td><td colspan="5">

Locations and operational status of food storage facilities and distribution centers.
Details on critical agricultural infrastructure and any dependencies on water or energy.
Supply chain information for essential food products.

Government Services:

Continuity plans for government operations.
Infrastructure and personnel critical to emergency management.
Security protocols for critical data and communications.

Emergency Services:

Locations, capabilities, and readiness status of emergency services like fire, police, and search and rescue.
Inventory of emergency equipment and vehicles.
Communication and coordination mechanisms among different emergency services.

Financial Services:

Maps and security assessments of major financial institutions and ATM networks.
Contingency plans for financial operations during disruptions.
Systems in place for emergency cash distributions.

Education Facilities:

Capacities and current uses of schools and universities as potential shelters or command centers.
Structural integrity and safety evaluations of buildings.
Emergency and evacuation plans specific to each facility.

Community Support Services:

Capacities and locations of shelters and community centers.
Resources available for public use during emergencies, including staffing and supplies.
Network of volunteer organizations and their capabilities and resources.

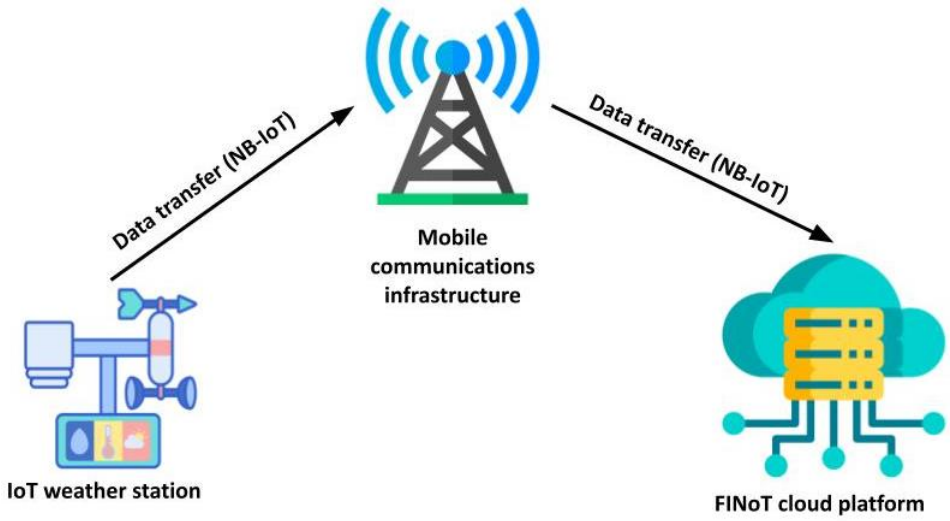
</td></tr>
<tr><td>Pilots</td><td colspan="5">Wildfire (GR), Fire (AT), Earthquake (GR), Heatwave (AT)</td></tr>
<tr><td>Scenario</td><td colspan="5">Emergency entails during to location of disaster</td></tr>
<tr><td>Module</td><td colspan="5">Input to the GIS Digital Twin (with attributes) and Metadata</td></tr>
<tr><td></td><td colspan="5">GIS systems allow for flexible unit handling, but it's crucial to maintain consistency in units across datasets to ensure accurate spatial analysis and data interoperability. Furthermore, GIS databases must incorporate INSPIRE metadata that specifies the units' formats used, which helps in maintaining clarity and precision in data management and analysis. This metadata is vital when datasets are shared between different organizations or used in multi-disciplinary projects, in the CASE of PANTHEON, between Firebrigade (fire) and Civil Protection (Earthquake)</td></tr>
<tr><td>Output</td><td>Input name</td><td>Input value type (GIS)</td><td>Input data format</td><td>Description</td><td>To</td></tr>
<tr><td></td><td>Energy Systems:
Facility location (latitude and longitude in degrees).
Capacity (megawatts, MW).
Fuel reserve levels (litres, gallons, or cubic meters).
Water and Sanitation:
Plant and network location (latitude and longitude in degrees).
Treatment capacity (litres per day).</td><td></td><td>When storing data on Geographic Information Systems (GIS), the units used can vary depending on the type of data being represented and the specific requirements of the project. Here are the common units used in GIS for storing and analyzing data across different categories of critical infrastructure:</td><td>From the manuals</td><td>Data Aggregation & Curation</td></tr>
</table>

	• Flood risk (expressed as return period in years or probability). **Telecommunications:** • Infrastructure locations (latitude and longitude in degrees). • Network redundancy (as a percentage or binary, yes/no). • Service coverage area (square kilometers or percentage of population covered). **Transportation:** • Structural integrity (rating scale, typically 1-5 or A-F). • Capacity (number of vehicles per hour for roads, tonnage for bridges). • Accessibility of evacuation routes (number of routes, average travel time in minutes). **Health Care:** • Hospital locations (latitude and longitude in degrees). • Bed capacity (number of beds). • Inventory of critical equipment (number of items like ventilators, MRIs). **Food and Agriculture:** • Storage facility locations (latitude and longitude in degrees). • Storage capacity (metric tons or cubic meters). • Dependency details (percentage of water, energy needed for operation). **Government Services:** • Location of critical facilities (latitude and longitude in degrees). • Data redundancy (as a percentage or binary, yes/no). • Employee headcount for critical operations (number of personnel). **Emergency Services:** • Facility locations (latitude and longitude in degrees). • Response time (minutes or seconds to reach an incident). • Equipment inventory (number of units, such as fire trucks or ambulances). **Financial Services:** • Location of financial centers (latitude and longitude in degrees). • Cash availability (amount in local currency). • Online transaction capacity (transactions per second). **Education Facilities:** • Building safety score (rating scale, typically 1-5 or A-F). • Shelter capacity (number of people the facility can accommodate). • Evacuation route efficiency (rated on a scale or average time to evacuate in minutes).	**Spatial Data** • Coordinates: Typically latitude and longitude are stored in degrees (decimal degrees for more precision). • Elevation: Often recorded in meters or feet. • Area and Distance Measurements: Area: Square meters (sq m), hectares, or square kilometers (sq km) for larger extents; acres may also be used depending on regional preferences. • Distance: Meters or kilometers; feet and miles are used in regions using the imperial system. • Capacity and Volume: • Water flow: Cubic meters per second (m^3/s) or liters per second (L/s). • Fuel storage: Liters, gallons, or cubic meters. • Electricity: Megawatts (MW) or kilowatt-hours (kWh) for energy production or consumption. • Quantitative Attributes: Population coverage: Typically expressed as a percentage or an absolute number. Vehicle capacity: Number of vehicles per hour. Bed capacity: Number of beds in facilities like hospitals. • Risk and Structural Integrity: Risk levels: Often qualitative, but can be quantified as probabilities (e.g., 1 in 100-year flood risk) or using rating scales (numeric or alphabetic). Structural ratings: Numeric scales, rating indices, or categorized levels (e.g., 1-5, A-F). • Temporal Data: Time: Seconds, minutes, hours, or days depending on the context (e.g., response times, evacuation times). • GIS systems typically allow for flexible unit handling, but it's crucial to maintain consistency in units across datasets to ensure accurate spatial analysis and data interoperability. Furthermore, GIS databases often incorporate metadata that specifies the units used, which helps in maintaining clarity and precision in data management and analysis.		

	Community Support Services: • Shelter locations and capacities (latitude and longitude in degrees, number of people accommodated). • Resource availability (quantities of food, water, medical supplies). • Volunteer numbers (total number of active volunteers).	This metadata is vital when datasets are shared between different organizations or used in multi-disciplinary projects.		

4.1.2 WEATHER IN-SITU IOT AGGREGATOR

Service Name	In-situ IoT-based weather monitoring				
Description	Providing local weather/environmental conditions monitoring				
Pilot	Athens; Vienna				
Scenario	1-Athens wildfire; 3-Vienna heatwave				
Module	T4.1 – Big data generation and harvesting; T6.3 – Integration of Copernicus services, satellite remote sensing and in situ PANTHEON aerial platform				
Submodule	N/A				
Input	**Input name**	**Input value type**	**Input data format**	**Description**	**From**
	IoT weather parameters data	Temperature: -40 to 125°C Relative humidity: 0-99% Wind speed: 0-80 m/sec Wind direction: 0-360° Solar radiation: 0-2000 W/m^2	REST API Json/csv	Sets of weather parameter data, gathered by the weather station gateway and transmitted by NB-IoT or IEEE 802.15.4	Json/csv From the weather station IoT gateway
Output *Describe the type of outputs expected by your service and the PANTHEON component they are expected to be consumed*	**Output name**	**Output value type**	**Output data format**	**Description**	**To**
	IoT weather monitoring	Temperature: -40 to 125 °C	REST API Json/csv	Sets of weather parameter data, gathered by the weather station	Data pre-processing and curation (T4.1)

		Relative humidity: 0-99% Wind speed: 0-80 m/sec Wind direction: 0-360° Solar radiation: 0-2000 W/m^2		gateway and transmitted by NB-IoT or IEEE 802.15.4	
Algorithm	N/A				
Technologies	NB-IoT, IEEE 802.15.4, Fiware				
Hardware Requirements	TBD				
Schematic					

4.1.3 UAV DATA AGGREGATOR

Service Name	UAV data aggregator				
Description	the service collects the data from the UAVs (payload and navigation data) and processes it ready for ingestion by component T4.1. the processing can consist of (not an exhaustive list): filtering, synchronising, taking into account latency and packet loss…				
Pilot	Vienna (man-made disaster) & Athens (wildfire and earthquake)				
Scenario	Vienna (man-made disaster) & Athens (wildfire and earthquake)				
Module	T4.1 and T6.3				
Submodule	N/A				
Input	**Input name**	**Input value type**	**Input data format**	**Description**	**From**

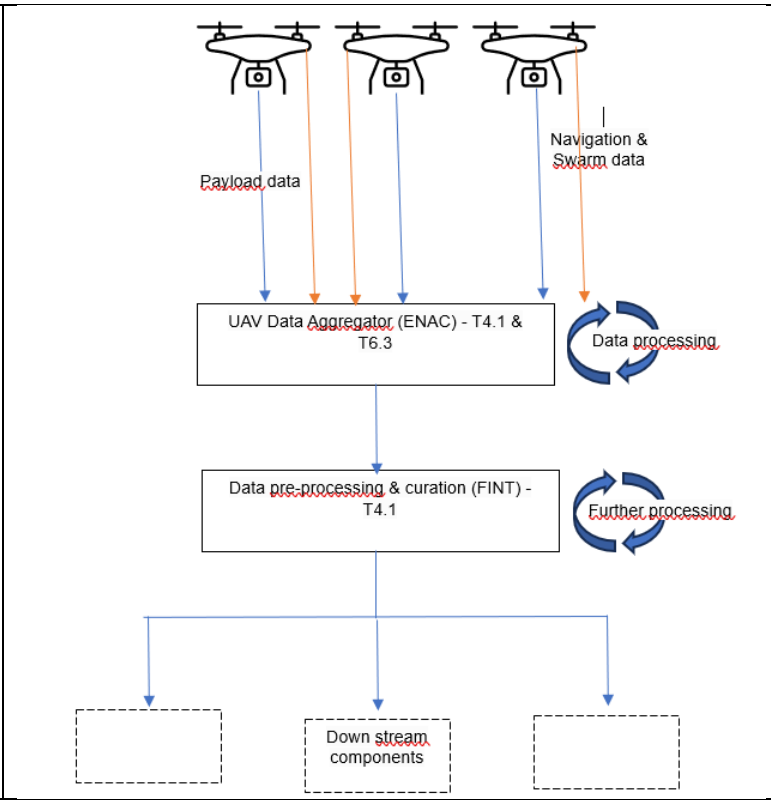
	UAV payload data	Depends on onboard sensor	Depends on onboard sensor and processing	All the data coming from payload sensors: EO / IR images, sniffer sensors, lidar…	UAVs

	UAV navigation & swarm data	State: {
		State [string] and Status [Vector of Strings] for each UAV in swarm,
		Current Position [float] of each UAV agent, remaining flight time, RSSI etc.
		}
			ROS2
			Publisher
			MQTT
			REST API
				all the data enabling, and pertaining to, the flight of the UAV inside the swarm (autopilot data)	UAVs

Output	**Output name**	**Output value type**	**Output data format**	**Description**	**To**
	UAV payload data	Depends on onboard sensor	Depends on onboard sensor and processing	All the data coming from payload sensors: EO / IR images, sniffer sensors, lidar…	T4.1

	UAV navigation & swarm data	State: {
		State [string] and Status [Vector of Strings] for each UAV in swarm,
		Current Position [float] of each UAV agent, remaining flight time, RSSI etc.
		}
			ROS2
			Publisher
			MQTT
			REST API
				all the data enabling, and pertaining to, the flight of the UAV inside the swarm (autopilot data)	T4.1

Algorithm	This component will centralise, store, and process incoming data in order to address desynchronisation, latency, packet loss… issues and then send the data on its way downstream. This component must not be a bottleneck and must be resilient to network degradations (thanks to server redundancies, relevant communication protocols, encryption etc).				
Technologies	Python, C++, ROS2, MQTT, KAFKA				
Hardware Requirements	*Recommended*: 8+ (v)CPU cores, 16+GB RAM, 960+GB storage				

Schematic	

4.1.4 TRAFFIC DATA AGGREGATOR

Service Name	Traffic Data Aggregator				
Description	The service offers a module for delivering traffic data into the PANTHEON platform. The service will aggregate traffic data from different vendors (Municipality of Attica, Attiki Odos, City of Vienna) and different modalities (JSON, CSV), aggregate and enrich them with geolocation and offer them in the data pre-processing and curation module in a standard format (JSON).				
Pilot	Attica, Vienna				
Scenario	All				
Module	Task 4.1 – Traffic Data Aggregator				
Submodule	N/A				
Input	**Input name**	**Input value type**	**Input data format**	**Description**	**From**
	Traffic data by Regional Municipal	Array of dictionaries with the following structure: { “deviceid”: string	JSON	Traffic dataset offered by the Region of Attica	PANTHEON platform filesystem

	ity of Attica	“countedcars”: int “approcesstime”: string “road_name”: string “road_info”: string “average_speed”: float }			
	Traffic data by Attiki Odos	Lines with the following structure [GR]: Date – string; Hour – int; Five-minute interval – int; Vehicle Detection Sensor (VDS) ID – string; Private Cars – int; Trucks – int; Total – int; Detection failure percentage – float; Occupation – int; Speed – int Below is an actual excerpt from the data (header + one line of data): ΗΜΕΡΟΜΗΝΙΑ;ΩΡΑ;ΠΕΝΤΑΛΕΠΤΟ;VDS;ΙΧ;ΦΟΡΤΗΓΑ;ΣΥΝΟΛΟ;ΑΣΤΟΧΙΑ;ΚΑΤΑΛΗΨΗ;ΤΑΧΥΤΗΤΑ 2021-12-31 00:00:00;23;11;VDS A 23.69;0;0;0;73,333333333333329;0;0	CSV	Traffic dataset as kindly offered by Attiki Odos S.A.	PANTHEON platform filesystem
	Traffic data by City of Vienna	TBD. Potential example on the structure is the following: { JAHR;MONAT;ZNR;ZNAME;STRTYP;STRNR;RINAME;FZTYP;DTVMS;DTVMF;DTVMO;DTVDD;DTVFR;DTVSA;DTVSF;TVMAX;TVMAXT; 2023;DEZ.;1075;Reichsbrücke;B;8;B_8 - km_2,6;Leopoldstadt;Kfz;157	JSON/CSV	Data from the regional authorities of Vienna	

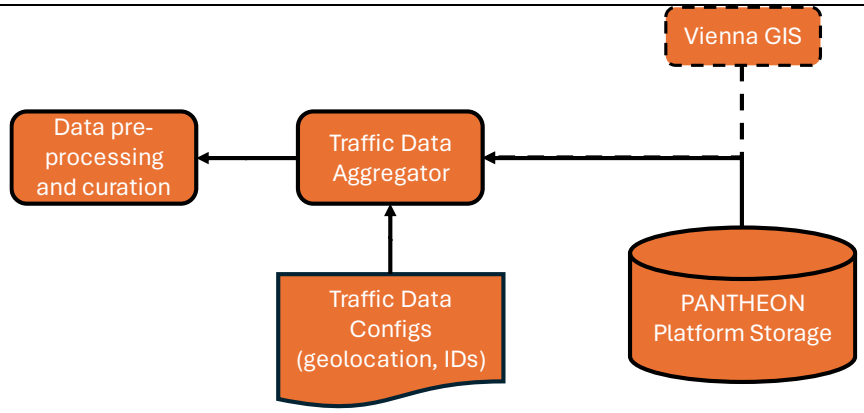
		94;18631;18590;18617;18700;13334;10948;21952;*Fr,15.12. }			
Output	**Output name**	**Output value type**	**Output data format**	**Description**	**To**

	Traffic data by Regional Municipality of Attica	Same as relevant input, with additional fields for geolocating the device:
		{
		“longitude”: float,
		“latitude”: float,
		…
		}
			JSON	Traffic dataset offered by the Region of Attica	Data preprocessing and curation module (T4.1)

	Traffic data by Attiki Odos	A JSON of the relevant input, with additional fields for geolocating the device:
		{
		“longitude”: float,
		“latitude”: float,
		…
		}
			JSON	Traffic dataset as kindly offered by Attiki Odos S.A, formatted in JSON	Data preprocessing and curation module (T4.1)

	Traffic data by City of Vienna	Same as relenvant input, with additional fields for geolocating the devices:
		{
		“longitude”: float,
		“latitude”: float,
		…
		}
			JSON	Data from the regional authorities of Vienna, augmented with data from Vienna GIS	Data preprocessing and curation module (T4.1)

Algorithm	The data will be read from the PANTHEON platform storage. Then they will be converted to JSON (if they are not) and configuration files along with potential APIs for online GIS (e.g. Vienna GIS) will be used to geolocate each traffic camera in the WGS84 coordinate system for simulating traffic in the SCDT.				

Technologies	Python
Hardware Requirements	*Recommended*: 512m CPU cores, 1GB RAM, 10 GB storage
Schematic	

4.1.5 COMMUNITY DATA AGGREGATOR

This component is dedicated to uploading and processing of community data. This kind of data are considered as statistical information describing some characteristic of a population or a community. This could be related to population density distributed by classes of age, population georeferenced distribution with respect to economic and social indicators, anonymized data linked to religious affiliation. These kinds of data are usually coming from sociological studies and are usually trying to give a macroscopic description of a population or a community with respect to some indicators that can be linked to individuals. The results are usually concretized by proportion of probabilities that can be more or less correlated to geographic parameters.

The main problem is the main problem is that this data is often imprecise in terms of geographical distribution or is in any case poorly formatted. The purpose of the Community Data Aggregator component is to translate any data batch into a formatted and agreed template, allowing them to be implemented and used into PANTHEON platform.

This function will be divided into three steps (see Figure 12):

- **Reading and interpreting raw data:** this block will be systematically specific to the data to be processed and must therefore be adapted and redeveloped for each type of data or for each source producing the input data;
- **Georeferenced interpolation on a formatted grid**: basic principles with remain the same as the principle is to traduce the poor geographic references into a systematic and structured geographic mapping grid (Regularly shaped grids can only consist of squares or hexagons, to be agreed with partners), but this block will probably have to be adapted for each batch of raw data as the input geographic information will probably be rough and varied depending on data and related sources. The main algorithm, still to be defined and something being specific to the delivered raw data, will be based on 2D interpolation functions or algorithms managing geographic redistributions;
- **Grid generation and output information implementation:** as soon as the data has been interpolated on a predefined geographic basis, the output file can be generated in an understood and shared format,

making the information exchangeable and interpretable. The output file will be in the form of a GEOJSON type text file delivering the redistributed data with their geographic references and all the information making it possible to describe and reconstruct the corresponding geographical distribution grid.

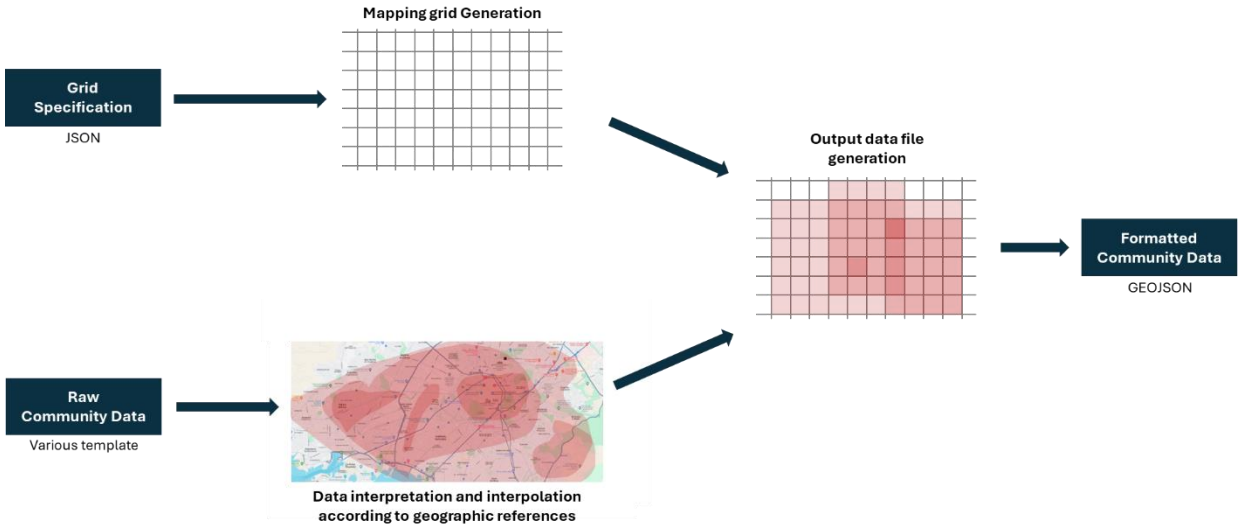


Figure 12 General description of the Community Data Aggregator workflow.

The input of the Community Data Aggregator component will be consisted of the raw data themselves, but also a JSON type text file delivered by the PANTHEON platform (API request initiated by the Community Data Aggregator component) to define the size and resolution of the geographic grid.

The mechanism used to deliver the outputs on the PANTHEON platform have still to be defined with the partners. The output can be push on the platform by the Community Data Aggregator component on its own initiative (e.g. pushing the output file in the KAFKA stream) when the raw data are made available, or the PANTHEON platform can send an API request to get the output file, receiving a specific flag when the data are not available.

