UNIVERSITY OF PATRAS

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

DIVISION OF TELECOMMUNICATIONS & INFORMATION TECHNOLOGY

LABORATORY OF WIRE COMMUNICATIONS & INFORMATION TECHNOLOGY

LEVERAGING DATA SPACES FOR COORDINATED DISASTER RESPONSE

D I P L O M A T H E S I S

KONSTANTINOS TSAMPRAS

SUPERVISOR: Spyros Denazis

PATRAS – October, 2025

University of Patras, Department of Electrical and Computer Engineering.

Konstantinos Tsampras

CERTIFICATION

It is certified that the Diploma Thesis titled

LEVERAGING DATA SPACES FOR COORDINATED DISASTER RESPONSE

of the Department of Electrical and Computer Engineering student

KONSTANTINOS TSAMPRAS

Registration Number: 1083865

was presented publicly at the Department of Electrical and Computer Engineering at

15/10/2025

and was examined by the following examining committee:

Spyros Denazis, Professor, Department of Electrical and Computer Engineering (supervisor)

Ioannis Tomkos, Professor, Department of Electrical and Computer Engineering (committee member)

Michael Logothetis, Professor, Department of Electrical and Computer Engineering (committee member)

The Supervisor	The Director of the Division
Spyros Denazis Professor	Stavros Koulouridis Professor

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΑΣ

ΕΡΓΑΣΤΗΡΙΟ ΕΝΣΥΡΜΑΤΗΣ ΤΗΛΕΠΙΚΟΙΝΩΝΙΑΣ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑ ΤΗΣ ΠΛΗΡΟΦΟΡΙΑΣ

ΑΞΙΟΠΟΙΗΣΗ ΧΩΡΩΝ ΔΕΔΟΜΕΝΩΝ ΓΙΑ ΣΥΝΤΟΝΙΣΜΕΝΗ ΑΝΤΙΜΕΤΩΠΙΣΗ ΦΥΣΙΚΩΝ ΚΑΤΑΣΤΡΟΦΩΝ

Δ Ι Π Λ Ω Μ Α Τ Ι Κ Η Ε Ρ Γ Α Σ Ι Α

ΚΩΝΣΤΑΝΤΙΝΟΣ ΤΣΑΜΠΡΑΣ

ΕΠΙΒΛΕΠΩΝ: ΣΠΥΡΟΣ ΔΕΝΑΖΗΣ

ΠΑΤΡΑ – ΟΚΤΩΒΡΙΟΣ 2025

Πανεπιστήμιο Πατρών, Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών.

Κωνσταντίνος Τσάμπρας

ΠΙΣΤΟΠΟΙΗΣΗ

Πιστοποιείται ότι η Διπλωματική Εργασία με τίτλο

ΑΞΙΟΠΟΙΗΣΗ ΧΩΡΩΝ ΔΕΔΟΜΕΝΩΝ ΓΙΑ ΣΥΝΤΟΝΙΣΜΕΝΗ ΑΝΤΙΜΕΤΩΠΙΣΗ ΦΥΣΙΚΩΝ ΚΑΤΑΣΤΡΟΦΩΝ

του φοιτητή του Τμήματος Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών

ΚΩΝΣΤΑΝΤΙΝΟΣ ΤΣΑΜΠΡΑΣ ΤΟΥ **ΛΑΜΠΡΟΥ**

Αριθμός Μητρώου: 1083865

Παρουσιάστηκε δημόσια στο Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών στις

15/10/2025

και εξετάστηκε από την ακόλουθη εξεταστική επιτροπή:

Σπύρος Δενάζης, Καθηγητής, Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών (επιβλέπων)

Ιωάννης Τόμκος, Καθηγητής, Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών (μέλος επιτροπής)

Μιχαήλ Λογοθέτης, Καθηγητής, Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών (μέλος επιτροπής)

Ο Επιβλέπων	Ο διευθυντής του Τομέα
Σπύρος Δενάζης Καθηγητής	Σταύρος Κουλουρίδης Καθηγητής

PREFACE

I would like to express my gratitude to my family and loved ones for the support they have given me and continue to provide. I would also like to thank my supervisor and the members of the laboratory for their guidance throughout the preparation of this thesis.

ABSTRACT

LEVERAGING DATA SPACES FOR COORDINATED DISASTER RESPONSE

KONSTANTINOS TSAMPRAS: **SPYROS DENAZIS:**

The modern digital world is flooded with data sources, and their utilization requires the adoption of integrated solutions that enable, organize, and secure data exchanges while maintaining the sovereignty of data owners over their assets. At the same time, new information regarding climate change developments makes it necessary for organizations to collaborate and take initiatives focused on citizen protection. The subject of this thesis is the study of the field of Data Spaces, the installation and configuration (with full documentation) of a Data Space, and the development of a demo application aimed at assisting authorities in organizing the evacuation of people with disabilities from high-risk areas in cases of natural disasters.

Keywords: Data Space, IDSA, IDS Testbed, Reference Architecture Model (RAM), Data Sovereignty, Interoperability, Disaster Management, Evacuation Planning, People with Disabilities, FIWARE, Google Cloud Fleet Routing (CFR).

ΕΚΤΕΤΑΜΕΝΗ ΕΛΛΗΝΙΚΗ ΠΕΡΙΛΗΨΗ

ΑΞΙΟΠΟΙΗΣΗ ΧΩΡΩΝ ΔΕΔΟΜΕΝΩΝ ΓΙΑ ΣΥΝΤΟΝΙΣΜΕΝΗ ΑΝΤΙΜΕΤΩΠΙΣΗ ΦΥΣΙΚΩΝ ΚΑΤΑΣΤΡΟΦΩΝ

ΚΩΝΣΤΑΝΤΙΝΟΣ ΤΣΑΜΠΡΑΣ: **ΣΠΥΡΟΣ ΔΕΝΑΖΗΣ:**

Οι σύγχρονες κοινωνίες κατακλύζονται από τεράστιους όγκους δεδομένων, καθώς και πληθώρα πηγών αυτών. Όμως, τα δεδομένα αυτά είναι συχνά δυσεύρετα και δεν παρέχεται ικανοποιητική περιγραφή για το είδος των δεδομένων, την ποιότητά τους, τους τρόπους πρόσβασης και τους όρους χρήσης τους.

Από την άλλη πλευρά, οι ιδιοκτήτες των δεδομένων, συχνά, δεν επιθυμούν να παραδώσουν τα δεδομένα τους σε οικοσυστήματα όπου δεν μπορούν να ελέγξουν τον τρόπο χρήσης τους και δεν μπορούν να εγγυηθούν τον ασφαλή διαμοιρασμό τους.

Κεντρικοποιημένες λύσεις, όπως κεντρικές βάσεις δεδομένων ή κεντρικοί brokers δεδομένων δίνουν λύσεις σε λίγα μόνο από τα υπάρχοντα προβλήματα.

1. Απαιτούν δημιουργία και συντήρηση υποδομής μεγάλου κόστους για τον διαμοιρασμό μεγάλου όγκου δεδομένων σε πολλούς ενδιαφερόμενους.
2. Βασιζόμενοι σε κεντρικοποιημένα συστήματα, τυχόν βλάβες σε ένα μόνο σημείο καθιστούν ολόκληρο το σύστημα μη λειτουργικό.
3. Η ενοποίηση όλων των δεδομένων σε ένα σύστημα απαιτεί τη χρήση ενός συγκεκριμένου και αυστηρού μοντέλου για κάθε είδος δεδομένων, κάτι που επιβαρύνει τους παρόχους δεδομένων με τη μετάφραση υπαρχόντων αποθηκευτικών συστημάτων με δεδομένα σε μια νέα μορφή. Αυτό απωθεί τους παρόχους από την συνεργασία με τέτοιες πρωτοβουλίες. Ταυτόχρονα, η αυστηρή ενοποίηση μοντέλων μπορεί να οδηγήσει σε απώλεια πληροφορίας.
4. Οι περισσότερες λύσεις δεν προσφέρουν ικανοποιητικές εγγυήσεις για τη διατήρηση της κυριαρχίας των δεδομένων από τους ιδιοκτήτες τους, αφού δεν προσφέρουν ιχνηλασιμότητα στις ανταλλαγές δεδομένων.
5. Πέρα από κοινές τεχνικές αυθεντικοποίησης, δεν παρέχεται ευελιξία στην πολιτική πρόσβασης δεδομένων.

Σε αυτή την πραγματικότητα η τεχνολογία των Data Spaces δίνει λύσεις στα κύρια προβλήματα που αντιμετωπίζει η κοινότητα ανταλλαγής δεδομένων.

Ένα Data Space είναι ένα οικοσύστημα εύρεσης και ανταλλαγής δεδομένων το οποίο εκπληρώνει τους παρακάτω κύριους στόχους:

1. Κυριαρχία δεδομένων: Οι ιδιοκτήτες των δεδομένων διατηρούν πλήρη έλεγχο των δεδομένων τους, ορίζοντας ποιος, υπό ποιες συνθήκες, για πόσο χρονικό διάστημα και με ποιο αντίτιμο, μπορεί να έχει πρόσβαση στα δεδομένα τους.
2. Διαλειτουργικότητα: Δεδομένα από ετερογενή μοντέλα συνυπάρχουν στο ίδιο οικοσύστημα, το οποίο με περιγραφή των μοντέλων και εμπλουτισμό με μεταδεδομένα, επιτρέπει σε τρίτους να χρησιμοποιούν τα δεδομένα με επαρκή τεκμηρίωση.
3. Ασφάλεια και εμπιστοσύνη: Κάθε ανταλλαγή δεδομένων συνοδεύεται από μηχανισμούς αυθεντικοποίησης, πολιτικές πρόσβασης και ιχνηλασιμότητα διασφαλίζοντας συμμόρφωση με νομικά πλαίσια.

4. Επεκτασιμότητα και ανθεκτικότητα: Η αποκεντρωμένη φύση των Data Spaces διασφαλίζει την δυνατότητα επέκτασης του οικοσυστήματος χωρίς να επιβαρύνεται ένα συγκεκριμένο σημείο, ενώ αποτρέπει και την αποτυχία ενός σημείου (single point of failure)

Ταυτόχρονα με την απουσία οργανωμένων οικοσυστημάτων δεδομένων, παρατηρείται και έλλειψη οργάνωσης κατά την εκκένωση περιοχών ιδίως όσον αφορά ευάλωτους πολίτες. Η απλή ενημέρωση -μέσω ενός γενικού μηνύματος- ενός ατόμου με αναπηρία ότι πρέπει άμεσα να εγκαταλείψει τον χώρο διαμονής του σε σύντομο χρονικό διάστημα, αγνοώντας τις ιδιαιτερότητες του ατόμου, δεν είναι μια ικανοποιητική πολιτική για μια κοινωνία που έχει τεράστιες δυνατότητες επεξεργασίας δεδομένων.

Με βάση τις παραπάνω παρατηρήσεις, προκύπτει το συμπέρασμα ότι η χρήση ενός Data Space για την εύρεση και αξιοποίηση ετερόκλητων δεδομένων που θα βοηθούν τις αρχές στην συντονισμένη οργάνωση εκκενώσεων ατόμων με ειδικές ανάγκες σε περιπτώσεις φυσικών καταστροφών θα ήταν ιδανική. Η προσήλωση στη διατήρηση της κυριαρχίας των προσωπικών δεδομένων και η διευκόλυνση εύρεσης και διαχείρισης διαφορετικών μοντέλων δεδομένων σε ένα Data Space απαντούν στις εύλογες ανησυχίες που εγείρονται για ένα τέτοιο εγχείρημα.

Ως εκ τούτου, οι κύριοι στόχοι της παρούσας διπλωματικής είναι:

1. Έρευνα στο περιβάλλον των Data Spaces και καταγραφή των διαδικασιών και τυχόν προβλημάτων ή ελλείψεων.
2. Εγκατάσταση και ρύθμιση ενός Data Space για την φιλοξενία ποικιλόμορφων δεδομένων γύρω από την εκκένωση περιοχών σε φυσικές καταστροφές.
3. Ανάπτυξη μιας δοκιμαστικής εφαρμογής η οποία θα καταναλώνει δεδομένα εντός του Data Space και θα συντονίζει την προσπάθεια εκκένωσης ατόμων με ειδικές ανάγκες από τις αρχές.

Για την επίτευξη των στόχων αυτών πραγματοποιήθηκε έρευνα σε πρωτοβουλίες δημιουργίας Data Spaces (Mobility Data Space, Smart Connected Supplier Network, Health-X dataLOFT), στον οργανισμό International Data Spaces Association (IDS/IDSA), και στην εγκατάσταση του IDS-Testbed ως κορμό της παρακάτω αρχιτεκτονικής για την υλοποίηση του περιβάλλοντος της δοκιμαστικής εφαρμογής.

Η αρχιτεκτονική αποτελείται από την ροή της αναζήτησης και παροχής πρόσβασης σε δεδομένα:

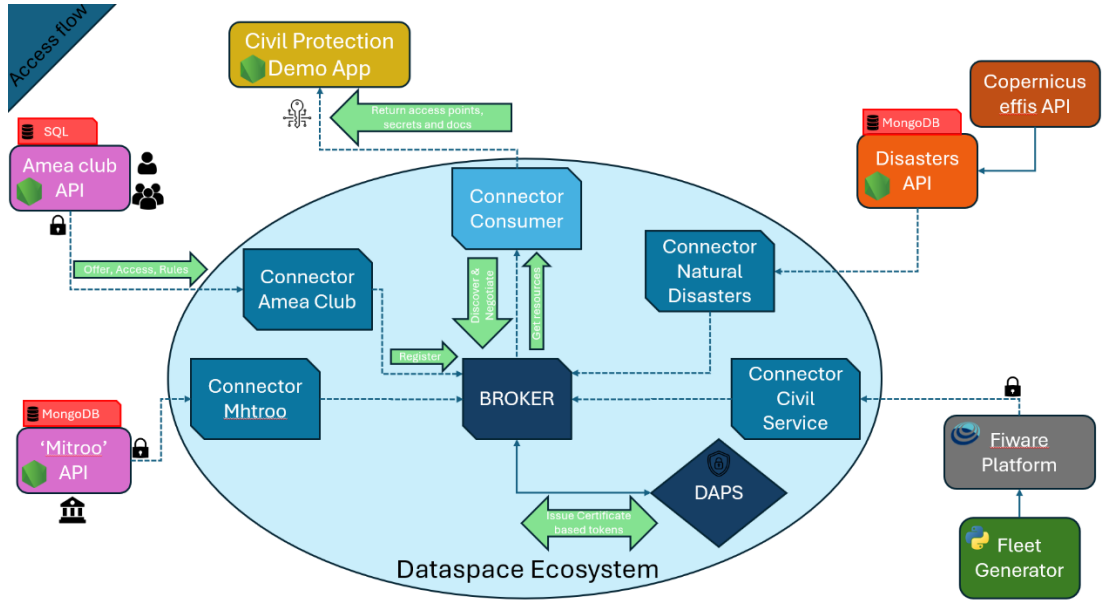


Figure 1 Ροή αδειοδότησης στο οικοσύστημα

Καθώς και από την ροή πρόσβασης στα δεδομένα:

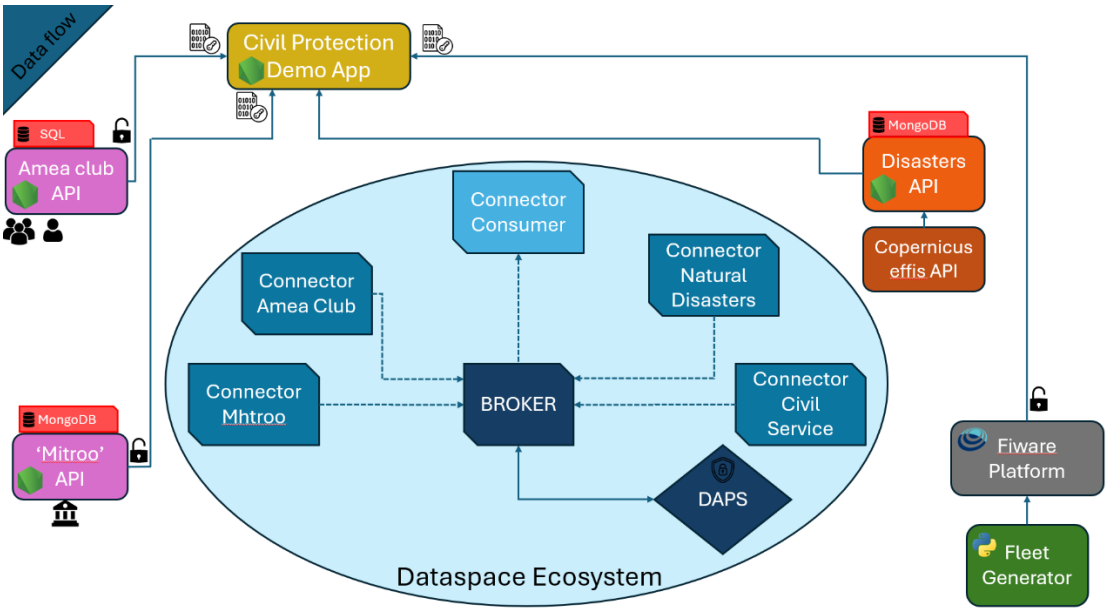


Figure 2 Ροή δεδομένων στο οικοσύστημα

Όπως γίνεται εύκολα αντιληπτό, κεντρικός πυλώνας της αρχιτεκτονικής είναι το οικοσύστημα του Data Space, όπου οι διάφορες πηγές δεδομένων παρέχουν ή αποκτούν πρόσβαση σε δεδομένα μέσω των Συνδετήρων (Connectors).

Πραγματοποιώντας αναζητήσεις στον διαμοιραστή μεταδεδομένων (Broker) η εφαρμογή εντοπίζει πηγές δεδομένων που την εξυπηρετούν και αποκτά πρόσβαση σε αυτές μέσω των διαπιστευτηρίων τους (Access token, encryption key). Ενώ η ασφάλεια διασφαλίζεται

μέσω του Συστήματος Παροχής Δυναμικών Χαρακτηριστικών (DAPS) το οποίο ελέγχει τα πιστοποιητικά των συμμετεχόντων και τους παρέχει tokens για την συμμετοχή τους στο οικοσύστημα.

Έχουν υλοποιηθεί APIs με τους εξής ρόλους:

1. Προσομοίωση δεδομένων για άτομα με ειδικές ανάγκες που έχουν εγγραφεί σε συλλόγους και δήμους (Ameaclub API). Αποθηκεύει και διαμοιράζει κρυπτογραφημένα δεδομένα, ενώ πρόσβαση δίνεται με access tokens.
2. Προσομοίωση δεδομένων για άτομα με ειδικές ανάγκες που βρίσκονται στο κεντρικό μητρώο του κράτους (Mhtroo API). Αποθηκεύει και διαμοιράζει κρυπτογραφημένα δεδομένα, ενώ πρόσβαση δίνεται με access tokens.
3. Προσομοίωση δεδομένων για τον στόλο οχημάτων (ασθενοφόρα, πυροσβεστικά, περιπολικά) και αποστολή τους σε πλατφόρμα FIWARE, ιδανική για την επικαιροποίηση δεδομένων οχημάτων σε πραγματικό χρόνο (Fleet generator).
4. Άντληση και παροχή πραγματικών δεδομένων από το EFFIS API του οργανισμού Copernicus για φυσικές καταστροφές (Disasters API).

Με βάση την παραπάνω αρχιτεκτονική υλοποιήθηκε η δοκιμαστική εφαρμογή (Civil Protection demo App) η οποία απευθύνεται στο συντονιστικό τμήμα της πολιτικής προστασίας (ή κάθε άλλης αρχής υπεύθυνης για την εκκένωση περιοχών σε περιπτώσεις φυσικών καταστροφών).

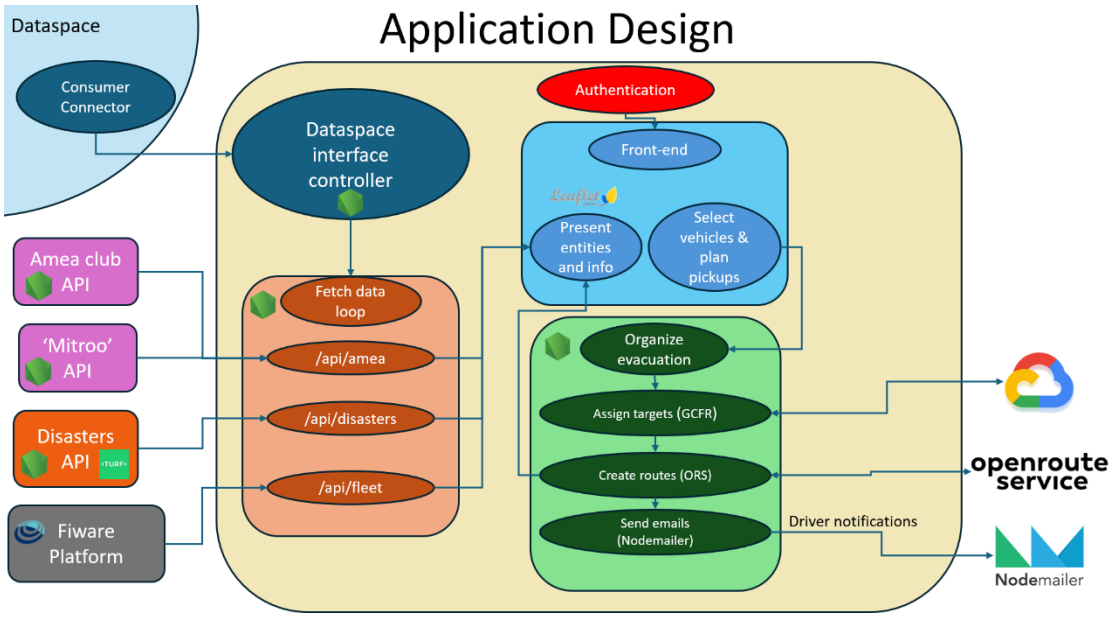


Figure 3 Αρχιτεκτονική της Εφαρμογής

Η δοκιμαστική εφαρμογή αποτελείται από μια επισκόπηση της εκάστοτε κατάστασης, παρουσιάζοντας στον χάρτη τις φυσικές καταστροφές, την περιοχή εκκένωσης πέριξ αυτών, τα οχήματα του στόλου καθώς και τις δηλωμένες οικίες των ατόμων με ειδικές ανάγκες.

Επιλέγοντας την κάθε οντότητα εμφανίζονται σημαντικές πληροφορίες για αυτήν.

Παράλληλα, ο διαχειριστής έχει την δυνατότητα να επιλέξει τα οχήματα τα οποία θα συμμετέχουν στην προσπάθεια εκκένωσης των ατόμων με ειδικές ανάγκες στις περιοχές εκκένωσης.

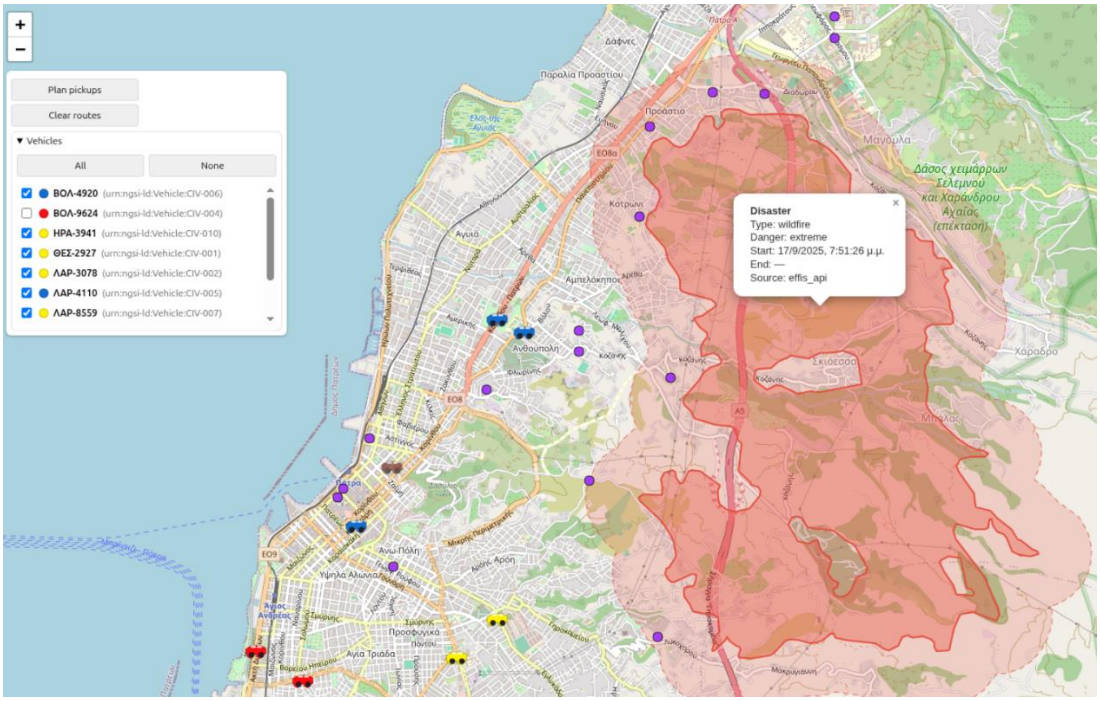


Figure 4 Εικόνα της εφαρμογής

Μετά την επιλογή των οχημάτων που θα συμμετέχουν στην εκκένωση, ο χειριστής μπορεί να επιλέξει την δημιουργία σχεδίου εκκένωσης και στην συνέχεια να εποπτεύσει τις διαδρομές που ανατέθηκαν σε κάθε όχημα, και τα σημεία όπου θα συναντήσει πολίτες με ειδικές ανάγκες.

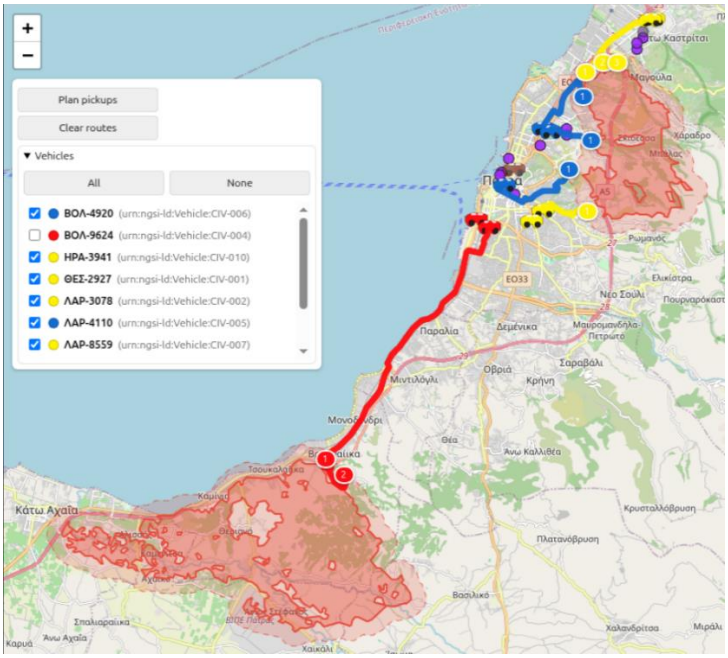
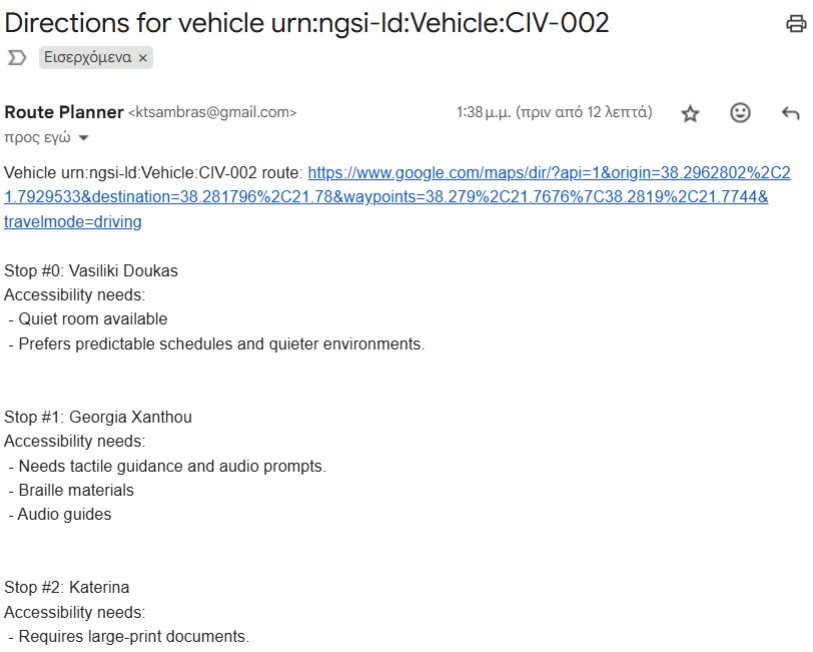


Figure 5 Δημιουργία σχεδίου εκκένωσης

Ταυτόχρονα αποστέλλεται στον κάθε χειριστή οχήματος email με αναλυτικές οδηγίες και πληροφορίες για τα άτομα που θα κληθεί να βοηθήσει στην διαδρομή αυτή.


Directions for vehicle urn:ngsi-ld:Vehicle:CIV-002

Εισερχόμενα ×

Route Planner <ktsambras@gmail.com> 1:38 μ.μ. (πριν από 12 λεπτά)

προς εγώ

Vehicle urn:ngsi-ld:Vehicle:CIV-002 route: https://www.google.com/maps/dir/?api=1&origin=38.2962802%2C21.7929533&destination=38.281796%2C21.78&waypoints=38.279%2C21.7676%7C38.2819%2C21.7744&travelmode=driving

Stop #0: Vasiliki Doukas
Accessibility needs:
- Quiet room available
- Prefers predictable schedules and quieter environments.

Stop #1: Georgia Xanthou
Accessibility needs:
- Needs tactile guidance and audio prompts.
- Braille materials
- Audio guides

Stop #2: Katerina
Accessibility needs:
- Requires large-print documents.

Figure 6 Αποστολή πληροφοριών σε χειριστές οχημάτων

Contents

1 Introduction

1.1 Motivation

In the age of information, we have witnessed an explosion in the production of datasets from diverse sources. Government agencies, private organizations, IoT devices, and even citizens themselves generate large volumes of data every single day. Despite the meteoric rise in the volume of data, data discovery has not become easier. Challenges in discovery, lack of descriptive metadata and documentation, and lack of access and usage policies become roadblocks in new initiatives that wish to utilize data to produce societal benefit.

On the other hand, data owners do not wish to simply waive their rights to their data and are unwilling to relinquish their data without having any control over how it is used.

Centralized systems like databases or data brokers have proven valuable in managing data, yet they suffer from critical limitations. They are effective at aggregating information in a single place, but this very centralization introduces high infrastructure costs, vulnerability to single points of failure, and strict requirements for data integration. Data providers are often forced to translate their models into a new, unforgiving format that often leads to the loss of information that was unique to the original data source.

Moreover, such centralized architectures do not usually offer many assurances regarding data sovereignty. The data owners are asked to upload their data onto a large dataset, losing visibility and control over who accesses it, under what terms, and for what purpose.

The motivation for this thesis stems from the need for a complete solution to these issues. That solution is the architecture of Data Spaces.

Data Spaces address the centralization concerns by employing a decentralized, but federated, architecture. In a Data Space, the data remain with their respective owners, who do not need to relinquish control of their data by blindly trusting a central authority to store and provide it to third parties.

Interoperability is achieved by allowing heterogeneous models to coexist. Metadata enrichment, as well as the use of vocabularies, allow for the integration of different systems into the Data Space ecosystem. This preserves the integrity of the original data, but also reduces the burden on providers, who do not need to translate historical or live datasets into a rigid schema.

Security and trust are built into the architecture through mechanisms such as authentication, encryption, certificate-based token issuing, and traceability. Every data

exchange is logged and can be traced, providing a secure framework for accountability and compliance.

To demonstrate the core features of a Data Space, while also providing societal value, this thesis focuses on its application in the field of disaster management. Natural disasters are increasing both in frequency and in severity due to climate change, which makes it necessary to make use of the aforementioned abundance of data to help the people in need when nature strikes.

1.2 Research Objectives

The goal of this thesis is to explore how Data Spaces can be used to enable secure and coordinated data sharing during natural disaster response. The work goes beyond theory by also showing their practical value through a real-world demonstration.

The research objectives are as follows:

1. Examine the current state of the Data Space ecosystem by reviewing existing initiatives and architectures.
2. Deploy and configure a Data Space environment to support heterogeneous data sources, including sensitive data.
3. Develop APIs and data Connectors that simulate or integrate real-world data providers (such as people with disabilities records, clubs, municipalities, fleet data, and natural disaster information)
4. Design and implement a demonstration application that consumes Data Space data and coordinates authorities and relevant agencies in evacuating people with disabilities.

1.3 Thesis Structure

Chapter 1 introduces the thesis by presenting the motivation, research objectives, and overall structure. It notes the advantages of Data Spaces and the societal need for initiatives in the field of natural disasters and protecting people with disabilities.

Chapter 2 provides a comprehensive literature review of the International Data Spaces Association's role, the theoretical aspect of Data Spaces, their architecture and key elements, while also presenting successful initiatives in the field.

Chapter 3 describes the methodology used to conduct this study, by providing the research design, and the key requirements behind the implementation of a project handling sensitive data.

Chapter 4 examines the main platforms used to power the entire implementation. It focuses on their main features, and how they were used in this current project.

Chapter 5 focuses on the data and the APIs developed and used in the prototype. It explains the role of the simulated entities (government, clubs, agencies) providing data and the use of real data when it comes to natural disasters, from the EFFIS API.

Chapter 6 dives into the IDS Testbed, the environment used to implement the Data Space. It covers the deployment, extension and configuration of the Testbed, addressing the addition of new Connectors to it and common issues. It serves as a detailed guide for future Data Space deployments.

Chapter 7 presents the implementation of the demo application. It explains the system's architecture, the technologies employed and showcases its features such as information retrieval, evacuation planning and providing information to vehicle operators.

Chapter 8 concludes the thesis by summarizing its contributions, evaluating the prototype and reflecting on limitations. With emphasis on the facilitation of secure and sovereign data exchanges it discusses scalability challenges and proposes directions for future work and wider adoption of Data Spaces.

Chapter 9 lists the sources used in this document and in the implementation.

1.4 Contributions

In the pursuit of these objectives, the thesis delivers several contributions to the study and deployment of Data Spaces, as well as providing a practical example on how they can be used to help people in need.

The main contributions are outlined below:

1. Detailed review of Data Spaces and related initiatives. Data Spaces are not just a theoretical concept but have already been employed by cities, countries and world-wide (IDS, Mobility Data Space in Germany, Smart Connected Supplier Network, Health-X dataLOFT). This analysis serves as a foundation for further research in the field.
2. Documentation of the deployment of a functional Data Space environment. Instructions on how to set up and deploy the IDS-Testbed, along with warnings about common issues or faulty instructions have been compiled, to serve as a guide for future Data Space deployment.
3. The development of the demo app for coordinating evacuation efforts in disaster response scenarios will serve as a working small-scale example of what government agencies, organizations, and businesses can achieve by employing Data Spaces (and data sharing in general) to serve society.

In summary, this thesis not only examines the current state of Data Spaces but also provides practical guidance for their deployment and a concrete demonstration of their potential in disaster management. Together, these contributions form a basis for further research and highlight how Data Spaces can support inclusive and effective responses to real-world emergencies.

2 Literature Review

2.1 Introduction

The concept of Data Spaces emerged as the answer to considerations about the growing volume and diversity of data sources, while also dealing with demands for interoperable and sovereign data exchanges across different systems. However, it has not remained a concept, as several initiatives have pioneered the way into implementing that technology into real-world applications. The foundation for this progress has been laid by the [1] (IDSA) which has set standards and created a testbed for developing and deploying Data Space initiatives. Many cities, countries and organizations have picked up on that effort and deployed their Data Spaces adhering to the standards set forth by the IDS.

In this chapter, we will explore the architecture of a Data Space, the role of the IDS and the initiatives that thrive in this age of information.

2.2 International Data Spaces Association

The International Data Spaces Association is a non-profit organization, founded in 2016, that has pioneered in creating standards and working examples, turning the concepts into tangible frameworks for secure and sovereign data exchanges.

The model used to describe the standards is the IDS Reference Architecture Model (RAM) [2], which serves as a blueprint for constructing and operating Data Spaces.

The model is defined by its layers, and by its perspectives.

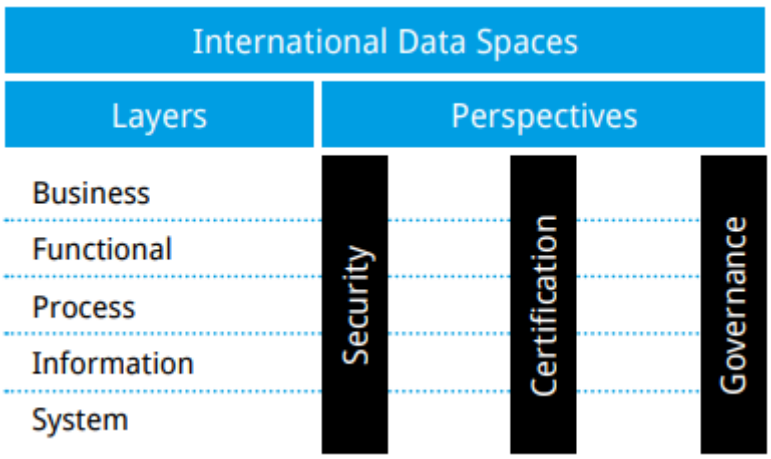


Figure 7 The correlation of Layers and Perspectives of the IDS RAM

2.2.1 Layers of the Reference Architecture Model

The layers of the model define levels of abstraction in the architecture. They range from most conceptual down to implementation. It uses a five-layer structure expressing various participants' concerns and viewpoints at different levels of granularity.

2.2.1.1 Business Layer

The business layer of the RAM describes the roles and key concepts of a Data Space, as well as how they interact with each other. These roles, along with some key concepts, are described below [2]:

Data owner: The entity holding the legal rights to data and is responsible for authorizing its sharing and usage (e.g. a person agreeing to share their car's telemetry data).

Data provider: The entity that supplies data to the Data Space on behalf of the owner with proper authorization (e.g. a car company collecting and sharing telemetry data through their servers).

Data consumer: The entity that searches for, accesses, and uses data available within the Data Space (e.g. an application utilizing telemetry data to predict traffic or accidents).

Data user: The individual or system that benefits from data-driven applications, possibly without interacting with the Data Space itself (e.g. a driver using the app to get notifications about possible traffic or accidents).

Broker service provider: Entity that enables data discovery by providing metadata and access mechanisms for available datasets.

Service provider: Entity that facilitates the data exchanges themselves. It is not a centralized entity, but rather participants in the Data Space (often the providers themselves) that also offer additional capabilities such as secure transfer channels, data transformation or analytics services.

Clearing House: Entity that maintains signed records of data exchanges, ensuring the parties involved follow their usage contracts and policies. A critical component of a Data Space that ensures auditability, billing capabilities and compliance checks.

Connector: The gateway that enables any third party to participate in the Data Space. It can be hosted directly by the data provider/consumer or, in some cases, by the Data Space through a Connector-as-a-Service model. There are two types of connectors: providers and consumers, although some Connectors can act as both. Provider Connectors are the ones that advertise and offer data into the Data Space. Consumer Connectors search for and get access to data offered. These two types of Connectors work together to negotiate the offered contracts and facilitate data exchanges.

Identity Provider: Issues and manages the basic digital identity of participants. Responsible for providing participants with certificates to prove who they are.

DAPS (Dynamic Attribute Provisioning Service): Builds on top of the identity provider's certificates by issuing tokens containing dynamic attributes that describe what each participant is allowed to do within the Data Space.

Certification Authority: Independent entity that certifies Connectors, participants, or services for compliance with IDS standards.

Vocabulary / Ontology Provider: The entity that provides sets of terms and definitions (vocabulary) that allows different systems to understand each other.

App Store / Marketplace Operator: Facilitates discovery of data services, applications, and connectors.

The IDSA offers a comprehensive table showcasing which roles interact with each other during certification or data exchanges in an International Data Space (IDS).

	Data Owner	Data Provider	Data Consumer	Data User	Broker	Clearing House	Identity Provider	Service Provider	App Provider	App Store	Vocabulary Provider	Certification Body	Evaluation Facility
Data Owner	-	X	-	-	-	(X)	-	(X)	(X)	(X)	(X)	-	(X)
Data Provider	X	-	X		X	(X)	X	(X)	(X)	(X)	(X)	-	X
Data Consumer	-	X	-	X	(X)	(X)	X	(X)	(X)	(X)	(X)	-	X
Data User	-	-	X	-	-	(X)	-	(X)	(X)	(X)	(X)	-	(X)
Broker	-	(X)	(X)	-	-	-	X	(X)	-	-	?	-	X
Clearing House	-	(X)	(X)	-	-	-	(X)	-	(X)	(X)	(X)	-	X
Identity Provider	-	X	X	-	X	(X)	Federation	-	(X)?	(X)?	-	-	X
Service Provider	(X)	(X)	(X)	(X)	(X)	-	-	-	(X)	(X)	(X)	-	X
App Provider	(X)	(X)	(X)	(X)	-	(X)	(X)	(X)	-	(X)	-	-	(X)
App Store	(X)	(X)	(X)	(X)	-	(X)	(X)?	(X)	(X)	-	(X)	-	(X)
Vocabulary Provider	(X)	(X)	(X)	(X)	?	(X)	-	(X)	(X)	(X)	-	-	X
Certification Body	-	-	-	-	-	-	-	-	-	-	-	-	X
Evaluation Facility	(X)	X	X	(X)	X	X	X	X	(X)	X	X	X	-

Figure 8 Table of interactions between roles in the IDS, X: mandatory, (X): optional [2]

While also offering a flow diagram describing the flow of data, metadata and exchange logging inside the IDS.

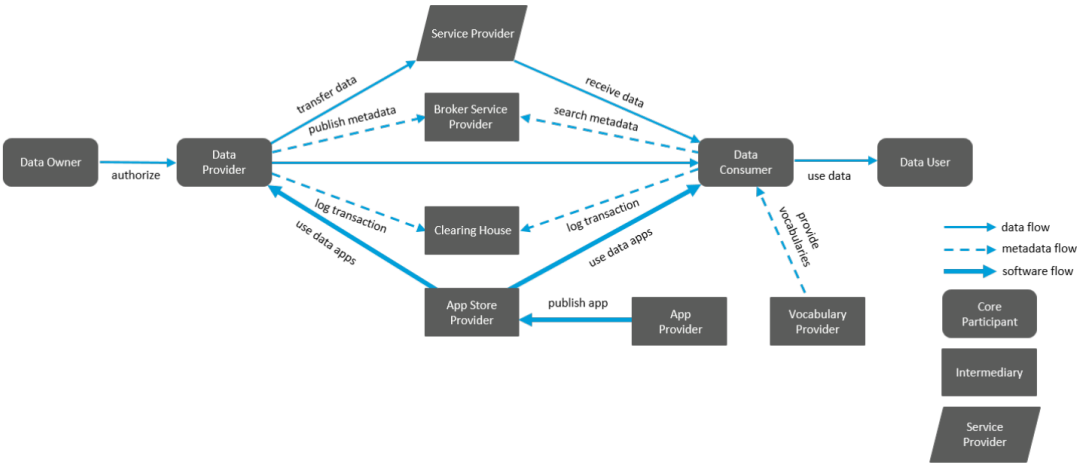


Figure 9 Diagram of interactions between roles in the IDS [2]

In the diagram above, we see how the Data Owner authorizes the Data Provider to advertise the data within the Data Space by publishing metadata in the Broker Service Provider.

At the same time, Data Consumers search for data by querying the Broker with specific metadata (like location, keywords, time).

The data transfer is facilitated by the Service Provider, and the Data Consumer can use Vocabularies to recognise the terms used by the Provider.

Each transaction is logged in the Clearing House, ensuring that the contract terms are met.

Finally, the Data User uses the App that consumes data discover and transferred through the Data Space.

2.2.1.2 Functional Layer

The functional layer outlines the functional requirements that an International Data Space must fulfil, regardless of the technologies in use. These requirements are as follows:

Trust: Each role in the IDS has certain rights and duties, as long as each participant follows their role, the Data Space will work as it is supposed to. For example, the identity provider must ensure that the participants are certified and validate their certifications through the certification process.

Security and Data Sovereignty: By employing concepts like authentication and authorization and enforcing usage policies, the Data Space becomes a secure and safe environment for participants to exchange data, while maintaining control over the data they contribute.

Ecosystem of Data: Each participant must offer adequate descriptions of their data, by providing metadata and clear usage policies, when advertising their data to the Broker. One way to achieve that goal is to employ the usage of vocabularies.

Standardized Interoperability: The Connector is the gateway for connecting the data provider to the Data Space, hence it must meet certain standards, such as secure connection and authentication when users connect to it and warning or alerts when issues arise.

Value Adding Apps: Other than harbouring secure data exchanges, the IDS enables data processing and transformation through modular applications, also known as Data apps. These apps offer useful data insight without compromising security by means of statistical analysis, anonymization or more complex data transformation.

Data Markets: The monetary aspect of data exchanges must not be overlooked. Providers must define clear pricing models (per day/month/year) and usage restrictions, reflected in the binding contracts negotiated in the Data Space.

2.2.1.3 Process Layer

The process layer describes the required steps in several key interactions with the Data Space.

Onboarding is the process by which a willing participant becomes a new member of the Data Space ecosystem.

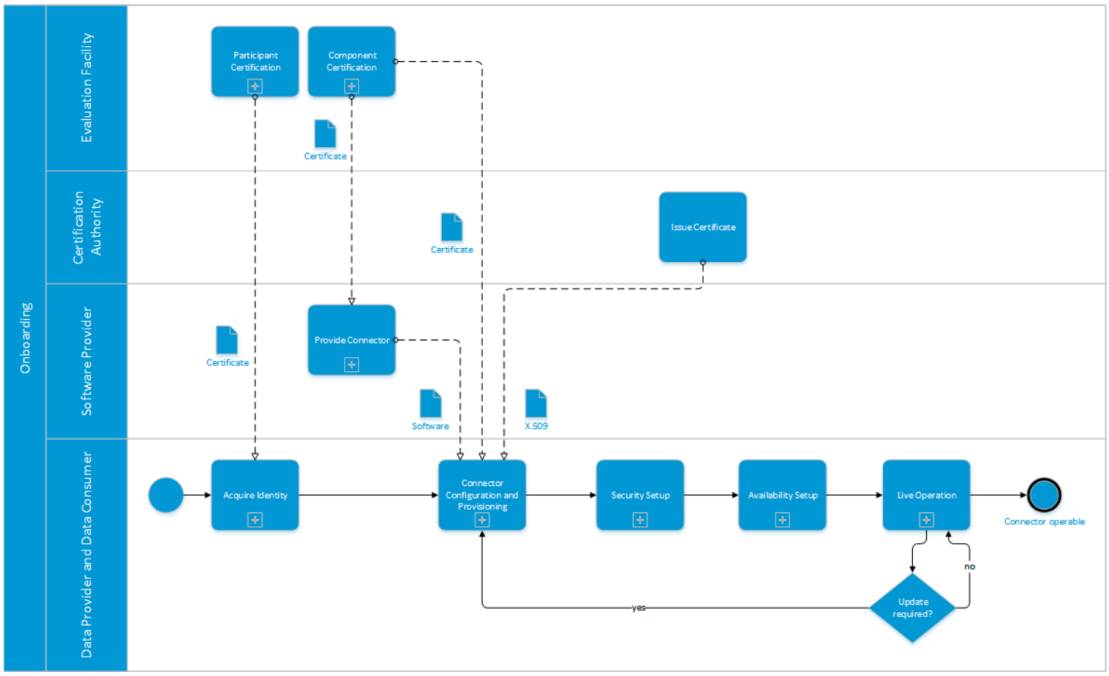


Figure 10 Onboarding overall process [2]

The first step is to acquire a certificate validating the identity of the organization willing to enter the Data Space, either as a consumer or as a provider of data. Next,

the organization must request a Connector from a Software Provider to serve as the gateway connecting them to the ecosystem. An X.509 certificate is required to certificate that the installation of the Connector conforms with the IDS standards.

Exchanging Data is the most important activity inside the Data Space.

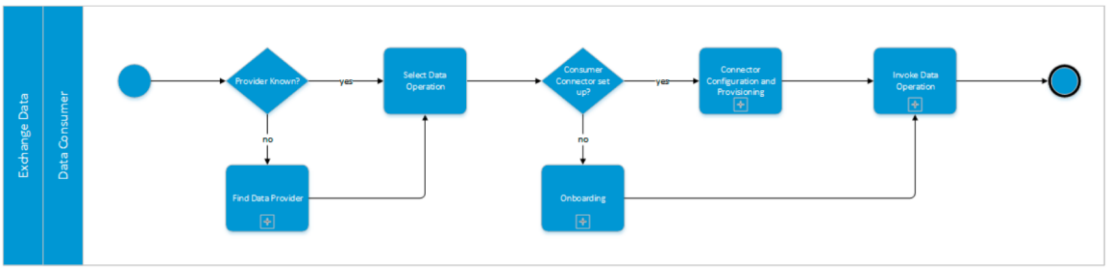


Figure 11 Data exchange overall process [2]

As seen in the diagram of the overall process, the data consumer searches for a data provider for their specific needs. After locating one (or more), and the consumer Connector has been properly set up and its identity has been verified (if not, it needs to onboard), it can negotiate an offer and proceed to receive the relevant data.

The subprocesses of “Finding a Data Provider” and “Invoking Data Operation” are described below.

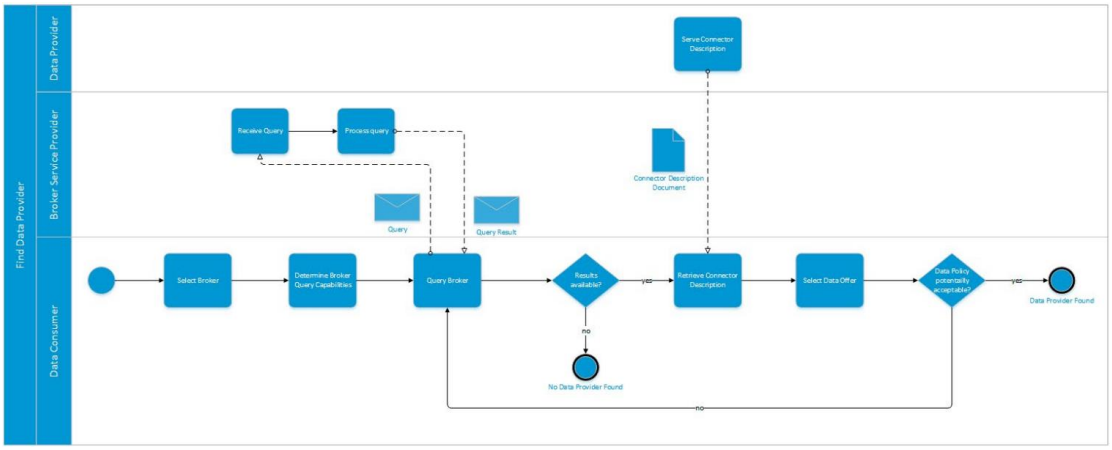


Figure 12 "Find Data Provider" sub process [2]

The consumer selects a Broker and queries it (according to its capabilities) to discover available Connectors and offers. The Broker goes through the list of registered provider Connectors and their descriptions. For each registered data offer, if the policy may agree with the query of the consumer, the result is returned to them.

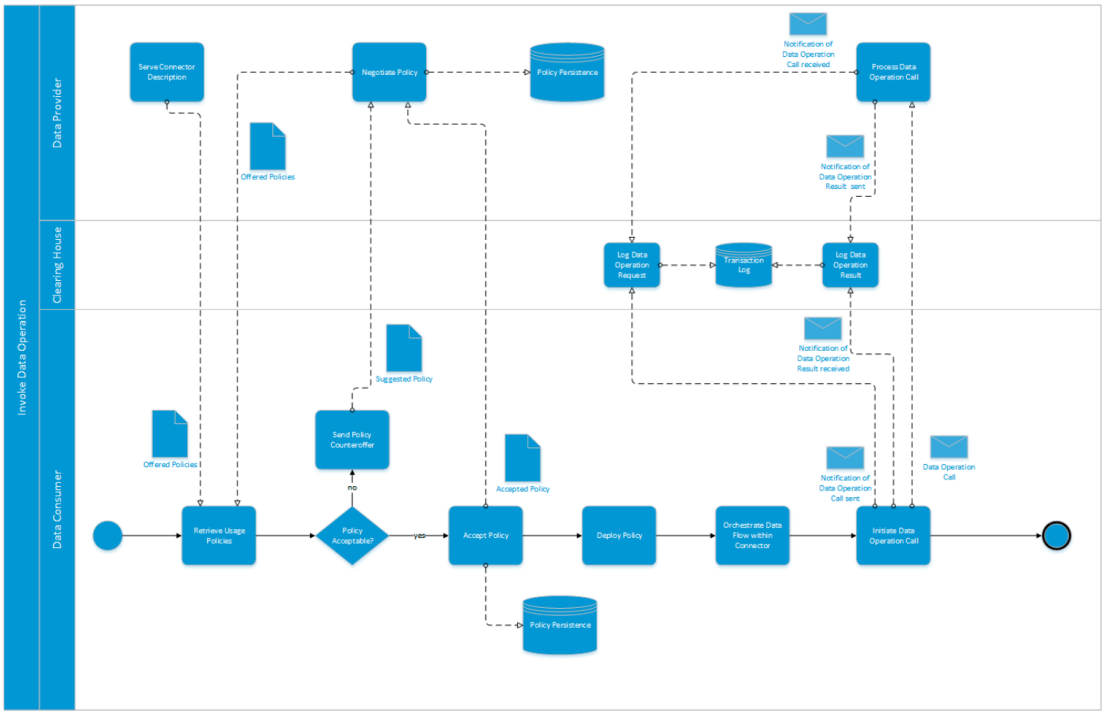


Figure 13 "Invoke Data Operation" sub process [2]

The data exchange is a procedure conducted between the consumer and the provider of data, with the Clearing House logging the transaction, if one occurs.