Service Name	Community Data Aggregator				
Description	Digital converter of Community Data, usually lightly formatted, in a predefined format that can be used by the PANTHEON platform. The main purpose will be to georeference these data and to implement them on a predefined geographic grid.				
Pilot	Attica				
Scenario	Wildfire				
Module	Community Data Aggregator				
Submodule	Wildfire				
Input	**Input name**	**Input value type**	**Input data format**	**Description**	**From**
	Population density (%) vs. mobility capability	Text	TBD	Statistic population density (%) vs. their capability to move and evacuate the area	TBD

	Grid requirement	Text	JSON	Give the dimension of the grid	PANTHEON Platform
	Result resolution	Text	JSON	General parameters defining the level of resolution to be applied on the results (e.g. size and number of age classes, number of digits for results)	PANTHEON Platform
	Density population in absolute value	Text	TBD	Give the population density according to some rough geographic reference (e.g. by neighbourhood)	TBD
Output	**Output name**	**Output value type**	**Output data format**	**Description**	**To**
	Mobility Mapping	Text	GEOJSON	Providing proportion of population density geographic distribution with their mobility capability (car, foot, limited…)	PANTHEON Platform
Algorithm	TBD, the statistic values will be interpolated and correlated on a geographic mapping grid to be defined with respect to PANTHEON platform requirements.				
Technologies	Python or C++				
Hardware Requirements	CPU, other TBD				
Schematic	TBD				

Service Name	Community Data Aggregator				
Description	Digital converter of Community Data, usually lightly formatted, in a predefined format that can be used by the PANTHEON platform. The main purpose will be to georeference the data and to implement them on a predefined geographic grid.				
Pilot	Austria				
Scenario	Heatwave				
Module	Community Data Aggregator				
Submodule	Heatwave				
Input	**Input name**	**Input value type***	**Input data format****	**Description**	**From****
	Population density (%) vs. age	Text	TBD	Statistic population density (%) vs. the age provided as numbers according to geographic reference (e.g. by neighbourhood)	TBD
	Grid requirement	Text	JSON	Give the dimension of the grid	PANTHEON Platform
	Result resolution	Text	JSON	General parameters defining the level of resolution to be	PANTHEON Platform

				applied on the results (e.g. size and number of age classes, number of digits for results)	
	Density population in absolute value	Text	TBD	Give the population density according to some rough geographic reference (e.g. by neighbourhood)	TBD
Output	**Output name**	**Output value type**	**Output data format**	**Description**	**To**
	Age Density Mapping	Text	GEOJSON	Providing proportion of population density geographic distribution for different age classes	PANTHEON Platform
Algorithm	TBD, the statistic values will be interpolated and correlated on a geographic mapping grid to be defined with respect to PANTHEON platform requirements.				
Technologies	Python or C++				
Hardware Requirements	CPU, other TBD				
Schematic	TBD				

4.2 DATA PRE-PROCESSING AND CURATION

Service Name	Data pre-processing and curation				
Description	Pre-processing of the (raw) data incoming from the aggregators				
Pilot	Athens; Vienna				
Scenario	1-Athens wildfire; 2-Athens earthquake; 3-Vienna heatwave; 4-Vienna cyber-terrorism				
Module	T4.1 – Big data generation and harvesting				
Submodule	N/A				
Input	**Input name**	**Input value type**	**Input data format**	**Description**	**From**
	Data from all the aggregators	N/A	REST API Json, csv	Sets data coming from all the available aggregators, such as satellite, in-situ IoT, infrastructure, traffic, UAV and community-based.	Aggregators

Output	Output name	Output value type	Output data format	Description	To
	Pre-processed data	N/A	REST API Json/csv	Pre-processed sets of the input data	Data analysis for simulation models (T4.2)
Algorithm	TBD				
Technologies	TBD				
Hardware Requirements	TBD				
Schematic	N/A				

4.3 DATA ANALYSIS FOR ML MODELS

The present chapter refers to the process of analysing and preparing data used by ML models.

Data analysis for machine learning (ML) models refers to the process of examining and understanding the data that will be used to train, validate, and test machine learning algorithms. It involves various techniques and methodologies to explore, clean, preprocess, and transform the data to make it suitable for training ML models.

The importance of analysing and processing data used by ML models comes from the fact that a model is better if the data used to train the model is most appropriate.

First, we'll make an introduction to ML models, and then we'll describe the data analysis technique proposed.

4.3.1 MACHINE LEARNING MODELS OVERVIEW

Machine learning (ML) models are computational algorithms or mathematical frameworks that learn patterns and relationships from data to make predictions or decisions without being explicitly programmed. ML models can be categorized into several types based on their learning approach and the nature of the tasks they perform[10]. Some common types of ML models include:

Supervised Learning Models: These models learn from labelled data, where each input is associated with a corresponding target output. Supervised learning models include:

- **Classification**: Predicting a categorical label or class for new instances based on training examples with known labels. Examples include logistic regression, decision trees, random forests, support vector machines (SVM), and neural networks.
- **Regression**: Predicting a continuous numerical value based on input features. Linear regression, polynomial regression, decision trees, support vector regression (SVR), and neural networks are examples of regression models.

[10] https://jakevdp.github.io/PythonDataScienceHandbook

Unsupervised Learning Models: These models learn from unlabelled data, aiming to discover patterns, structures, or relationships within the data. Unsupervised learning models include:

- **Clustering**: Grouping similar data points together based on their characteristics. K-means clustering, hierarchical clustering, and Gaussian mixture models (GMM) are common clustering algorithms.
- **Dimensionality Reduction**: Reducing the number of features or variables in the data while preserving important information. Principal Component Analysis (PCA), t-distributed Stochastic Neighbor Embedding (t-SNE), and autoencoders are popular dimensionality reduction techniques.

Reinforcement Learning Models. Reinforcement Learning (RL) is a machine learning paradigm where an agent learns to make decisions by interacting with an environment.

The ML models are defined in the Pantheon project as part of:

- Task 4.2: Analysis and representation of collected data & Conceptual Models
- Task 4.3: Enhanced Intelligence & Self-adaptive Simulations related to Community Disaster Management Models

The present chapter only mentioned the ML models, as the target is Data Analysis.

4.3.2 MACHINE LEARNING DATA MODELS

An important aspect when using ML models refers to how data is organized in such models. The organization of data depends on the type of model, and the type of data processed. The group of data models is given by the type of model, and the details are given by the type of processing. The definition and design of data models is part of task 4.2. In this chapter, we are summarizing the data models, for a good understanding of what data analysis will refer to. The choice of data models depends on the specific task, the nature of the data, and the algorithms being used. Some of the most frequently used data models for machine learning include:

Relational (Tabular) Data: Tabular data is structured data organized in rows and columns, commonly found in relational databases or spreadsheet formats. This type of data is used in many machine-learning tasks such as classification, regression, and clustering. Popular libraries like pandas in Python are frequently used to work with tabular data. However, the main software tools used to treat relational data are the RDBMS, like Oracle[11], PostgreSQL[12], MySQL[13], and SQL Server[14].

When working with the design of a system using relational data, the concept used is the Entity-relationship diagram (ERD).

[11] https://www.oracle.com
[12] https://www.postgresql.org
[13] https://www.oracle.com
[14] https://www.microsoft.com/en-us/sql-server

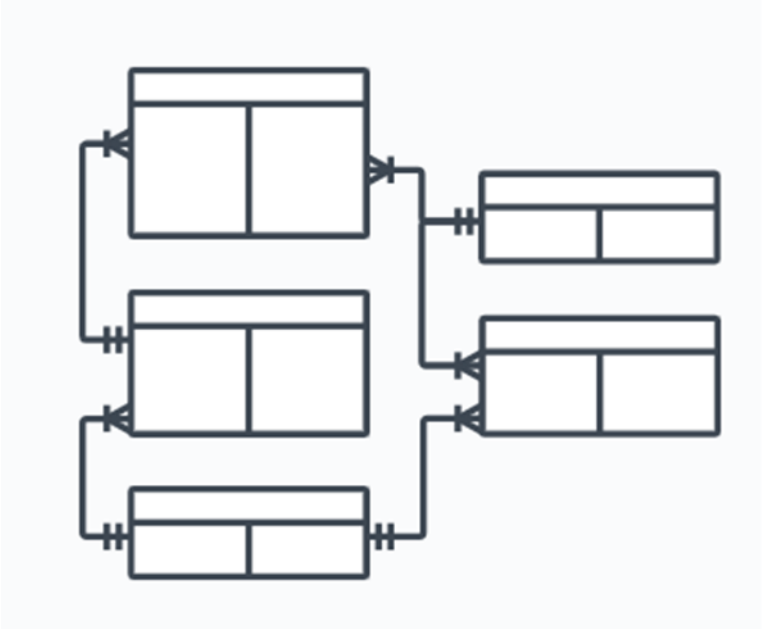


Figure 13 ERD representation

In PANTHEON, we'll use tabular data to store data received from IoT (excluding time series), simulated data, and data representing extracted features.

Image and Video Data: Image data consists of pixel values representing visual information. Video data is considered as a sequence of images. Convolutional Neural Networks (CNNs) are commonly used for tasks such as image classification, object detection, and image segmentation. Image data models often involve processing images in formats like JPEG, PNG, or TIFF. Image and video data will be processed by a file system and stored as files protected in a data lake mechanism. In PANTHEON, processing Image data comes from T4.1-developed mechanisms.

Document and Text Data: Document Text data includes natural language text in various forms such as data documents, articles, tweets, or emails. Techniques such as Natural Language Processing (NLP) and Deep Learning models like Recurrent Neural Networks (RNNs) and Transformer models (e.g., BERT, GPT) are frequently applied for tasks like sentiment analysis, text classification, named entity recognition, and machine translation. In PANTHEON, the document and text come from WP2 and T4.1 developed mechanisms. Document and text data will be managed by MongoDB.

Time Series Data: Time series data consists of observations collected over time, such as stock prices, sensor data, or weather data. Time series models like Autoregressive Integrated Moving Average (ARIMA), Seasonal Decomposition of Time Series (STL), and Long Short-Term Memory (LSTM) networks are commonly used for tasks such as forecasting, anomaly detection, and pattern recognition in time series data. In PANTHEON, processing time series data comes from IoTs (T4.1 developed mechanisms). Time Series Data will be managed by InfluxDB[15].

Graph Data: Graph data represents relationships between entities in a network, such as social networks, knowledge graphs, or citation networks. Graph Neural Networks (GNNs) are used to learn patterns and make

[15] https://www.influxdata.com

predictions on graph-structured data for tasks like nodes. In PANTHEON, processing graph data comes from IoTs (T4.1 developed mechanisms). Time Series Data will be managed by GraphDB[16].

Audio Data: Audio data represents sound waves and is commonly used in applications such as speech recognition, music classification, and sound event detection. Deep Learning models like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) are often applied to process audio data. In PANTHEON, processing graph data comes from IoTs (T4.1 developed mechanisms). Audio data will be processed by a file system and stored as files protected in a data lake mechanism.

Geospatial Data: Geospatial data includes information about locations on the Earth's surface, such as GPS coordinates, maps, satellite imagery, and geotagged data. Machine learning techniques are used for tasks like geospatial analysis, spatial prediction, and remote sensing applications. In PANTHEON, processing graph data comes from IoTs (T4.1 developed mechanisms). Geospatial data in PANTHEON will be managed by the extension to PostgreSQL, PostGIS[17].

4.3.3. Data analysis for machine learning (ML) models

Data analysis for machine learning (ML) models refers to the process of examining and understanding the data that will be used to train, validate, and test machine learning algorithms. It involves various techniques and methodologies to explore, clean, preprocess, and transform the data to make it suitable for training ML models. The data analysis for ML models is the subject of T4.2. We are presenting here a summary of data analysis as is it detailed by the outputs of T4.2. The overall placement of data analysis in the whole architecture is presented in Figure 14.

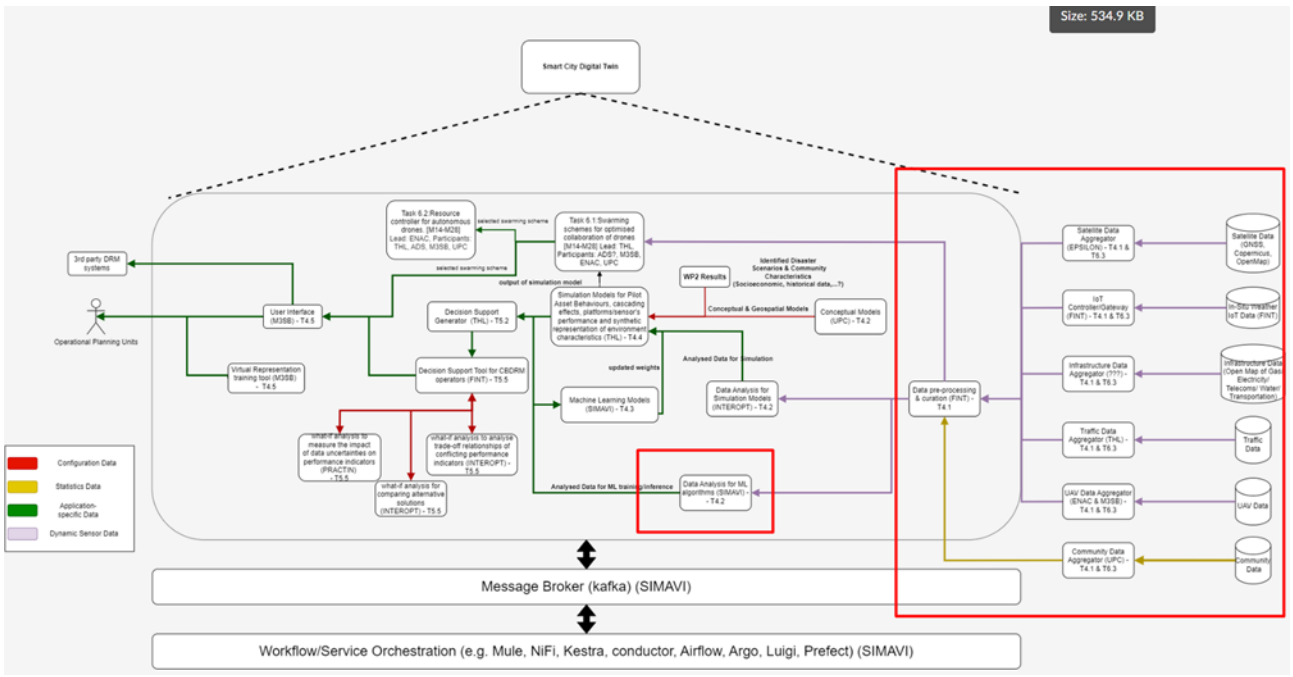


Figure 14 Data analysis in Pantheon Architecture

In the PANTHEON project, we plan to perform the following data processing:

[16] https://graphdb.ontotext.com
[17] https://postgis.net

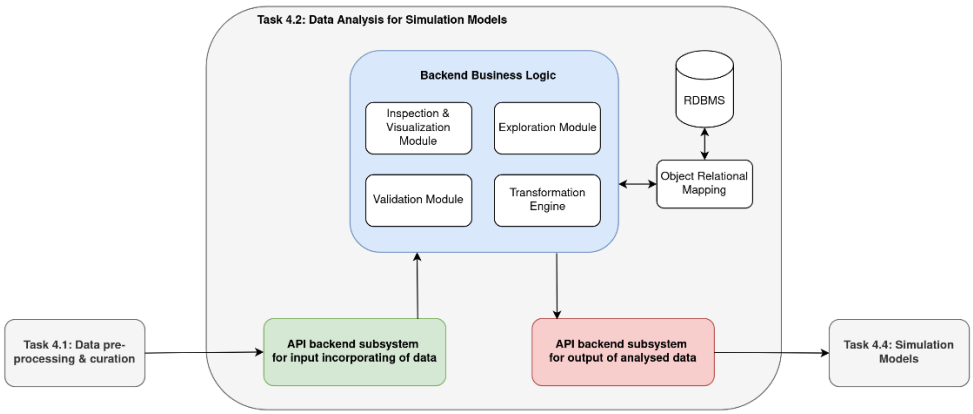
- **Data Collection**: Gathering the relevant data from various sources such as databases, files, APIs, and web scraping. In PANTHEON, data collection is part of T4.1
- **Data preprocessing:** Identifying and handling missing values, outliers, duplicates, and inconsistencies in the data. This step ensures that the data is accurate and reliable for analysis. Standardizing, normalizing, or scaling the features, handling categorical variables, and splitting the data into training, validation, and test sets. In Pantheon, data cleaning is part of T4.1
- **Exploratory Data Analysis (EDA):** Exploring the data through statistical summaries, visualization techniques (e.g., histograms, scatter plots, box plots), and correlation analysis to understand the relationships and patterns within the data. The main processing mechanisms will detect statistical elements like correlations, percentiles, and median. Based on the statistical elements, data will be filtered, to be used in training only data with real significance. In Pantheon, EDA is part of T4.2
- **Feature Engineering**: Selecting, transforming, or creating new features (variables) from the raw data to improve the performance of ML models. This may include feature scaling, encoding categorical variables, and generating new features through mathematical operations or domain knowledge. In Pantheon, feature engineering is part of T4.2
- **Model Evaluation and interpretation**: Assessing the performance of ML models using appropriate evaluation metrics (e.g., accuracy, precision, recall, F1-score, ROC-AUC) on the validation and test datasets. This step helps in comparing different models and selecting the best-performing one. In Pantheon, Model evaluation and interpretation is part of T4.2

Data analysis for ML models is an iterative process where various steps are revisited and refined based on the insights gained during the analysis. This iterative approach helps in improving the quality of the data and the performance of ML models.

4.4 DATA ANALYSIS FOR SIMULATION MODELS

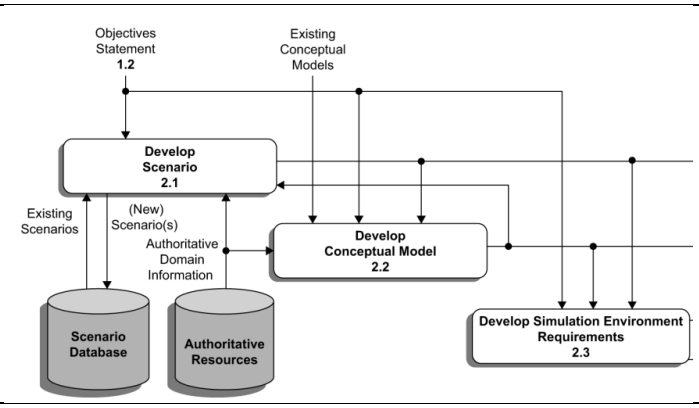
Service Name	Data inspection, visualization, exploring, transformation, and validation service				
Description	The service involves inspecting and overviewing the raw data aggregated, pre-processed, and curated by the PANTHEON project. Data includes Satellite Data, In-Situ Weather IoT Data, Infrastructure Data, Traffic Data, UAV Data, and Community Data. The service also involves understanding the structure of the data, exploring variables, and identifying any anomalies or irregularities. Furthermore, by visually and statistically summarizing the data main characteristics, the service offers creating visualizations, computing descriptive statistics, and identifying patterns to gain an initial understanding of the simulation data. Visual representations, such as charts, graphs, and dashboards, facilitate the communication of complex patterns and trends within the simulation data. An optional transformation feature capability is also provided, for performing simple conversions into similar formats for enhanced convenience. This may include simple data aggregations, creating derived variables, or applying mathematical transformations to enhance the usefulness of the data.				
Pilot	All pilots				
Scenario	All scenarios				
Module	T4.2 - Data Analysis for Simulation Models				
Submodule	N/A				
Input	**Input name**	**Input value type**	**Input data format**	**Description**	**From**

	Satellite Data	Boolean Integer String Float Timestamp	REST API (JSON) or CSV/Excel file import	Sets of satellite entities and relevant metrics, such as imagery and landmark data.	T4.1 – Data pre-processing and curation (FINT)
	In-Situ Weather IoT Data			Sets of weather entities and relevant metrics, such as temperature, wind speed, wind direction, atmospheric pressure, humidity, UV, visibility, precipitation, cloud coverage	
	Infrastructure Data			Sets of infrastructure entities and relevant details, such as landmarks and points of interest.	
	Traffic Data			Sets of traffic entities and metrics, such as points of interest, speed, and throughput variables.	
	UAV Data			Sets of flight entities and relevant metrics, such as flight route, duration, distance, direction vectors, height, points of interest.	
	Community Data			Sets of community entities and relevant metrics, such as residential population, age, gender, ethnicity distributions, social activities, events, points of interest, social media activity.	
Output	**Output name**	**Output value type**	**Output data format**	**Description**	**To**
	Same as input	Boolean Integer String Float Timestamp	REST API (JSON) or CSV/Excel file export	Same as input	T4.4 – Simulation Models (THL)
Algorithm	Exploratory Data Analysis process for uncovering patterns, relationships, and trends. Descriptive Statistics process for summarization and measuring mean, median and				

	averages. Correlation Analysis methodology for identifying mutual influences. Regression Analysis methodology for modelling impact changes among variables. Moving average time-series modelling methodology for analysing seasonality and cycles. Sensitivity Analysis for identifying parameters significant impact.
Technologies	PHP, R, Python, Java, Laravel, Spring Boot, PostgreSQL, PL/pgSQL, Bootstrap, HTML5, React.js, jQuery, Timescale, D3.js, Dygraphs, Plotly, Leaflet, MapLibre, OpenStreetMap, Open API.
Hardware Requirements	Minimum: 4 (v)CPU cores, 8GB RAM, 200GB storage Recommended: 8+ (v)CPU cores, 16+GB RAM, 960+GB storage
Schematic	

4.5 CONCEPTUAL MODELS

Service Name	Conceptual Models				
Description	Abstract representation of the systems and modules to be simulated. *Not really a running service*.				
Pilot	Attica forest fire, Attica earthquake, Viena heatwave and Viena Terrorist attack				
Scenario	Attica forest fire, Attica earthquake, Viena heatwave and Viena Terrorist attack				
Module	T4.2				
Submodule	Conceptual models				
Input	**Input name**	**Input value type**	**Input data format**	**Description**	**From**
	D3.3	SoA on Conceptual models	DOC/PDF	Existing conceptual models for digital twins	T3.3
	D3.6	Scenarios description	DOC/PDF	Details of the entities and processes of the scenarios	T3.6
Output	**Output name**	**Output value type**	**Output data format**	**Description**	**To**
	CM	Conceptual model	PNG/JPEG/drawio	Graph of the involved data and its relation, and state machine describing the processes involved in each scenario	T4.4
Algorithm	Methodology according to standard IEEE-1730-2022-DSEEP				
Technologies.	Technology agnostic				