After the consumer receives the provider's policy agreement, they can either accept or counteroffer to negotiate the policy. After an agreement on policy has been made, the provider must abide by it and proceed to make the actual data request from the provider. The request is logged into the Clearing House (along with the transaction) and the data is received.

The last process detailed in the process layer of the RAM is the one of **Publishing and using data apps.**

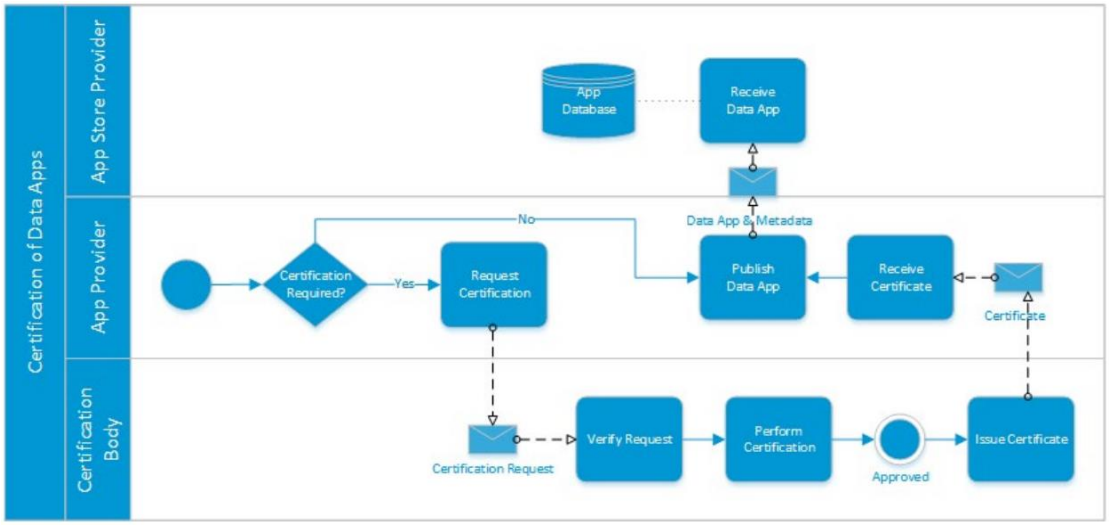


Figure 14 "Data App Certification" process [2]

The process revolves around getting an app certified by a Certification Body and publishing it -along with its metadata- on an App Store Provider.

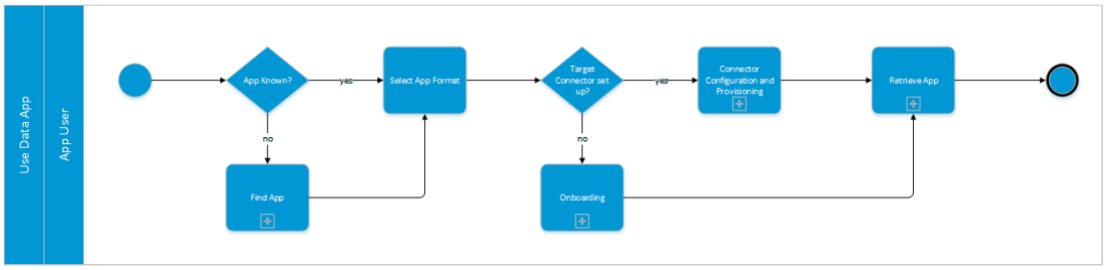


Figure 15 "Use Data App" process [2]

On the other side we have the process of employing a data-using app. After selecting the app format, it is ensured the relevant Connector is properly set up (as explained above) and, if it is, the app is ready to use and can retrieve data.

2.2.1.3 Information Layer

The information layer defines the Information Model, a domain-agnostic, common vocabulary. It is a “common language” that enables semi-automated data “translation” between different systems.

The Information Model is generic; it is not tied to any specific field. Field-specific details are provided via outside documentation of offered resources.

It offers three layers of representation, each with its own degree of human readability and machine integration.

The **Conceptual Representation** is a high-level demonstration of resources, participants, infrastructure and processes. Its target audience is the wider public, media and management boards, as it provides basic and human readable information.

The **Declarative Representation** provides a normative view of the Information Model. Based on W3C standards and modeling vocabularies it serves as a bridge between Conceptual and Programmatic representations, as it can be interpreted by machines while remaining understandable to humans.

The **Programmatic Representation** targets Software Providers, as it is a representation of the Information Model in programming languages (e.g., Java, Python, C++). It allows developers to easily integrate the model in their codebase without burdening them with the efforts of ontology translation.

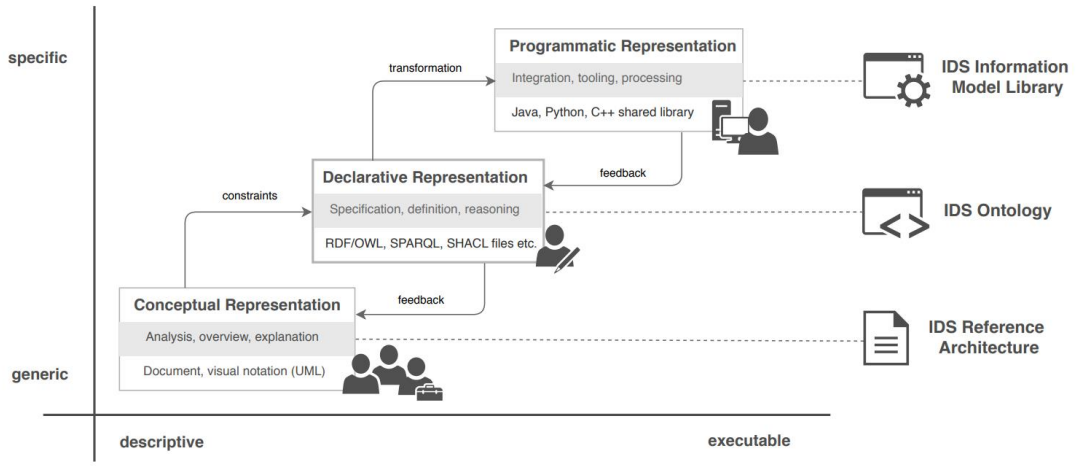


Figure 16 Representations of the Information Model [2]

As seen in the diagram above, the Representations range from generic and human-readable, to specific and machine executable, while feedback from each representation can flow back, refining each layer.

2.2.1.5 System Layer

The system layer "maps" the theoretical concepts from higher layers (Business, Functional, Information, Process) into actual technical components, assigning to each one their respective roles. Additionally, it describes the architecture of those components.

2.2.1.5.1 Connector

The **Connector**, as we have seen, is the building block of the entire Data Space architecture. It is a technical component that every participant needs to either host or hire a Connector-as-a-Service to participate in the ecosystem. Its role ranges from being certified by the Identity authority, to publishing/searching for metadata and facilitating data exchanges.

At the technical level, the Connector is a container-based system composed of core services and isolated execution environments. It manages identity and trust by securely storing credentials, integrates with the IDS Identity Provider for certification, and enforces policies by embedding them into the data transfer procedure. The communication between Connectors is secured end-to-end and each one of them is verified according to their certification. Furthermore, the Connector exposes configurable data endpoints and metadata descriptions to the IDS Broker, enabling discovery from possible data consumers.

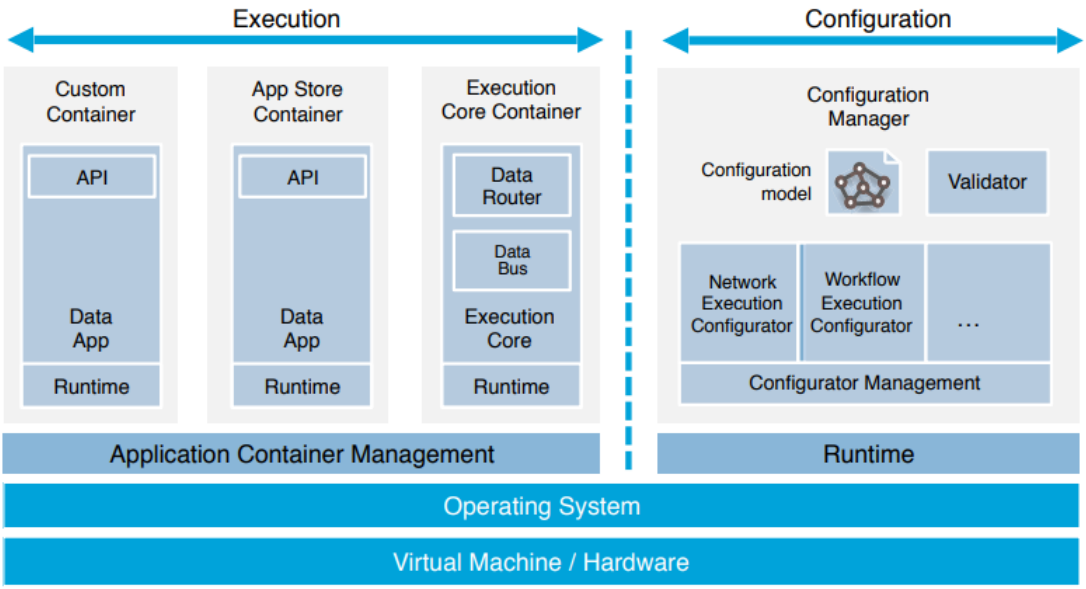


Figure 17 Architecture of a Connector [2]

The Connector Architecture runs on top of any hardware/virtualization and Operating System, splitting its functions over two domains: Execution and Configuration.

In the Execution domain, the Application Container Management hosts three distinct types of containers:

1. Custom Containers: a runtime for self-developed or non-certified Data Apps tailored to the needs of the organization that created them.
2. App Store Containers: a container downloaded from, and certified by, the App Store.
3. Execution Core Containers: responsible for the core functions of the Connector. It includes the Data Router and the Data Bus core elements, responsible for data discovery and transfer, according to the configuration.

The configuration phase involves the configuration, validation and deployment of the connector. There exist several execution configurators, each one is technology dependent and is responsible for translating the abstract configuration model into concrete runtime environments.

The Validator is responsible for validating that the configuration model complies with the standards and rules set forth by the IDS. If the validation fails, the Connector will not launch.

The Configuration Model of the Connector describes several important aspects of the Connector and is technology independent.

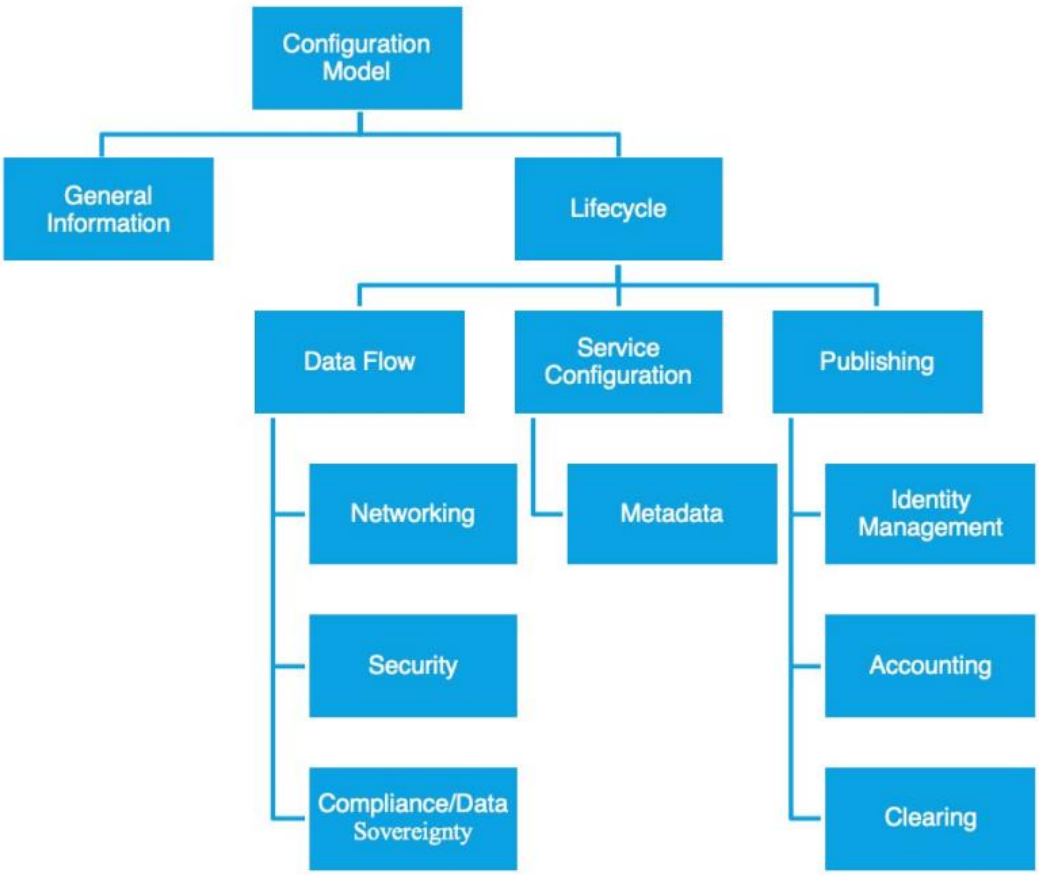


Figure 18 Connector Configuration Model [2]

It comprised of General Information -such as the Connector type (Base, Trusted, Mobile, Embedded, Developer), the version, timestamps for changes- and Lifecycle configurations, which are used to set up the flow of data, the security requirements, metadata formatting, and aspects of publishing offers.

2.2.1.5.2 Broker

The **Broker** is a special implementation of the IDS Connector. It must facilitate data source registration, publication, maintenance and queries. The interactions with the Broker have been described in the layers above. Specifically, the Information Layer describes the processes for Broker registration and query.

2.2.1.5.3 Data Apps and App Store

Data Apps extend the functionality of IDS Connectors by enabling data processing, transformation or integration within the Data Space. They can be self-developed (internal to the provider, without certification), third-party apps (retrieved from the IDS App Store, usually certified) or consumer-side apps allowing for data filtering or aggregation before the data is exchanged.

The apps can also be separated into three categories depending on their purpose. System Adapters (provider side) are responsible for transforming internal data models into standardized formats, while also enriching them with metadata. Smart Data Apps (consumer side) which perform any kind of data processing or storage for usage. Other Apps may provide specialized functions for either provider or consumer.

The IDS App store is the secure platform responsible for distributing Data Apps. It supports certification processes, registration procedures and billing support.

2.2.2 Perspectives of the Reference Architecture Model

The perspectives (Security, Certification, Governance) are cross-cutting concerns that must be considered and addressed by each layer of the RAM.

2.2.2.1 Security Perspective

Security is one of the prominent promises that the IDSA has made regarding its data-sharing policies. To show how these security requirements are met, it defines how each layer addresses security concerns, provides key security concepts that safeguard the Data Space, and highlights the end-to-end enforcement across all interactions.

Layer-wise security: Each layer must address different security concerns.

- On the **Business layer**: Security aspects are crucial for the definition of roles and basic interactions in the IDS.
- On the **Functional layer**: Security requirements can sometimes restrict certain transactions or operations. Meanwhile some other operations are meaningless without addressing security concerns (trusting other participants with sensitive data).
- On the **Process layer**: Emphasis is given on the continuous upholding of security standards by all processes in the Data Space, as they are continuously validated. That can be achieved by using a central Public Key infrastructure for Connector certification, or using the DAPS to verify attributes before issuing Dynamic Attribute Tokens.
- On the **Information layer**: In the common vocabularies used in this layer, it is important to also define and specify usage control policies in a way that is understood by all participants.
- On the **System layer**: The Connector is the system that must conform to all the security requirements that data exchanges must adhere to. The trusted Connector is the extension of the base Connector that conforms to all those standards.

On the **key security concepts** domain, the IDSA lists the following:

- **Secure communication** between all participants, achieved by ensuring point-to-point encryption between Connectors and end-to-end authorization on communication endpoints.

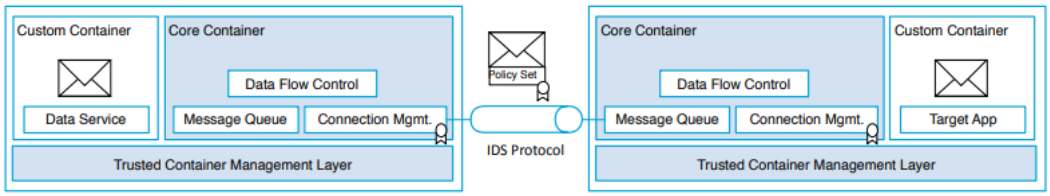


Figure 19 Communication between Connectors in the IDS

- **Identity management** is a key part of the ecosystem. As we will see in the deployment of the Testbed in chapter 6, each participant must provide X.509 certificates to prove its identity, and here the IDS provides the format they must follow for that process.

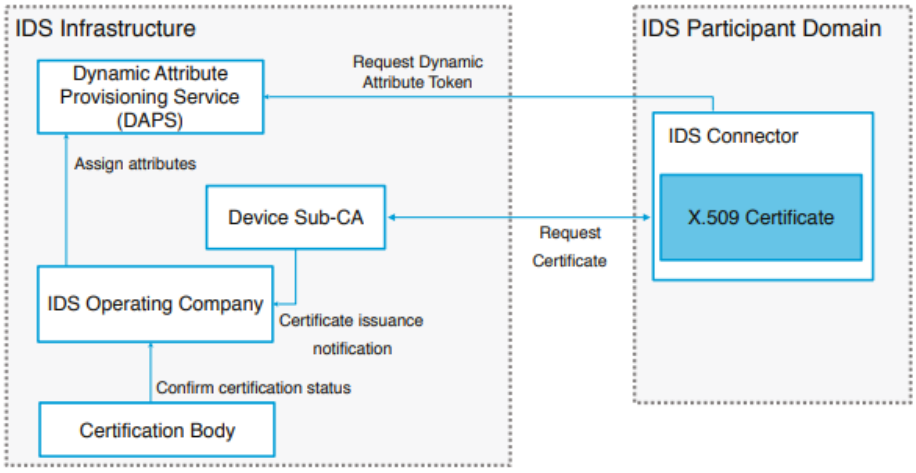


Figure 20 Embedding the Connector Certificate

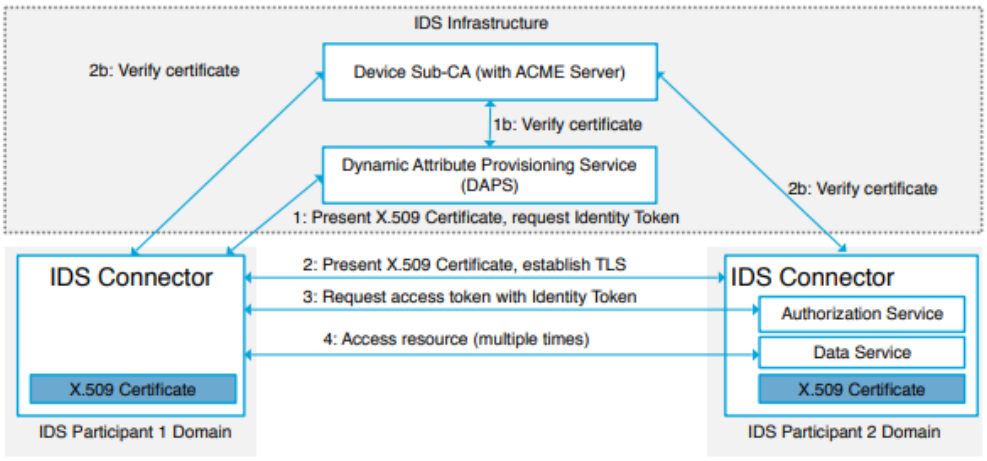


Figure 21 Use of certificates in access workflow

- **Trust management**: To prevent any sort of man-in-the-middle attacks that lead to fraud, the ecosystem requires for each participant to participate in the Public Key infrastructure by having a private key. That allows other participants to verify any messages stemming from them.

- **Trusted Platforms** are needed to verify minimal requirements for participation in the Data Space, verifying Data Apps and establishing a trustworthy relationships between participants.

2.2.2.2 Certification Perspective

As we have seen, certification is an integral part of the IDS ecosystem and necessary for providing security and preventing fraud. It is addressed by each layer, just like the security perspective:

- **Business layer:** The bodies responsible for the certification process are the Certification Body and Evaluation Facility. They are responsible for verifying Participants, intermediaries and Software providers.

- The **Functional layer** highlights that the implementation of participant-systems interacting in the Data Space must be compatible with the certification processes offered by the Certification Bodies of the Data Space.

- For a component to be certified, it must perform all its functions according to the requirements of the **Process layer**.

- Similarly for the **Information layer**, a core component must comply with functionality and protocols defined in the RAM (and more specifically to the Information Model, where applicable), in order to receive certification.

- By defining the interactions between components, the **System layer** plays an important role in the successful certification of the security aspect of a component.

The **Certification process** for any applicant-participant is shown in the diagram below.

Applicants are the willing participants that wish to be certified. They contract an Evaluation Facility (third-party auditors) that “test” the applicant to see if they meet all standards.

The Certification Body of the IDS oversees the Evaluation, and if both entities agree that the applicant can be certified, the X.509 certification is granted.

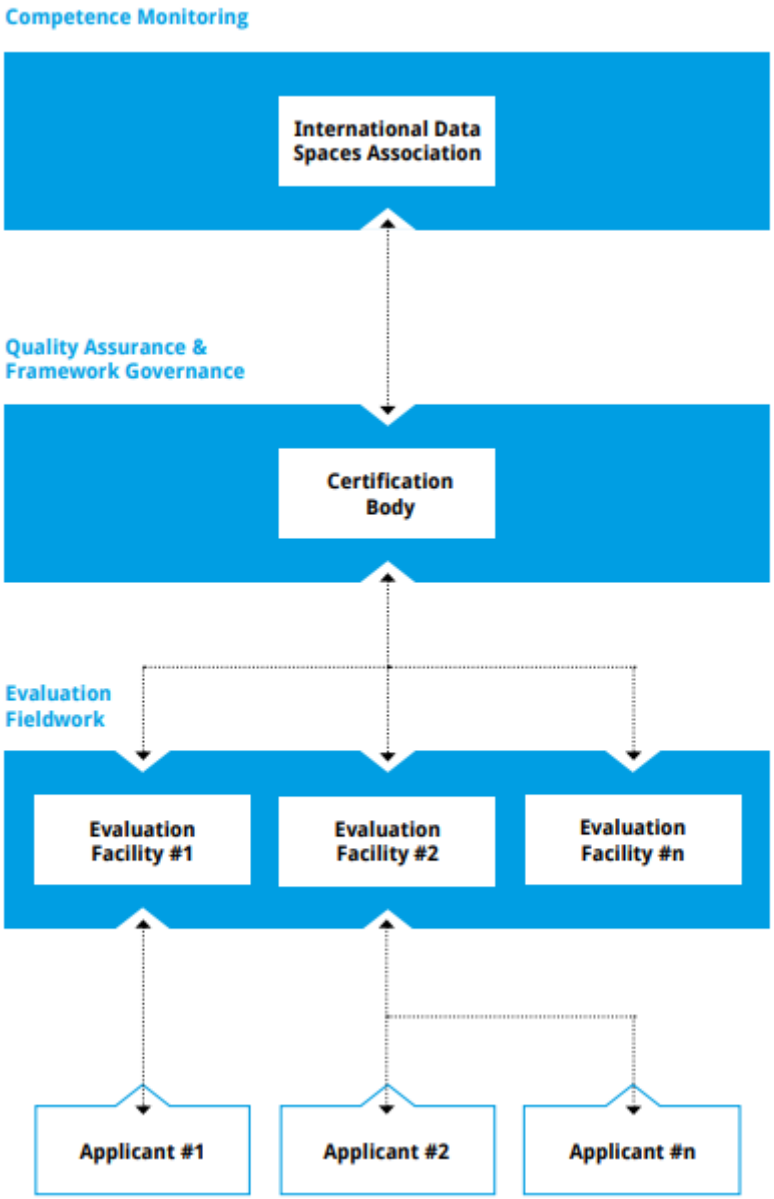


Figure 22 Certification process [2]

2.2.2.3 Governance perspective

The Governance perspective of the IDS-RAM defines how rules, roles and responsibilities are organized so that participants can collaborate securely and reliably while retaining control over their data. It provides a flexible framework for negotiation and accountability across organizational boundaries.

This perspective, just like the others, is addressed by each layer of the RAM:

- The **Business layer** defines the business models and roles to be applied by IDS participants, therefore it is directly related to the governance perspective by

touching upon subjects such as data ownership, data provision and data consumption.

- From a Governance perspective, interoperability and connectivity must be guaranteed to uphold trust, security and data sovereignty, the requirements for which are defined on the **Functional layer**.
- The **Process layer** defines the interactions between components and the three major processes (onboarding, exchanging data and publishing Data Apps), actions directly related to the Governance perspective, as they define its scope regarding the technical architecture.
- The **Information layer** defines a common vocabulary that lets participants describe data in a standardized way. This supports collaboration, contract definition, and governance by ensuring that metadata is expressed consistently across the IDS.
- The **System layer** relates to the Governance Perspective due to its technical implementation of different security levels for data exchange between Data Endpoints in the IDS.

2.3 Data Space Initiatives

The standards and examples set forth by the IDSA have already been put to use by several initiatives which are working examples of communities benefitting from the employment of Data Space, and the ecosystem of apps that are made possible by it.

2.3.1 Mobility Data Space (Germany)

A national, decentralized data-sharing marketplace, where mobility actors (businesses, government agencies, private sector) publish and exchange data according to the rules and standards of a modern Data Space [3].

Funded by Germany's Federal Ministry of Digital Affairs and Transport (BMDV), it stands as a lighthouse [4] example of what federated data sharing can accomplish.

The MDS builds on the IDS Reference Architecture Model, by using IDS Connectors and employing usage policies attached to data. Following the architecture, it is not a central repository, but rather a network of Connectors, allowing the data to stay under the physical and digital control of the owners and providers.

Figure 23 Catalogs of offered data in the MDS [5]

With over 200 participants [6] in the data sharing ecosystem, Germany has seen several initiatives taking advantage of the data in order to provide meaningful work [7].

PrioBike-HH is an initiative by Yunex Traffic, centered around bicycle safety in Germany. By utilizing data from the MDS it offers several improvements in the everyday life of cyclists [8]:

1. Bicycle totem: a sign placed a distance away from intersections, displaying the recommended speed for cyclists in order to avoid red lights ahead.

Figure 24 Bicycle Totem in Hamburg [9]

2. App for cyclists: An app (currently in beta) that stores cycling routing data and offers cyclists multicriteria route planning, not only according to shortest path, but also considering comfort and safety.

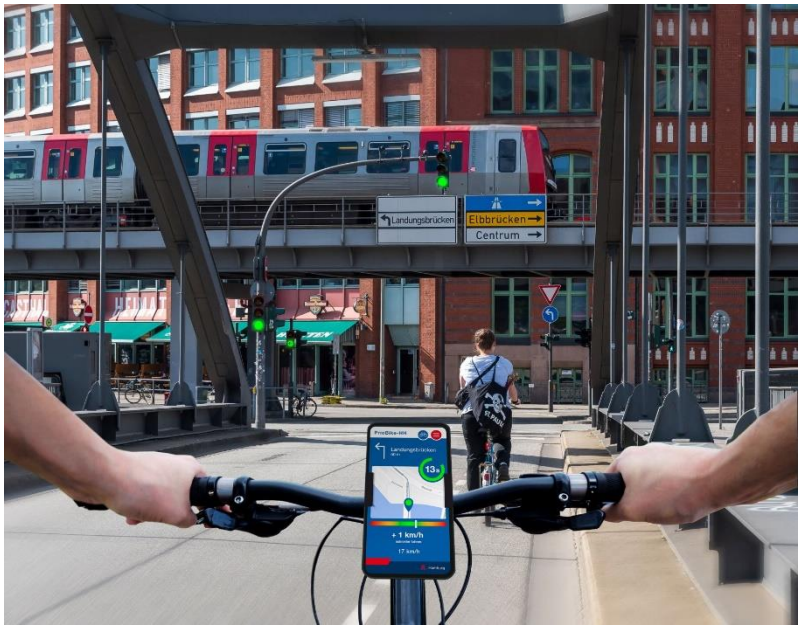


Figure 25 Beta version of the PrioBike App [10]

Esri's Living Digital Twin [11] aims at upgrading the concept of a "Digital Twin" into a living one, by integrating live telemetry data from car manufacturers. Aggregation of live vehicle data allows for a comprehensive overview of the current traffic situation, while historical data provide insights into areas with increased accident frequencies.

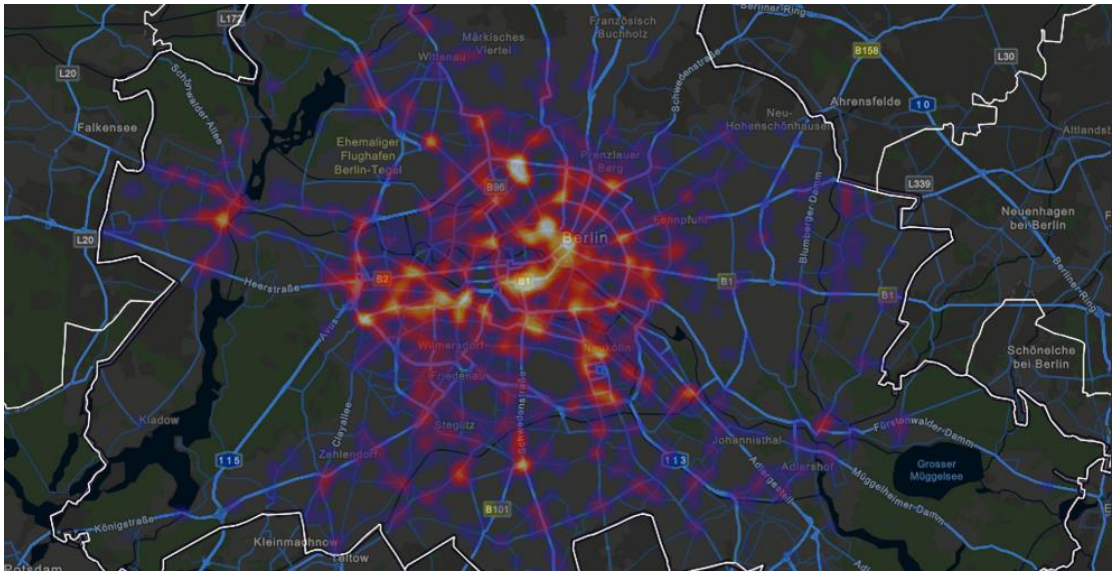


Figure 26 Map of accidents provided by Esri [12]

The **DeepVolt Location Intelligence Assistant (DLIA)** [13] is an AI powered web tool that enables cities and businesses to locate optimal spots for new EV Charging Stations. By leveraging data analytics, it picks spots with maximum impact according to factors like, population density, median income, EV registrations and existing EV infrastructure.

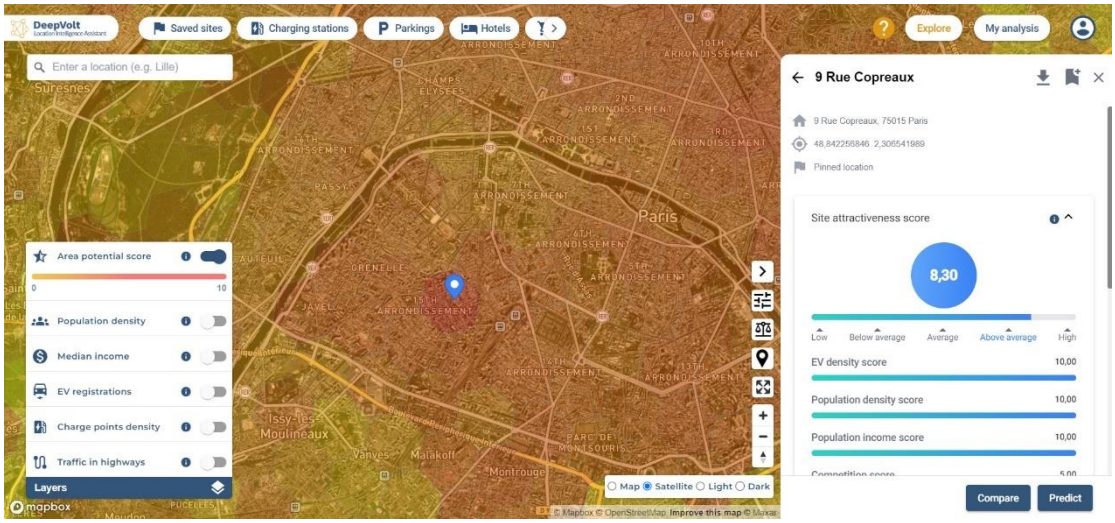


Figure 27 Strategic placemanet of new EV charging locations [13]

2.3.2 Smart Connected Supplier Network

The Smart Connected Supplier Network (SCSN) [14] is a Netherlands-based Data Space for the high-tech manufacturing supply chain. It enables seamless inter-company exchange of orders, forecasts, invoices, technical product data, and other supply chain related information across manufacturers and their suppliers.

The SCSN technical infrastructure conforms to the IDS Reference Architecture. Following the decentralized “four-corner” model [15], its participants only need to connect to the network once but can communicate with everyone. The model is comparable to the telecom sector, every participant is free to choose their service provider, but in the end, all service providers agree to use the same standards, allowing for seamless communication between any two participants of the network.

The underlying SCSN infrastructure is based on IDS components developed specifically for the network’s needs. Those components (Connectors and data apps) were created by the SCSN Foundation and are offered to participants to install and connect to the network, or perform work with data apps, ensuring compliance with IDS standards across the field.

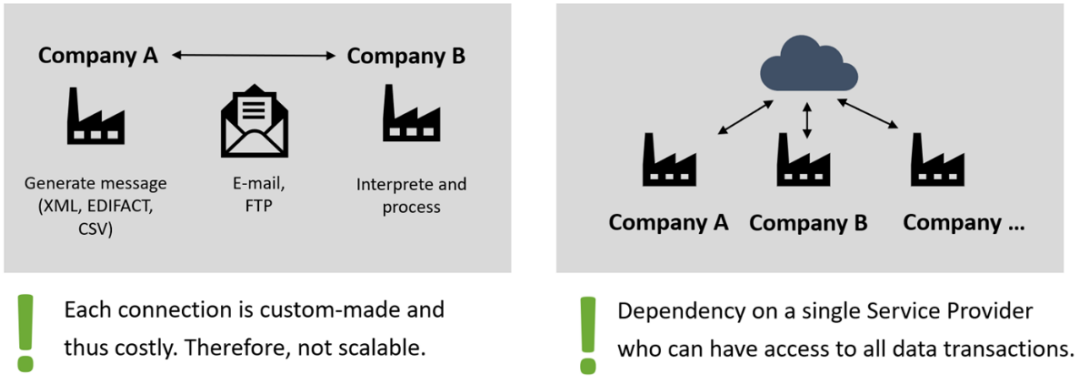


Figure 28 Two corner and Three corner design [15]

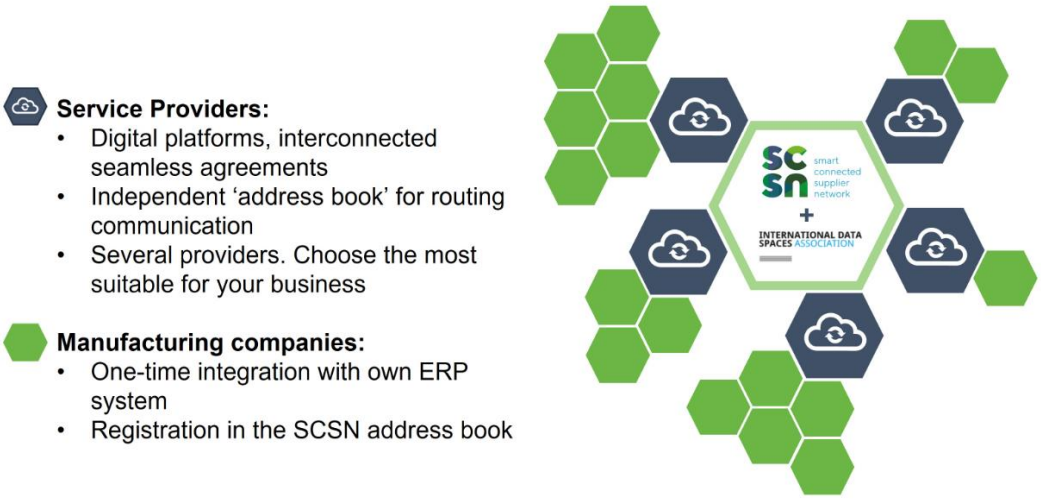


Figure 29 Four corner design [15]

Founded in 2014, it continues to grow and build an interconnected ecosystem. By 2024, over 400 participants, and 10 service providers have joined the project, making it one of the most mature implementations of Data Spaces in real-world scenarios.

Figure 30 The SCSN network [14]

2.3.3 Health-X dataLOFT

Health-X dataLOFT [16] is a collaborative data space initiative in the healthcare sector. It is a project that launched in 2021 in Germany and has received funding by the German Federal Ministry for Economic Affairs. Its mission is to create a common health data space where citizens, healthcare providers, and researchers can share and access health data in a sovereign, privacy-preserving way. It specifically focuses on bridging personal wellness data (from fitness trackers, smartwatches, health apps) with clinical data. For example, citizens could volunteer smartwatch activity data to their doctors or research institutes via the Health-X platform, under clear consent and usage policies.

The project architecture is fully compliant with IDS and Gaia-X principles. The backbone of the technical infastructue is the IDS-compatible Connector technology. Fraunhofer ISST has developed a custom Health-X Data Space Connector based on the Eclipse Data Space Connector (EDC) framework .

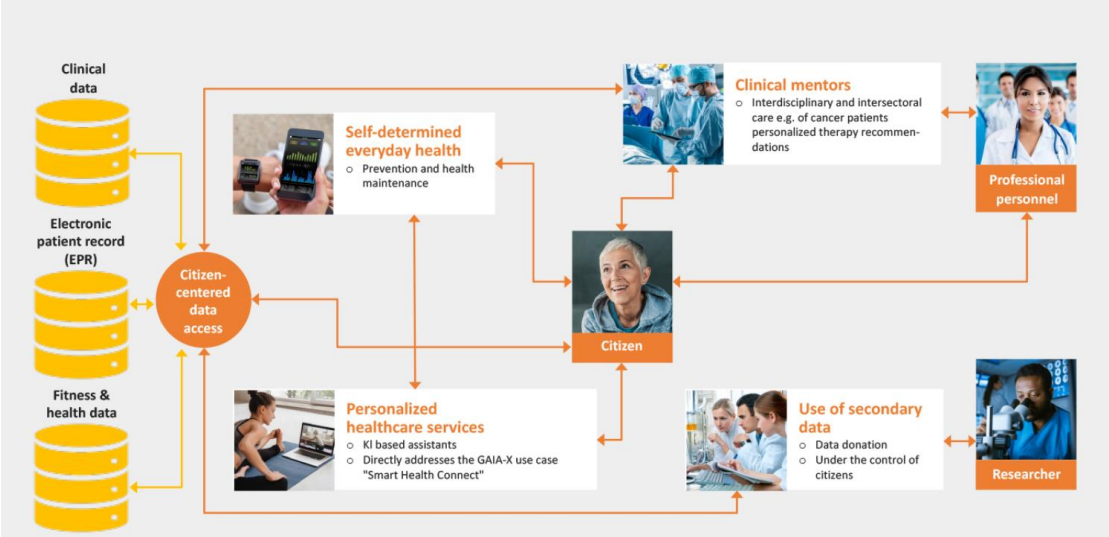


Figure 31 Health-X evosystem of data around the citizen [17]

While still being in a pilot stage, Health-X has demonstrated how extremely sensitive data (personal health documents) can be safely incorporated into an IDS-compliant data space.

2.4 Concluding Remarks on Data Spaces

The review of existing Data Space initiatives demonstrates that the concept is not limited to research but has already been applied in diverse domains, including the mobility, manufacturing and healthcare sectors. These projects highlight recurring patterns: they usually begin as pilots and expand gradually as trust is established, and

security is proven. Technical interoperability is necessary, but it is not enough, a big community of participants in a trusted ecosystem is what Data Space are made for.

At the same time, the underlying IDS Reference Architecture provides consistent technical foundation that allows these initiatives to function. The modular set of components (Connectors, Brokers, Clearing Houses and Trust Services) offer balance between flexibility and accountability. In the context of disaster response this architecture is particularly relevant: the Connector ensures that sensitive data about vulnerable citizens remains under the provider's control, the Broker allows for easy discovery of resources during a crisis, and the DAPS guarantees that only certified participants can enter the ecosystem. Taken together, the lessons from current initiatives and capabilities of the IDS framework underline why Data Spaces are a promising approach for coordinating information in emergencies.

3 Methodology

3.1 Research Design

The design of the implementation of the Data Space ecosystem and its relevant actors will be a small-scale demonstration of real-world scenarios with different entities participating in the exchange of data. However, the modularity of the Data Space platform chosen leaves ample room for growth and further development, as will be discussed in chapter 5.

The scope of the design is also limited due to the sensitive data that are required for creating an evacuation planning app for people with disabilities (Personal info, fleet data). We don't have access to data like that so we have to simulate the entities involved (Clubs, Government agencies, Municipalities) and the data sharing that would occur if they participated in the Data Space. However, for data freely available (like natural disaster data) we can use public APIs and connect them to the ecosystem. For more information on the data and APIs see chapter 5.

After simulating the needed data, the next step will be to connect them to the Data Space ecosystem according to the standards and requirements set forth by the IDSA.

Finally, the App will connect to the Data Space and will be able to search for relevant data, in order to fulfil its role of evacuation planning for people in need.

Following this procedure allows us to discover possible hurdles in the implementation of the aforementioned technologies and document the process to create a comprehensive guide for future developers.

3.2 Requirements

Possibly the most important requirement is upholding the core principles of sensitive data sharing in Data Spaces: Data sovereignty, Confidentiality (only authorized parties have access to data), Integrity (No data alteration in transit) and Accountability (existence of signed logs of transactions).

Additionally, we have set forth some functional requirements for the implementation, also important features in any Data Space:

1. Data discovery: The participants must be able to discover data they are interested in by querying the Broker.
2. Interoperability: APIs from heterogeneous sources must provide documentation for any third party to be able to use their data.
3. Demo app features: Data visualization, Planning/Dispatching of forces, Providing relevant information to dispatched forces.

Furthermore, the architecture must be scalable and offer a clear path into becoming operational in real-world scenarios.

By ensuring these requirements are met, we can guarantee that the proposed implementation does not simply remain a technical experiment but can be the foundation for new Data Space initiatives in areas that involve sensitive data.

3.3 Evaluation Methodology

In order to assess whether the system developed in this thesis actually meets its intended goals, it is important to define a clear evaluation methodology. Since the prototype is not deployed in a real emergency management agency but rather in a controlled academic context, the evaluation cannot rely on large-scale field experiments. Instead, the assessment focuses on two complementary aspects:

- Verifying that the functional and non-functional requirements defined earlier are fulfilled.
- Demonstrating, through realistic scenarios, that the system provides tangible value in coordinating evacuations.

4 Platforms Used

In developing the Data Space ecosystem focused on coordinating evacuation scenarios for people with disabilities during natural disasters, several modern platforms were used. This chapter provides an overview of each key technology covering their purpose, main features and usage in our scenario.

More detailed information about specific libraries or tools that these platforms offer will be provided in later chapters (such as chapter 5 and subchapter 7.2)

4.1 Docker

Docker [18] is an open platform for containerization that enables developers to package applications and their dependencies into isolated units called containers. A container bundles together everything required for the application to function exactly as intended, including code, runtime, system tools and settings. Docker allows developers to detach their app from the underlying system and architecture, and only focus on developing in a known environment.

Some of the **main features and benefits** of Docker and its containerized architecture are that the containers are **lightweight** (sharing the OS kernel, so more efficient than full virtual machines) and **isolated** (each container has its own filesystem and process space). This allows running many containers on the same host with minimal overhead. These characteristics also enable **scalability** since the host is allowed to host multiple containers without fear of them interfering with each other or affecting its resources more than needed.

Docker has an essential **role in the implementation of this thesis**. The International Data Spaces (IDS) Reference Testbed, which forms the backbone of our Data Space, was set up using Docker containers for each core component. As we are going to see in chapter 6, the entities Connector, Broker, DAPS and other essential participants of the Data Space are container instances that only require proper configuration to be deployed, thanks to the architecture of Docker.

Docker Compose [19] is used in cases where several services or applications need to be deployed at the same time. The orchestration of such a deployment is configured via a docker-compose.yml which contains the configurations for each container to be deployed. These configurations may regard port usage, log policies, file mountings from the host or any other environmental variable.

In our case Docker Compose is exactly what's needed for the deployment of all the key entities that comprise our Data Space. The configuration file includes necessary information for the deployment of all those entities, like certificates and truststores, docker image sources, authentication credentials, ports, aliases and necessary URLs.

With the use of the docker compose up and docker compose down commands, we can instantly deploy or take down all the necessary parts of the Data Space.

Finally, the modularity of the Docker architecture is ideal for such initiatives, since adding more participants to the network is as easy as configuring one more container to be deployed along with every other.

4.2 FIWARE

FIWARE [20] is an open-source framework of software components designed for building smart applications with a strong focus on context data management. The platform provides a set of Generic Enablers (reusable building blocks) that developers can assemble to create solutions in domains like smart cities, smart mobility, and IoT. At its core is the **FIWARE Context Broker**, which manages and stores contextual information (sensor data, location of assets) in a structured way. The Context Broker exposes a standard API (NGSI-V2, and more recently NGSI-LD) through which applications can create, update, query and subscribe to context information in real-time [21].

The three pillars that make up the FIWARE ecosystem are:

1. The standard API for context management (NGSI).
2. Open-source components to avoid vendor lock-in and encourage collaboration.
3. A global community set on defining common data models and best practices.

Typical use cases for this platform include Smart City systems (integrating data from traffic sensors, public transport, air quality monitors), Smart Agrifood (data about crops and weather), and any other scenario where standardized data modeling or real time updates are required.

In this thesis project, FIWARE was employed as a way to share real-time information about a fleet of vehicles, to be used in evacuation attempts. Specifically, the system uses the FIWARE Orion Context Broker (NGSI-V2) to provide such information (including position, speed, seat availability) from the agencies' side, and query that information from the application's perspective.

The FIWARE platform used in this project is hosted by the Lab [22] of the University of Patras. It provides this service for students to develop their applications but also for more mature projects.

4.3 Node.js

Node.js [23] is an open-source, cross-platform JavaScript runtime environment. Built on Google's V8 JavaScript engine, Node.js introduced JavaScript to the realm of

backend development, allowing developers to use a single language across both client and server. Its asynchronous, event-driven architecture has led to widespread usage in the backend ecosystem, particularly for applications requiring scalability and responsiveness under high concurrency. The built-in package manager (npm) provides access to a vast ecosystem of open-source libraries, ranging from web frameworks (Express, Koa) and database clients (MongoDB, SQL drivers) to testing tools and utility modules. These characteristics make Node.js a natural choice for building lightweight, efficient, and maintainable services. Common use cases include web servers and RESTful APIs, real-time applications, and microservices architectures, where low latency and high throughput are critical.

In this thesis, Node.js plays a central role as it is the runtime chosen to implement multiple components of the implementation ecosystem.

Firstly, the central web application (Civil Protection Demo App) is built on Node.js. As we will cover in detail, this application runs an HTTP server to serve a dynamic map-based interface. The choice of Node for the backend of this app allowed seamless use of JavaScript on both the client (in the browser) and server, using the same formats (JSON) and benefiting from the vat number of mapping and geospatial packages available in the JavaScript ecosystem. More detailed examination of the specific libraries used in the main app will be discussed in chapter 7.2.

Along with the main app, three out of four data providing APIs also employ Node to provide data using several packages themselves, such as Swagger, MongoDB and SQL drivers.

4.4 Python

Python [23] is a high-level interpreted programming language known for its readability, versatility and extensive ecosystem of libraries. It supports multiple programming paradigms, including object-oriented, functional, and procedural programming, which makes it adaptable to a wide range of problem domains. Its cross-platform status, along with the support for a wide range of tools—from HTTP requests and data processing to geospatial analysis—make it a great platform for quick development of APIs and applications. In addition, Python benefits from an active open-source community, which continuously contributes frameworks, modules, and resources that accelerate development and foster innovation. This ecosystem, combined with Python's relatively gentle learning curve, has positioned it as one of the most widely adopted languages in both academia and industry.

Specifically in this implementation, Python was used for the development of the fleet_simulator.py program. As seen in subchapter 5.3 it is a program responsible for managing the fleet data on the Orion Context Broker and running a simulation of the fleet on the road network.

5 Data and APIs

In this chapter, we will analyze the data used in the implementation, its structure, the sources in real-world applications and the APIs that simulate organizations and government agencies. The APIs and other files mentioned below can be found in the thesis repository [24].

5.1 Government catalog API

Every country has agencies that aid people in need. The most common way of helping those people is by having a central catalog of people that require assistance (usually monetary or caretaking) [25]. Such a catalog must store essential information about each person regarding their name, home location, caretaker information, contact information, age, disability % and description.

Lacking access to that resource in the scope of this thesis, an API simulating such a catalog has been created, along with the necessary model, which holds key information about each person. The API is a Node.js server and MongoDB is used for storing the data according to the following model.

```
const ameaSchema = new mongoose.Schema({
  name: { type: String },
  surname: { type: String },
  email: emailSchema,
  phoneNumber: phoneNumberSchema,
  landNumber: landNumberSchema,
  mandatoryCommunication: { type: String },
  loc: {
    type: { type: String },
    coordinates: { type: [] }
  },
  region: {
    administrative: { type: String },
    municipality: { type: String }
  },
  disabilities: { type: Array },
  disabilitiesDesc: { type: String },
  disabilityPct: { type: Number },
  floor: { type: Number },
  owner: [{ type: mongoose.Schema.Types.ObjectId, ref: 'Account' }],
  club: [{ type: mongoose.Schema.Types.ObjectId, ref: 'Clubs' }],
  birthday: { type: Date },
  created: { type: Date, default: new Date() },
  updated: { type: Date },
  address: { type: String },
  caretaker: caretakerSchema,
  status: { type: String }, // pending, cancelled, active
  group_club: { type: String, default: '' },
  activity_problem: { type: Number, default: 0 },
  cardAmeaNumber: { type: String, default: '' },
  verifyUser: [{ type: mongoose.Schema.Types.ObjectId, ref: 'Account' }],
  verifyClub: [{ type: mongoose.Schema.Types.ObjectId, ref: 'Clubs' }],
  mustVerify: { type: Boolean, default: false }
});
```

Figure 32 The model of a person with disabilities in the Government catalog

As mentioned before, the security requirements for such sensitive data are extremely high. Therefore, all the sensitive fields are encrypted at rest [26], meaning that upon a leak of the entire database, the data would be inaccessible without the decryption key which must be stored separately. That is possible with the use of the mongoose-field-encryption npm package [27].

For demonstration purposes, a populator of the database has been created (MhtrooApi/populate_seeded.js) that injects data into the catalog database from a configurable file (MhtrooApi/amea_seed.json).

The API itself (MhtrooApi/server.js) offers an endpoint to retrieve all entries for people with disabilities. However, that endpoint can only be accessed by providing a valid token, and the entries returned are encrypted, with a different secret key than the database encryption.

That way the API ensures that no third party can have access to the data without receiving a valid token, and even if there is a leakage of a response, there will not be a data breach, since the payload is encrypted itself.

The encryption algorithm used is AES-256-GCM [28], a highly secure encryption algorithm, guaranteeing confidentiality, integrity and authenticity of the data transferred.

```
function encryptUtf8(plaintext) {
  const iv = crypto.randomBytes(12); // 96-bit nonce for GCM
  const cipher = crypto.createCipheriv('aes-256-gcm', AES_KEY, iv);
  const ciphertext = Buffer.concat([cipher.update(plaintext, 'utf8'), cipher.final()]);
  const tag = cipher.getAuthTag();
  return {
    alg: 'AES-256-GCM',
    iv: iv.toString('base64'),
    tag: tag.toString('base64'),
    ciphertext: ciphertext.toString('base64'),
    encoding: 'utf8',
    issuedAt: new Date().toISOString(),
  };
}
```

Figure 33 Encryption process of AES-256-GCM in JS environment

Sufficiently documenting the responses of APIs participating in a Data Space is crucial. For that purpose, all the APIs offer documentation using the Swagger library [29]. For the case of encrypted responses, the documentation first shows the encrypted form, and then an example of the decrypted model of the response. Swagger not only offers documentation, but also interactive testing of endpoints for developers or third parties.