Hardware Requirements	None
Schematic	

4.6 SIMULATION MODELS & DECISION SUPPORT GENERATOR

Service Name	Transportation sector based SCDT				
Description	An SCDT based on the transportation sector				
Pilot	All				
Scenario	Earthquakes or wildfires affect the road network, affecting multiple roads in the affected area and cascades into multiple junctions. Decision support on: (1) which routes historically are of highest risk to be delayed/blocked (analysis) (2) on routes that will be most affected during a disaster type (prediction) (3) shortest path to reach specific affected locations (4) highest risk (in terms of impact and likelihood of blockage) paths to reach specific location				
Module					
Submodule	Import module				
Input	**Input name**	**Input value type**	**Input data format**	**Description**	**From**
	Road network nodes (junctions)	• node_name • road • connected_node • label (example: AL1000 A38 between A513 and A5127 (AL1000))	csv file	Node Creator	
	Road network routes (edges - roads)	• :START_ID • edge_name • road • :END_ID • :TYPE • date • hour	csv file	**Relationship Creator**	

		• flow • impactLevel • type			
Output	**Output name**	**Output value type**	**Output data format**	**Description**	**To**
	CSV dataset (2 files)	**Nodes.csv Relationships.csv**	**csv**	Creates the node and relationship files, required by Graph input builder module	Graph input builder
Algorithm	Node Creator • Read a line from the input file (skipping csv header) • Isolate the road names stated in the Link Description and store as unique nodes in output file using regex. • Write unique stored nodes to the output node csv file. Relationship Creator • Read a line from the input file & keep only data good quality entries (provided at scale in dataset) • Check for missing flow variable and store flow for calculating the impact and type metrics. • Every four entries, store date and time periods, and model the four 15-minute entries into an hourly entry. • After the input file's traversing is done, calculate impact level for each hourly entry and determine its type. • Write fully formed relationships to output relationship csv file.				
Technologies	Python, Neo4J, Javascript, TypeScript, FireBase API				
Submodule	Earthquake simulation input				
Input	**Input name**	**Input value type**	**Input data format**	**Description**	**From**
	Earthquake data	Geographic Data: • Location (lat., long.) • Magnitude of earthquake • Depth of earthquake. • Fault Line location. • Historical Seismic Activity Data	REST POST request		

	Building / infrastructure data	• Infrastructure Sensitivity • Building Codes and types • Structural Vulnerability (e.g. age, etc)			
	Response resources	- Locations of firefighting resources - Location of water sources • Infrastructure Exposure and Vulnerability Data (e.g. age, type etc.)	REST POST request		
Algorithm	• Convert earthquake data to impact scale • Convert response resources exposure to vulnerability scale • Convert historical data to threat chance (likelihood) • Calculate risk per area based on nodes affected.				
Technologies	Javascript, Neo4J, Python (TBD)				
Submodule	Wildfire simulation input				
Input	**Input name**	**Input value type**	**Input data format**	**Description**	**From**
	Land data	- Vegetation Types - Vegetation Distribution - Vegetation distance from nodes - Topographic slopes - Topographic elevation - Natural barriers	REST POST request		
	Wildfire data	- Wind Speed - Wind Direction - Temperature - Humidity - Fire History Data (causes, durations, extents, and areas impacted)			

	Response resources	- Locations of firefighting resources - Location of water sources - Infrastructure Exposure and Vulnerability Data (e.g. age, type etc.)			
Algorithm	• Convert land data to threat change and vulnerability scale • Convert response resources exposure to vulnerability scale • Convert wildfire data to impact scale • Convert historical data to threat chance (likelihood) • Calculate risk per area based on nodes affected.				
Technologies	Javascript, Neo4J, Python, C/C++ (TBD)				
Submodule	Heatwave simulation input				
Input	**Input name**	**Input value type**	**Input data format**	**Description**	**From**
	Heatwave data	- Temperature levels - Duration - Historical Temperature Data. - Humidity - Timing.	REST POST request		
	Area data	- Energy Demand - Population in affected area - Public Health Infrastructure capacity (beds)	REST POST request		
	Infrastructure exposure	- Amount of vulnerable populations (% or absolute num.) - Building vulnerability/exposure (Google Solar API) - Green space data per sq/m.	REST POST request		
Algorithm	• Convert heatwave data to threat change and impact scale • Convert infrastructure exposure to vulnerability scale				

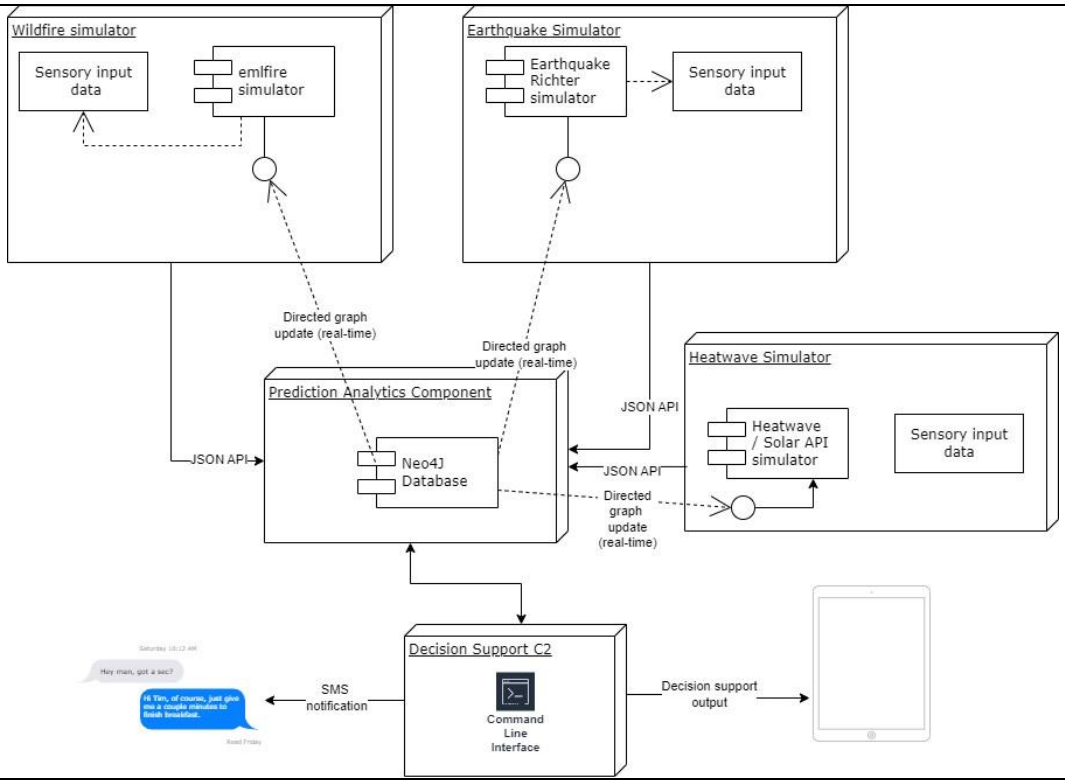
<table>
<tr><td></td><td colspan="5">• Convert area data to impact and vulnerability scale
• Calculate risk per area based on population affected.
○ Express population affected as first responder infrastructure risk.
○ Express population affected as infrastructure impact.</td></tr>
<tr><td>Technologies</td><td colspan="5">Javascript, Neo4J, Google Solar API (TBD)</td></tr>
<tr><td>Submodule</td><td colspan="5">Graph network builder</td></tr>
<tr><td rowspan="3">Input</td><td>Input name</td><td>Input value type</td><td>Input data format</td><td>Description</td><td>From</td></tr>
<tr><td>Road network information (nodes file)</td><td>List of traffic data, each line including:
- Junction name
- Junction ID
- Connected junction 1
- Connected junction 2
Label (type of junction)</td><td>File (CSV)</td><td>Graph node builder</td><td></td></tr>
<tr><td>Traffic Flow information (relationships file)</td><td>List of junction relationships (connections with flow):
- Start ID
- Edge name
- Road
- End ID
- Type
- Date
- Hour
- Flow
- Impact level
Type (bad/good)</td><td>File (CSV)</td><td>Flow (edge/weight) population</td><td></td></tr>
<tr><td rowspan="3">Output</td><td>Output name</td><td>Output value type</td><td>Output data format</td><td>Description</td><td>To</td></tr>
<tr><td>Neo4J model</td><td>- Neo4J database file</td><td></td><td>This module imports each modified and processed graph into a Neo4j graph database. It creates the new database from data in CSV files</td><td></td></tr>
<tr><td>Graph metrics</td><td>Betweeness infrastructure influence</td><td>csv</td><td>Identifies infrastructure nodes that serve as critical bridges within</td><td></td></tr>
</table>

				the network. High betweenness centrality nodes, if impacted, could significantly disrupt the flow of resources or information, highlighting priority areas for protection or rapid response.	
		Closeness infrastructure influence	**csv**	Measures how close an infrastructure node is to all other nodes, indicating how quickly resources or information can be distributed from or to that node. This metric could help prioritize nodes for faster disaster response or recovery efforts.	
		Eigenvector influence	**csv**	Highlights nodes that are connected to other highly connected nodes, suggesting infrastructures that, if compromised, could have a widespread impact on the network due to their influential position	
	Risk Chains	**Shortest Path Analysis**	**csv**	Determines the shortest paths or least risk chains between nodes, useful for planning evacuation routes or deploying emergency services to affected areas. It can also identify alternative routes if primary paths are compromised.	
		Impact Propagation Paths	**csv**	Simulates how a disaster's impact spreads through the network, identifying chains of interconnected nodes that are at risk of	

				cascading failures. This analysis can help in preemptive actions to protect or reinforce vulnerable points in the network.	
		Resilience Analysis	**csv**	Evaluates the network's ability to maintain connectivity and function under stress, identifying nodes whose removal (due to disaster impact) would most severely affect the network's resilience.	
Submodule	Decision Support Generator				
Input	**Input name**	**Input value type**	**Input data format**	**Description**	**From**
	Graph metrics	**Betweeness infrastructure influence**	**csv**	Identifies infrastructure nodes that serve as critical bridges within the network. High betweenness centrality nodes, if impacted, could significantly disrupt the flow of resources or information, highlighting priority areas for protection or rapid response.	
	Traffic Flow information (relationships file)	**Closeness infrastructure influence**	**csv**	Measures how close an infrastructure node is to all other nodes, indicating how quickly resources or information can be distributed from or to that node. This metric could help prioritize nodes for faster disaster response or recovery efforts.	
	Output name	**Eigenvector influence**	**csv**	Highlights nodes that are connected to other highly connected nodes, suggesting	

				infrastructures that, if compromised, could have a widespread impact on the network due to their influential position	
	Risk Chains	**Shortest Path Analysis**	**csv**	Determines the shortest paths or least risk chains between nodes, useful for planning evacuation routes or deploying emergency services to affected areas. It can also identify alternative routes if primary paths are compromised.	
Output	**Curated List (metrics)**	**Machine Learning-Enhanced Prioritization**		Developing algorithms that prioritize communications based on the criticality of the information	
		Optimization of Response Strategies		Using supervised learning to optimize response strategies based on simulations of different scenarios. This approach can help in determining the most effective actions First Responders can take to mitigate the impact	
	Curated List (chains)	**Resource Allocation Optimization**		Applying machine learning to optimize the allocation of resources (e.g., personnel, equipment) based on the severity and location of infrastructure risks.	
		Anomaly Detection in Cascading Effects		Employing machine learning algorithms to detect anomalies in the patterns of cascading effects. This can highlight unusual or unexpected risk changes	
		Prioritized Response Lists		Utilizing machine learning to classify risk	

				chains by their total risk score and the length of the path. This classification can help in identifying the most critical chains that could lead to severe cascading effects, enabling prioritized attention to those chains.	

Algorithm	**Centrality Metrics for Infrastructure Nodes** - Betweenness Centrality Calculation: Brandes' algorithm to calculate the number of shortest paths passing through each node. - Closeness Centrality Measurement: Dijkstra's algorithm for each node is calculated as the inverse of the sum of the shortest path distances from a node to all other nodes in the graph. - Eigenvector Centrality Computation: Power Iteration method to compute the centrality of a node based on the centrality of its neighbours. - PageRank Calculation: PageRank algorithm, adapted for networks. **Risk Chains and Paths** • Shortest Path Analysis: Algorithm: Dijkstra's or A* algorithm to calculate the shortest paths between nodes, factoring in weights that represent risk levels. • Impact Propagation Paths: Monte Carlo simulations or Agent-based modelling.
Technologies	Javascript, Neo4J
Hardware Requirements	TBD
Schematic	

Each node is modeled using the following properties:

Property	Description
node_name	The node's displayed name. Also used as the node's ID.
road	The road which links the node with his pair in the same junction.
connected_node	The node paired with our current node in the same junction.
label	The label used to describe nodes.

Each relationship is modeled using the following properties:

Property	Description
:START_ID	The name of the first node in the current junction.
edge_name	The name of the junction.
road	The road linking the two nodes in this junction.
:END_ID	The node paired with :START_ID node in the same junction.
:TYPE	Type label for relationships. (e.g. a binary value for Road_Congestion)
date	Date of travel.
hour	(1-24) Hour of travel during the day.
flow	Average flow of the four 15-minute entries combined into this hourly entry.
impactLevel	Impact level for the current hourly entry, as calculated from impact assessment algorithm.
type	{good, bad}. An entry is good iff it is proportionally fair to the other entries for the same edge.

4.7 DECISION SUPPORT & WHAT-IF ANALYSIS

4.7.1 DECISION SUPPORT TOOL

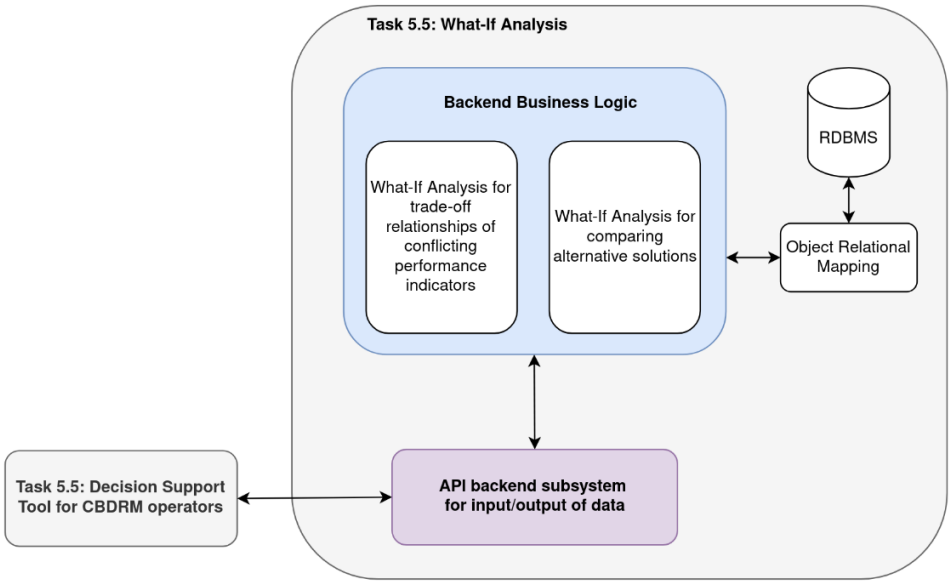
Service Name	Decision support for CBDRM operators
Description	Providing recommendations for actions to the CBDRM operators
Pilot	Athens; Vienna
Scenario	1-Athens wildfire; 2-Athens earthquake; 3-Vienna heatwave; 4-Vienna cyber-terrorism
Module	T5.5 – Decision support tool for CBDRM operators

Submodule					
Input	**Input name**	**Input value type**	**Input data format**	**Description**	**From**
	Multiple scenarios	N/A	REST API Json, csv	Multiple disaster and impacts scenarios generated by the DSS generator, according to the simulations	Decision support generator (T5.2)
Output	**Output name**	**Output value type**	**Output data format**	**Description**	**To**
	Decision recommendations	N/A	REST API Json/csv	Recommendations on the actions of the CBDRM operators	User interface (T4.5)
Algorithm	TBD				
Technologies	TBD				
Hardware Requirements	TBD				
Schematic	TBD				

4.7.2 WHAT-IF ANALYSIS

Service Name	What-if Analysis service				
Description	The service involves PANTHEON's scenario and sensitivity analysis, exploring the potential outcomes of different scenarios by altering one or more input variables. Its purpose is to assess the impact of changes in certain factors on the final results, helping PANTHEON's decision-makers understand the sensitivity of a decision to variations and conflicts in key parameters and performance indicators. This analysis is particularly useful in evaluating different possibilities, exploring the consequences of different scenarios, making informed decisions, and planning for uncertainty.				
Pilot	All pilots				
Scenario	All scenarios				
Module	T5.5 – What-if analysis to analyse trade-off relationships of conflicting performance indicators. T5.5 – What-if analysis for comparing alternative solutions.				
Submodule					
Input	**Input name**	**Input value type**	**Input data format**	**Description**	**From**
	Simulation and Decision Support Data	Boolean Integer String Float	REST API (JSON) or CSV/Excel file import	Sets of information generated, processed, and analysed during the simulation and the	T5.5 – Decision Support Tool for

		Timestamp		subsequent decision-making process, such as physical quantities, states of entities, (key) performance indicators, efficiency measures, validation and calibration data, and historical records.	CBDRM operators (FINT)
Output	**Output name**	**Output value type**	**Output data format**	**Description**	**To**
	Same as input	Boolean Integer String Float Timestamp	REST API (JSON) or CSV/Excel file export	Same as input	T5.5 – Decision Support Tool for CBDRM operators (FINT)
Algorithm	Parameter variation process for modifying input parameters systematically, randomly creating different scenarios, adjusting numerical values, introducing new conditions, and changing assumptions. Statistical techniques, hypothesis testing, and confidence intervals for assessing the significance of observed differences. Risk Analysis methodology for assessing the risks associated with different scenarios by considering uncertainties and variations in input parameters, as well as analysing the likelihood and consequences of various outcomes. Comparative Analysis methodology for comparing the performance or outcomes of different scenarios and for identifying the most favourable or robust options. Feedback loop iterative process where insights gained inform further adjustments to scenarios for improving the quality of decision-making over time. Interactive Exploration process for creating an interactive environment where decision-makers can explore scenarios dynamically, adjusting parameters and observing the impact on outcomes.				
Technologies	PHP, R, Python, Java, Laravel, Spring Boot, PostgreSQL, PL/pgSQL, Bootstrap, HTML5, React.js, jQuery, Timescale, D3.js, Dygraphs, Plotly, Leaflet, MapLibre, OpenStreetMap, Open API.				
Hardware Requirements	Minimum: 4 (v)CPU cores, 8GB RAM, 200GB storage Recommended: 8+ (v)CPU cores, 16+GB RAM, 960+GB storage				

Schematic	

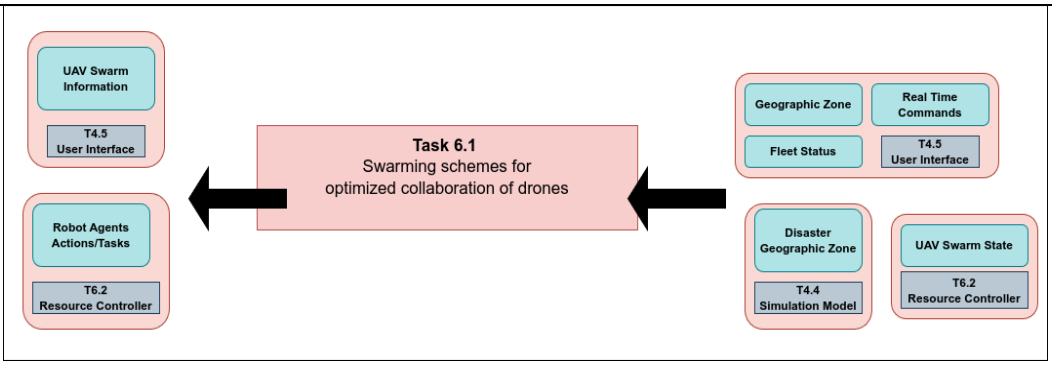
4.8 UAV SWARMING SCHEMES

Service Name	UAV Swarming Scheme				
Description	This service offers a module for creating UAV swarming techniques for various operational contexts. Libraries containing features such as centralized/decentralized control, leader-follower dynamics, task allocation and adaptive behavior, along with the generation of swarm commands and actions should be implemented for the coordinated operation of multiple UAVs for target effectiveness and achievement.				
Pilot	Vienna, Athens				
Scenario	Man made disaster, Wildfire, Earthquake				
Module	Task 6.1 - Swarming schemes for optimized collaboration of drones				
Submodule	N/A				
Input	**Input name**	**Input value type**	**Input data format**	**Description**	**From**
	Mission definition	Scenario description, type of geographic zone, etc	text	Algorithm conception is based on mission definition	See scenario description in D3.6
	Fleet Description	A Struct that will describe for each UAV - Type of UAV [Integer/String] - Flight Time [Float]	REST API MQTT ROS2 Subscriber	PANTHEON UAV fleet information regarding each UAV's capabilities: flight	User Interface [T4.5] or see scenario description in D3.6

		- Attached Sensors [Array of Strings/Integers] -Maximum Velocity & Acceleration [Float] - ID [Integer]		dynamics, payload.	
	Real Time Commands	A struct that will describe - Action [String] - POI Location XYZ [Float] -Priority [Integer/String]	REST API MQTT ROS2 Subscriber	Real Time action updates from user regarding swarm	Resource controller [T6.2]
	Swarm Information	A struct that will be describe -Pose (XYZ and Quaternion) of the neighbours [Float] - Current Action [String]for each robot agent in swarm	ROS2 Publisher MQTT REST API	Information about robotic agent state	Resource Controller [T6.2]
	Geographic zone of disaster, generated from digital twin model	Array of POI Location XYZ [Float]	REST API MQTT ROS2 Subscriber KAFKA	Geographic coordinates of the boundaries of predicted disaster	Simulation Models [T4.4]
	No fly zones, obstacles, operational constraints	Array of POI Location XYZ [Float] Constraint[string]/[float]	REST API MQTT ROS2 Subscriber KAFKA	Geographic coordinates, time slots	resource controller [T6.2]
Output	**Output name**	**Output value type**	**Output data format**	**Description**	**To**
	Robot Agent Actions	Actions : { XML that will contains Action [String], Location	ROS2 Publisher KAFKA	Actions in a defined Language (XML) for each agent	Resource Controller (T6.2)

		[XYZ] and macros for various tasks [String] }			

	Swarm Information	State: {
		State [string] and Status [Vector of Strings] for each UAV in swarm,
		Current Position [float] of each UAV agent, remaining flight time, RSSI etc.
		}
			ROS2 Publisher
			MQTT
			REST API
			KAFKA
				Inform user about swarm agents' state as well as visualize information about current agent position and status	User Interface [T4.5] & Resource Controller (T6.2)

Algorithm	the objectives or tasks to be accomplished determine the algorithms to be used. Based on these objectives, the algorithm calculates safe trajectories and control inputs for each UAV, considering factors like obstacle avoidance, energy constraints, and mission priorities. Throughout the mission, the algorithm continuously updates the UAVs' trajectories and control commands based on real-time sensor data and feedback from neighbouring UAVs, ensuring coordination, synchronization, and adaptation to dynamic environmental conditions. Finally, the algorithm terminates the mission once the objectives are achieved.				
Technologies	Python, C++, ROS2				
Hardware Requirements	*Recommended*: 8+ (v)CPU cores, 16+GB RAM, 960+GB storage				
Schematic					

4.9 RESOURCE CONTROLLER

Service Name	Resource controller for drone swarms
Description	The service offers a module for orchestrating and managing UAV swarms in various operational contexts. This service enables control of multiple UAVs through advanced allocation and coordination algorithms, optimizing their collective behavior to achieve mission objectives efficiently.
Pilot	Vienna (man made disaster) & Athens (wildfire and earthquake)

Scenario	Vienna (man made disaster) & Athens (wildfire and earthquake)				
Module	Task 6.2 – resource controller for autonomous drones				
Submodule					
Input	**Input name**	**Input value type***	**Input data format****	**Description**	**From****

	Mission	Mission Abstraction:
		{
		XML description [String]
		}
			REST API
			MQTT
			ROS2 Subscriber
			KAFKA
				Mission that needs to be executed in a high-level approach	User Interface [T4.5]

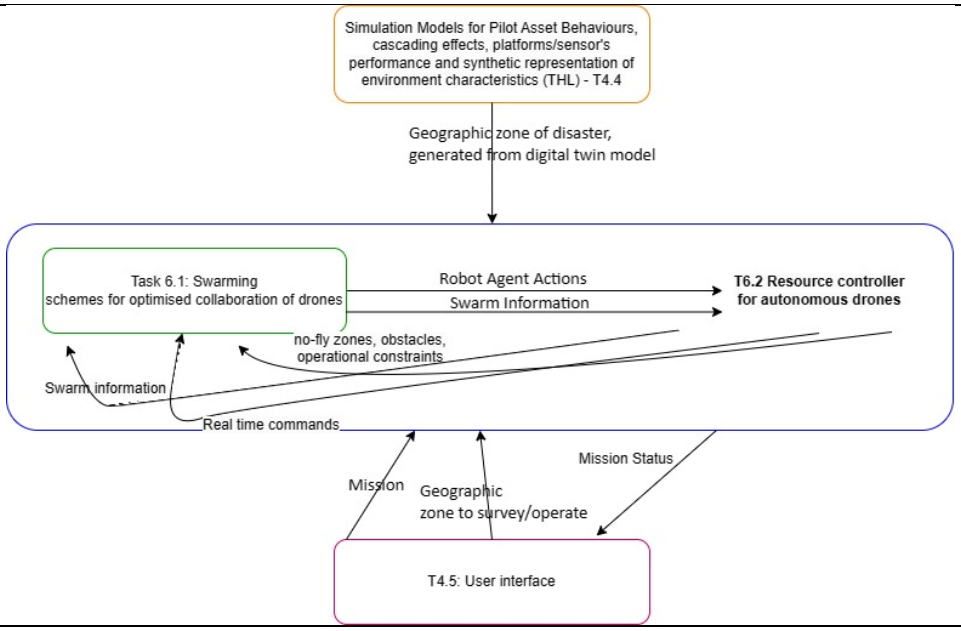
	Robot Agent Actions	Actions : {
		XML that will contains Action [String], Location [XYZ] and macros for various tasks [String]
		}
			ROS2 Publisher
			KAFKA
				Actions in a defined Language (XML) for each agent	T6.1

	Swarm Information	State: {
		State [string] and Status [Vector of Strings] for each UAV in swarm,
		Current Position [float] of each UAV agent, remaining flight time, RSSI etc.
		}
			ROS2 Publisher
			MQTT
			REST API
			KAFKA
				Inform user about swarm agents' state as well as visualize information about current agent position and status	**T6.1**

	Geographic zone to survey/operate	Array of POI Location XYZ [Float]	REST API MQTT ROS2 Subscriber KAFKA	Geographic coordinates of the boundaries of the zones, including buffer zones/safety	User Interface [T4.5]
	Geographic zone of disaster, generated from digital twin model	Array of POI Location XYZ [Float]	REST API MQTT ROS2 Subscriber KAFKA	Geographic coordinates of the boundaries of predicted disaster	Simulation Models [T4.4]
Output	**Output name**	**Output value type**	**Output data format**	**Description**	**To**
	Real Time Commands	A struct that will describe - Action [String] - POI Location XYZ [Float] -Priority [Integer/String]	REST API MQTT ROS2 Subscriber KAFKA	Real Time action updates from user regarding swarm	Swarming schemes [T6.1]

	Swarm Information	A struct that will be describe -Pose (XYZ and Quaternion) of the neighbours [Float] - Current Action [String]for each robot agent in swarm	ROS2 Publisher MQTT REST API KAFKA	Information about robotic agent state	Swarming schemes [T6.1]
	No fly zones, obstacles, operational constraints	Array of POI Location XYZ [Float] Constraint[string]/[float]	REST API MQTT ROS2 Subscriber KAFKA	Geographic coordinates, time slots…	Swarming schemes [T6.1]

	Mission status	Mission Report:
		{
		XML description [String]
		}
			REST API
			MQTT
			ROS2 Subscriber
			KAFKA
				Mission status in a high-level approach	User Interface [T4.5]

Algorithm	T6.2 is a state machine acting as an outer loop for fleet management (T6.1).				
Technologies	Python, C++, ROS2, MQTT, KAFKA (TBD)				
Hardware Requirements	*Recommended*: 8+ (v)CPU cores, 16+GB RAM, 960+GB storage				
Schematic					

4.10 USER INTERFACE

The objective of the User Interface component is to provide a visualization of the results of simulations carried out by the backend of the PANTHEON platform. Considering the large areas that will need to be considered (an entire city or region) and given the significant amount of information to be represented, this representation will be offered in the form of a GIS (Geographic Information System) providing 2D maps of the region or city affected by the crisis, and including a whole series of operational information that can help the operator better understand the situation. Beyond an improved understanding of a large-scale crisis, the tool should assist the operator in better understanding the effects or impacts of the parameters being represented, with the objective of enabling them to anticipate the future evolution of the situation and make decisions accordingly.

The main structure of the application will therefore be a 2D map with a base map, including all the information useful to the user. A series of fields will be organized to allow interaction with the data and interpretation of the simulation results provided by the PANTHEON backend.

Figure 15.a illustrates the general organization of the visual interface, while Figure 15.b represents the first action of the user, which is to log in with their username and associated password. Each user will be assigned a role essentially related to their role within the rescue teams, which will determine the filling of the interaction fields of the application. For example, a member of firefighting teams will only see information related to firefighting activities in the drop-down menus, while a member of medical teams will have their information limited to hospitals or any other capacities related to healthcare.

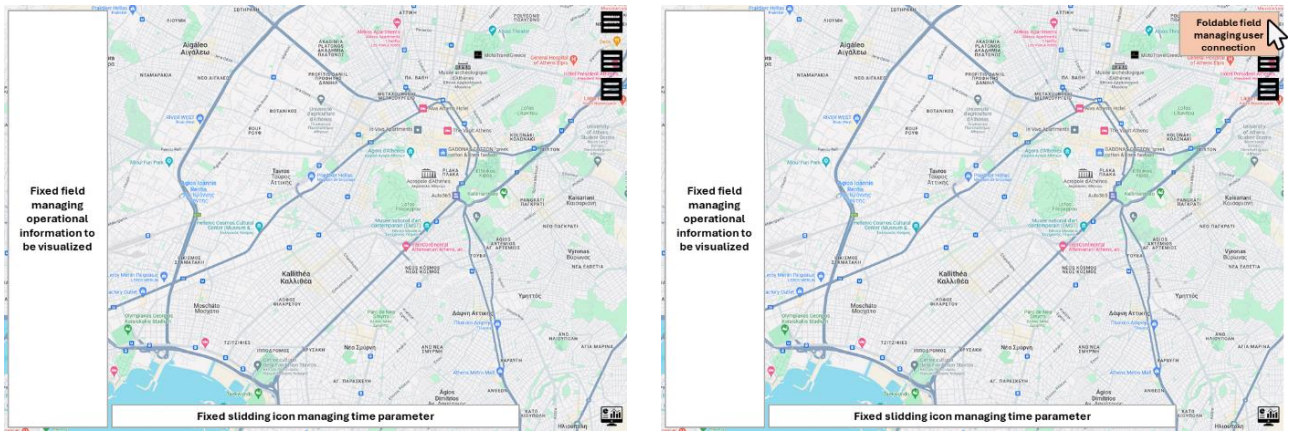


Figure 15 (a) – left Illustration of User Interface lay-out and (b) – right initiation of the user's session via the credential introduction.

User management will be handled by the backend of PANTHEON platform, which will send a token upon connection, which token will then be used to secure exchanges between the backend and the frontend. Each dataset will be assigned a code, with each user being assigned a list of codes to which they will have access. It will be the responsibility of the backend to sort the data that can be sent, with this information being integrated into the token at the time of its creation. A user will be assigned a default role but may be assigned multiple roles to access a wider range of information.
The user will also be assigned a security class that will filter accessible information. Each dataset will be assigned a class, with users not having a sufficient security level not receiving the information. The management of the security class will once again be handled by the backend of PANTHEON platform and integrated into the token.

The list of information accessible to a particular role or user can be modified by a particular user with the administrator role through a specific field available in their own interface. This administrator will be responsible for creating users and assigning them a security class (which they may eventually modify), as well as deleting them. The creation of roles and data types is part of the internal functioning of the PANTHEON application and is not directly accessible to the administrator.

A fixed field located on the left will contain a series of nested dropdown menus that will allow selection of the information to be visualized. These will be grouped by role for data directly related to assets or team types (for example, firefighting teams, police, healthcare teams, drone deployment teams) or by data type (for example, critical infrastructures, sociological data). The definition of the list of available data and its nesting is the responsibility of the backend of PANTHEON platform, which sends (and possibly updates) this information upon connection. Figure 16 (a) and (b) illustrate two ways of presenting community or sociological data mainly related to areal densities.

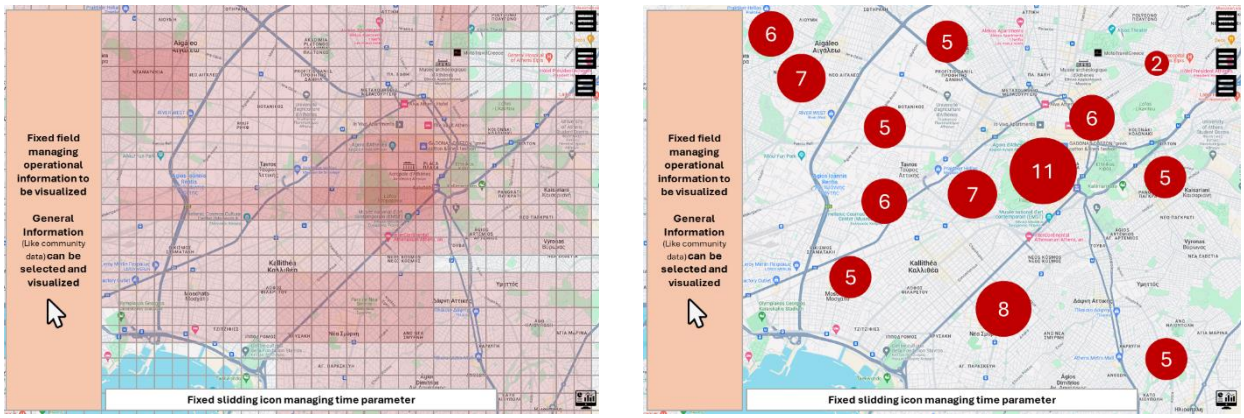


Figure 16 (a) - left and (b) - right: Different way to present community or sociological data.

The fixed field on the left thus allows the user to select crisis management assets or critical infrastructure and visualize them on the map (see Figure 17.a). A deployable field located on the right allows for a list of assets or infrastructure that can be added to the map (For example, adding a firefighting station) by selecting them and dragging them onto the map (see Figure 17.b).

Some crisis management assets will require some configuration when they are created. For example, the deployment base of a drone system can be implemented in the map and must be configured by indicating the targeted deployment zone. These parameters may need to be updated in the case of a simulation involving another type of crisis, or an identical crisis but taking place in another location. The application will accept or not accept the deployed system configuration parameters based on feasibility assessment done by the backend. For example, the application may refuse to deploy a drone system consisting of a single drone over an area of 100km² for which the user has requested a revisit every 10 minutes, inviting the operator to increase the number of drones available in this deployment station.

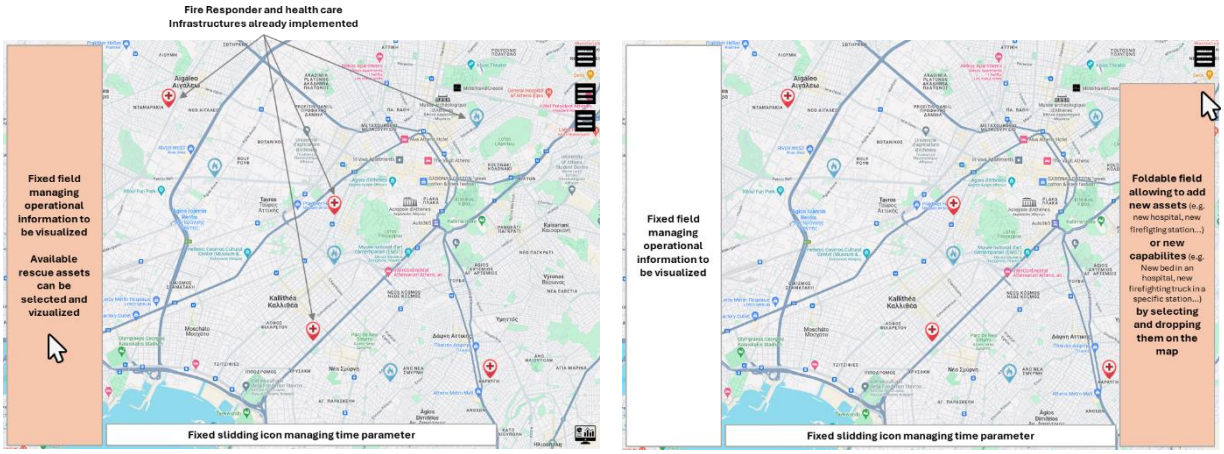


Figure 17 (a) left - Visualization of crisis management assets and (b) right - activation of the window allowing to create new assets.

By clicking on a crisis management asset or critical infrastructure visible on the map, the user will be able to activate a foldable window containing all the information characterizing the asset or infrastructure and its capabilities (see Figure 18.a). In the case of crisis management assets, this window will also allow for the modification of their capabilities (for example, by increasing the total number of beds in a specific hospital).

This window can also be used to select areas on the map (thus not selected via the fixed left window) and gather information about them (for example, knowing the number of floors in a specific building, see Figure 18.b).

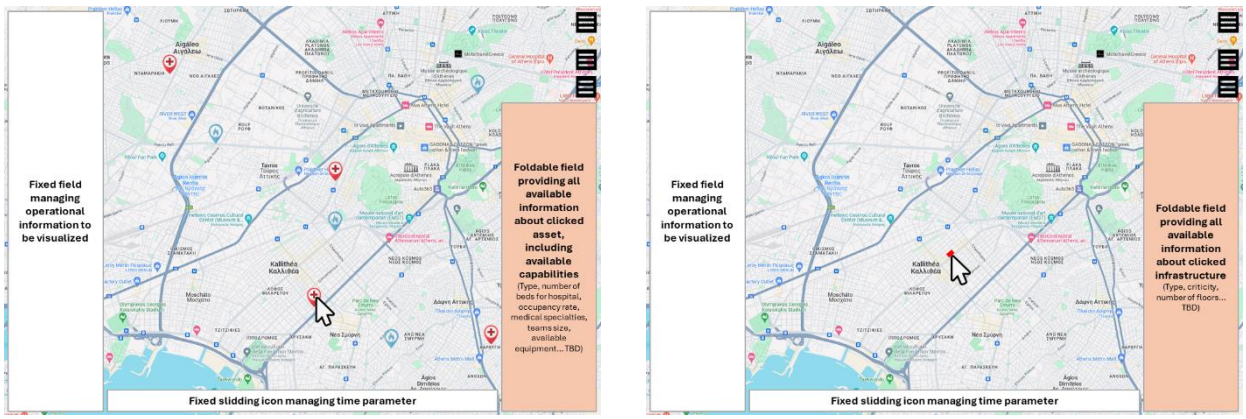


Figure 18 (a) left - Analysis of the characteristics of an asset or infrastructure visible on the map and (b) – right Retrieval of information on an area of the map.

A second deployable field located on the right (see Figure 19.a) will allow configuring the crisis to be simulated by identifying the type of crisis, its magnitude, position, extent, etc. The parameters will involve

initiating the crisis, with the definition of its temporal and geographical evolution being the responsibility of the PANTHEON backend.

Once a crisis is activated, new dropdown menus will appear again in the fixed left window to select and visualize certain effects and impacts of the crisis (For example, destroyed buildings, extent of a flood, fire fronts, and areas destroyed by fires) (see Figure 19.b).

The activation of a crisis, and its temporal evolution, will impact all the data and may change their appearance on the screen or the data available in the right window. For example, during the propagation of a flood, the icon of a power plant may be modified to indicate its shutdown at a given time. By clicking on a visible impact of the crisis on the map (see Figure 19.c), the fixed right window will reappear and provide a series of information about the type of infrastructure and the damage suffered.

Similarly, it will be possible via the fixed left window to obtain new information directly related to the crisis, such as the movement of disaster-affected people and the saturation of communication paths (see Figure 19.d).

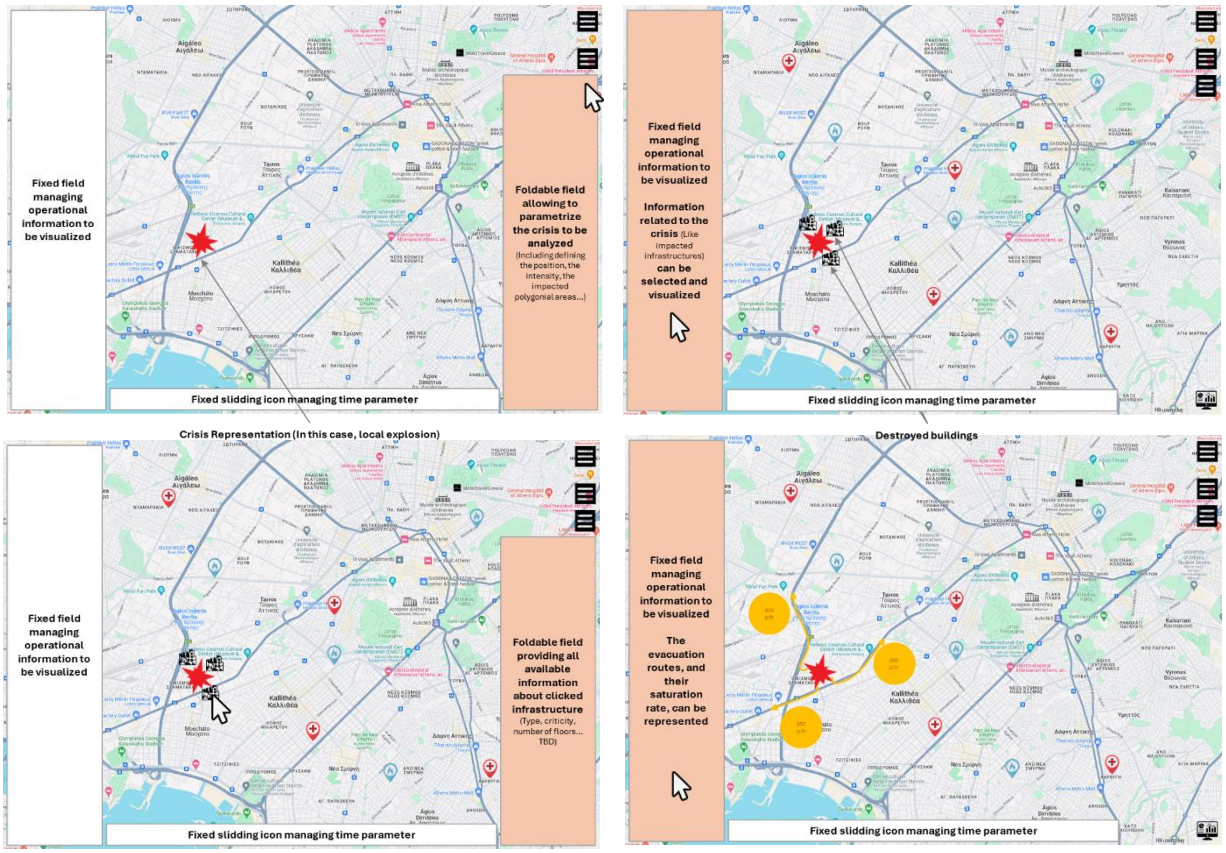


Figure 19 (a) – top left Parameterization of a crisis to be simulated, (b) – top right Visualization of the damage caused by the crisis, (c) bottom left Recovery of more precise information on certain damage and (d) bottom right Visualization of refugee flows following the initiation of the crisis.

Once a crisis has been initiated and configured, selecting a crisis management asset on the screen will still bring up the information window on the right of the interface, but the latter will see new features appear. The available capabilities and equipment will still be present, but it will be possible to deploy them on the map in order to better understand the impact that their action could have on the unfolding of the crisis (see

Figure 20). Each deployment must be assigned a given initial time in relation to the initiation of the crisis, to allow the backend to calculate the temporal effects associated with each event.