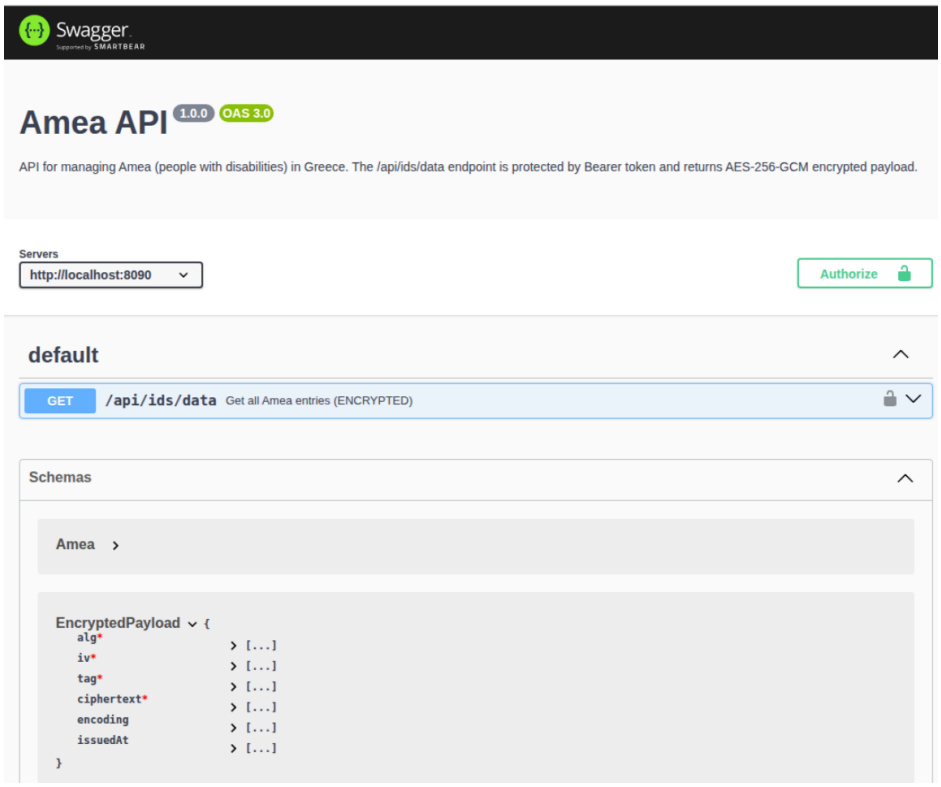


Figure 34 Mhtroo API encrypted response documentation

With the decrypted model example being as follows.

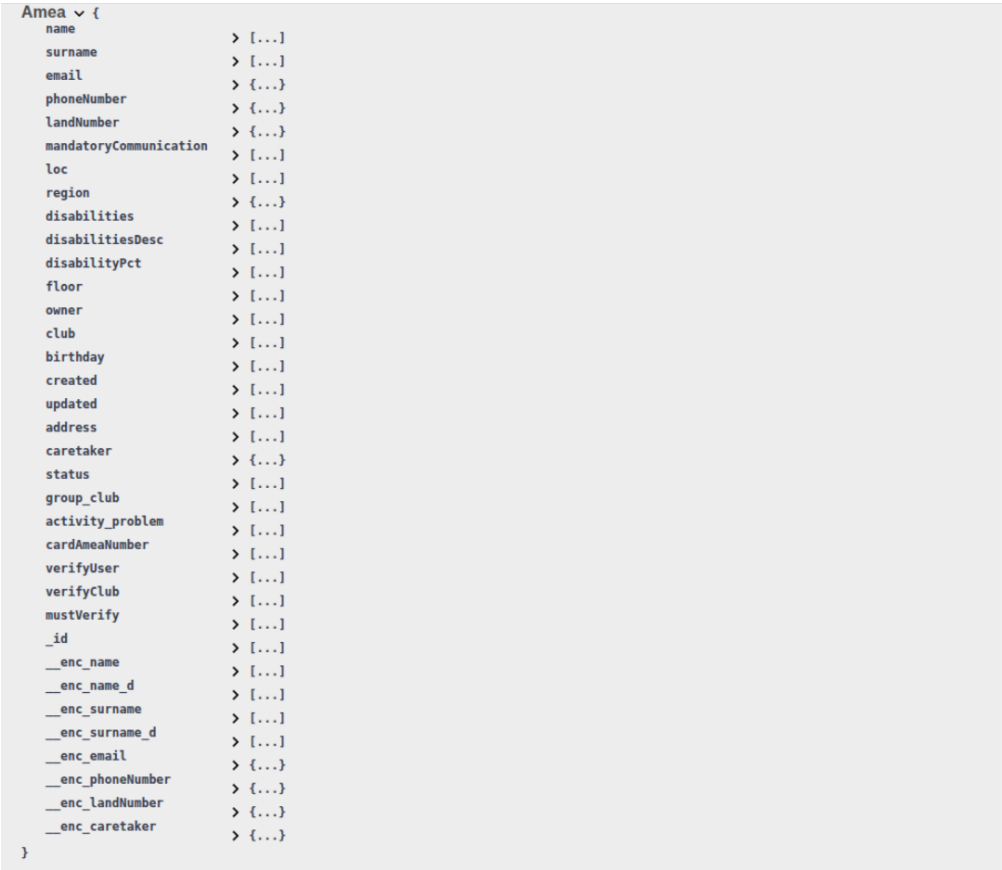


Figure 35 Mhtroo API decrypted response documentation

5.2 Clubs and municipalities API

To be registered in the government's catalog for people with special needs, one needs to have over 67% disability percentage, to be eligible for financial aid [30]. That means that the catalog is far from a complete registry of all the people that require special attention in the context of a natural disaster.

That means that there must be other sources of records in order to get the full picture of the people that require assistance in evacuation scenarios. Those other sources are clubs of people with disabilities and municipalities, who keep their own records.

To simulate those different sources, another API has been developed. This API is also written in Node.js but uses a different database technology: SQLite [31]. This showcases that the participants are not required to conform to a strict model in order to enter the ecosystem, any datasets can be introduced, as long as they are accessible and documented.

A populator script exists here as well (AmeaclubApi/populate_seeded.js) drawing data from (AmeaclubApi/amea_seed.json). Meanwhile the data are encrypted at rest too, using the same algorithm AES-256-GCM as before, ensuring integrity, confidentiality and authenticity.

Again, we see the important documentation done with Swagger

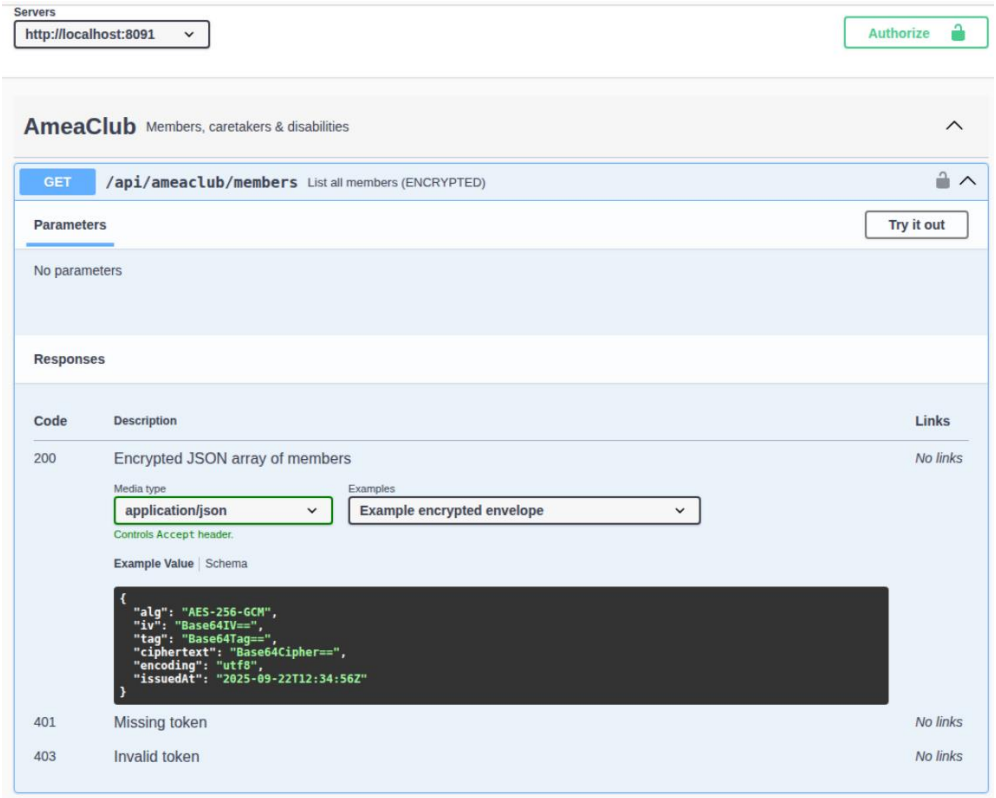


Figure 36 Ameaclub API documentation of encrypted response

And the decrypted response describing the model used.

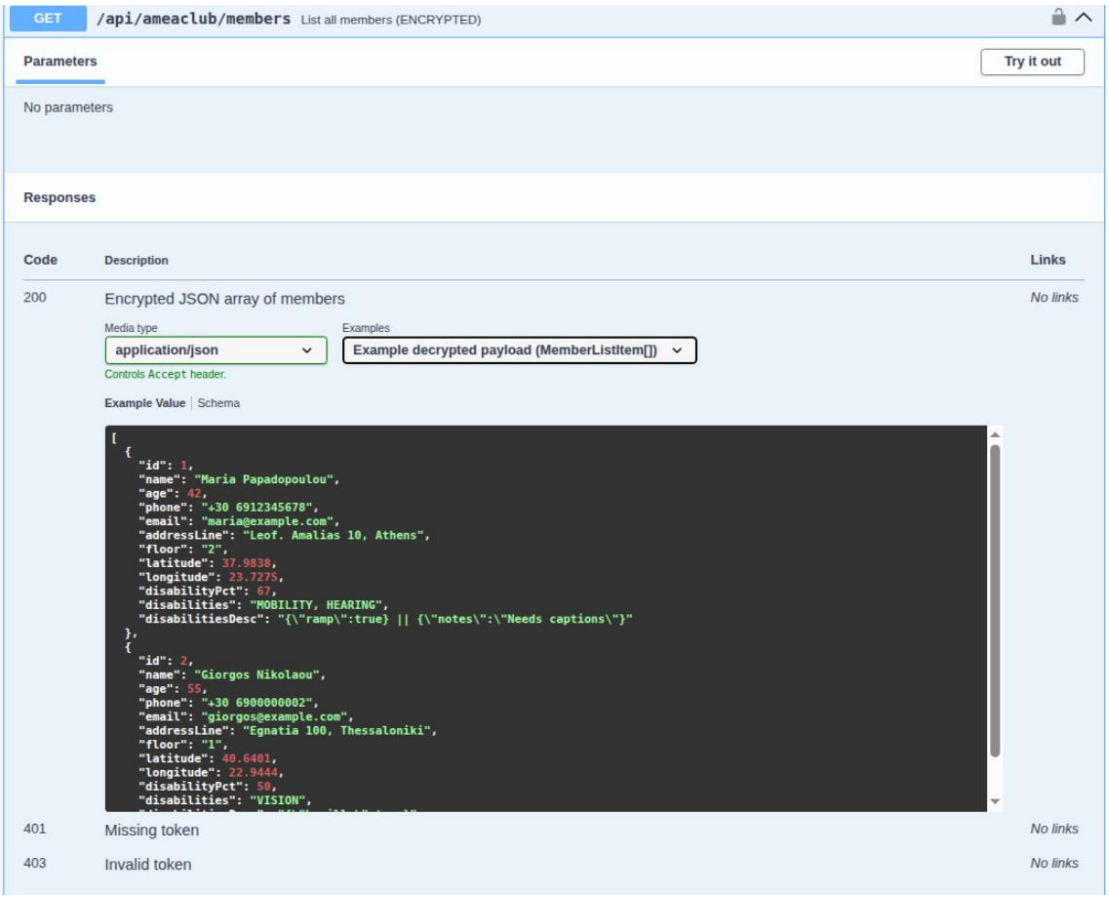


Figure 37 Ameaclub API documentation of decrypted response

With the first two APIs we have examined, the importance of decentralization in the maintenance of data sovereignty is becoming clear. The agencies and organizations are not required to give up their data into centralized repositories and hope they are not leaked or taken advantage of. Instead, they control the access and the safety measures taken to prevent data leakage or misuse by third parties.

5.3 Fleet Generation

Limited, again, by the scope of this thesis, we do not have access to telemetry data from vehicles belonging to agencies such as the police, fire fighting force or ambulances. For that reason, the decision was made to simulate such telemetry data by creating FIWARE entities according to the NGSI-V2 model for vehicles (with added properties necessary for the scenario) and update their speed and position.

FIWARE is an open-source stack for building context-driven applications (smart cities, mobility, IoT). The key concept is a Context Broker that exposes a standardized API (NGSI versions, in this case, NGSI-V2 is used) so producers can publish and update data, while consumers can subscribe to real-time entity data. The platform used is hosted by the Lab [22] of the Department.


```
{
      "id": "vehicle:WasteManagement:1",
      "type": "Vehicle",
      "vehicleType": "lorry",
      "category": ["municipalServices"],
      "location": {
         "type": "Point",
         "coordinates": [ -3.164485591715449, 40.62785133667262 ]
      },
      "name": "C Recogida 1",
      "speed": 50,
      "cargoWeight": 314,
      "serviceStatus": "onRoute",
      "serviceProvided": ["garbageCollection", "wasteContainerCleaning"],
      "areaServed": "Centro",
      "refVehicleModel": "vehiclemodel:econic",
      "vehiclePlateIdentifier": "3456ABC"
}
```


Figure 38 Vehicle model example according to NGSI-V2 [32]

On top of basic attributes (speed, location, id) of the standardized vehicle data model, we append necessary attributes for an evacuation scenario, such as the number of occupied and total seats, as well as a contact point for the crew.


```
{
  "id": "urn:ngsi-v2:Vehicle:CIV-001",
  "type": "Vehicle",
  "areaServed": "Patras",
  "category": [
    "municipalServices"
  ],
  "contactPoint": {
    "email": "ktsambras@gmail.com",
    "contactType": "email"
  },
  "crew": 3,
  "homeCity": "Patras",
  "lastUpdated": "2025-09-23T10:29:09.460Z",
  "location": {
    "type": "Point",
    "coordinates": [21.7463313, 38.234516]
  },
  "name": "CIV-001",
  "occupiedSeats": 3,
  "owner": "urn:ngsi-v2:PublicOrganization:CivilService",
  "refVehicleModel": "vehiclemodel:ambulance",
  "serviceStatus": "idle",
  "speed": 41.6,
  "totalSeats": 4,
  "vehiclePlateIdentifier": "KOP-1988",
  "vehicleType": "ambulance"
}
```


Figure 39 Extended vehicle data model

The goal of the fleet generation is to produce reasonable locations for the vehicles according to their type. Thus, each vehicle has a spawning station where it spawns near to. These stations include the University General Hospital of Patras (Rio), the General Hospital of Agios Andreas and others.

```
STATIONS: Dict[str, Dict[str, List[Dict[str, float]]]] = {
    "ambulance": {
        "Patras": [
            {"name": "General Hospital Agios Andreas", "lon": 21.748008, "lat": 38.234512},
            {"name": "University Gen. Hospital of Patras (Rio)", "lon": 21.795547, "lat": 38.294240},
        ]
    },
    "fire_truck": {
        "Patras": [
            {"name": "Patras Fire Brigade HQ", "lon": 21.728747, "lat": 38.234359},
            {"name": "Pyrosvestio", "lon": 21.742785, "lat": 38.252295},
        ]
    },
    "police_car": {
        "Patras": [
            {"name": "B' police station", "lon": 21.737566, "lat": 38.245579},
            {"name": "A' police station", "lon": 21.754060, "lat": 38.261352},
        ]
    }
}
```

Figure 40 Stations for vehicles to spawn near

Additionally, to simulate a real-world scenario, the vehicles might move if an evacuation scenario is not presented. Hence, using the Overpass API, we gather data about the road network and move the vehicles according to an arbitrary speed, across the roads, until they are presented with the evacuation orders.

The fleet simulator (CivilserviceAPI/fleet_simulator) is a Python script that is responsible for the generation and updating (patching) of the vehicle data -as seen above- for the duration of the simulation. However, consumers can only access the data via the FIWARE platform and its Context Broker via standardized queries.

The script offers a variety of options, including creating the entities and starting the simulation of their movement across the road network.

```
📋 Options:
1. Generate and post new fleet (on roads)
2. Delete all vehicle entities
3. Delete specific vehicles (by ID)
4. Run continuous road-based simulation
5. View current entities info
6. Load roads for specific Greek city
7. Show available cities
8. Exit
```

Figure 41 Fleet_simulator options

5.4 Natural Disasters API

In contrast to the previously examined sensitive nature of data, information about natural disasters has no reason to not be publicly available.

The first source of such information that needs to be investigated is the 112 warning service of the Civil Protection in Greece [33]. It is a system that sends SMS messages close to areas that need to be evacuated, warning civilians. However, using such a service would come with several disadvantages:

1. Ambiguous geography: The message sent only contains references to neighborhoods or toponyms which -while useful to evacuating locals- don't always have an accurate translation into affected areas -or even locations- to be used in evacuation planning.
2. No geometry: Similarly, the simple message does not offer machine-readable polygons, or other forms of describing the exact area that needs to be evacuated. Knowing the exact region is crucial to the planner: wider area means the forces will not be focused on where they should be, smaller area and the forces might not evacuate places they should.
3. No official source: While messages are posted on platforms such as X [34] and Facebook [35], that is not a verified or trusted way to retrieve data used for planning evacuation routes for people with disabilities. Additionally, access to the APIs of those platforms is not free.
4. Inconsistent format: The message is usually humanly readable, but it doesn't have a standard format for machines to interpret properly.
5. Missing information: Information like the severity of the disaster, its historic or projected areas of affect are not shared.

The Civil Protection Agency surely has the information missing but has not set up a proper platform for people to monitor disasters in real time.

For those reasons, the search for natural disasters data moved on to international organizations.

Copernicus is the EU's Earth-observation programme [36]. It is built around the EU's Sentinel satellite family (examples include Sentinel-1 radar for smoke/cloud-penetrating imagery [37] and Sentinel-2 for optical [38]) plus ground data. The Copernicus Emergency Management Service (CEMS) offer real-time mapping of live natural disasters, along with damage grading and early warning systems [39]. The most well-known platform of this service is the European Forest Fire Information System, specifically for wildfire disasters [40].

Other than providing live data (which is the most important aspect for the scope of this thesis) it also provides historic data of burnt areas, analytical reports of previous wildfires and wildfire risk viewers.

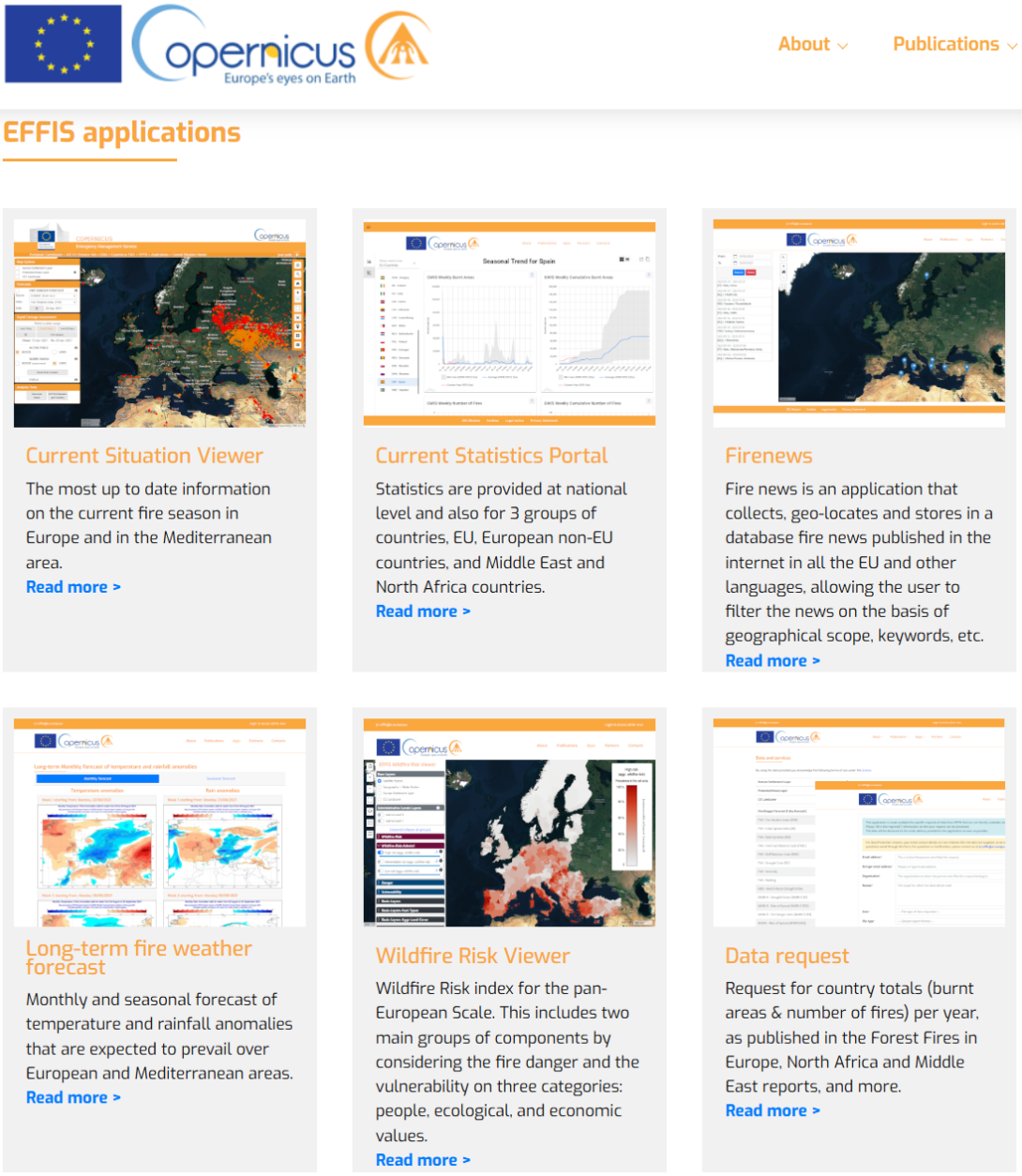


Figure 42 Copernicus EFFIS applications [40]

In order to demonstrate the evacuation planner in an area around the city of Patras, historical data about the recent fires in Achaia are requested from the EFFIS API.

The EFFIS API provides a detailed polygon of the burnt areas, which we are going to use as an example of currently occurring wildfires. The wildfires that occurred during August of 2025 in Achaia (at the same time) can be seen below.

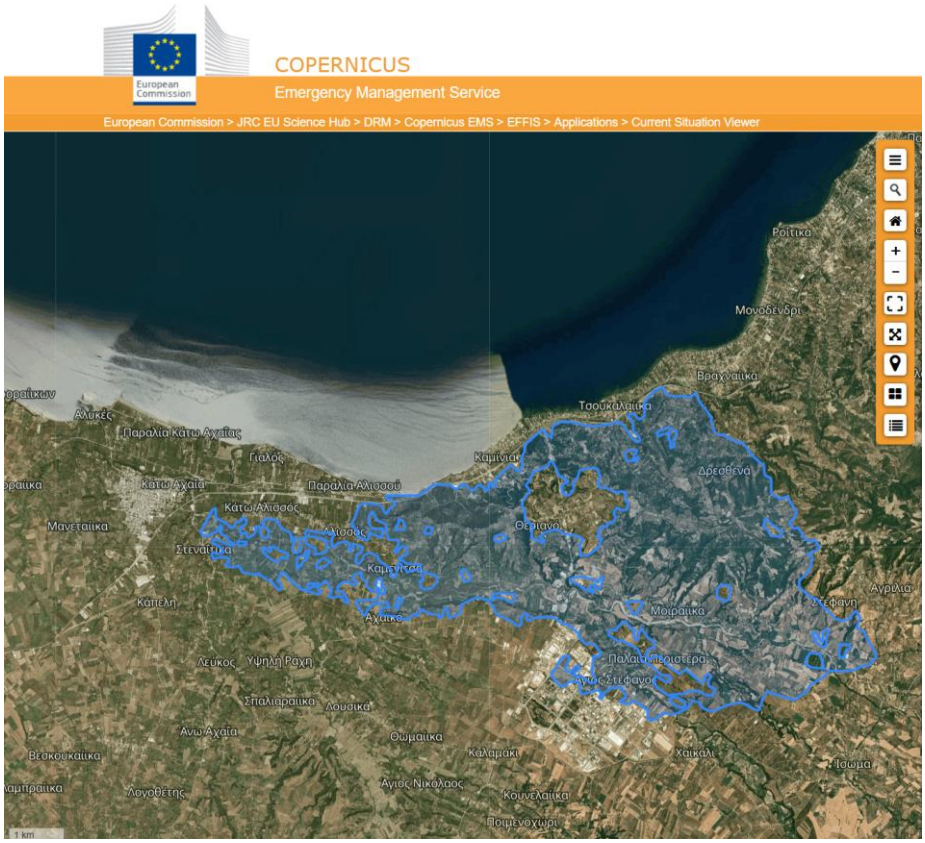


Figure 43 Burnt area at Vrachneika, August 12, 2025

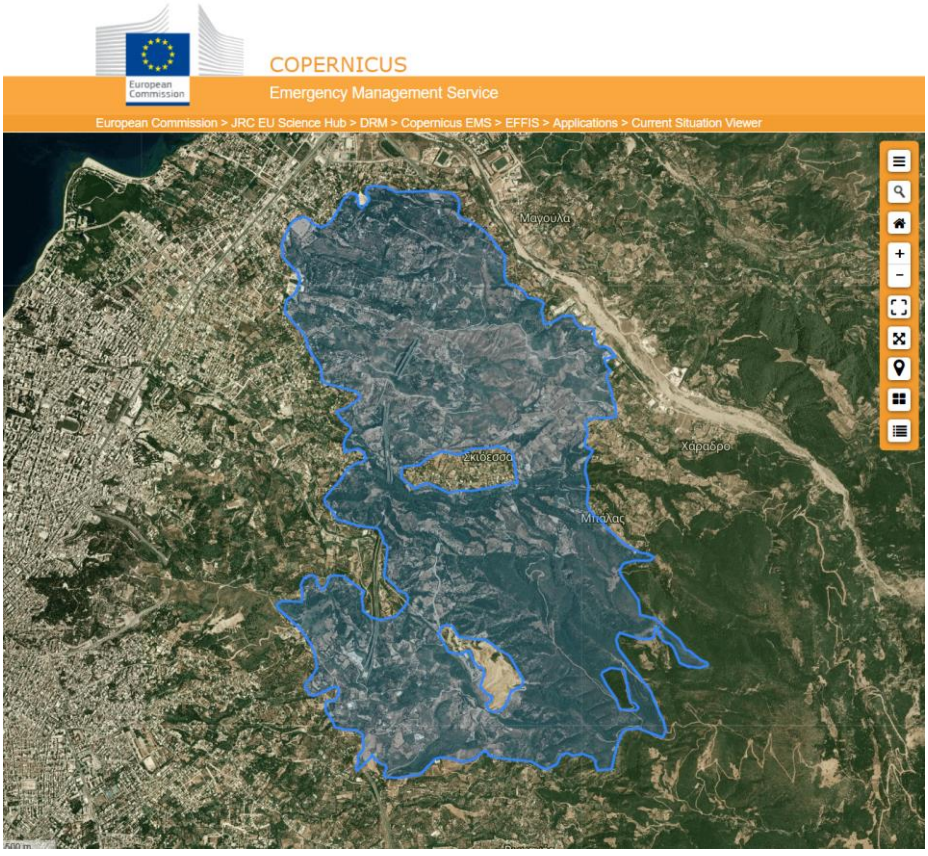


Figure 44 Burnt area at Patras' suburbs, August 13, 2025

According to the above, a script has been constructed (NaturaldisasterApi/populate_copernicus.js) that fetches the relevant data from the EFFIS API and stores them in a local database according to the following model.

```
const mongoose = require('mongoose');

const geoJSONGeometry = new mongoose.Schema({
  type: {
    type: String,
    enum: ['Polygon', 'MultiPolygon'],
    required: true
  },
  coordinates: {
    type: Array,
    required: true
  }
}, { _id: false });

const naturalDisasterSchema = new mongoose.Schema({
  type: { type: String },
  description: { type: String },
  dangerLevel: { type: String, enum: ['low', 'moderate', 'high', 'extreme'] },
  areaOfEffect: {
    type: geoJSONGeometry,
    required: true
  },
  startDate: { type: Date },
  endDate: { type: Date },
  historicalAreasOfEffect: {
    type: [geoJSONGeometry],
    default: []
  },
  projectedAreasOfEffect: {
    type: [geoJSONGeometry],
    default: []
  },
  updatedAt: { type: Date, default: Date.now },
  source: { type: String },
  sourceId: { type: String }
}, { timestamps: true });

naturalDisasterSchema.index({ areaOfEffect: '2dsphere' });

module.exports = mongoose.model('Disaster', naturalDisasterSchema);
```

Figure 45 Model of Natural Disaster

As we can see, the most important information about any possible natural disaster is stored: type of disaster, description, danger level (four different levels), start, end, last update dates and the source of the information (currently the only source is the EFFIS API). Additionally -and very importantly- current, historical, and projected areas of

effect are stored for each disaster. Those areas can be polygons or multi-polygons to cover all cases, and geospatial queries can be performed on them.