Figure 20 Deployment on the map of emergency resources, from crisis management assets.

All the data mentioned below, and their evolution calculated by the backend, will be characterized by a temporal tag given according to the initiation of the crisis. This will allow the PANTHEON backend to calculate all parameters while ensuring their temporal synchronization. A temporal management icon will be present at the bottom of the application to allow the user to navigate back and forth in time (see Figure 21), to better understand the temporal evolution of the information presented on the screen.

Deploying crisis management assets or rescue devices may influence the crisis outcomes. Therefore, all parameters characterizing the evolution of the crisis will be continuously refreshed to account for the latest decisions, and the user will have the ability to travel into the future and the past to assess the impact and relevance of the decisions made.

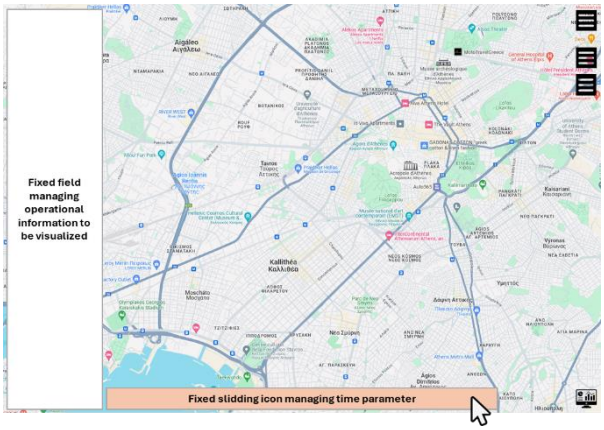


Figure 21 Position of the temporal management icon.

Finally, the application will make it possible to view a set of information related to the crisis and presented in a synthetic manner through a dashboard that can be activated using an icon located at the bottom right (see Figure 22). This dashboard will display a series of macroscopic information linked to the final result of the crisis but can also present specific information with respect to a specific moment. The time management icon will therefore always be accessible, to allow the user to obtain an inventory at a given time.

Figure 22 Activation of the crisis management dashboard.

4.10.1 VIRTUAL REPRESENTATION TRAINING TOOL (M3SB)

The Virtual Representation Training Tool is considered as fully integrated in the User Interface component, and some of its capabilities have been already described. For the operator, this involves implementing new strategies and verifying their impact on the unfolding of the crisis. This possibility should allow it to test its decision-making and improve its reactions to large-scale crises. The dashboard will provide a means of quantifying the positive or negative impacts of the actions undertaken and will therefore aim to improve conditioned reflexes. Furthermore, being able to visualize the actions of other emergency services will allow each operator to learn to better integrate their action with those of others and will improve general coordination.

Service Name	User Interface / Learning tool				
Description	User interface concretized by a GIS visual interface and allowing the operator to visualize the results of PANTHEON platform and to interact with the parameters of the simulation				
Pilot	Attica				
Scenario	Wildfire				
Module	User Interface / Learning tool				
Submodule	Wildfire User Interface				
Input	**Input name**	**Input value type**	**Input data format**	**Description**	**From**
	Position of flame fronts, with their expected movement (including speed) and historic	Text	GEOJSON	Exact position, history, movement, speed for all flames fronts	PANTHEON Platform
	Infrastructures to be impacted function of time	Text	GEOJSON, JSON	Give the infrastructures (road, building…) that will be destroyed or unreachable with respect to time	PANTHEON Platform
	Characterization of related areas vs.	Text	GEOJSON	Give some characteristic in terms of surface	PANTHEON Platform

	mobility capabilities			status, describing how (and at what speed) any vehicle or pedestrian could move on these surfaces	
	Modification of evacuation strategy **(LEARNING)**	Dynamic interaction with visual interface, and input boxes to implement number	Dynamic interaction with visual interface	Provide privileged paths to evacuate disaster victims	Operator
Output	**Output name**	**Output value type**	**Output data format**	**Description**	**To**
	Visualization of flame fronts, with clear information about future movement and impacted areas	Visualization	KML/KMZ (Vector data)	Providing clear information illustrating how the situation will evolute	Operator
	Privileged paths to be considered for population evacuation	Text	GEOJSON	Give proposal in terms of evacuation paths, allowing the PANTHEON platform to recalculate the population movements	PANTHEON Platform
Algorithm	TBD				
Technologies	Geoserver (TBD) and visual interface based on QT				
Hardware Requirements	CPU for the HMI itself, and geoserver installed on a delocalized server.				
Schematic	TBD				

Service Name	User Interface / Learning tool				
Description	User interface concretized by a GIS visual interface and allowing the operator to visualize the results of PANTHEON platform and to interact with the parameters of the simulation				
Pilot	Austria				
Scenario	Heatwave				
Module	User Interface / Learning tool				
Submodule	Heatwave User Interface				
Input	**Input name**	**Input value type**	**Input data format**	**Description**	**From**
	Dynamic occupancy level of each hospital	Text	JSON	Number of beds still available vs. their	PANTHEON Platform

				characteristics as a function of time	
	Result resolution	Text	JSON	General parameters defining the level of resolution to be applied on the results (e.g. type of hospital capabilities to be considered, number of digits for results)	PANTHEON Platform
	List of hospital and their general characteristics	Text	JSON	Give the total number of beds, list of health specialties or hospital capabilities, available infrastructure and assets…	PANTHEON Platform
	Modification of the disaster victim's distributions **(LEARNING)**	Dynamic interaction with visual interface, and input boxes to implement number	Dynamic interaction with visual interface	Can influence and modify the distribution of victims, eventually function of pathologies	Operator
Output	**Output name**	**Output value type**	**Output data format**	**Description**	**To**
	Visualization of dynamic occupancy level of each hospital as a function of time with respect to beginning of the crisis	Visualization	KML/KMZ (Vector data and informative popup)	Providing clear information related to each hospital that can be viewed on the map	Operator
	Quantified dispatching of disaster victims in existing hospitals **(LEARNING)**	Text	JSON	Give the proposal from operator to modify the distribution of victims, eventually function of pathologies	PANTHEON Platform
Algorithm	TBD				
Technologies	Geoserver (TBD) and visual interface based on QT				
Hardware Requirements	CPU for the HMI itself, and geoserver installed on a delocalized server.				
Schematic	TBD				

4.11 MESSAGE BROKER

The message broker is a mechanism used for asynchronous data exchange. In the context of the current project, it will be used in scenarios where one or more of these situations appear:

- Data processing for one involved tool is not deterministic. It can take milliseconds or hours, depending on the context. For this situation, in order not to block other processing, asynchronous data exchange is used.
- There are manual interventions in the flow of processing. Waiting for manual processing should not stop other processing, and asynchronous communication is the best approach.
- There are use cases where the choreography model best suits the need.
- There are streams of data as input, without predefined length.

4.11.1 STATE OF THE ART

For the current project, we have studied the usage of three well-known message brokers:

1. Apache ActiveMQ[18]
2. RabitMQ[19]
3. Apache Kafka[20]

The description of the message brokers taken from their official site is:

Apache ActiveMQ: *"Apache ActiveMQ® is the most popular open source, multi-protocol, Java-based message broker. It supports industry-standard protocols so users get the benefits of client choices across a broad range of languages and platforms. Connect with clients written in JavaScript, C, C++, Python, .Net, and more. Integrate your multi-platform applications using the ubiquitous **AMQP** protocol. Exchange messages between your web applications using **STOMP** over WebSockets. Manage your IoT devices using **MQTT**. Support your existing **JMS** infrastructure and beyond. ActiveMQ offers the power and flexibility to support any messaging use case."*[21]

RabitMQ: *"RabbitMQ is a reliable and mature messaging and streaming broker, which is easy to deploy on cloud environments, on-premises, and on your local machine. It is currently used by millions worldwide. RabbitMQ supports several open standard protocols, including AMQP 1.0 and MQTT 5. There are multiple client libraries available, that can be used with your programming language of choice, just pick one. No vendor lock-in! RabbitMQ provides many options you can combine to define how your messages go from the publisher to one or many consumers. Routing, filtering, streaming, federation, and so on, you name it. With the ability to acknowledge message delivery and replicate messages across a cluster, you can ensure your messages are safe with RabbitMQ. "*[22]

Apache Kafka: *"Kafka combines three key capabilities so you can implement your use cases for event streaming end-to-end with a single battle-tested solution:*

1. *To **publish** (write) and **subscribe to** (read) streams of events, including continuous import/export of your data from other systems.*

[18] https://activemq.apache.org/
[19] https://www.rabbitmq.com/
[20] https://kafka.apache.org/
[21] https://activemq.apache.org/
[22] https://www.rabbitmq.com/

2. *To **store** streams of events durably and reliably for as long as you want.*
3. *To **process** streams of events as they occur or retrospectively.*

And all this functionality is provided in a distributed, highly scalable, elastic, fault-tolerant, and secure manner. Kafka can be deployed on bare-metal hardware, virtual machines, containers, and on-premises as well as in the cloud. You can choose between self-managing your Kafka environments and using fully managed services offered by a variety of vendors."[23]

"Apache Kafka is an open-source streaming data platform originally developed by LinkedIn. As it expanded Kafka's capabilities, LinkedIn donated it to Apache for further development.

Kafka operates like a traditional pub-sub message queue, such as RabbitMQ, in that it enables you to publish and subscribe to streams of messages. But it differs from traditional message queues in 3 keyways:

1. *Kafka operates as a modern distributed system that runs as a cluster and can scale to handle any number of applications.*
2. *Kafka is designed to serve as a storage system and can store data as long as necessary; most message queues remove messages immediately after the consumer confirms receipt.*
3. *Kafka handles stream processing, computing derived streams and datasets dynamically, rather than just passing batches of messages*."[24]

4.11.2 CHOSEN TECHNOLOGY

The investigation of the three technologies mentioned before, and the experience of using similar solutions in other projects requesting asynchronous data exchange leads us the selection of Apache Kafka Broker. The decision is based on the following:

Even if ActiveMQ is a good and proven messaging mechanism, it does not offer the required complexity, especially related to the distributed architectures, and large amount of data streaming.

Kafka and RabitMQ cover a lot of similar functionalities, and there are some differences for which it was possible to make the final decision to use Kafka.

"One important difference between Kafka and RabbitMQ is that the first is pull-based, while the other is push-based. In pull-based systems, the brokers wait for the consumer to ask for data ('pull'); if a consumer is late, it can catch up later. With push-based systems, messages are immediately pushed to any subscribed consumer. This can cause these two tools to behave differently in some circumstances.
Kafka uses a pull model. Consumers request batches of messages from a specific offset. Kafka permits long-pooling, which prevents tight loops when there is no message past the offset and aggressively batches messages to support this.
RabbitMQ uses a push model and stops overwhelming consumers through a prefetch limit defined on the consumer. This can be used for low-latency messaging.
The push model aims to distribute messages individually and quickly, to ensure that work is parallelized evenly and that messages are processed approximately in the order in which they arrived in the queue. However, this can also cause issues in cases where one or more consumers have 'died' and are no longer receiving messages."[25]

[23] https://kafka.apache.org/
[24] https://www.upsolver.com/blog/kafka-versus-rabbitmq-architecture-performance-use-case
[25] https://www.upsolver.com/blog/kafka-versus-rabbitmq-architecture-performance-use-case

But the main reason we decided to use Kafka relates to the persistence of the messages. Kafka is a log, which means that it retains messages by default. You can manage this by specifying a retention policy. On the other hand, RabbitMQ is a queue, so messages are removed once consumed, and acknowledgment is provided.

Considering that also the performance is better for Kafka, due to its usage of sequential disk I/O to boost performance. It can achieve high throughput (millions of messages per second) with limited resources, a necessity for big data use cases.

4.11.3 KAFKA MAIN ENTITIES

Once Kafka technology is chosen, we can present the basic elements used by Kafka, and map our project needs on these elements.

The elements are described in the official documentation[26]. The following elements are considered:

- Events
- Producer
- Consumer
- Topics
- Partitions
- Replications

An **event** records the fact that "something happened" in the world or your business. It is also called a record or message in the documentation. When you read or write data to Kafka, you do this in the form of events. Conceptually, an event has a key, value, timestamp, and optional metadata headers. Here's an example event:

- Event key: "Alice"
- Event value: "Made a payment of $200 to Bob"
- Event timestamp: "Jun. 25, 2020 at 2:06 p.m."

Producers are those client applications that publish (write) events to Kafka, and **consumers** are those that subscribe to (read and process) these events. In Kafka, producers and consumers are fully decoupled and agnostic of each other, which is a key design element to achieve the high scalability that Kafka is known for. For example, producers never need to wait for consumers. Kafka provides various guarantees such as the ability to process events exactly once.

Events are organized and durably stored in **topics**. Very simplified, a topic is similar to a folder in a filesystem, and the events are the files in that folder. An example topic name could be "payments". Topics in Kafka are always multi-producer and multi-subscriber: a topic can have zero, one, or many producers that write events to it, as well as zero, one, or many consumers that subscribe to these events. Events in a topic can be read as often as needed—unlike traditional messaging systems, events are not deleted after consumption. Instead, you define for how long Kafka should retain your events through a per-topic configuration setting, after which old events will be discarded. Kafka's performance is effectively constant concerning data size, so storing data for a long time is perfectly fine.

Topics are **partitioned,** meaning a topic is spread over a number of "buckets" located on different Kafka brokers. This distributed placement of your data is very important for scalability because it allows client applications to both read and write the data from/to many brokers at the same time. When a new event is

[26] https://kafka.apache.org/

published on a topic, it is appended to one of the topic's partitions. Events with the same event key (e.g., a customer or vehicle ID) are written to the same partition, and Kafka guarantees that any consumer of a given topic partition will always read that partition's events in exactly the same order as they were written. An example of this concept can be found in Figure 23.

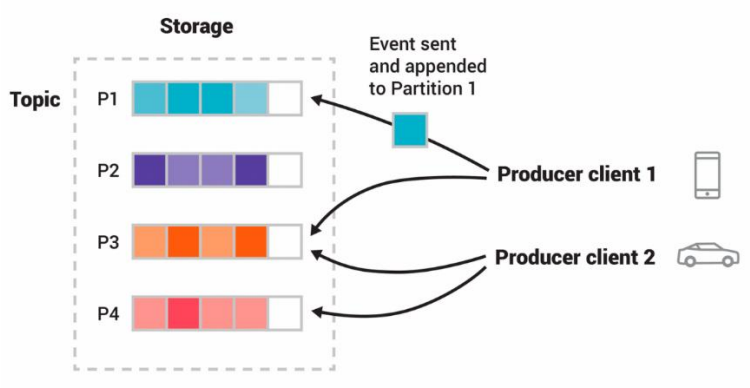
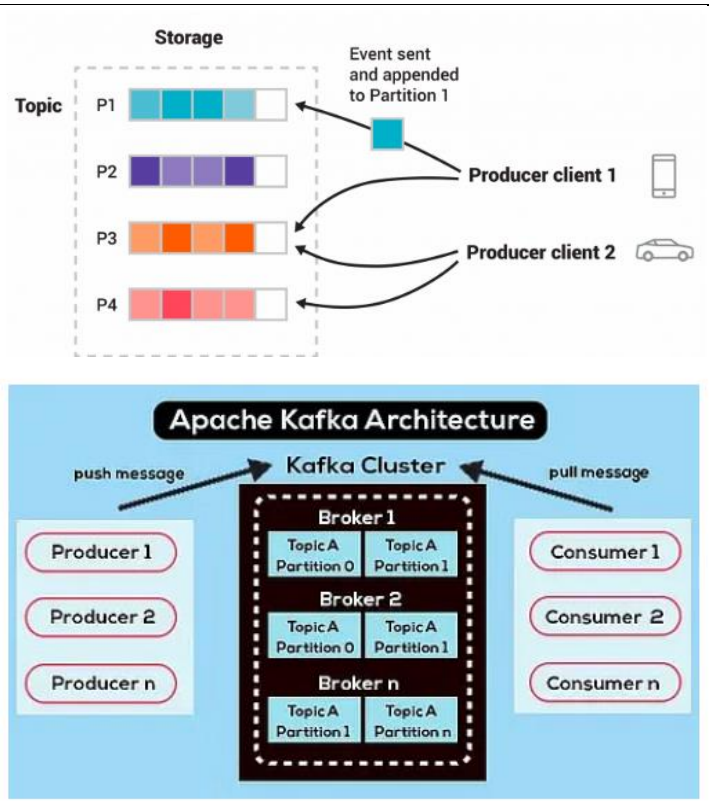


Figure 23 Kafka concepts.

This example topic has four partitions P1–P4. Two different producer clients are publishing, independently from each other, new events on the topic by writing events over the network to the topic's partitions. Events with the same key (denoted by their colour in the figure) are written to the same partition. Note that both producers can write to the same partition if appropriate. To make your data fault-tolerant and highly available, every topic can be **replicated**, even across geo-regions or data centres, so that there are always multiple brokers that have a copy of the data just in case things go wrong, you want to do maintenance on the brokers, and so on. A common production setting is a replication factor of 3, i.e., there will always be three copies of your data. This replication is performed at the level of topic-partitions[27]. The Kafka component can be summarized in the following table.

Service Name	Message Broker				
Description	Message Broker used for asynchronous data exchange				
Pilot	All pilots. It is used by platform for all scenarios and pilots				
Scenario	All scenarios				
Module	Communication				
Submodule	Asynchronous communication				
Input	**Input name**	**Input value type**	**Input data format**	**Description**	**From**
	Asynchronous messages	- JSON type with specific content for any topic	Publish JSON or XML message	Data to be exchanged asynchronously between components	Any module able to publish data in JSON or XML format
Output	**Output name**	**Output value type**	**Output data format**	**Description**	**To**
	Asynchronous messages	- JSON type with	Subscribe to JSON or	Data to be exchanged	Any module able to

[27] https://kafka.apache.org/

		specific content for any topic	XML message	asynchronously between components	subscribe to broker
Algorithm	Asynchronous data exchange based on publish subscribe mechanism. LIFO.				
Technologies	Kafka[28]				
Hardware Requirements	4 CPU, 8G RAM, 250 GB SSD				
Schematic	 Kafka architecture[29]				

4.11.4 KAFKA TOPICS

Considering that Kafka is a pull mechanism, we propose to use a topic for each functional consumer of data. The consumers will subscribe to one topic, but producers can write for several topics. There will also be three general-purpose topics: One for general messages, one for errors, and one for acknowledgments.

Based on this, our proposed topics are:

1. PTH_01_DLML (topic used for data models for machine learning)
2. PTH_02_ML (topic used as input for machine learning tools)
3. PTH_03_DMSIM (topic used as input by data models for simulations)
4. PTH_04_SIM (topic used as input for simulations)
5. PTH_05_AIENG (topic used as input for AI engine)
6. PTH_06_DSS (topic used as input for DSS)

[28] https://kafka.apache.org/
[29] https://kafka.apache.org/21/documentation/streams/architecture.html

7. PTH_07_DSSCBDRM (topic used as input for DSS for CBDRM)
8. PTH_08_WHATIF_A (topic used as input for what-if scenarios alternative solutions)
9. PTH_09_WHATIF_U (topic used as input for what-if scenarios uncertainties)
10. PTH_10_WHATIF_C (topic used as input for what-if scenarios conflicting performance)
11. PTH_11_DAP (topic used as input for data analysis and presentation)
12. PTH_12_MON (topic used as input for system monitoring)
13. PTH_00_GENERAL (topic used for notifications, alerts, or uncategorized input)
14. PTH_99_ERR (topic used for errors and alerts)
15. PTH_88_ACK (topic used for the acknowledge mechanism)

4.12 SERVICE AND WORKFLOW ORCHESTRATOR

4.12.1 SERVICE ORCHESTRATION

The service orchestration refers to the management of services deployed on the platform.

Considering the presence in the system of several modules, developed on different software platforms, and having a well-defined functionality, we propose to package the components in containers.

For our experience and also considering the wide usage, we have chosen to deploy the Docker[30] engine and to have all components placed in Docker containers.

"Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications. By taking advantage of Docker's methodologies for shipping, testing, and deploying code, you can significantly reduce the delay between writing code and running it in production."[31]

Now considering the usage of containers, run on top of the Docker engine, the next step is to select a container orchestration mechanism. The container orchestration mechanism is implicitly a service orchestration mechanism, as the containers are built to run services.

The main orchestration mechanisms described in the literature for Docker containers are Docker Swarm[32] and Kubernetes[33].

Docker Swarm is a more lightweight and simpler platform, while Kubernetes is a more powerful and feature-rich platform. Docker Swarm may better fit smaller-scale deployments or teams with limited resources or expertise, while Kubernetes may be more suitable for larger-scale deployments and complex applications.

A good presentation of the differences between the two systems can be found in the public discourse[34].

"**What is Docker Swarm?**

[30] Docker: Accelerated Container Application Development
[31] https://docs.docker.com/get-started/overview/
[32] https://www.geeksforgeeks.org/introduction-to-docker-swarm-mode
[33] https://kubernetes.io
[34] https://phoenixnap.com/blog/kubernetes-vs-docker-swarm

Docker Swarm (or simply **Swarm**) is an open-source platform for container orchestration popular for its quick setup and ease of use. Not to be confused with Docker (a tool for containerization), Swarm is the platform for managing "Dockerized" containers and is a native mode of Docker.

- A Swarm cluster (set of computers that operate as a single system) consists of:
- Docker Engine-deployed Swarm manager nodes that manage the cluster.

Worker nodes that execute tasks assigned by the manager node.

An admin uses the Docker CLI (the same one used to build images and run containers) to manage the Swarm. This feature makes the tool a logical choice for users already familiar with Docker containers and Docker commands.

Docker Swarm excels within smaller apps with fewer containers.

What is Kubernetes?

Kubernetes (**K8s** or **Kube**) is an open-source platform for managing containers and their workloads. K8s is currently the most popular container orchestration tool on the market and will continue to be so for the foreseeable future.

A K8s cluster is composed of compute hosts called worker nodes managed by a Kubernetes master that controls cluster resources. A cluster can span hosts across various IT systems (on-prem, virtual machines (VMs), public cloud, hybrid architecture, etc.).

K8s architecture is more complicated than Swarm as the platform has master/worker nodes and **pods** that can contain one or more containers. Kubernetes is ideal for complex apps that can benefit from automatic scaling"[35]

Considering the complexity of our system, and the number of services and containers proposed, our selection for the service orchestration is Docker Swarm.

4.12.2 DOCKER SWARM PRESENTATION

A short description of Docker Swarm is the following: *"Docker swarm is a container orchestration tool. Swarm Mode in Docker was introduced in version 1.12 which enables the ability to deploy multiple containers on multiple Docker hosts. For this Docker uses an overlay network for the service discovery and with a built-in load balancer for scaling the services. One of the key advantages of docker swarm is that the configuration of the docker services such as volume and network can be modified without the need to manually restart the service Docker will update the configuration, stop the service tasks with the out-of-date configuration, and create new ones matching the desired configuration. When you want to deploy a container in the swarm first, you have to launch services. Service consists of multiple containers of the same image. These services are deployed inside a node so to deploy a swarm at least one node has to be deployed. As you see below diagram the manager node is responsible for the allocation of the task, dispatch the tasks, and schedule the tasks. API in the manager is the medium between the manager node and the worker node to communicate with each other by using the HTTP protocol. The service of one cluster can be used by the other. All the execution of the task is performed by the worker node."*[36]

[35] https://phoenixnap.com/blog/kubernetes-vs-docker-swarm
[36] https://www.geeksforgeeks.org/introduction-to-docker-swarm-mode/

The main features of Docker Swarm include:

- Cluster management
- Multi-host networking
- Load balancing
- Scaling

In summary, the Docker Swarm is described as: *"Docker swarm is a container orchestration tool that is used to Docker containers and scale them. Instead of a single host with the help of Docker Swarm, we can manage multiple nodes which are called clusters where we can deploy and maintain our containers in multiple hosts. Docker Swarm assures that our application is always available even if the one of nodes fails by creating the container in another node which is available. Based on the incoming traffic we can scale the containers up and down by adding to the multiple nodes. If there are multiple containers the incoming load will be balanced automatically by the Docker Swarm. Docker Swarm has several security features, including traffic encryption between nodes and mutual TLS authentication. Docker Swarm will automatically take care of failed containers and nodes. By this, we can maintain high availability. Simply Docker Swarm is mainly used to deploy, scale, and manage the containers and nodes that are available in the cluster."*[37]

4.12.3 WORKFLOW ORCHESTRATION

Data orchestration is necessary for the Pantheon project, considering the large number of data providers, producers, and processors. There is a wide variety of data, and for an end-to-end functionality, the flow of processing should be well-defined and followed. This data processing on well-defined flows indicates the usage of service orchestration, where each service is responsible for a very well-defined set of inputs, outputs, and processing. The services are implemented mainly as microservices, but SOAP web services are also possible. Service orchestration tools are placed in backend layer, and play a central role, taking care of all your data pipelining. Considering the complexity of the PANTHEON project, we have studied some existing open-source service orchestrators, and finally, we have decided to use one of them. We are presenting in this chapter three of the candidates, the chosen one, and the motivation for the selection. Also, we'll describe the place of the service orchestrator in the overall architecture.

4.12.4 STATE OF THE ART FOR OPEN-SOURCE WORKFLOW ORCHESTRATORS

We have considered two open-source service orchestrators and one commercial:

- Airflow[38]
- Apache NiFi[39]
- MuleSoft's Anypoint Platform[40]

Apache Airflow: "*Apache Airflow is an open-source tool to programmatically author, schedule, and monitor workflows. It is used by Data Engineers for orchestrating workflows or pipelines. One can easily visualize your data pipelines' dependencies, progress, logs, code, trigger tasks, and success status. Complex data pipelines are managed using it. These data pipelines are used to deliver datasets that are easily used by business intelligence applications or machine learning models where a huge amount of data is required. It is one of the most robust platforms for data engineers. Batch-oriented workflows are developed, scheduled, and*

[37] https://www.geeksforgeeks.org/introduction-to-docker-swarm-mode
[38] Apache Airflow
[39] Apache NiFi
[40] Enterprise Hybrid Integration Platform | Anypoint Platform | MuleSoft

monitored efficiently. Apache Airflow is a workflow engine that easily schedules and runs complex data pipelines. Airflow has four important components that are very important to understand how Airflow works.

- *Dynamic: Airflow allows dynamic pipeline generation and configures using Python programming.*
- *Extensible: Airflow is very extensible. User can easily define their operators as the requirement and suit the environment.*
- *Elegant: Airflow pipelines are lean and explicit.*
- *Scalable: Airflow uses a message queue for communication. It has a modular architecture.*

Workflow refers to the process of achieving some goal. They always have an end goal which could be something like creating visualizations for some data as given here. Directed Acyclic Graphs (abbreviated as DAG) are used to represent the workflow."[41]

Apache NiFi: "*Put simply, NiFi was built to automate the flow of data between systems. While the term 'dataflow' is used in a variety of contexts, we use it here to mean the automated and managed flow of information between systems. This problem space has been around ever since enterprises had more than one system, where some of the systems created data and some of the systems consumed data. The problems and solution patterns that emerged have been discussed and articulated extensively.*"[42]
MuleSoft's Anypoint Platform: *"MuleSoft's Anypoint Platform is a leading integration platform for SOA, SaaS, and APIs. It connects applications, data, and devices both on-premises and in the cloud with an API-led approach, providing exceptional business agility to companies. Anypoint Platform is a single solution for developing, deploying, securing, and managing APIs and integrations. Anypoint Monitoring is the standard method of monitoring Mule application and API performance.."*

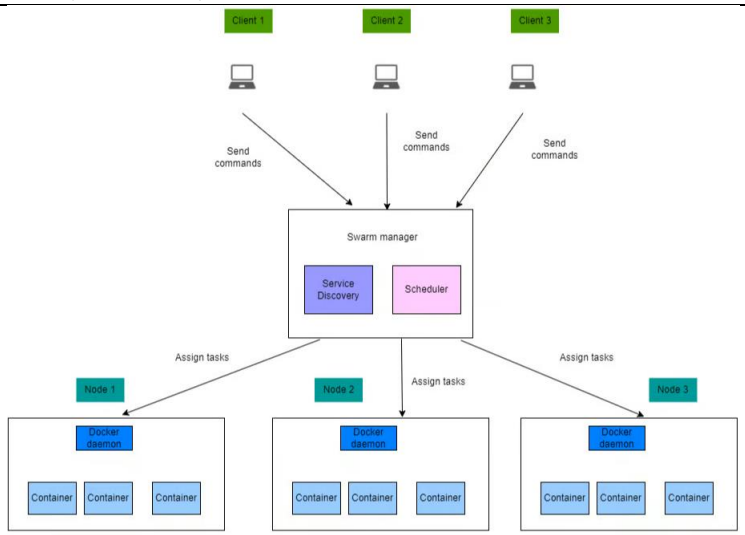
Based on the study of existing systems, and considering our experience in deploying service workflows, we decided to choose Airflow as the technology enabler for the Pantheon project.

The main reasons are:

- The Airflow is open source and the community is very large and is still growing and there's a lot of support available.
- Apache Airflow is highly extensible which allows it to suit any environment.
- The pipelines are generated dynamically and are configured as code using Python programming language.
- Rich scheduling and execution semantics are used to easily define complex pipelines and keep them running at regular intervals.
- Easy to deploy (Python knowledge necessary)
- Good management tools for workflows

Service Name	Service and workflow orchestrator
Description	Service orchestrator for services placed in containers. Workflow orchestrator for events and processes.
Pilot	All pilots
Scenario	All scenarios

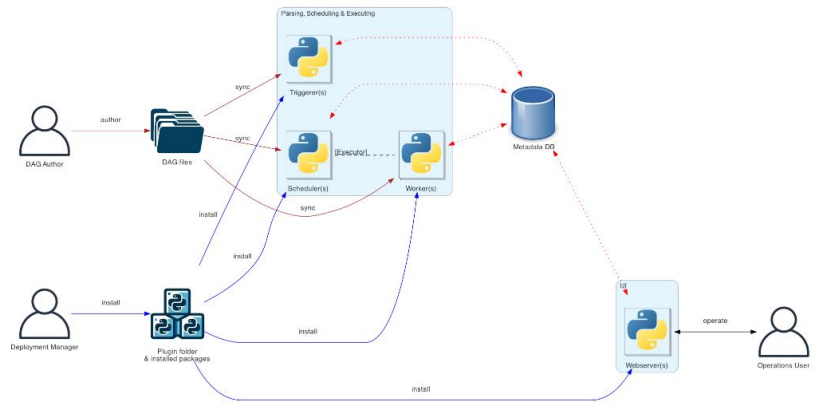
[41] https://www.geeksforgeeks.org/what-is-apache-airflow/
[42] https://nifi.apache.org/

Module	Service and workflow orchestrator				
Submodule					
Input	**Input name**	**Input value type**	**Input data format**	**Description**	**From**
	Containers	- List of images or containers	Container (Docker)	Containers with services embeded	docker
	Processes	Processes	Workflow description in JSON Format		
Output	**Output name**	**Output value type**	**Output data format**	**Description**	**To**
	Run a service in a container	Service run	Depending on service	Used to manage the services in a distributed environment	
	Enable a step in a workflow	Action for a step in workflow	Action command in JSON format	Launcg an action for a state machine	
Algorithm	Finite State Machine				
Technologies	Docker SWARM[43] Airflow[44]				
Hardware Requirements	Cluster with at least 4 nodes, each one with 8CPU, 32 GRAM, 250 G SSD				
Schematic	 Docker swarm architecture[45]				

[43] Docker: Accelerated Container Application Development

[44] Apache Airflow

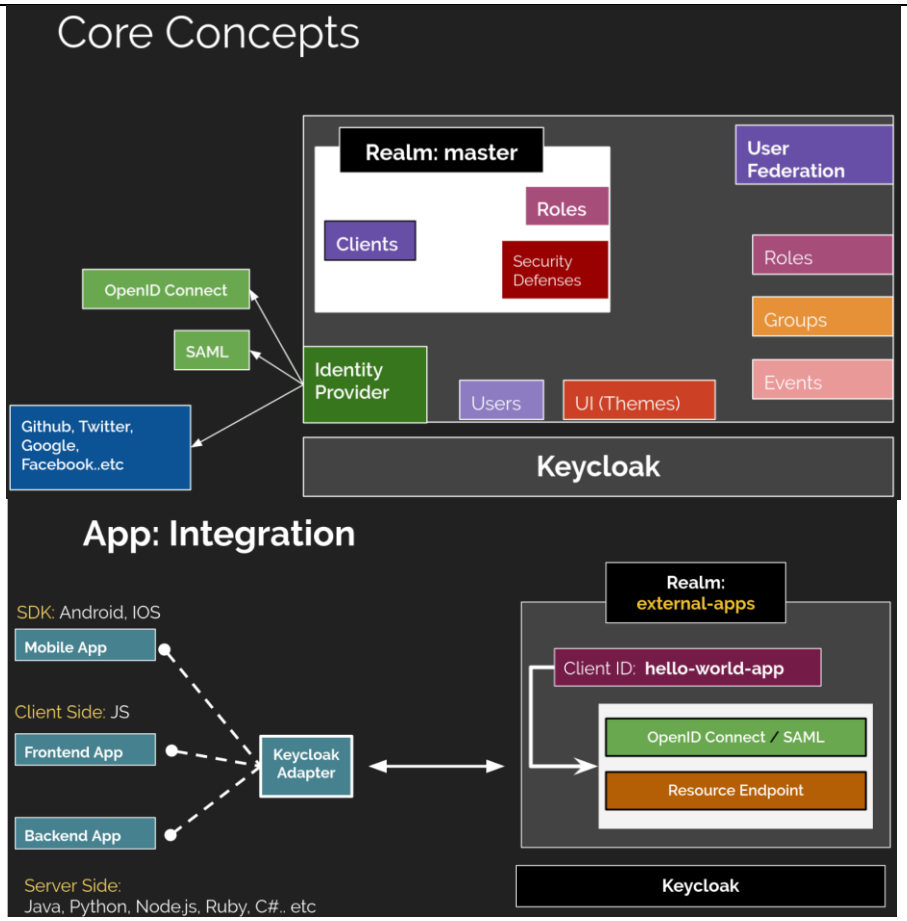
[45] Docker: Accelerated Container Application Development

	 Airflow architecture[46]

4.13 USER MANAGEMENT

Service Name	User Authorization and Authentication				
Description	System used for user registration, role management, user authentication, user authorization based on roles. SSO enabled. Oauth2 compatible.				
Pilot	All pilots. It is used by the platform in all implementations				
Scenario	All scenarios				
Module	User Authorization and Authentication				
Submodule	--				
Input	**Input name**	**Input value type**	**Input data format**	**Description**	**From**
	roles	List of roles in the system	Manually entered in UI	The roles are used to allow different functionality for different users and roles	UI (User Interface)
	users	List of users	Manually entered in UI		UI (User Interface)
	Authentication request	User Name and password	Manually entered in UI		UI (User Interface)
	Second Factor authentication	Code	Manually entered in UI	Request received by email	UI (User Interface)
Output	**Output name**	**Output value type**	**Output data format**	**Description**	**To**
	Registered roles	List of roles			
	Registered users	List of users			
	Authorization to use the system	Permission to use the system			

[46] https://airflow.apache.org/docs/apache-airflow/stable/core-concepts/overview.html

	Access Token	Token	JWT (Oauth2)	Access token (Bearer token)	Main portal or API client
Algorithm	Oauth2, OpenID				
Technologies	Keycloak[47]				
Hardware Requirements	4 CPU, 8G RAM, 10 G SSD.				
Schematic	 Keycloak concepts[48]				

[47] https://www.keycloak.org/docs/latest/securing_apps/
[48] https://www.keycloak.org/docs/latest/securing_apps/

5. OPERATION FLOW

5.1 CORE PROCEDURES

5.1.1 USER AUTHORIZATION

Access of users to the system will be:

- Public, not restricted, for a limited number of functionalities, where any citizen can obtain information
- For registered users, based on their roles. Specific functionality will be accessible to each role registered in the system
- API access for registered users, based on tokens.

The system will be secured by an Authentication and Authorization system, responsible for the secured access to each layer of the system.

The proposed security will be compatible with Oauth2 specifications.

The user access will be guarded by a Single Sign-on mechanism, where also second-factor authentication can be enabled.

The technology supporting the aforementioned requirements and proposed based on our experience in deploying large and highly secured systems is Keycloak[49].

"Keycloak is a single sign-on solution for web apps and RESTful web services. The goal of Keycloak is to make security simple so that it is easy for application developers to secure the apps and services they have deployed in their organization. Security features that developers normally have to write for themselves are provided out of the box and are easily tailorable to the individual requirements of your organization. Keycloak provides customizable user interfaces for login, registration, administration, and account management. You can also use Keycloak as an integration platform to hook it into existing LDAP and Active Directory servers. You can also delegate authentication to third-party identity providers like Facebook and Google."

Keycloak's main features used in this project, as described in the official documentation are:

- Single-Sign-On and Single-Sign Out for browser applications.
- OpenID Connect support.
- OAuth 2.0 support.
- SAML support.
- Identity Brokering - Authenticate with external OpenID Connect or SAML Identity Providers.
- Social Login - Enable login with Google, GitHub, Facebook, Twitter, and other social networks.
- User Federation - Sync users from LDAP and Active Directory servers.
- Kerberos bridge - Automatically authenticate users that are logged in to a Kerberos server.
- Admin Console for central management of users, roles, role mappings, clients, and configuration.
- Account Console that allows users to centrally manage their accounts.
- Theme support - Customize all user-facing pages to integrate with your applications and branding.
- Two-factor Authentication - Support for TOTP/HOTP via Google Authenticator or FreeOTP.

[49] https://www.keycloak.org

- Login flows - optional user self-registration, recover password, verify email, require password update, etc.
- Session management - Admins and users themselves can view and manage user sessions.
- Token mappers - Map user attributes, roles, etc. how you want into tokens and statements.
- Not-before revocation policies per realm, application, and user.
- CORS support - Client adapters have built-in support for CORS.
- Service Provider Interfaces (SPI) - Several SPIs to enable customizing various aspects of the server. Authentication flows, user federation providers, protocol mappers, and many more.
- Client adapters for JavaScript applications, WildFly, JBoss EAP, Tomcat, Spring, etc.
- Supports any platform/language that has an OpenID Connect Relying Party library or SAML 2.0 Service Provider library.

Keycloak core concepts used in this project, as described in the official documentation are:

users
Users are entities that can log into your system. They can have attributes associated with themselves like email, username, address, phone number, and birthday. They can be assigned group membership and have specific roles assigned to them.
authentication
The process of identifying and validating a user.
authorization
The process of granting access to a user.
credentials
Credentials are pieces of data that Keycloak uses to verify the identity of a user. Some examples are passwords, one-time passwords, digital certificates, or even fingerprints.
roles
Roles identify a type or category of user. Admin, user, manager, and employee are all typical roles that may exist in an organization. Applications often assign access and permissions to specific roles rather than individual users as dealing with users can be too fine-grained and hard to manage.
user role mapping
A user role mapping defines a mapping between a role and a user. A user can be associated with zero or more roles. This role mapping information can be encapsulated into tokens and assertions so that applications can decide access permissions on various resources they manage.
composite roles
A composite role is a role that can be associated with other roles. For example, a superuser composite role could be associated with the sales-admin and order-entry-admin roles. If a user is mapped to the superuser role they also inherit the sales-admin and order-entry-admin roles.
groups
Groups manage groups of users. Attributes can be defined for a group. You can map roles to a group as well. Users who become members of a group inherit the attributes and role mappings that the group defines.
realms
A realm manages a set of users, credentials, roles, and groups. A user belongs to and logs into a realm. Realms are isolated from one another and can only manage and authenticate the users that they control.
clients
Clients are entities that can request Keycloak to authenticate a user. Most often, clients are applications and services that want to use Keycloak to secure themselves and provide a single sign-on solution. Clients can

also be entities that just want to request identity information or an access token so that they can securely invoke other services on the network that are secured by Keycloak.

client adapters

Client adapters are plugins that you install into your application environment to be able to communicate and be secured by Keycloak. Keycloak has a number of adapters for different platforms that you can download. There are also third-party adapters you can get for environments that we don't cover.

consent

Consent is when you as an admin want a user to give permission to a client before that client can participate in the authentication process. After a user provides their credentials, Keycloak will pop up a screen identifying the client requesting a login and what identity information is requested of the user. The user can decide whether or not to grant the request.

client scopes

When a client is registered, you must define protocol mappers and role scope mappings for that client. It is often useful to store a client scope, to make creating new clients easier by sharing some common settings. This is also useful for requesting some claims or roles to be conditionally based on the value of the scope parameter. Keycloak provides the concept of a client scope for this.

client role

Clients can define roles that are specific to them. This is a role namespace dedicated to the client.

identity token

A token that provides identity information about the user. Part of the OpenID Connect specification.

access token

A token that can be provided as part of an HTTP request that grants access to the service being invoked on. This is part of the OpenID Connect and OAuth 2.0 specification.

assertion

Information about a user. This usually pertains to an XML blob that is included in a SAML authentication response that provided identity metadata about an authenticated user.

service account

Each client has a built-in service account which allows it to obtain an access token.

session

When a user logs in, a session is created to manage the login session. A session contains information like when the user logged in and what applications have participated within single-sign on during that session. Both admins and users can view session information.

authentication flows

Authentication flows are work flows a user must perform when interacting with certain aspects of the system. A login flow can define what credential types are required. A registration flow defines what profile information a user must enter and whether something like reCAPTCHA must be used to filter out bots. Credential reset flow defines what actions a user must do before they can reset their password.

events

Events are audit streams that admins can view and hook into.

The sequence of actions when using Keycloak (after its installation) for the current project will be:

1. Creation of a realm
2. Creation of specific clients for main backend components
3. Creation of roles for each client
4. Creation of groups of users
5. Creation of users, and association with groups
6. Assign roles to groups and/or individual users.

As a separate action, we consider enabling second-factor authentication, based on emails.

It is important to consider that the users, roles, and groups, will be created at the central level for all modules, including the access to API.

A functional module will be associated in Keycloak with a client.

By having SSO, when a user is authenticated, he/she can access any module based on the roles associated.

5.1.2 USER CREATION (BY THE ADMINISTRATOR)

User creation is possible using the Keycloak system, which is the main security mechanism used by Pantheon.

The prerequisites are:

1. A realm is created
2. Clients are created in the realm
3. Roles are created for clients.

The creation of the realm can be done like in Figure 24.

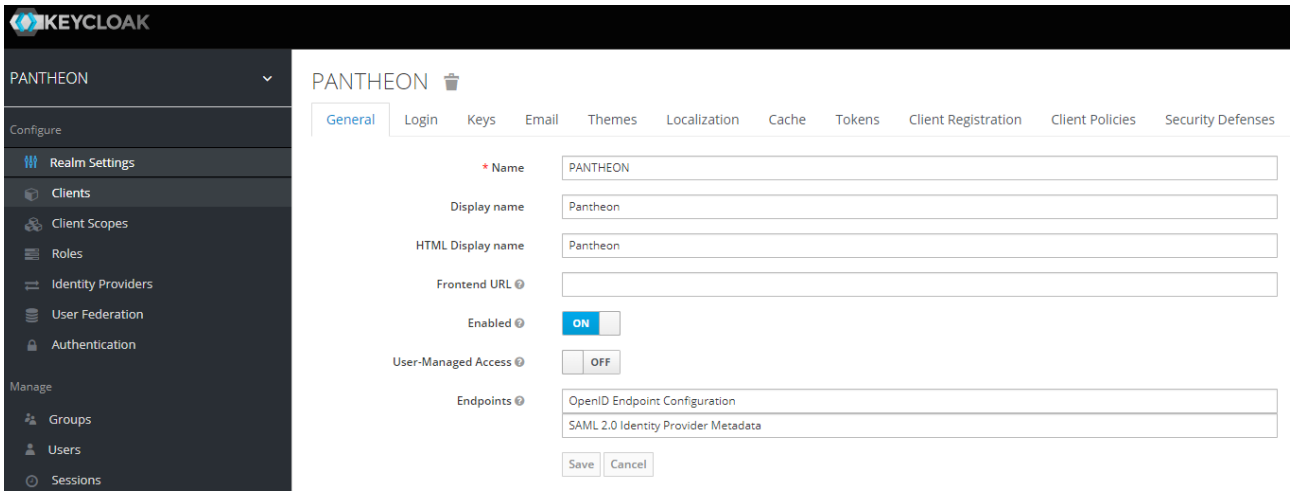


Figure 24 Create Realm.

The creation of clients can be done in a similar manner as shown in Figure 25.