Since the examples of wildfires mentioned above are static (they are historic burnt areas), an estimated 500m buffer zone has been created around them, serving as the projected areas of affect, which will be the evacuation areas in our scenario. That buffer zone is created using the Turf toolkit. Turf.js is a powerful JavaScript library for advanced geospatial analysis. It provides functions for creating buffers, measuring distances, clipping, and manipulating GeoJSON data [41].

```
async function makeEvacuationGeometry(geometry, meters = 500) {
  const { buffer, cleanCoords } = await ensureTurf();

  // Turf takes Features or Geometries; we wrap as a Feature
  const feature = { type: 'Feature', geometry, properties: {} };

  // Buffer outward by 0.5 km. 'steps' controls roundness of corners.
  const buffered = buffer(feature, meters / 1000, { units: 'kilometers', steps: 16 });

  // Clean up any tiny artifacts/self-intersections
  const cleaned = cleanCoords(buffered);

  return cleaned.geometry;
}
```

Figure 46 Turf calculates buffer zone for evacuation

The danger level of the (past wildfires) is calculated according to their size.

```
function chooseDanger(areaHa) {
  const a = Number(areaHa || 0);
  if (a >= 1000) return 'extreme';
  if (a >= 300) return 'high';
  if (a >= 50) return 'moderate';
  return 'low';
}
```

Figure 47 Danger level calculation

The Natural Disaster API offers several endpoints, that return the status of the server, the list of disasters, specific disasters by id, disasters as a feature collection, and a map to project them.

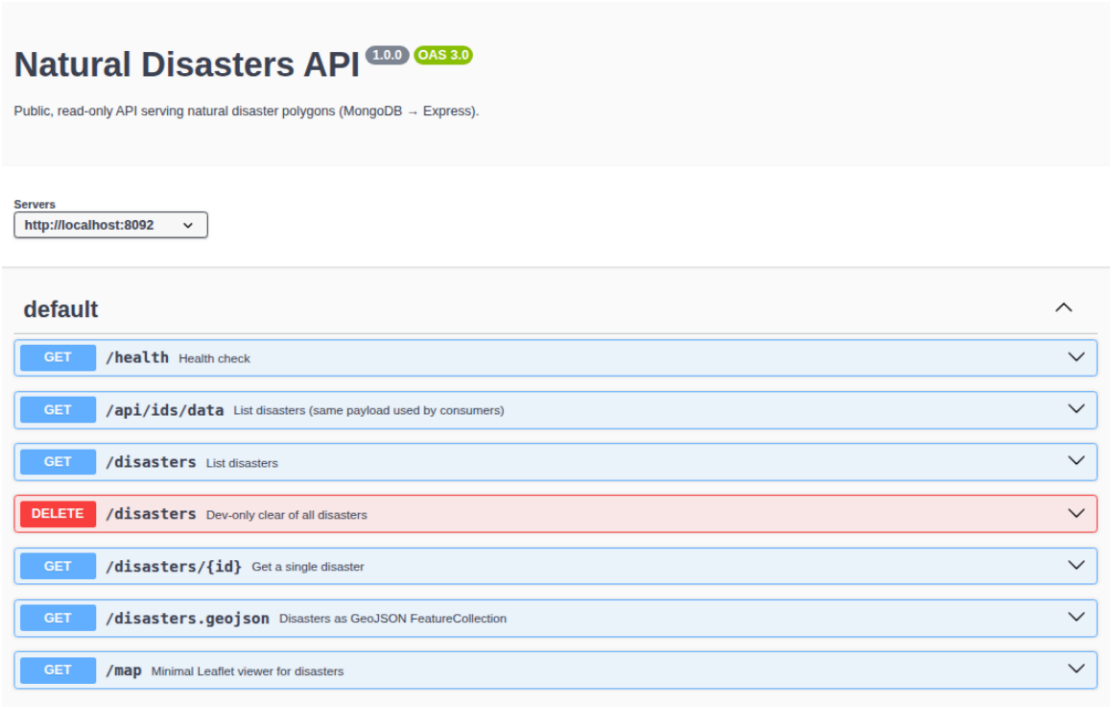


Figure 48 Natural Disasters API documentation

Endpoints that list disasters also have bounding box parameters to only search for disasters inside the box, as seen in the documentation.

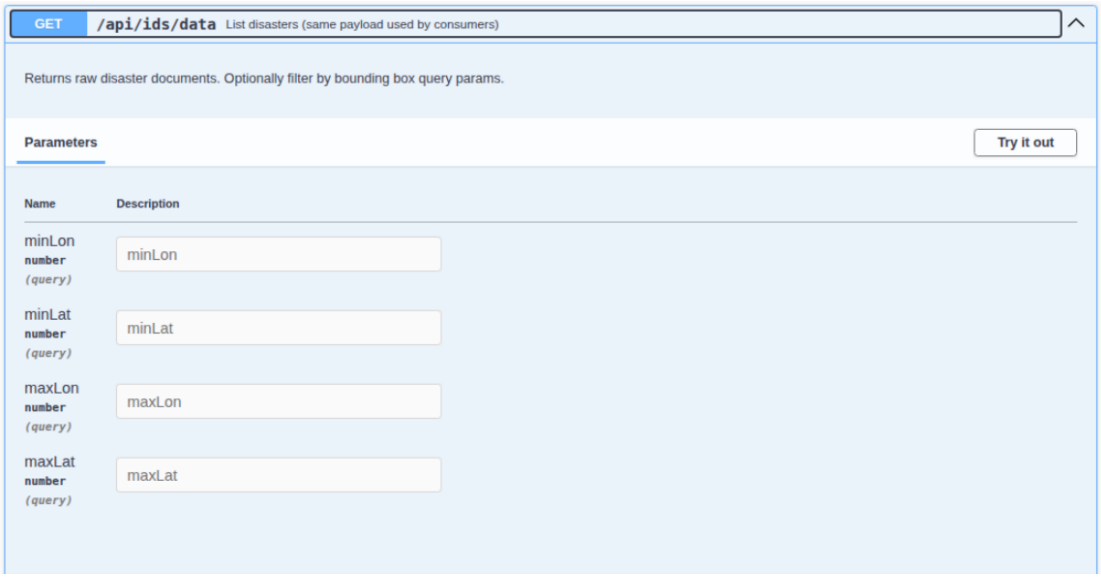


Figure 49 Natural Disasters IDS data endpoint documentation: parameters

Finally, the documentation also offers an example of a natural disaster being returned to a valid request.

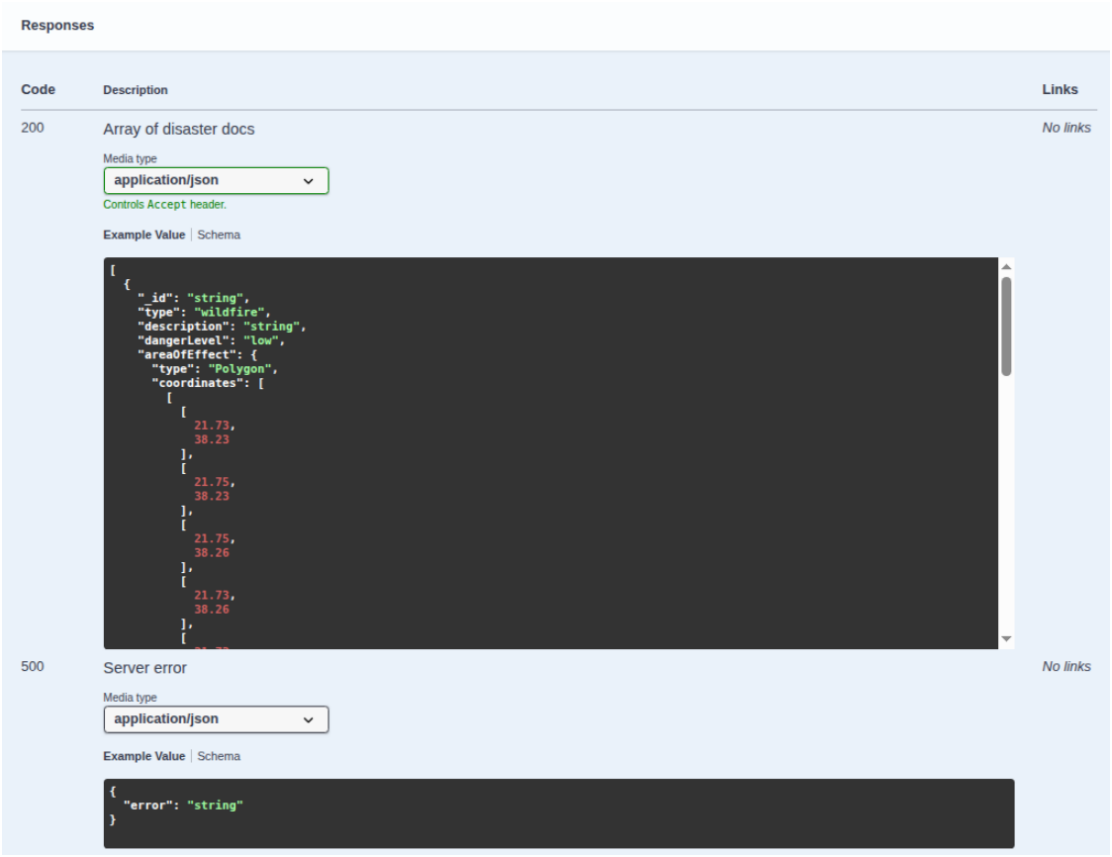


Figure 50 Natural Disasters IDS data endpoint documentation: response

The inclusion of Copernicus data in the prototype is not only a technical convenience but also an important demonstration of how open sources can complement national services. While the Civil Protection's 112 system is indispensable for alerting citizens at large, it is designed for communication rather than machine-readable planning. By integrating EFFIS data, the application gains precise geometries, severity indicators and historical context, all of which are essential for structured evacuation planning. The combination of availability, reliability and technical richness of such an open platform makes Copernicus a strong backbone for the disaster dimension of the system. However, that does not exclude the possibility of further sources that provide disaster information in the future, since the nature of a Data Space is the diversity of the resources provided.

6 IDS Testbed

6.1 Introduction to the Testbed

The **IDS Reference Testbed** [42] is a development and experimentation environment created by the International Data Spaces Association (IDSA) to demonstrate and validate the principles of Data Spaces. It provides a framework that implements the key elements of the Reference Architecture Model (RAM) [43].

Its main goal is to be used to develop and test new components -such as new connectors- and verify that they can work in a verified IDS environment.

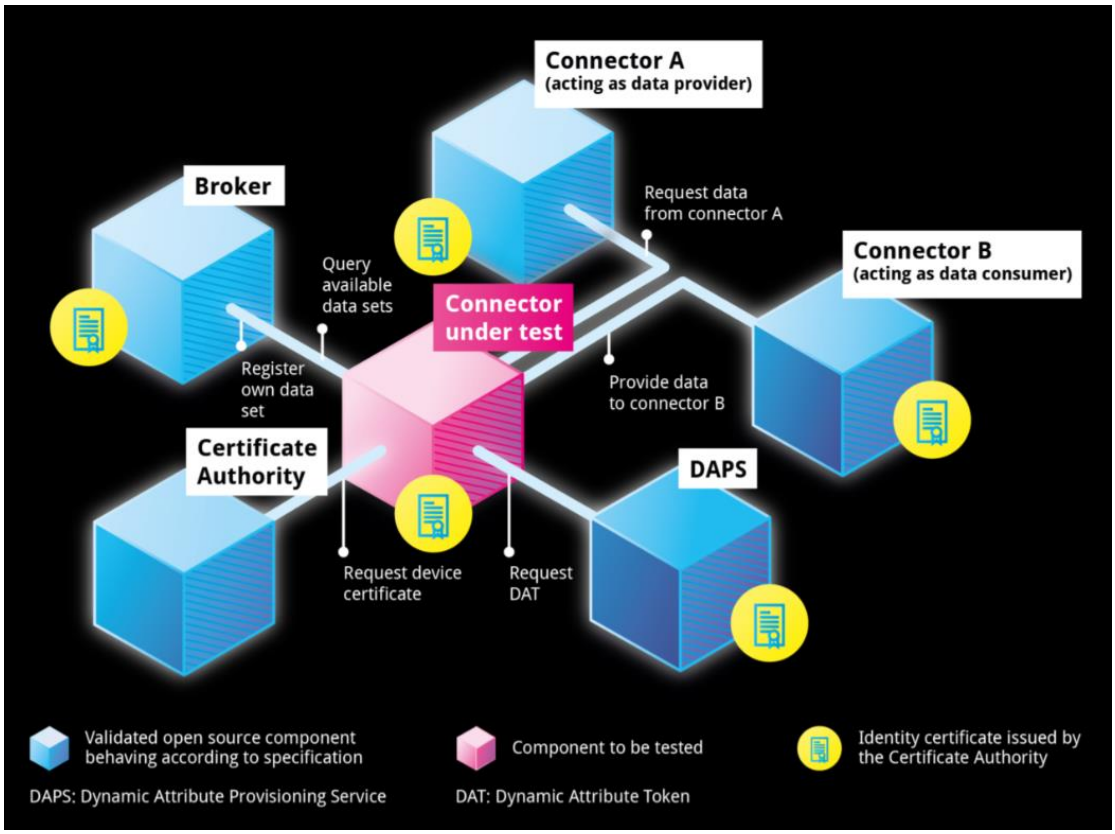


Figure 51 Testing of new components in the Testbed environment [42]

However, it can also be used as a **Minimum Viable Data Space** (**MVDS)** [44], a minimalistic Data Space that has the necessary parts to create a working Data Space environment.

Its modular nature allows for the integration of several open-source components to facilitate (or simulate) several participants.

At its core state, it has the essential components:

1. Two Connectors: One acting as a data provider and one acting as a data consumer.
2. A Broker: Responsible for data discovery.

3. The Certificate Authority: Stores, verifies and signs the certificate of each component.
4. The Dynamic Attribute Provisioning System (DAPS): Provides Dynamic Attribute Tokens that are used to verify participants at every step.

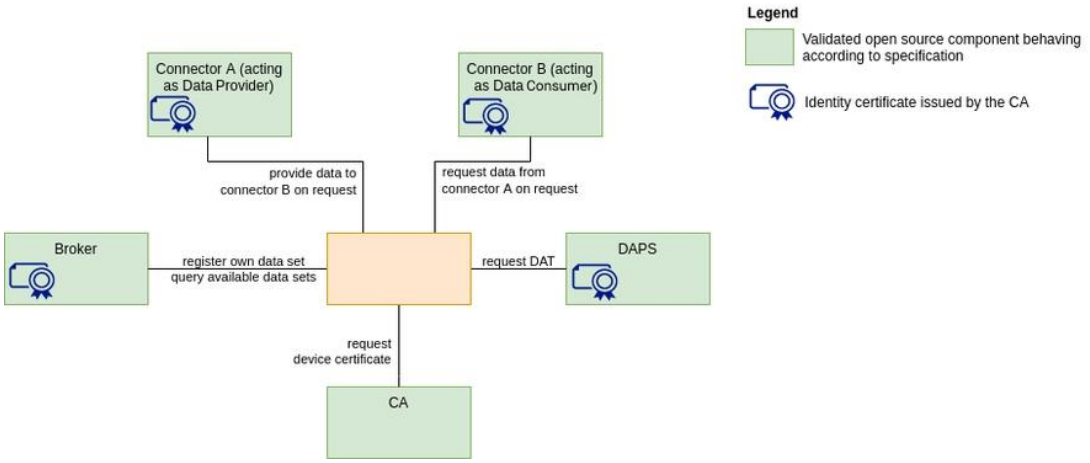


Figure 52 Technical diagram of entities in the IDS Reference Testbed

In the context of this thesis, the IDS Testbed was used as the backbone for deploying the Civil Protection demo application. Its extensibility made it possible to integrate simulated APIs, enforce security through certificates, and validate the controlled exchange of sensitive and heterogeneous data sources relevant to evacuation planning.

6.2 Deployment and Extension

In this subchapter, we will cover the deployment of the Testbed in a Linux environment.

6.2.1 Requirements

Below are presented several dependencies and tools that will be used for the swift execution of the steps ahead.

System:

1. 4GB+ of RAM
2. 50GB storage
3. The guide was created in the Ubuntu 24.04.2 LTS environment, but any Linux environment should work similarly.

Tools:

1. Git: Required to clone the repository but is also extremely useful for version tracking in any developing scenario [45].

```
sudo apt-get install git-all
```

2. Docker Engine: The engine that runs each component [18].

```
sudo apt-get install docker.io
```

3. Docker compose: The tool that orchestrates multi-container stacks [19].

```
sudo apt-get install docker-compose
```

4. CFSSL & cfssljson: CloudFlare's toolkit for Public Key Infrastructure (PKI) [46].

```
sudo apt install -y golang-cfssl
```

5. OpenSSL: Library also used for certificate handling [47].

```
sudo apt install -y openssl
```

Apps:

1. Postman: Used to run some basic scripts to confirm the installation and configuration of the testbed [48].
2. (Optional) Visual Studio Code: For editing and inspecting files [49].

Ports: Although customizable, by default ports like 443, 444, 8080, 8081 are taken up by the Testbed.

6.2.2 Testbed Deployment

After installing the tools and dependencies, the second step in the deployment is the cloning of the IDS Testbed repository. So, after changing to the directory that the project must be placed in, we start by running the following commands on a terminal.

```
git clone https://github.com/International-Data-Spaces-Association/IDS-testbed.git
cd IDS-testbed
```

After the execution we are in the root directory of the Testbed project.

Before attempting to run it, we must edit the docker-compose.yml file and for each Connector (so currently connectora and connectorb), under 'environment:' we must add one line (one space after the dash). This solves a common crashing issue.

```
-
SPRING_AUTOCONFIGURE_EXCLUDE=org.springframework.boot.actuate.autoconfi
gure.metrics.SystemMetricsAutoConfiguration
```

Figure 53 Disabling metrics for spring

The next step is to run the command to launch the docker containers.

```
docker compose up
```

Figure 54 Running the docker containers

If we don't encounter any errors, the Testbed should be up and running.

To test the core functions (posting offers, discovering data, registering connectors) we can import the IDS Testbed Postman collection for pre-configuration of the Testbed [50].

After importing the collection, we can see several requests that we can make to interact with it. At this point it is crucial to understand the offer model architecture of the IDS Testbed, which is based on IDS standards.

The key entities that revolve around a resource offering are the following:

- Catalog: A collection of relevant offers offered by a Connector
- Offer: A record of the offer with its basic elements (title, description, keywords and others)
- Representation: Metadata about the dataset/api (format, language)
- Artifact: The accessing of the dataset/api (offers title, description, value). It can either offer the data or provide access to other APIs.
- Rule: Machine readable policy statements. Examples may include the following:

```
"ids:constraint": [{
  "@type": "ids:Constraint",
  "ids:leftOperand": { "@id": "idsc:PURPOSE" },
  "ids:operator": { "@id": "idsc:EQUALS" },
  "ids:rightOperand": "RESEARCH"
}]
```

Figure 55 Example rule that limits use purpose to research

```
"ids:constraint": [{
  "@type":"ids:Constraint",
  "ids:leftOperand":{"@id":"idsc:ELAPSED_TIME"},
  "ids:operator":{"@id":"idsc:LTEQ"},
  "ids:rightOperand":"P30D"
}]
```

Figure 56 Example rule that limits use time to only 30 days after acquiring

- Contract: The sum of rules restricting the usage of an offer.

The relations between these entities are the following:

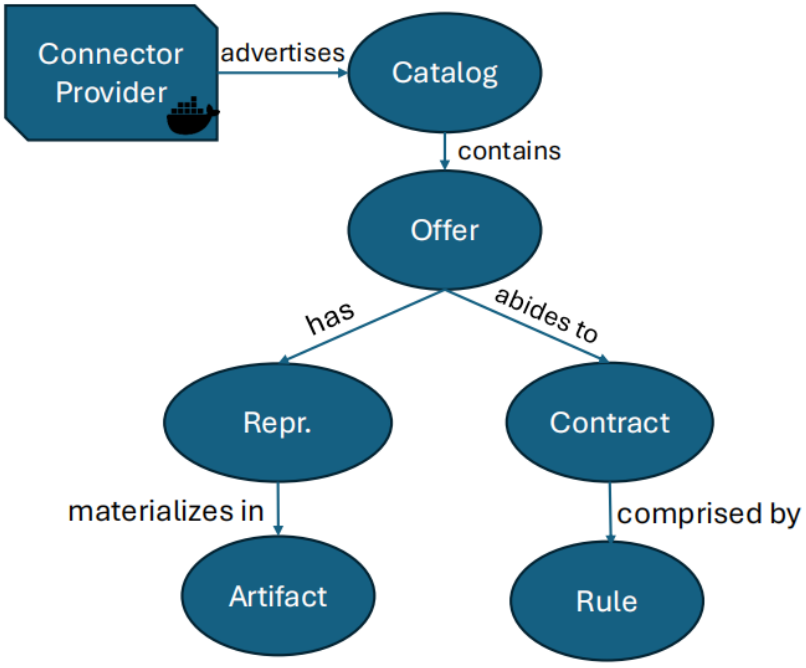


Figure 57 Relations between offer entities

Those relations can also be inferred by the overall process of posting a resource through the IDS Testbed postman collection [50].

First the offer and the catalog are created. The next step is to add the offer to the catalog (it can contain more than one offer).

Next, the rules are created. In the collection only one rule is created for the sake of showcasing the entire process, but several rules can be created and then banded to a single, newly created, contract (also mentioned as contract template).

The next step is to create a representation of the offer, and then the artifact that materializes that representation. These two entities are then bonded together.

The last step is to connect the representation (that already has the artifact registered to it) to the offer, as well as bind the offer to the contract, with all the rules it is comprised of.

Other than creating the offer, and its relevant entities, the collection shows the requests that:

- Register the Connectors at the metadata Broker. This makes the connectors discoverable, making their descriptions reachable from other connectors.
- Request self-descriptions of Connectors. This is how consumers get access to provider's catalogs. Then they can search through them to find desired resources.
- Request information about a resource. After finding an interesting resource, the consumer can request all the relevant information regarding it (rules, contracts, artifacts, representations).
- Negotiate contracts. The rules of the contract are negotiated and if they match with the desired policies from the consumer, an agreement is reached, and its ID is returned.
- Request data based on the agreement reached. After the creation of an agreement and acquisition of an agreementID, the consumer can request the data that the contract offers.
- Request Broker self-description. Multiple metadata Brokers may exist in the Data Space, and querying each one to see its information and capabilities can be useful.

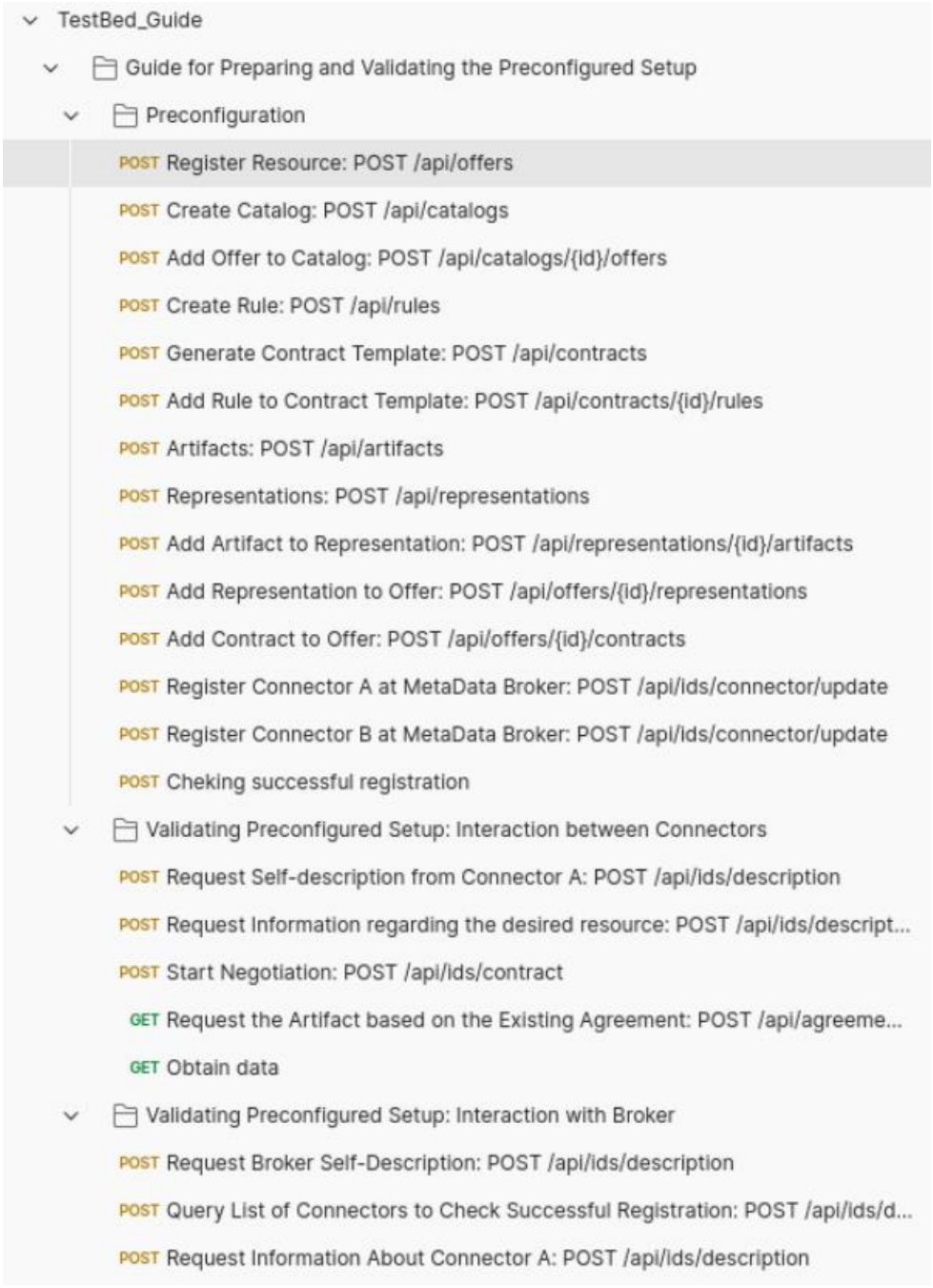


Figure 58 Postman collection for pre-configuration of the Testbed [50]

6.2.3 Adding Connectors

The building block of any Data Space is its connectors. The Testbed offers 2 Connectors (one consumer and one provider) out-of-the-box and, while sufficient for basic data exchanges, do not provide the diversity of providers that a real-world Data Space initiative would offer.

Hence, the next step is to explore the process of adding more Connectors to the ecosystem.

We will examine the case where a provider Connector (with the name of Connector C) will be added to the working Data Space.

The relevant files of the provider will be placed in the root directory in a specific folder, to prevent confusion, although in a real-world scenario, such files would be placed to the facility that hosts the Connector (which would typically be the data provider or consumer, with the exception of cases where connector-as-a-service is employed).

The first step is to create the folder structure for the new Connector in the root directory.

```
DataspaceConnectorC/conf/
```

Create a file that will contain the self-description of the connector.

```
DataspaceConnectorC/selfDescription.json
```

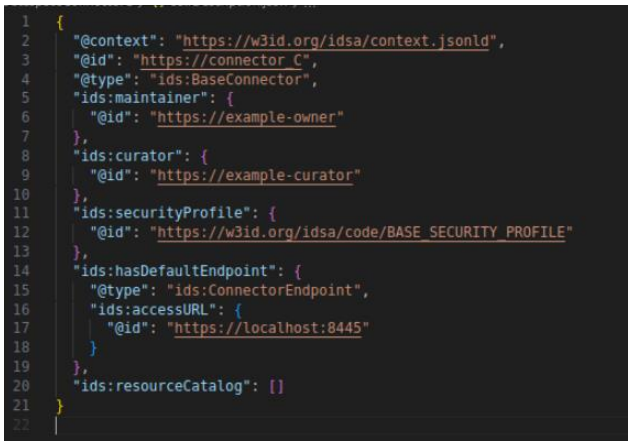


Figure 59 Example self-description of Connector C

Create a file that contains important information about the Connector such as its type, its context model, version, truststore, keystore, status and more.

```
DataspaceConnectorC/conf/config.json
```

DataspaceConnectorC > conf > {} config.json > ...

```
{
  "connector": {
    "title": "Connector C",
    "description": "Connector C with static example resources",
    "curator": "https://example-curator",
    "maintainer": "https://example-owner",
    "securityProfile": "BASE_SECURITY_PROFILE",
    "version": "1.0.0",
    "modelVersion": "4.2.7",
    "inboundModelVersion": ["4.2.7"]
  },
  "default": {
    "agreementValidity": 60,
    "target": "http://connectorc:8086/api/ids/data"
  },
  "@context": {
    "ids": "https://w3id.org/idsa/core/",
    "idsc": "https://w3id.org/idsa/code/"
  },
  "@type": "ids:ConfigurationModel",
  "@id": "https://w3id.org/idsa/autogen/configurationModel/connector-c-uuid",
  "ids:configurationModelLogLevel": {
    "@id": "idsc:MINIMAL_LOGGING"
  },
  "ids:connectorDeployMode": {
    "@id": "idsc:PRODUCTIVE_DEPLOYMENT"
  },
  "ids:connectorDescription": {
    "@type": "ids:BaseConnector",
    "@id": "https://connector_C",
    "ids:publicKey": {
      "@type": "ids:PublicKey",
      "@id": "https://w3id.org/idsa/autogen/publicKey/connector-c",
      "ids:keyType": { "@id": "idsc:RSA" },
      "ids:keyValue": "MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8A..."
    },
    "ids:description": [
      {
        "@value": "IDS Connector C with static example resources",
        "@type": "http://www.w3.org/2001/XMLSchema#string"
      }
    ],
    "ids:version": "1.0.0",
    "ids:hasDefaultEndpoint": {
      "@type": "ids:ConnectorEndpoint",
      "@id": "https://w3id.org/idsa/autogen/connectorEndpoint/connector-c",
      "ids:accessURL": {
        "@id": "https://connectorc:8086/api/ids/data"
      }
    },
    "ids:outboundModelVersion": "4.2.7",
    "ids:inboundModelVersion": [
      "4.0.0", "4.1.0", "4.1.2", "4.2.0", "4.2.1",
      "4.2.2", "4.2.3", "4.2.4", "4.2.5", "4.2.6", "4.2.7"
    ],
    "ids:title": [
      {
        "@value": "Dataspace Connector C",
        "@type": "http://www.w3.org/2001/XMLSchema#string"
      }
    ],
    "ids:securityProfile": { "@id": "idsc:BASE_SECURITY_PROFILE" },
    "ids:curator": { "@id": "https://example-curator" },
    "ids:maintainer": { "@id": "https://example-owner" }
  },
  "ids:trustStore": {
    "@id": "file:///config/truststore.p12"
  },
  "ids:keyStore": {
    "@id": "file:///conf/connectorC.p12"
  },
  "ids:connectorStatus": {
    "@id": "idsc:CONNECTOR_ONLINE"
  }
}
```

Figure 60 Example config.json of Connector C

Next, we change the working directory to:

```
CertificateAuthority/pkiInput
```

Create a file that will be used for creating the connector's certificates.

```
CertificateAuthority/pkiInput/connectorC.json
```

Figure 61 Example Connector C configuration file for certificate generation

Run the command:

```
cfssl genkey connectorC.json | cfssljson -bare connectorC
```

```
kostas@mvds:~/Desktop/Tutorial/IDS-testbed/CertificateAuthority/pkiInput$
cfssl genkey connectorC.json | cfssljson -bare connectorC
```

Figure 62 Generating key and certificate request for Connector C

This creates the files:

```
connectorC-key.pem
connectorC.csr
```

These are the private key of the Connector and the certificate signing request built from it. These must be moved to the following directory:

```
CertificateAuthority/data-scssl/certs
```

We change working directory that directory too.

Next step is to sign the request with the certificate authority's key, receiving the X.509 certificate [51].

Run:

```
cfssl sign \
 -ca=../ca/ca.pem \
 -ca-key=../ca/ca-key.pem \
 connectorC.csr | cfssljson -bare connectorC
```

```
kostas@mvds:~/Desktop/Tutorial/IDS-testbed/CertificateAuthority/data-cfssl/certs$
cfssl sign \
  -ca=../ca/ca.pem \
  -ca-key=../ca/ca-key.pem \
  connectorC.csr | cfssljson -bare connectorC
```

Figure 63 Creating X.509 certificate for Connector C

The .pem extension is a generic one used for all Base-64 encoded certificates and keys, so for clarity we will change the extension of the resulting file to .crt which is used specifically for X.509 certificates.

Renaming

```
connectorC.pem
```

to

```
connectorC.crt
```

```
kostas@mvds:~/Desktop/Tutorial/IDS-testbed/CertificateAuthority/data-cfssl/certs$
cp connectorC.pem connectorC.crt
```

Figure 64 Renaming certificate

Next, we combine the private key and the certificate into a single connectorC.p12 file (keystore):

```
openssl pkcs12 -export \
 -out connectorC.p12 \
 -inkey connectorC-key.pem \
 -in connectorC.crt \
 -passout pass:password
```

```
kostas@mvds:~/Desktop/Tutorial/IDS-testbed/CertificateAuthority/data-cfssl/certs$
openssl pkcs12 -export \
  -out connectorC.p12 \
  -inkey connectorC-key.pem \
  -in connectorC.crt \
  -passout pass:password
```

Figure 65 Creating the keystore

Finally, we export the certificate (this is what the DAPS will use to verify the connector).

```
openssl pkcs12 -in connectorC.p12 \
 -out connectorC.cert \
 -nokeys -nodes \
 -passin pass:password
```

```
kostas@mvds:~/Desktop/Tutorial/IDS-testbed/CertificateAuthority/data-cfssl/certs$
openssl pkcs12 -in connectorC.p12 \
  -out connectorC.cert \
  -nokeys -nodes \
  -passin pass:password
```

Figure 66 Exporting the certificate

After creating all the certificates necessary, we need to move (or copy) the file

```
connectorC.p12
```

into the directory

```
DataspaceConnectorC/conf
```

And the file

```
connectorC.cert
```

into

```
DAPS/keys
```

The next step would be to extract the Subject Key Identifier (SKI) and Authority Key Identifier (AKI) [52] from our certificates and add them to the client list of the DAPS. However, the Testbed does offer a script for that.

To run it we change the working directory to DAPS and run:

```
./register_connector.sh connectorC
```

A new input should appear in the following file about the newly added connector.

```
DAPS/config/clients.yml
```

The last step of the certifications process is to tell the Connector C that it must trust the DAPS. That is possible by providing the truststore certificate of DAPS. The file responsible for that is found in

```
DataspaceConnectorA/conf/truststore.p12
```

Or

```
DataspaceConnectorB/conf/truststore.p12
```

And must be copied to the following directory so it can be mounted on the new container

```
DataspaceConnectorC/conf
```

And the last step of the entire process of adding a new Connector to the Data Space is the addition and configuration of the container that will run.

For that we will edit the docker-compose.yml file found at the root of the project.

Three changes need to be made:

1. Insert a Postgres container named postgresc (modeled by postgresa and postgresb). Special attention is needed to avoiding using ports that other processes or containers might be already using.

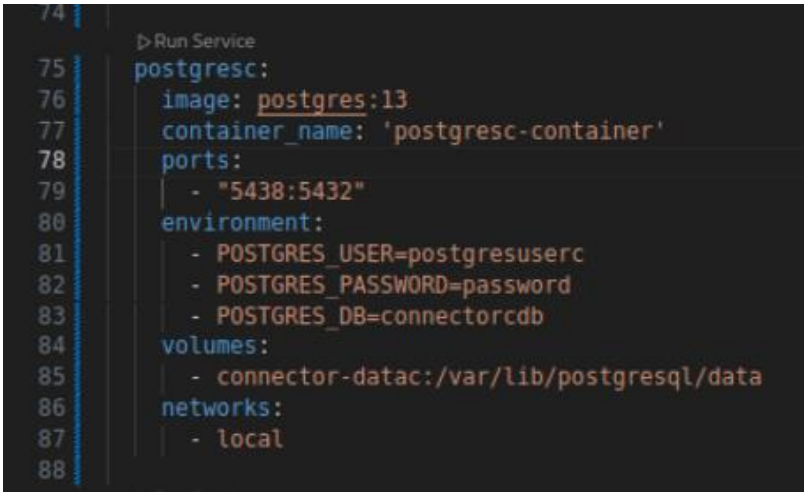


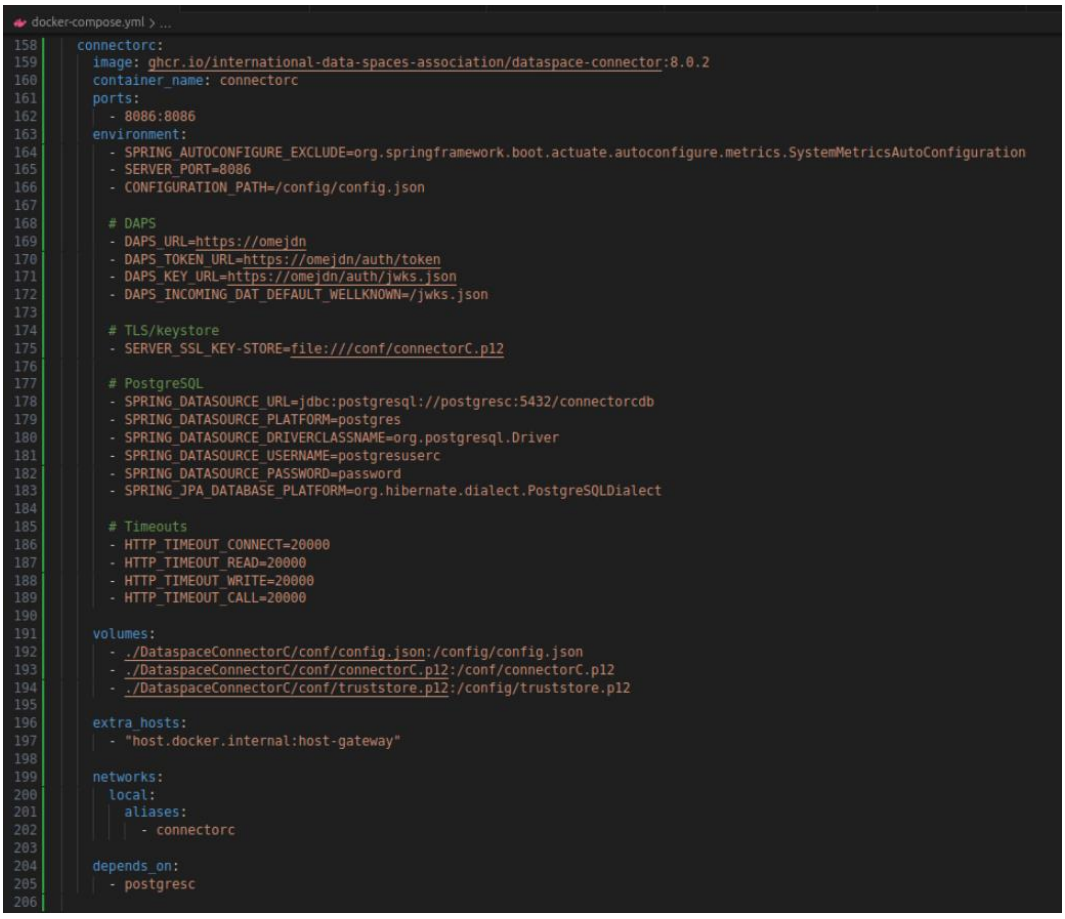
Figure 67 Postgresc container configuration

2. Add a volume for the data of the new Connector, so its data can be saved on restart.

```
247  volumes:
248    broker-fuseki: {}
249    connector-dataa: {}
250    connector-datab: {}
251    connector-datac: {}
252
```

Figure 68 Volume addition for the new connector

3. Add and configure the Connector container itself.
   The most notable configuration variables in its environment are the DAPS URLs (responsible for providing valid token-DATs), the keystore (being the path where the connector's certificate will be stored inside the connector), the postgres configurations for its data storing needs and the aliases through which it can be known in the local network.



```
docker-compose.yml > ...
158   connectorc:
159     image: ghcr.io/international-data-spaces-association/dataspace-connector:8.0.2
160     container_name: connectorc
161     ports:
162       - 8086:8086
163     environment:
164       - SPRING_AUTOCONFIGURE_EXCLUDE=org.springframework.boot.actuate.autoconfigure.metrics.SystemMetricsAutoConfiguration
165       - SERVER_PORT=8086
166       - CONFIGURATION_PATH=/config/config.json
167
168       # DAPS
169       - DAPS_URL=https://omejdn
170       - DAPS_TOKEN_URL=https://omejdn/auth/token
171       - DAPS_KEY_URL=https://omejdn/auth/jwks.json
172       - DAPS_INCOMING_DAT_DEFAULT_WELLKNOWN=/jwks.json
173
174       # TLS/keystore
175       - SERVER_SSL_KEY-STORE=file:///conf/connectorC.p12
176
177       # PostgreSQL
178       - SPRING_DATASOURCE_URL=jdbc:postgresql://postgresc:5432/connectorcdb
179       - SPRING_DATASOURCE_PLATFORM=postgres
180       - SPRING_DATASOURCE_DRIVERCLASSNAME=org.postgresql.Driver
181       - SPRING_DATASOURCE_USERNAME=postgresuserc
182       - SPRING_DATASOURCE_PASSWORD=password
183       - SPRING_JPA_DATABASE_PLATFORM=org.hibernate.dialect.PostgreSQLDialect
184
185       # Timeouts
186       - HTTP_TIMEOUT_CONNECT=20000
187       - HTTP_TIMEOUT_READ=20000
188       - HTTP_TIMEOUT_WRITE=20000
189       - HTTP_TIMEOUT_CALL=20000
190
191     volumes:
192       - ./DataspaceConnectorC/conf/config.json:/config/config.json
193       - ./DataspaceConnectorC/conf/connectorC.p12:/conf/connectorC.p12
194       - ./DataspaceConnectorC/conf/truststore.p12:/config/truststore.p12
195
196     extra_hosts:
197       - "host.docker.internal:host-gateway"
198
199     networks:
200       local:
201         aliases:
202           - connectorc
203
204     depends_on:
205       - postgresc
206
```

Figure 69 Container configuration for the new connector

After following these steps and restarting the docker processes:

```
docker compose down
```

and

```
docker compose up
```

The new connector start successfully.

For quick testing and validation, it is possible to change the URL of Connector A in the Postman collection variables to the identifier of Connector C (in this example change the variable CONNECTORA_URL= https://localhost:8080 to https://localhost:8086).

In the repository of the final implementation [24] there are Node.js scripts that set up each Connector and simulate a simple procedure of offer posting, negotiation, and data acquisition, for example:

```
ConfigureConnector/mhtroo.js
```

6.2.4 Common Issues

The instructions on the official IDS Testbed installation guide do offer a general direction for the deployment process, but in some cases are either outdated, or assuming familiarity with specific concepts which an aspiring developer might find difficult to understand.

Some of the most common issues will be mentioned in this subchapter to prevent new developers from encountering them or help them deal with them.

1. As mentioned before, trying to run the out-of-the-box components will cause an error. That error is caused by the incompatibility of the Spring Boot version (mainly 2.6+) with some JDK metrics processes inside the JDK environment. The easiest fix is to disable such metrics as seen in subchapter 6.2.2.
2. The certificate process, although straightforward for developers familiar with the standards of modern certification procedures, is not friendly to developers who want to experiment with the Data Space ecosystem to see if it fits their needs for a data sharing platform. For that reason, the certificate process for registering a new Connector has been detailed in subchapter 6.2.2 and explained for novices in the field.
3. On the same issue as point number 2, in the official instructions there is the mention of a pki.py file that handles part of the certification process [53]. However, that file does not currently exist in the repository and one needs to search in the issues [54] of the repository to see that the recommended

approach has been changed to use cfssl (as we have examined) without any further instructions provided, or the official instructions changed.

4. Another issue with the official guide for the Testbed installation [53] is the usage of the deprecated command

```
docker-compose up
```

instead of the new version

```
docker compose up
```

As we have used it before.

5. One of the most common issues when adding new Connectors is the “unable to retrieve valid DAT”. The inability to retrieve a Dynamic Attribute Token can have many causes, but the most common one is the misconfiguration of the

```
DAPS/config/clients.yml
```

or incorrect configuration of the certificates. In such cases it is recommended to delete previous configurations follow the process described above carefully and in the correct order.

6. Ports conflict is also a common issue, especially if the Testbed is being installed on a machine that is used by other services. As an example, this is the error we get when port 80 (common port) is already in use:

```
failed to bind host port for 0.0.0.0:80: address already in use
```

And to fix this issue we need to find the process that is using that port by using:

```
sudo lsof -i :80
```

Finally, we can kill the process using its Process ID:

```
sudo kill <PID>
```

Or edit the docker-compose.yml file in order to map the port to something else in the machine. Specifically, port 80 is used by the omejdn service.

7. Another issue that often arises -although not strictly related to the Testbed itself- is lack of permissions of the docker processes to read each connector’s configuration files (mainly those in the directory:

```
DataspaceConnectorC/conf/
```

). The simple fix is to allow everyone to have read access to those files using the chmod command with 644 as the argument for the new permissions [55].

Overall, the process of deploying the Testbed is not particularly complex, but it is held back by a few practical obstacles. The available documentation is sometimes outdated, or not detailed enough, which means that certain steps require a lot of trial and error to resolve. Even missing files were encountered during the process. Additionally, the lack of a “common issues” page that would resolve most of the issues by pointing developers in the right direction is also a negative aspect of the documentation. All these suggest that the technical foundation is solid, but better supporting material would make the setup process smoother, especially for new developers in the field.

7 Demo App Implementation

After examining the environment used and the data that will be available through the Data Space, the next step is to review the Demo Application that shows a working example of the possibilities that arise from the creation of a data sharing community based on trust and security.

The application will discover and gather data from the sources we have mentioned, while combining them and creating an evacuation plan for the people with disabilities in areas affected by natural disasters.

The source code for the application is in the “Civil Protection” folder at the root of the thesis repository [24].

7.1 App Architecture

The participants of the Data Space have already been mentioned, but in short, they are the following:

- The Amea club API: An example of clubs, municipalities and individuals registering in catalogs of people with disabilities. A Node API exposing an SQL database with encrypted data. The access is only allowed with an access token, while the transmitted data are also encrypted with a different key. The API is connected to the Data Space via the Connector Amea Club (in place of Connector C in our previous example). Through that Connector it registers with the Broker and becomes discoverable, according to the self-description of the Connector.
- Mhtroo API: A server simulating the exposing of the government’s registry (a MongoDB database). Similarly to the Amea club API, it is a Node app that protects sensitive data both at rest, and in transit (using encryption and token-based access). Having its own Connector, it connects to the Data Space and the metadata Broker.
- Fleet generator: A Python script that simulates the fleet of civil service vehicles posting updates about their status. It creates and updates the vehicle entities in the Lab’s [22] FIWARE platform, while offering the option to run a simulation with the vehicles moving across the road network until they are given directions for evacuation planning.
  The relevant Connector advertises access to the FIWARE platform, and not the fleet generator itself.
- Disasters API: A server that provides information about natural disasters. Drawing data from the Copernicus API, it stores and publicly shares the information without restricting access. The Natural Disasters Connector advertises access to this data offering to the metadata Broker.
- Civil Protection Demo App: A Node.js application that searches for and uses the data in the Data Space to provide an overview of the situation on a map, and

options to organize evacuation of affected areas. The app technologies and features will be discussed in the next subchapters.

The access flow of the implementation is described by the diagram below.

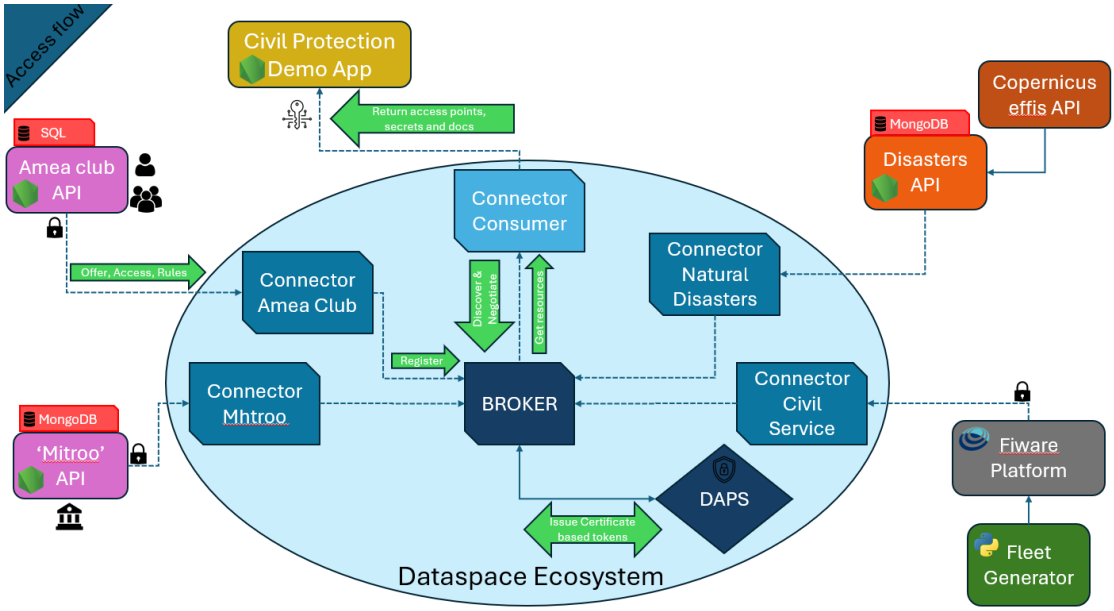


Figure 70 Access flow of the implementation

The procedures for posting and discovering offers has been examined on a theoretical level (subchapter 2.2) and on a general level (subchapter 6.2.3). Next we will examine the specific example of Connector Mhtroo posting an offer and the app discovering and consuming it. The process can be found in the ConfigureConnector/mhtroo.js script.

The first part of the script covers the provider's process.

1. The first step is registering the Connector to the network, that is achieved by the certificate generation process (subchapter 6.2.3) and the request from the Connector to receive a valid Dynamic Attribute Token (DAT) based on those certificates.
2. The provider Connector then starts creating the offer, special attention to the keywords which will be used in the discovery phase.

```
58      const offerResp = await axios.post(
59        `${PROVIDER_URL}/api/offers`,
60        {
61          title: 'Amea Data Offer',
62          description:
63            'Directory of Amea entries (secured access; configuration delivered as JSON artifact)',
64          keywords: ['AMEA', 'directory', 'secured'],
65          publisher: DATA_API_URL,
66          language: 'EN',
67          sovereign: DATA_API_URL,
68          endpointDocumentation: DATA_API_URL,
69          paymentModality: 'undefined',
70        },
71        { baseURL: PROVIDER_URL }
72      );
```

Figure 71 Offering of data from Mhtroo Connector

3. As seen in subchapter 6.2.2 the offer is placed in a catalog, and is linked to a representation, artifact as well as a contract comprised of rules. So, the next steps are the creation and linkage of those to the offer.
   Special attention is needed in the creation of the artifact, where the access method is described.

```
// Build a minimal, self-describing secured config object
const artifactConfig = {
  type: 'secured-config',
  version: 1,
  endpoint: DATA_API_URL, // upstream data API (no secrets in URL)
  auth: {
    scheme: 'Bearer',
    token: AMEA_ACCESS_TOKEN, // sensitive
  },
  crypto: {
    alg: 'AES-256-GCM',
    secret: AMEA_DECRYPTION_SECRET, // sensitive
    // You can add iv/nonce rotation guidance if your consumer uses envelope encryption
    encoding: 'utf8',
  },
  meta: {
    service: 'amea',
    issuedAt: new Date().toISOString(),
    note: 'Confidential. Do not log/store in plaintext.',
  },
};

const artResp = await axios.post(`${PROVIDER_URL}/api/artifacts`, {
  title: 'Amea Secured Config Artifact',
  description:
    'JSON configuration containing access token and decryption secret for AMEA data access',
  value: JSON.stringify(artifactConfig), // JSON as string; provider will deliver as-is
  automatedDownload: false, // not a URL fetch; deliver literal JSON
});
```

Figure 72 Configuration of the artifact object linked to the offer of Connector Mhtroo

The testbed allows for the data to be stored in the container as plain json files, but providing the access method through tokens is a way more flexible approach, especially for sensitive data.