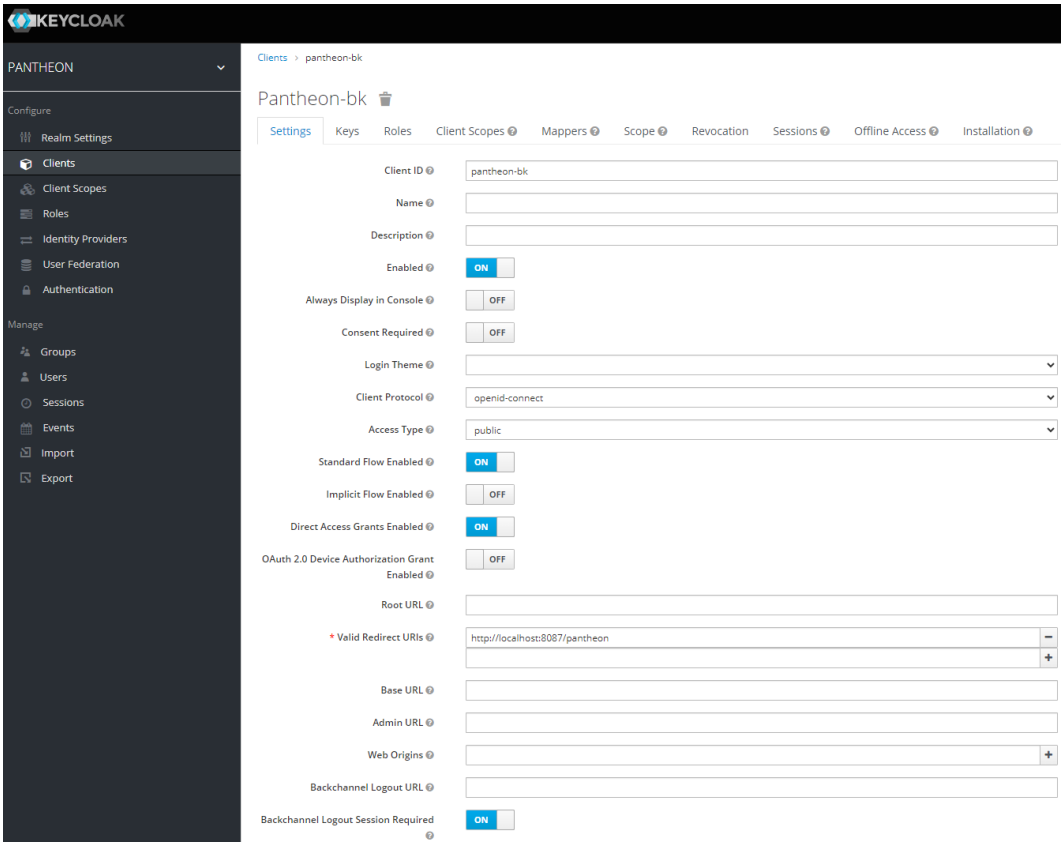


Figure 25 Create Client.

In our example we are considering the roles defined like in Figure 26.

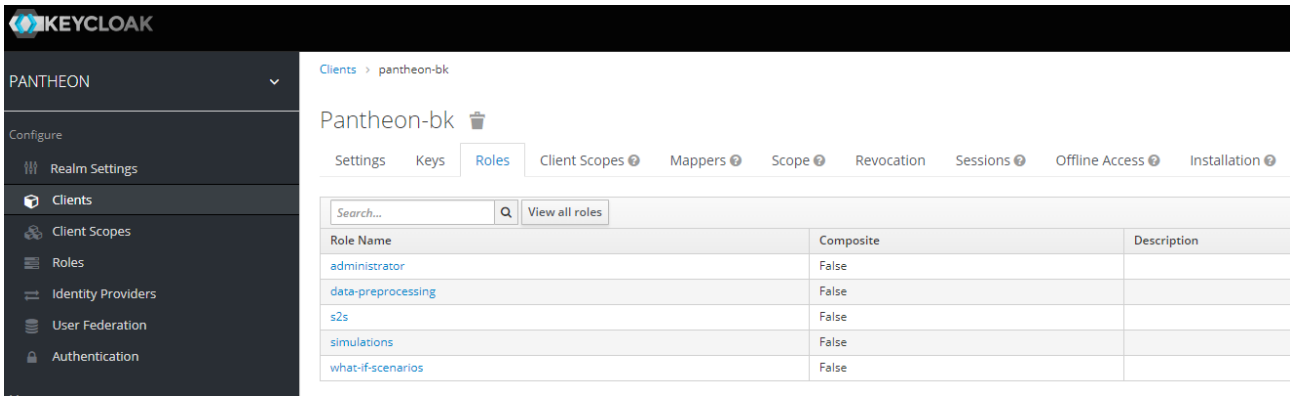


Figure 26 Roles for a client.

Having the required elements, the creation of a user is done also in Keycloak administration console

First, the “Add User” button is pressed (as shown in Figure 27).

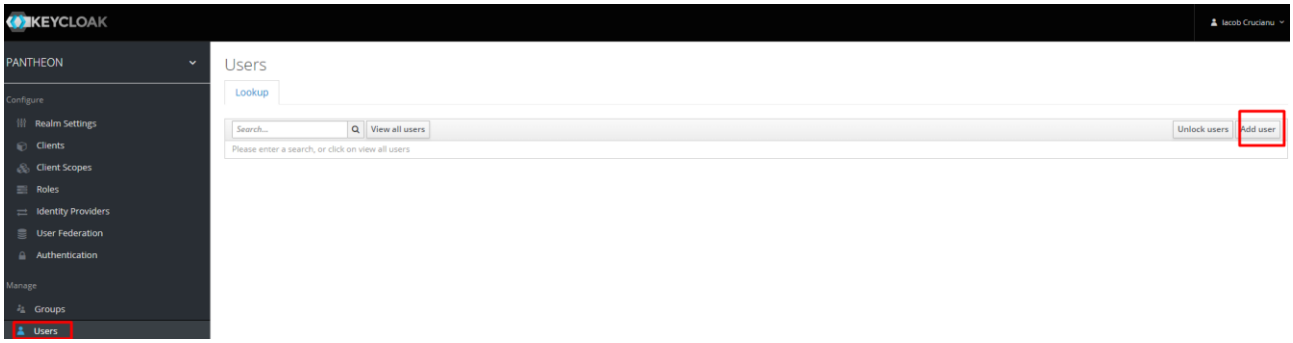


Figure 27 Add User.

Next, data about a user can be entered (Figure 28).

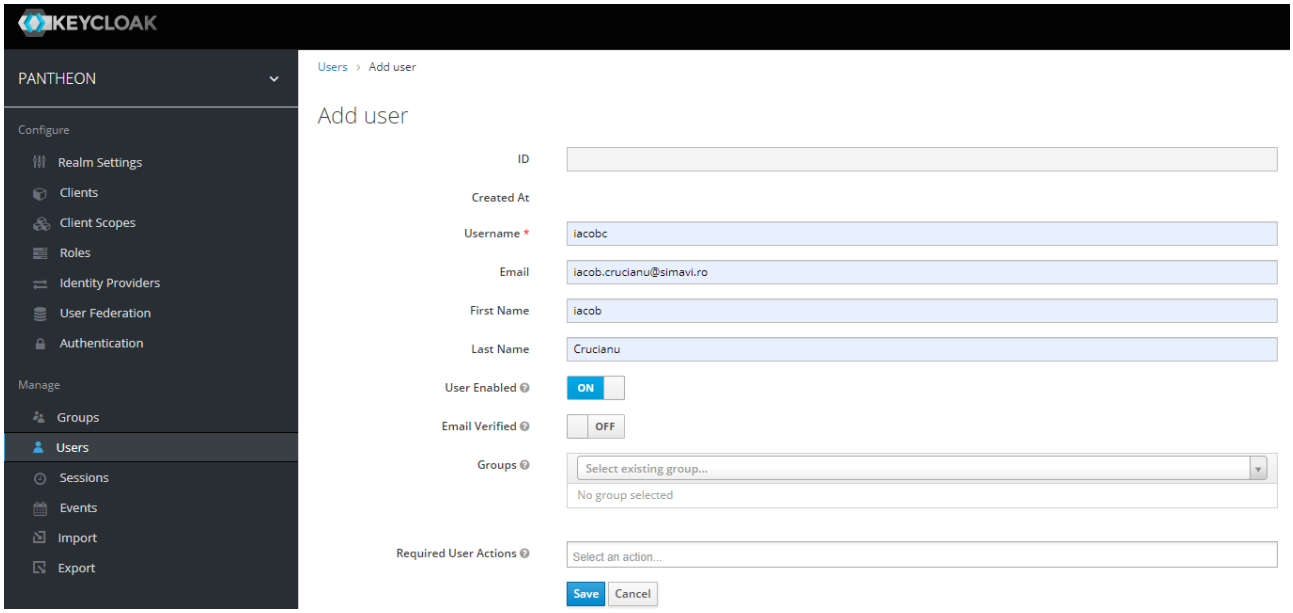


Figure 28 User data.

Next, roles can be assigned (Figure 29).

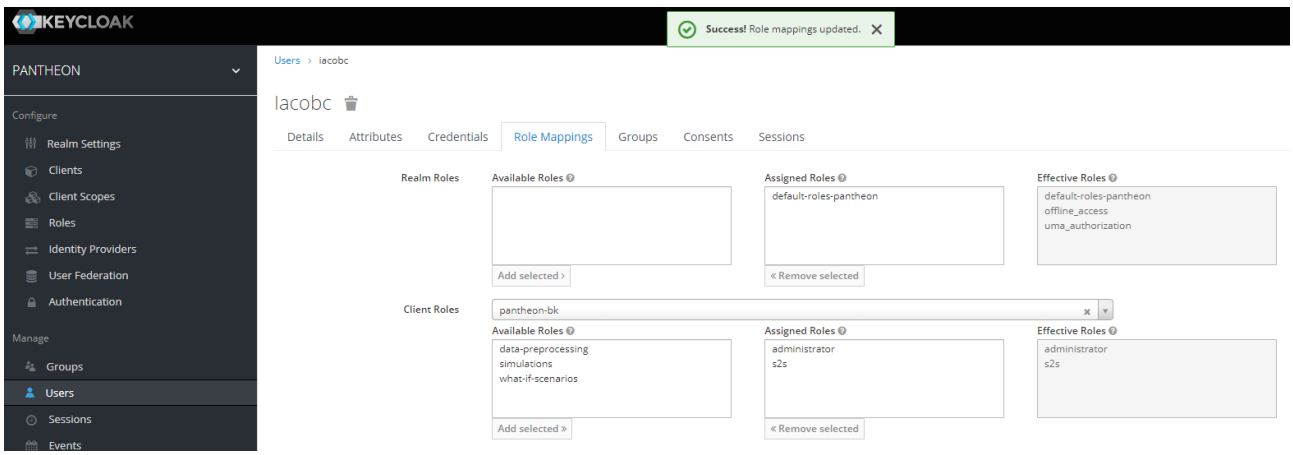


Figure 29 Assign roles to user.

Setting the credentials can be done as shown in Figure 30.

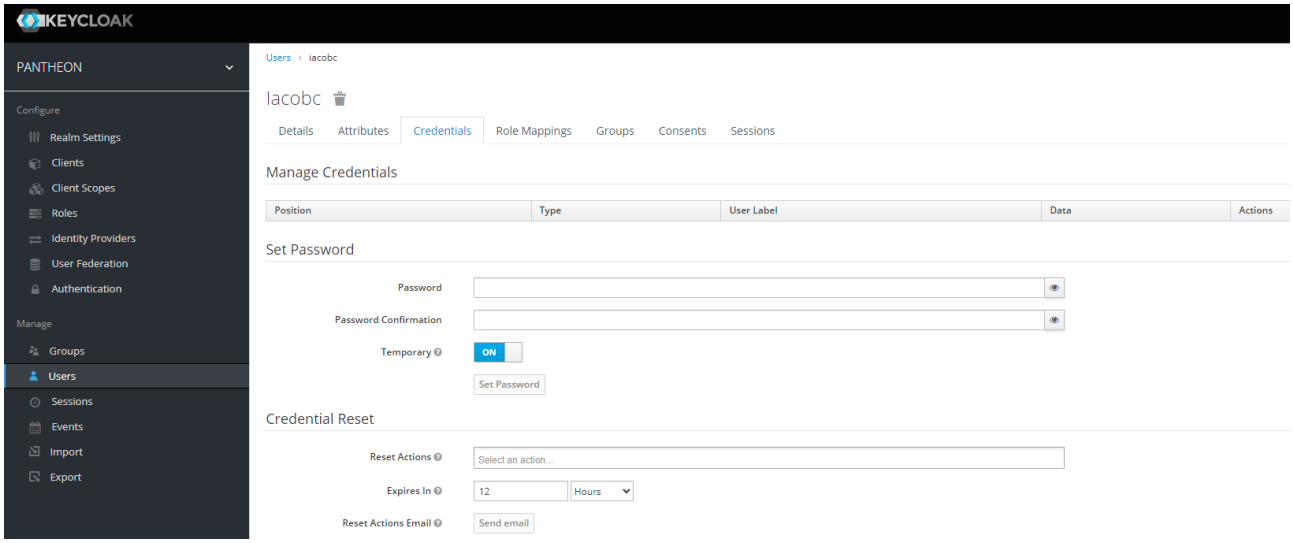


Figure 30 Setting credentials.

In any moment of system functionality, the administrator can modify users, roles, rights, and other attributes of users.

5.2 EXAMPLES FROM THE PANTHEON USE CASES

Starting from the Usage Scenarios and Use Cases as indicated in D3.6, we further describe the sequence of steps followed in the PANTHEON platform for each Disaster/Usage scenario in PANTHEON. The flow starts with the user logging in and starting the simulation, which initiates a process in a loop with the user in the loop (Human in the loop – HITL). The process starts with the relevant, analysed, and curated data being ingested and provided to the simulation, which after an iteration with the ML models offers a result. This result is then propagated to the user, the swarm schemes and resource controller, the decision support system and the what-if analysis. The additional input by the user is also considered before the next iteration of the simulation. After several iterations, the simulation concludes, and the final results (final simulation, decision support and swarm scheme) are provided to the user. A preliminary set of sequence diagrams for each disaster scenario in PANTHEON is presented in Figure 31, Figure 32, Figure 33 and Figure 34 respectively.

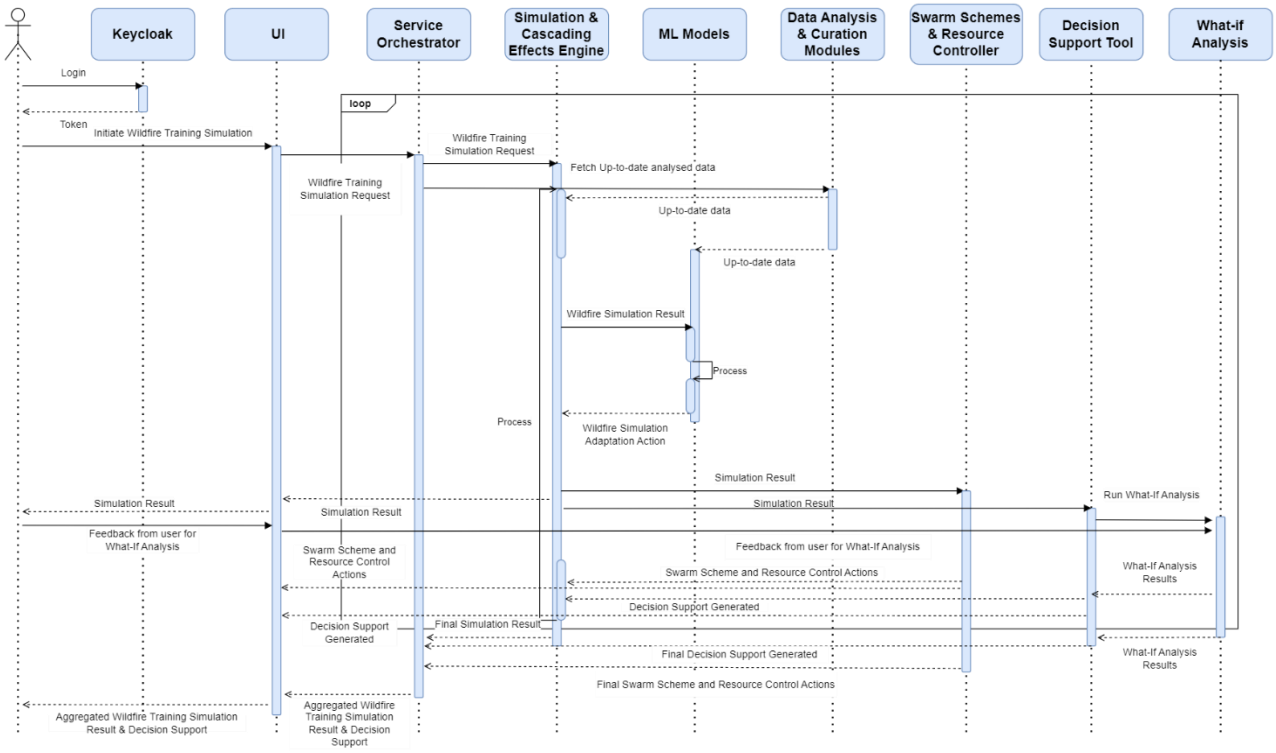


Figure 31 A preliminary sequence diagram of the PANTHEON platform for DS-ATH-B.

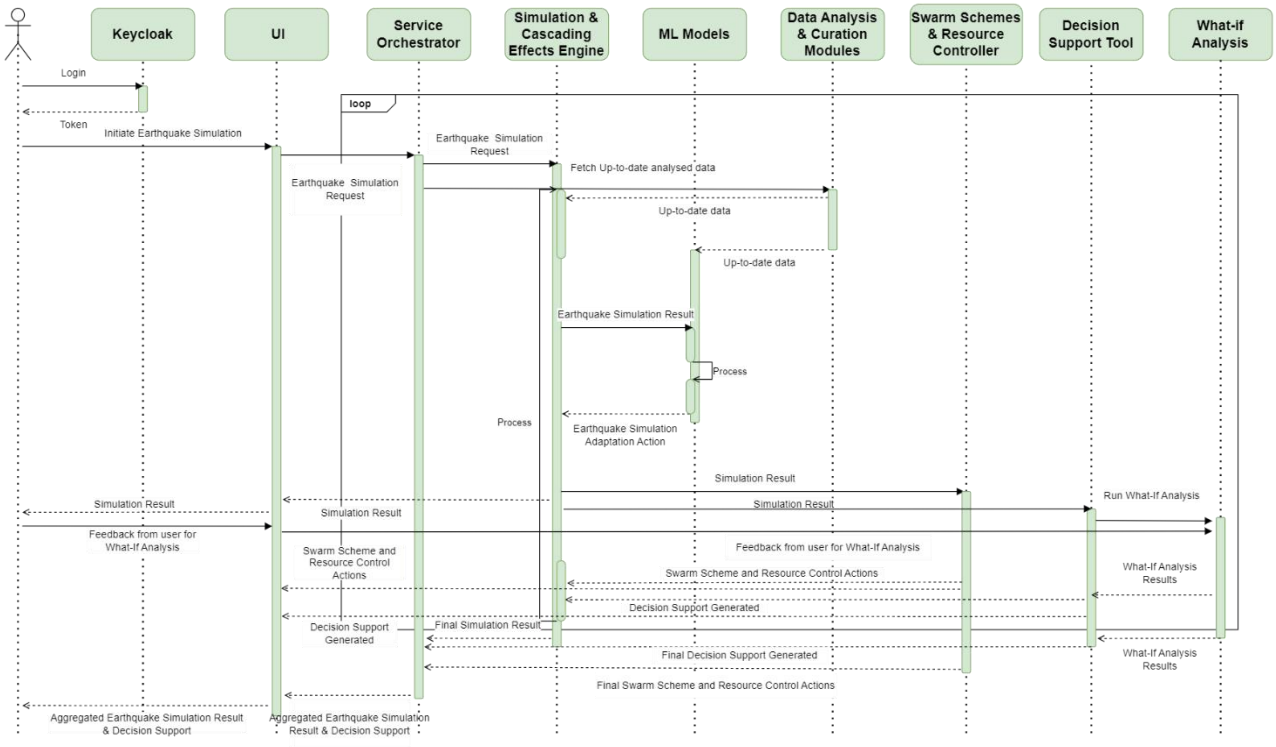


Figure 32 A preliminary sequence diagram of the PANTHEON platform for DS-ATH-A.

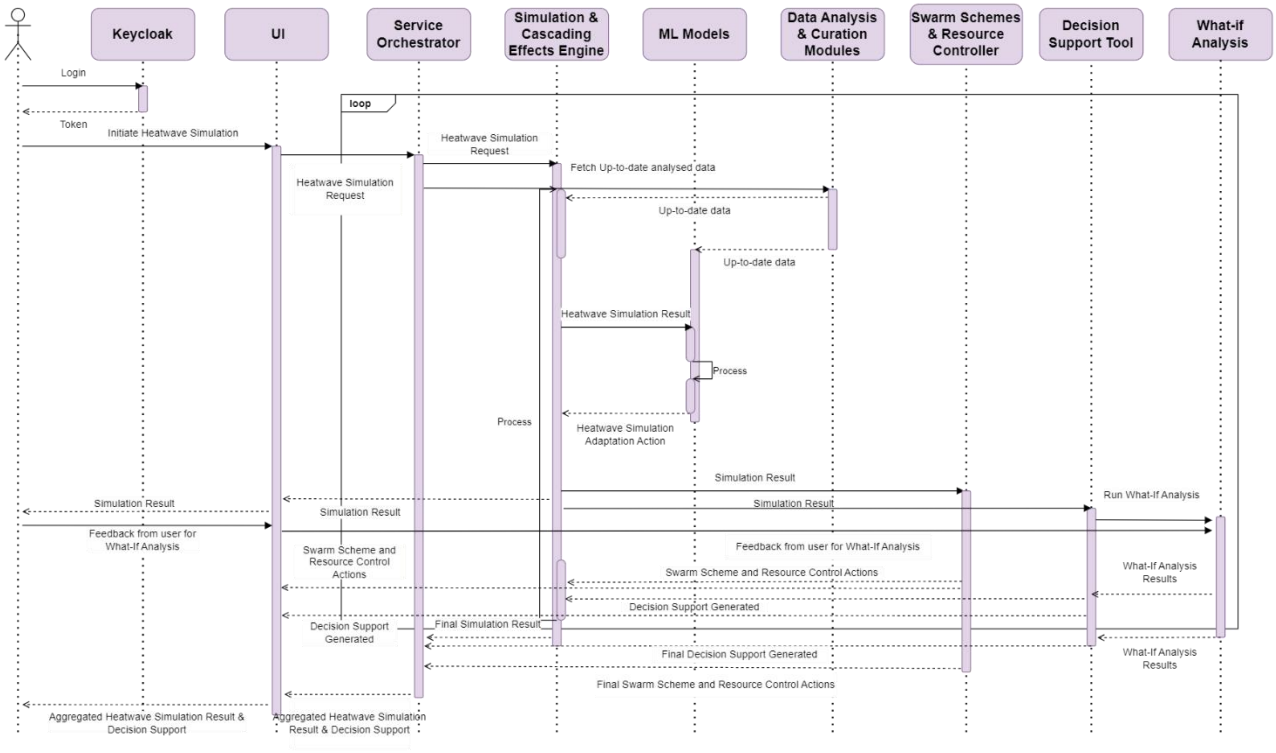


Figure 33 A preliminary sequence diagram of the PANTHEON platform for DS-VIE-A.

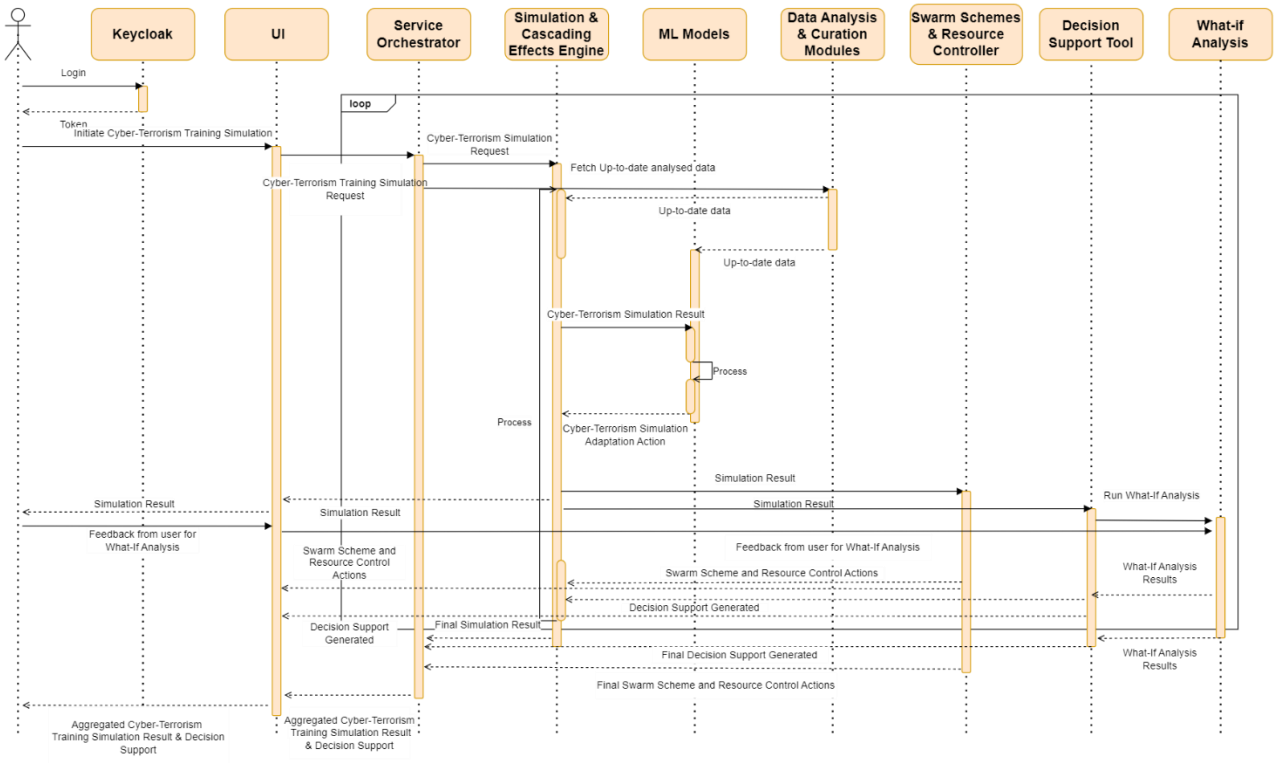


Figure 34 A preliminary sequence diagram of the PANTHEON platform for DS-VIE-B.

6. PLATFORM INTEGRATION

6.1 OVERVIEW

The Platform Integration view describes in detail the constraints of the system elements in terms of hardware and/or software resources, compatibility with standards, etc. The distribution of the functionalities among the several components of the architecture as well as the interconnections and flow of information among them is described. This section presents the significant aspects concerning the integration approach specific to the Pantheon project, based on the architectural design and technological insights gathered in the previous tasks (T3.1 and T3.3). The PANTHEON ecosystem is expected to support systems of systems. System integration used in the current project uses the definitions from ISO/IEC/IEEE 24748-6[50]. This system requires complex interconnections (equipment and functions); and interoperability (networks, systems, devices, applications, and components exchanging a diversity of data in terms of formats, frequency of generation, and delivery). Considering the Standard recommendations and the PANTHEON reference architecture, the PANTHEON Consortium proposed an integration design process based on the following steps:

- Integration planning and application guidelines
    1. General
    2. Integration strategy
    3. Efficiency considerations in the integration strategy
    4. Integration context
    5. Roles and competencies of the integration team
    6. Methods used to perform integration

6.2 INTEGRATION PLANNING AND APPLICATION GUIDELINES

We are presenting in this section the planning and the design of the integration process. We are defining the strategy, the responsibilities, the context, and the methods used for the integration of the system.

6.2.1 GENERAL

The flow proposed looks into a high-level integration flow, i.e. assumes the existence of all necessary prerequisites together with their capabilities to adapt to a common framework. In the following section details about each step to be taken to achieve the proposed integration are depicted.

It is important to note that the integration will use a central platform and a set of modules and tools.

The central platform is responsible for data persistence, process orchestration, choreography of asynchronous processes, user access, security mechanisms, system monitoring and administration, and data presentations.

6.2.2 INTEGRATION STRATEGY

This report follows mainly the IT facet of integration defined as a system of systems but looks also into smart grid-specific aspects whenever necessary. This is why when discussing the approaches of integration the scope of the integration concept ranges from the integrations defining a smart grid (e.g. integration of smart

[50] https://www.iso.org/obp/ui/en/#iso:std:iso-iec-ieee:24748:-6:ed-1:v1:en

appliances, consumer devices, advances electricity storage, renewable resources) to the connection, interconnection, operability, and interoperability of specific equipment and communication protocols.

Considering the integration of IT solutions, two major prospects are considered:

i. The deployment side: This approach is focused on how the final (integrated) system/solution will look and how its specific parts will exchange and handle data and trigger processes.
ii. The development side: This approach is focused on how to develop the different parts of the system (as a framework of tools) in an orchestrated way, taking into consideration that different technologies (tools) are delivered by different technology providers.

Chronologically, the following integration approaches should be pursued:

1. Initially, the main framework of deployment should be established.
2. Second, the proceeding between technology providers should be established.
3. Finally, the repositories and locations for the development of tools and the deployed components should be established.

The above-mentioned aspects concerning the integration approach are depicted hereinafter, specifically for the Pantheon project.

Based on the Pantheon architecture and requirements, various options for integration were explored and finally an iterative approach to integration was pursued. Thus, each integration cycle will have a specific set of development and testing activities.

As a basis of our assessment of various options for integration, we used a survey dedicated to deployment tools and integration analytical framework (development environment for integration) which was filled out by each tool provider.

Considering the overall integration concept described within Pantheon we are looking at a four-level approach to integration:

- Integration at the conceptual Level (uniformly define concepts, requirements, and validation mechanisms)
- Integration at the data level (define the structures and protocols for data exchange)
- Integration at the service level (define service signatures uniformly, and make them compatible between modules and tools)
- Integration at the presentation level (have a uniform presentation theme)

6.2.3 EFFICIENCY CONSIDERATIONS IN THE INTEGRATION STRATEGY

When integrating the system, several aspects related to the efficiency are considered. They refer to:

Resource Utilization: Efficient integration ensures optimal use of resources such as hardware, software, and human capital. This involves reusage whenever possible of data and processing between system modules.

Performance Optimization: Integration processes are designed to maximize performance and minimize latency. This includes optimizing data transfer speeds, pull and push mechanisms, query processing times, and overall system responsiveness.

Scalability: The system will be fully scalable by placing the services in containers and using Swarm container management.

Flexibility and Adaptability: Integration solutions will accommodate changes in business requirements, technology landscapes, and data formats. This includes the ability to quickly integrate new modules, data sources, and models as needed.

Cost Efficiency: The whole platform is developed using open-source components. The selection of components was based on their maturity, and community of users, able to assist in supporting the functionality.

Automation: The integrated system uses a service orchestrator, able to command and automate the whole integrated functionality of the system.

Monitoring and Optimization: The platform is implementing a monitoring tool, which allows the understanding of resource usage and their optimization.

6.2.4 INTEGRATION CONTEXT

Each tool/component being an atomic software component, the integration can be done by placing them as services in containers and then using an orchestration or message-based choreography to enable data exchange and the flow of processing. We consider that the data element exchanged between components is a message. This message can be exchanged synchronously in the orchestrated environment or can be asynchronously placed on message queues.

The following entities are considered:

- Messages (data about an entity)
- Synchronous Messaging Systems (REST APIs managed by orchestrator),
- Asynchronous Messaging Channels (Topics, Queues),
- Message Routing,
- Message Transformation,
- Orchestration engine
- Message brokers.

This integration approach facilitates the exchange of information between tools/components and the reuse of information.

Considering the integration capabilities of each tool and use cases, the integration platform will implement several methods to achieve cross-component exchange of information There are two main approaches for exchanging information, both implemented in the PANTHEON integration platform:

- Online – exchange information using web services or asynchronous messages governed by brokers.
- Off-line – exchange information using files.

On-line integration of tools

For online integration, the Pantheon integration platform supports web services in two approaches:

Push (Endpoint) – the web service is exposed by the integration platform and invoked by the tools.
Pull (Client) – the web service is exposed by the tool and invoked by the integration platform.

On-line push integration

In this approach, the tool pushes information into the integration platform. This is the desirable approach from a *business* point of view because the tools know best about the dynamics of the data they manage and can implement different strategies to send (push) the information to the integration platform, e.g.

periodically or when some events occur. It is the responsibility of the tool (client) to take care of the information already sent.

In the synchronous mode, Push means the usage of HTTP POST or PUT methods.

In asynchronous mode, Push means the placement (Publish) of a message on the message broker queue.

On-line pull integration

In this approach, the integration platform pulls information from the tool. The endpoint being implemented on the tool side and the client on the integration platform side the only strategy for pooling data from the tool is periodically. Is the responsibility of the integration platform to request only information that is new and for that purpose, the endpoint implementation must have a parameter of type timestamp which will be used to distinguish the information already sent.

In the synchronous mode, Pull means the usage of HTTP GET methods.

In asynchronous mode, Pull means the receive (Subscribe) of a message on the message broker queue.

Off-line integration of tools

From a technical point of view, some tools do not have support for web services or messaging. For these cases, the integration platform implements functionalities for processing files uploaded by the tools in a well-specified location (shared folders) or via FTP.

This approach requires defining at least the following:

File format – the information must be uploaded in a structured and well-established format.
Upload period – the upload service implemented at the level of the tool prepares and transmits the information with a well-established frequency.
Errors – on the integration platform side, files are processed and some (part) of them can have errors, for instance, the format is wrong. For this situation, the integration platform will produce error files, also in a well-specified format, linked with the original file, from which the tool can find what was wrong.

6.2.5 ROLES AND COMPETENCIES OF THE INTEGRATION TEAM

The following roles and responsibilities are proposed for the integration of the platform:

- **Integration manager**. Responsible for the whole integration of the system (WP7 leader)
- **Concepts integration manager**. Responsible for the definition of the uniform concepts used by all components (WP3 leader)
- **Data integration manager**. Responsible for the definition of data structures and mechanisms used between modules. (WP4 leader)
- **Service integration manager**. Responsible for the definition of communication protoco9ls between modules (WP5 leader)
- **Presentation integration manager**. Responsible for the definition of presentation themes. (WP8 Leader)
- **Test integration manager**. Responsible for the integration tests. (WP7 leader)

Also, for each module, there will be the responsibility to comply with the integration rules, protocols, structures, and mechanisms defined.

6.2.6 METHODS USED TO PERFORM INTEGRATION

Under our research for choosing the appropriate approach to integration *(conceptual, methodological, logical, and technological)*, the following aspects were relieved: the need for a uniform approach to the integration process together with the appropriate process.
The need for a uniform approach to the integration process, in line with the principles of integration, as depicted hereinbefore and fully compliant with the reference architecture of Pantheon solution is outstanding.

Moreover, the appropriate process of integration should follow the steps below:
Use case definition. The definition of Use Cases is oriented on the formalization of the integration requirements. The uniformity of the requirements within the proposed use cases is achieved on the conceptual level on each architectural layer (L1 to L5 defined in 3.1):
Core functions. These core functional modules are applicable and potentially re-usable by other actors of the Pantheon platform. The core functional modules will become accessible to other actors from in Pantheon and have the facilities provided by the Pantheon platform for their own data set infused/registered into the Pantheon.
Framework of tools. The framework of tools (integration platform) represents the possibility of evaluating the results of the Pantheon project in a tangible, not only conceptual way. Implementation of the integration platform is achieved through the technologies hosted by the platform.

7. PLATFORM DEPLOYMENT

The PANTHEON architecture depicted in Figure 2 corresponds to a single and specific installation of the PANTHEON platform.

7.1 PANTHEON DEPLOYMENT TYPES

The core components of the PANTHEON platform are the same regardless of the characteristics of the deployment type. The following possible types of deployments are foreseen in PANTHEON:

- On-premises: This is one of the main target deployment sites. As a deployment site, the first responder organisation's premises is a challenging environment since multiple organisation restrictions are taking place (mostly related to organisation's privacy due to data sensitivity) and the PANTHEON platform would need to relate to the internal first responder organisation's databases to get access to organisational data. One big drawback of this type is the limited availability of resources to run locally,
- Cloud: This environment is the most prominent for the PANTHEON platform deployment. The platform is installed in cloud virtual machines, with the purpose of offering enough resources to run data- and resource- intensive simulations along with ML models. Back-end and front-end applications, AI algorithms and databases are among the kinds of resources that can be deployed in such a cloud PANTHEON platform. In this deployment, access to the first responder organisation's sensitive data won't be possible, so PANTHEON will probably need to work with simulated data and open-source 3rd party DRM platforms (e.g. SAHANA EDEN[51]).
- Hybrid: This type of deployment is combining the availability of resources in the cloud with the availability of sensitive data from the first responder organisations. The use of such a deployment is the most technically challenging (e.g., networking with VPN, scheduling, latency etc) and a feasibility check will need to be performed prior to proceeding with such a type.

7.2 PANTHEON CLOUD DEPLOYMENT CONSIDERATIONS

If the cloud deployment type is selected, private or public cloud VMs will be used to remotely host the PANTHEON platform. The base of the platform instance will be either a Docker Swarm (as described in detail in section 3.3) or a Kubernetes cluster. To accomplish that goal several VMs would need to be provisioned to create a functional cluster. The number of VMs and the resources allocated for each VM will vary depending on the platform needs and the workloads that will run in the cluster.

In case Kubernetes is chosen as the deployment platform, a minimum Kubernetes cluster would include:

- A master node. This is where the control plane and etcd services will run. No workloads are recommended to be run in this type of nodes. Optionally another node can be provisioned to provide high availability.
- Two worker nodes. This type of nodes is where the workloads are executed. Most likely more than just two nodes will be necessary for the PANTHEON platform. Adding more nodes to a running cluster is feasible without much hassle.

[51] https://sahanafoundation.org

Additionally, it is also recommendable to have a management platform for the cluster such as Rancher[52]. That would increase the number of necessary VMs. Nevertheless, where regulations permit the use of public cloud services, it is an option to be considered, although traditionally first responder organisations are very reluctant to deploy IT services in the public cloud. Most major cloud infrastructure providers offer a managed Kubernetes cluster service (EKS[53], AKS[54], GKE[55], OKE[56]). In addition, these providers also offer the option to connect to their services via a site-to-site VPN or use a direct connection (AWS Direct Connect[57], Azure ExpressRoute[58], Google Cloud Interconnect[59], Oracle Cloud Infrastructure FastConnect[60]) which makes connectivity a lot faster than through a VPN.

[52] Rancher, https://rancher.com/
[53] Elastic Kubernetes Service, https://aws.amazon.com/eks/
[54] Azure Kubernetes Service, https://azure.microsoft.com/en-us/services/kubernetes-service/
[55] Google Kubernetes Engine, https://cloud.google.com/kubernetes-engine
[56] Oracle Kubernetes Engine, https://www.oracle.com/es/cloud-native/container-engine-kubernetes/
[57] https://aws.amazon.com/directconnect/
[58] https://azure.microsoft.com/en-us/services/expressroute/
[59] https://cloud.google.com/hybrid-connectivity
[60] https://www.oracle.com/cloud/networking/fastconnect/

8. PLATFORM VERIFICATION

Platform verification is the subject of task: Task 4.6: Verification, Validation and Accreditation of SCDT Platform and Simulation Model. This task will cover all aspects of verification and validation.

Platform verification will also be part of the testing activities Task 7.5: Full-scale deployments, operational testing and refinement. The aspects related to platform verification presented in the present document are limited to the ones influencing the system architecture of the system. For this reason, we are presenting here the general aspects related to:

- The specifications of objectives of the verification process
- The planning of the verification process
- The global content of verification activities.

Objectives of the verification process.

The objectives of the verification process are to check if the complete system, offers end to end functionality, able to cover all the requirements (functional and technical) and complies with the performance, accuracy, security and reliability characteristics.

The objectives will follow to prove the reach of the KPIs established in task T4.6 and grouped in:

- Operational usability and effectiveness vs standard simulation interaction interfaces;
    - social infrastructures in disaster management and their interactions with physical infrastructures;
    - potential for simulation to forecast future conditions in given multi-hazards scenarios
- Data used
    - number of data sources;
    - data of relevance to be generated;
    - number of remote sensors;
    - quality and validity of data
- Integration aspects
    - Time required to render Operational Picture;
    - Ease of use / end-user satisfaction;
    - Number of integrated components;
    - Random Access Memory (RAM) Utilization;
    - Central Processing Unit (CPU) Utilization;
    - Network (Input/Output) Observed Traffic;
    - Cloud Deployment Cost (per month/year).

The level of each KPI is subject to next tasks (especially T4.6)

Planification of the verification

The verification will be an iterative process, with each version of the platform being verified. The outputs of the verification process will be used in the development of the next versions.

The first iteration will include:

- Definition of the methodology used. It will be informed by the best practices identified.
- Definition of the KPIs.

- Selection of verification tools. They will include tests developed in the programming language used (java unit tests or Python unit tests), Selenium[61], Jmeter[62], Jira[63].
- Selection of testing methods to be used. They will include Unit testing, Manual testing, Automatic testing, testing at limits.

Verification activities

- **Functional Testing**: Verify that the software platform meets its functional requirements through various testing techniques such as unit testing, integration testing, system testing, and user acceptance testing (UAT). This involves testing individual components as well as the integrated system.
- **Security Testing**: Evaluate the security of the software platform to identify and mitigate potential vulnerabilities and threats. This may involve penetration testing, vulnerability scanning, code analysis, and security reviews.
- **Compatibility Testing**: Ensure the software modules can interact and exchange data between them. The tests will be done on the integrated system.
- **Usability Testing**: Assess the usability of the software platform to ensure it is intuitive and easy to use for its intended users. Usability testing involves gathering feedback from end-users through surveys, interviews, and usability testing sessions.
- **Regression Testing**: Perform regression testing to verify that new changes or updates to the software platform have not introduced any unintended side effects or regressions in existing functionality. This will ap0ply to individual modules and to the integrated system also.
- **Performance Testing**: Assess the performance of the software platform under different load conditions to ensure it can handle expected workloads. Performance testing may include load testing, stress testing, scalability testing, and endurance testing. The tests will be done with data acquired or simulated data.
- **Compliance Verification**: Ensure the software platform complies with relevant standards, regulations, and industry best practices. The standard considered here is IEEE 1730.2. (IEEE, 2022)
- **Feedback Collection and Iteration**: Gather feedback from stakeholders, including end-users, developers, and testers, and use it to iterate on the software platform to address any identified issues or areas for improvement. The feedback will be collected and registered in an issue tracking mechanism (as Jira[64])
- **Final Verification and Approval**: Once all verification activities have been completed satisfactorily, finalize the verification process, and obtain approval from relevant stakeholders to release the software platform for production use. (This will be part of task T7.5)

[61] https://www.selenium.dev/
[62] https://jmeter.apache.org/
[63] https://www.atlassian.com/software/jira
[64] https://www.atlassian.com/software/jira

9. CONCLUSIONS

This deliverable elaborates on the design of the high-level reference architecture and the technical specifications along with a description of each technical component. It describes the architecture and preliminary integration, deployment and verification plan as well as considerations to facilitate the kick-start of the implementation phase of the PANTHEON platform.

Specifically, the work done was based in D2.4, D3.1, D3.2, D3.3, D3.4 and D3.6, by further specifying: (i) the necessary system requirements to fulfil the user requirements described in D3.2 and D3.6 and (ii) the technical components that comprise the architecture along with the sequence diagrams of their interconnection after analysis on: (a) the existing technological landscape (D2.4 & D3.1), (b) the conceptual model of the SCDT (D3.3), (c) the data delivery schemes in D3.4 and (d) the usage scenarios and use cases as documented in D3.6.

Based on the user requirements as documented in D3.2 and D3.6, the system requirements were created that would further facilitate the design of the high-level reference architecture of the PANTHEON platform. For the platform, two views were created; (i) the layered view which helps encapsulate and decouple the PANTHEON platform by grouping the distinct segments using their functional role and (ii) the functional view which defines the architectural elements that deliver the functions of the system being described and documents the system's functional structure, including the key functional elements, their responsibilities, the interfaces they expose, and the interactions between them.

From the description of the components, we created the operation flows for the core procedures (user authorisation and creation) as well as the PANTHEON usage scenarios.

Finally, this deliverable incorporates a preliminary version of the integration, deployment and verification plans and considerations, which will be further elaborated and documented in future relevant deliverables (D4.5, D7.1, D7.2 and D7.4).