The second part of the script validates that the process has been successful by targeting the offer through the consumer Connector and gaining access to the data.

However, we will examine the consuming process from the perspective of the app, discovering all the relevant data sources for its goal.

The script that serves that purpose is /Civil Protection/discover-negotiate-save.js.

The process is as follows:

1. Loading the configuration variables (such as the Broker URL and authentication credentials)
2. Registering the consumer Connector (same as the provider)

```
    try {
      console.log('🔗 Registering consumer at broker (optional but common)...');
      await axios.post(`${CONSUMER_URL}/api/ids/connector/update`, null, {
        params: { recipient: BROKER_API },
      });
    } catch (e) {
      console.warn('⚠️  Consumer registration failed/ignored:', e.response?.status || e.message);
    }
```

Figure 73 Registering consumer Connector at Broker

3. Discovers all available Connectors and goes through their self-descriptions and catalogs, searching for the keywords provided.

```
    try {
      const listResp = await axios.post(`${CONSUMER_URL}/api/ids/description`, null, {
        params: { recipient: BROKER_API, elementId: `${BROKER_REV_PROXY}/connectors/` },
      });

      const data = listResp.data;
      const graph = data['@graph'] || [];
      const byId = Object.fromEntries(graph.map(n => [n['@id'], n]));

      const connectors = graph.filter(n => n['@type'] === 'ids:BaseConnector');
      const results = connectors.map(c => {
        const urls = extractAccessUrlsFromConnector(c, byId);
        const chosen = pickPublicUrl(urls);
        return {
          connectorId: c['@id'],
          sameAs: c['ids:sameAs'] ?? c.sameAs,
          accessURLs: urls,
          accessURL: chosen
        };
      }).filter(r => r.accessURL);

      accessUrls = new Set(results.map(r => r.accessURL));
      console.log(`   • Connector access URLs discovered: ${accessUrls.size}`);
    } catch (e) {
      console.warn('⚠️  Broker connector listing failed:', e.response?.status || e.message);
    }

```

Figure 74 Querying the Broker for list of Connectors

4. Negotiate the contract rules of each relevant offer and receive and agreement.

```
        // Negotiate contract
        await wait(WAIT_MS);
        const permissions = [
          {
            '@type': 'ids:Permission',
            'ids:title': [{ '@value': 'Usage Policy', '@type': 'http://www.w3.org/2001/XMLSchema#string' }],
            'ids:action': [{ '@id': 'https://w3id.org/idsa/code/USE' }],
            'ids:target': artifactId,
          },
        ];

        let agreementId = null;
        try {
          const contractResp = await axios.post(`${CONSUMER_URL}/api/ids/contract`, permissions, {
            params: {
              recipient,
              resourceIds: resourceId,
              artifactIds: artifactId,
              download: false,
            },
            headers: { 'Content-Type': 'application/ld+json' },
          });
          const agreementHref = contractResp.data?._links?.self?.href || '';
          agreementId = agreementHref.split('/').pop();
        } catch (e) {
          console.warn(`    ⚠️  Contract negotiation failed for ${idSuffix(resourceId)}:`, e.response?.status || e.message);
          runSummary.errors.push({
            provider: baseUrl,
            resourceId,
            step: 'contract',
            error: e.response?.data || e.message,
          });
          continue;
        }
```

Figure 75 Contract negotiation

5. Fetch the data link and get access to the payload (access method and secrets) based on the agreementId.

```
339             // Fetch data link
340             await wait(WAIT_MS);
341             let dataLink = null;
342             try {
343               const dl = await axios.get(`${CONSUMER_URL}/api/agreements/${agreementId}/artifacts`);
344               dataLink = dl.data?._embedded?.artifacts?.[0]?._links?.data?.href || null;
345             } catch (e) {
346               console.warn(`    ⚠  Getting data link failed:`, e.response?.status || e.message);
347               runSummary.errors.push({
348                 provider: baseUrl,
349                 resourceId,
350                 agreementId,
351                 step: 'data-link',
352                 error: e.response?.data || e.message,
353               });
354               continue;
355             }
356
357             if (!dataLink) {
358               console.warn('    ⚠  No data link returned.');
359               continue;
360             }
361
362             // Download payload
363             await wait(WAIT_MS);
364             let payload;
365             try {
366               const payloadResp = await axios.get(dataLink);
367               payload = await normalizePayload(payloadResp.data);
368             } catch (e) {
369               console.warn('    ⚠  Payload fetch/normalize failed:', e.response?.status || e.message);
370               runSummary.errors.push({
371                 provider: baseUrl,
372                 resourceId,
373                 agreementId,
374                 step: 'payload',
375                 error: e.response?.data || e.message,
376               });
377               continue;
378             }
379
```

Figure 76 Fetch data link and payload

6. Finally, store in a .json file the access method and secrets for each connector. That file will be used by the app to access the data according to the data flow diagram.

After discovering the relevant Connectors and their offers, and getting access to the respective APIs, it can start consuming data based on the secrets provided in the discovery process.

The data flow is as follows:

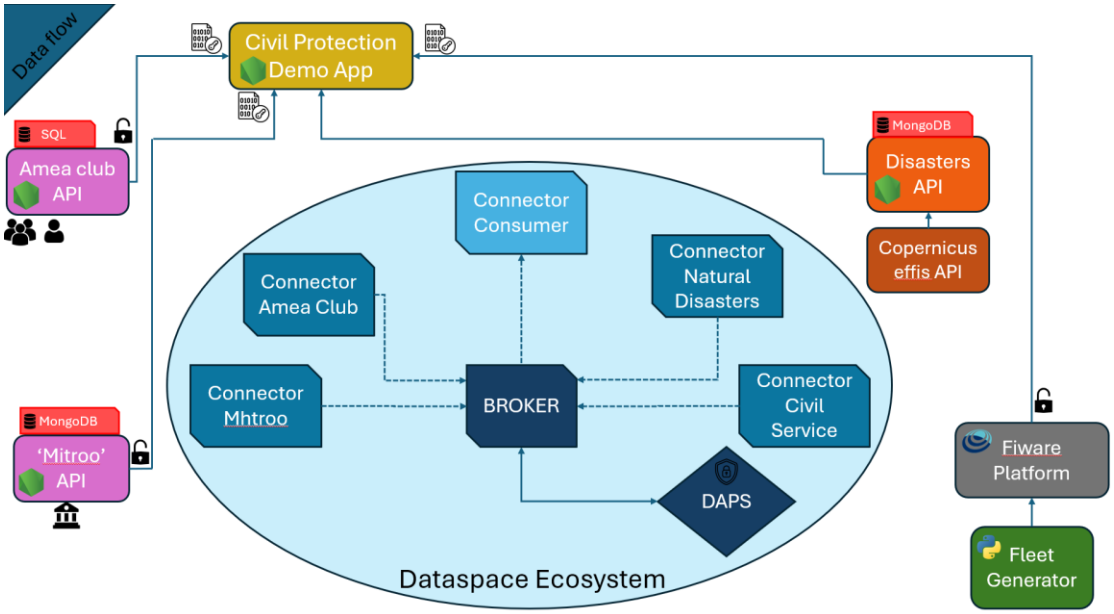


Figure 77 Data flow according to access secrets

The App provides the correct tokens to gain access to the APIs and it can decrypt the transmitted data according to the secret provided in the offer.

After examining the Ecosystem surrounding the demo application, all that is left is to delve into the architecture of the application itself.

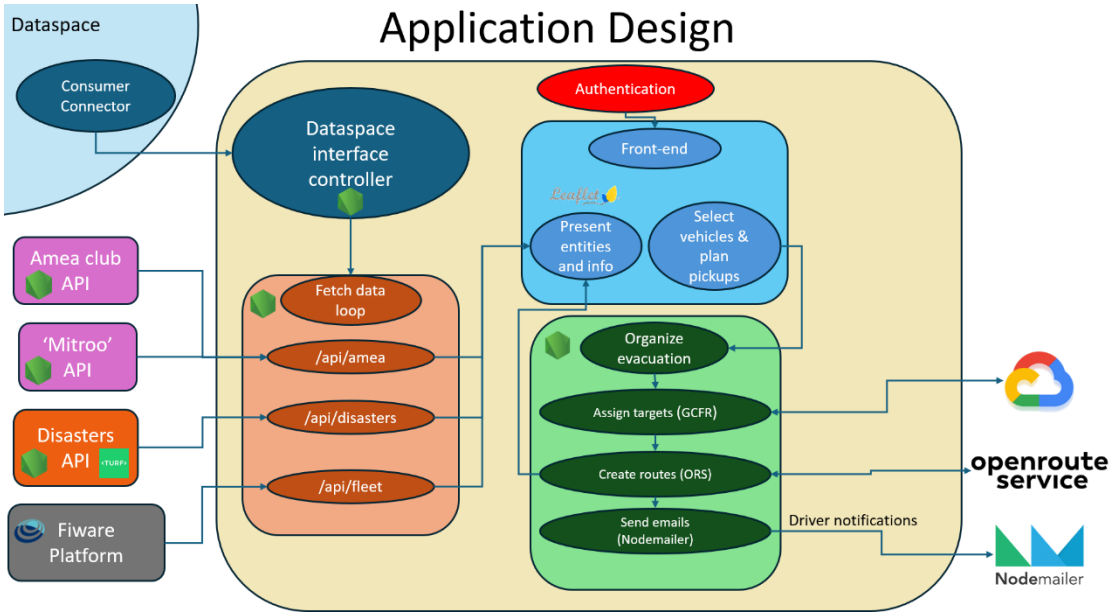


Figure 78 App architecure

The app includes a Data Space interface controller which communicates with the Data Space through the Consumer Connector, obtaining the access payloads as seen previously.

The access payloads are fed forward to the data fetching loop which communicates with the discovered APIs, receiving the data necessary for the functionality of the application.

The three types of data (People with disabilities, vehicle fleet, natural disasters) are then presented in the front end. The operator is then able to initiate the evacuation procedure that will be discussed in the following subchapters.

7.2 Technologies Used in the App

Other than the platforms mentioned in chapter 4, and the technologies discussed in chapter 5 (Data and APIs), the demo application itself employs a number of modern web technologies and third-party APIs in order to provide the operator with the tools he needs to organize an efficient evacuation plan, and have an overview of the current situation.

7.2.1 Web Mapping and Visualization Technologies

The central dashboard of the evacuation coordination system is delivered via Node.js, serving a client application written in HTML [56] and CSS [57]. To render the interactive map interface the project employs Leaflet [58], a popular open-source JavaScript library for development of map related projects. It provides the core map rendering engine and a simple API for adding layers, markers, and user interactions. In our implementation the positions of emergency vehicles (fetched from FIWARE Orion Context Broker) are displayed as SVG-based icons, for their scalability and clarity at different zoom levels. The map tiles themselves were sourced from OpenStreetMap, the collaborative open-data mapping project that provides detailed geographic basemaps [59].

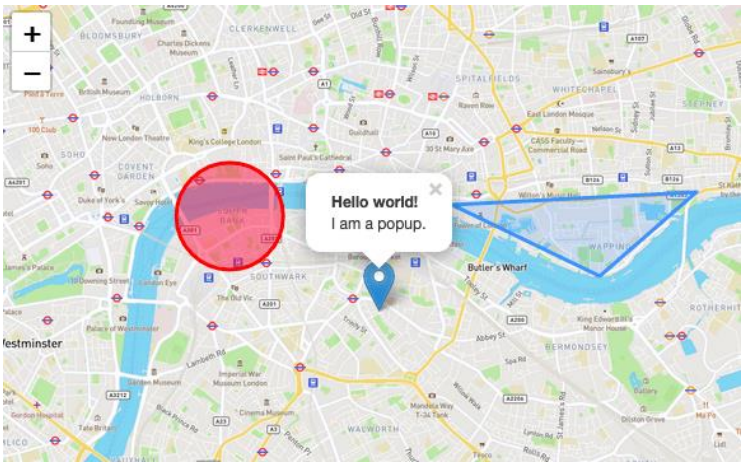


Figure 79 Leaflet showcase [60]

7.2.2 Routing and Optimization APIs

Optimization of the target assignment and routing is handled by a combination of specialized third-party APIs:

- Google Cloud Fleet Routing (CFR) [61]: CFR is a cloud-based optimization service designed for complex vehicle routing problems. It supports modeling vehicles with capacity constraints, defining targets (pickup locations), and optimizing assignments based on distance, time or resource constraints. In this thesis, CFR is used to assign people with disabilities requiring evacuation to the most suitable vehicles, ensuring that no vehicle exceeds its seating capacity.

```
453  async function optimizeTours(model) {
454    if (!GOOGLE_PROJECT) {
455      console.warn('GOOGLE_PROJECT not set; skipping optimization');
456      return null;
457    }
458    const url = `https://${ROUTE_OPTIMIZATION_ENDPOINT}/v1/projects/${encodeURIComponent(GOOGLE_PROJECT)}:optimizeTours`;
459
460    try {
461      const token = await getGcpAccessToken();
462      const r = await axios.post(
463        url,
464        { model, searchMode: 'CONSUME_ALL_AVAILABLE_TIME' }, // you can add optional knobs here
465
466        { headers: { Authorization: `Bearer ${token}`, 'Content-Type': 'application/json' }, timeout: 30000 }
467      );
468      return r.data || null;
469    } catch (e) {
470      console.warn('OptimizeTours failed:', e.response?.status, e.response?.data || e.message);
471      return null;
472    }
473  }
```

Figure 80 Google CFR request

- OpenRouteService (ORS) [62] and Open Source Routing Machine (OSRM) [63]: While CFR provides the assignment of targets to vehicles, the actual geographic routes were calculated using ORS and OSRM. Both services leverage OpenStreetMap data to generate realistic driving directions. Both services are employed as backup in case either is down during an evacuation scenario, since both are free services, the uptime is not 100%.

```
async function orsRoute(coords) {
  if (!Array.isArray(coords) || coords.length < 2 || !ORS_API_KEY) return null;
  try {
    const url = `${ORS_BASE}/v2/directions/driving-car/geojson`;
    const r = await axios.post(
      url,
      { coordinates: coords, instructions: false }, // full path via waypoints in given order
      { headers: { Authorization: ORS_API_KEY, 'Content-Type': 'application/json' }, timeout: 30000 }
    );

    const feat = r.data?.features?.[0];
    const geom = feat?.geometry;
    const sum  = feat?.properties?.summary; // { distance (m), duration (s) }
    // console.log(`ors route: ${JSON.stringify({ geom, sum })}`);

    if (!geom || !sum) {
      return null;
    }
    return {
      geometry: geom,                              // GeoJSON LineString
      distanceKm: (Number(sum.distance) || 0) / 1000,
      durationMin: (Number(sum.duration) || 0) / 60
    };
  } catch (e) {
    console.warn('ORS route failed:', e.response?.status || e.message);
    return null;
  }
}
```

Figure 81 Creating OSR route for each vehicle

7.2.3 Notification Tool

To notify field agents (or vehicle operators) about their assigned route and provide other essential info, Nodemailer was employed. Nodemailer is a Node.js library that allows the application to dispatch emails using SMTP [64].

```
async function sendRouteEmail(vehicleId, directionsUrl, recipient, stopsInfo) {
  let _text = `Vehicle ${vehicleId} route: ${directionsUrl}`
  let counter = 0;
  for (const stop of stopsInfo) {
    _text += `\n\nStop #${counter++}: ${stop.name}`;
    _text += `\n${stop.disability_info}\n`;
  }

  await transporter.sendMail({
    from: `"Route Planner" <${process.env.GMAIL_USER}>`,
    to: recipient,
    subject: `Directions for vehicle ${vehicleId}`,
    text: _text,
  });
}
```

Figure 82 Email dispatching

7.2.4 Security Technologies

As we have discussed in precious chapters, the APIs that provide the data essential for the purpose of the application, use encryption and token-based access. That means

the application itself also needs to employ decryption algorithms (mainly AES-256-GCM) to decrypt the incoming data have them serve their purpose. AES-256-GCM is widely regarded as a secure symmetric encryption/decryption algorithm offering confidentiality and integrity.

7.3 Showcase of the Demo App

The goal of the developed app is to provide a platform for a central Civil Protection operator to have an overview of the situation, get information about events and entities, plan evacuation efforts and dispatch directions and relevant information to vehicle operators.

7.3.1 App Overview

Since the app is directed only towards the central operator (or coordinator) of the Civil Protection agency, a basic authorization is required to enter the app.

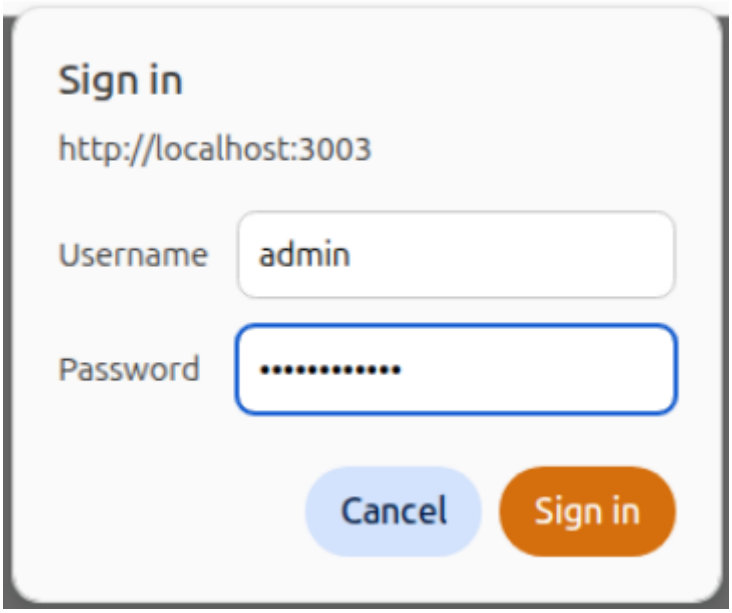


Figure 83 Signing in to the app

After logging in, the operator is presented with an overview of the current situation.

The areas that are currently affected by the natural disaster are marked as deep red, while the projected areas of effect -the evacuation zones- are shown as pink.

The available fleet is shown on the map, with the respective color for each type of vehicle (yellow for ambulances, red for fire trucks, blue for police cars), as well as in a list on the left side.

Each person with disabilities is also shown as a purple dot on the map.

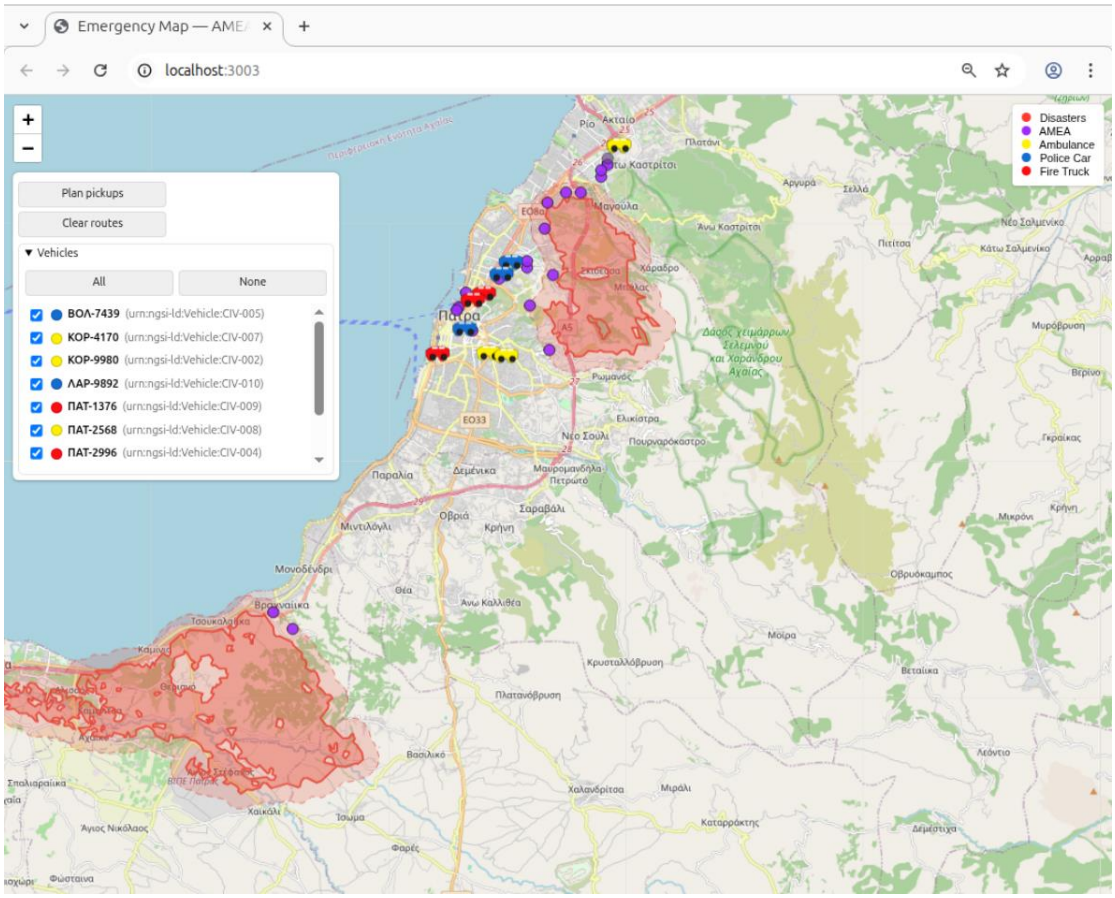


Figure 84 App overview

7.3.2 Information

By clicking on each entity on the map, important information regarding it is displayed.

1. Disasters and evacuation areas: The information displayed are the type of disaster, its danger level, its start (and possibly end) and the source that served this disaster.

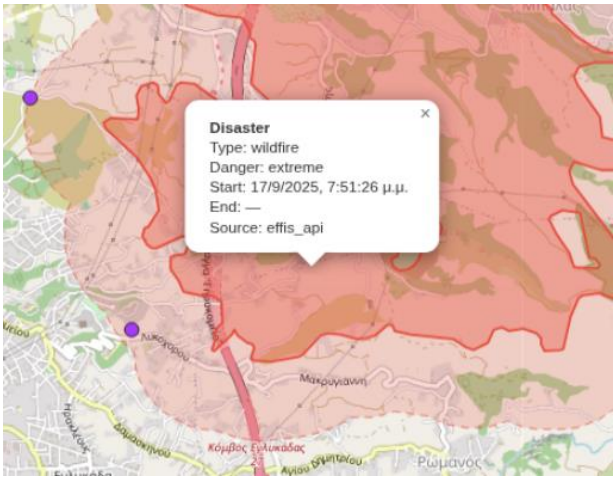


Figure 85 Disaster info

2. Fleet: Each vehicle displays its ID, its plate number, the type of vehicle, its speed and the available seats that can be used to transport people in need.

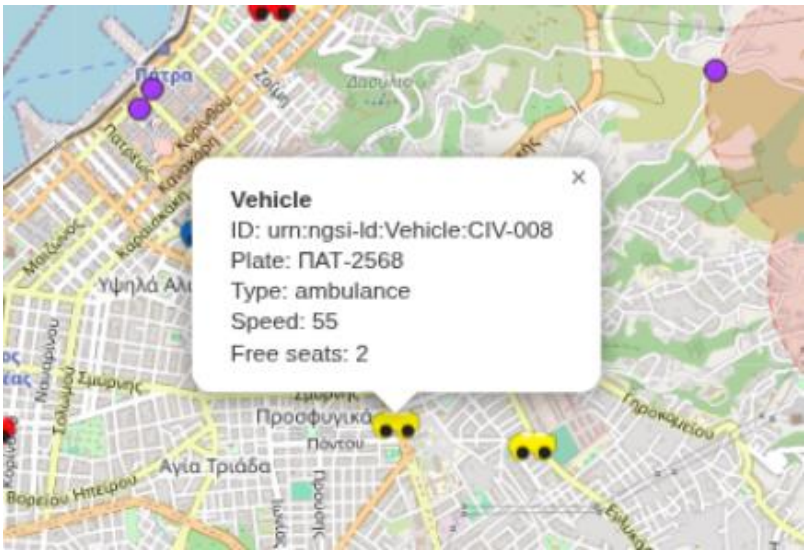


Figure 86 Vehicle info

3. People with disabilities (AMEA): The name, disability percentage, and contact points are displayed.

Figure 87 Info for people with disabilities.

Meanwhile the map itself offers information about street names, important buildings and local place names.

7.3.3 Planning and Dispatching

The operator is presented with a list of vehicles on the left-side of the screen. This list allows them to select which vehicles will take part in the planning and execution of the evacuation attempt. For example, fire trucks may not be able to assist in evacuation attempts since they will be needed to battle against the wildfire.

By selecting a vehicle, it becomes bright, while it dims if it is not selected.

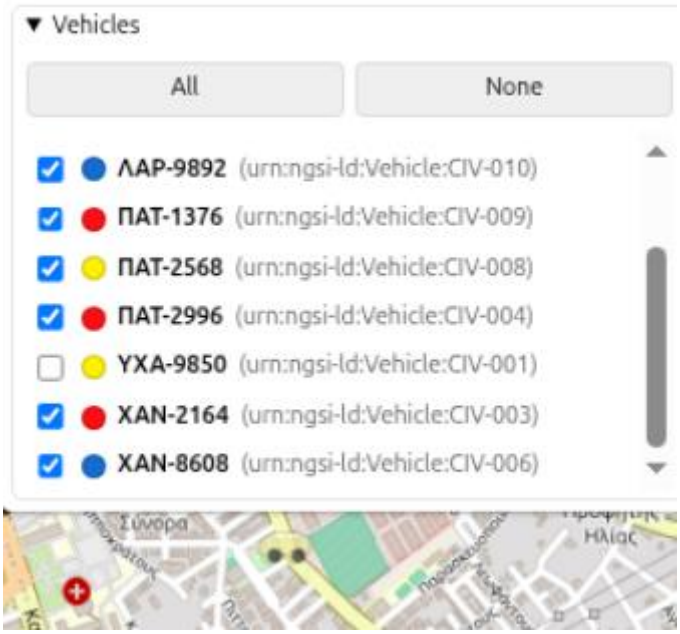


Figure 88 Deselecting a vehicle

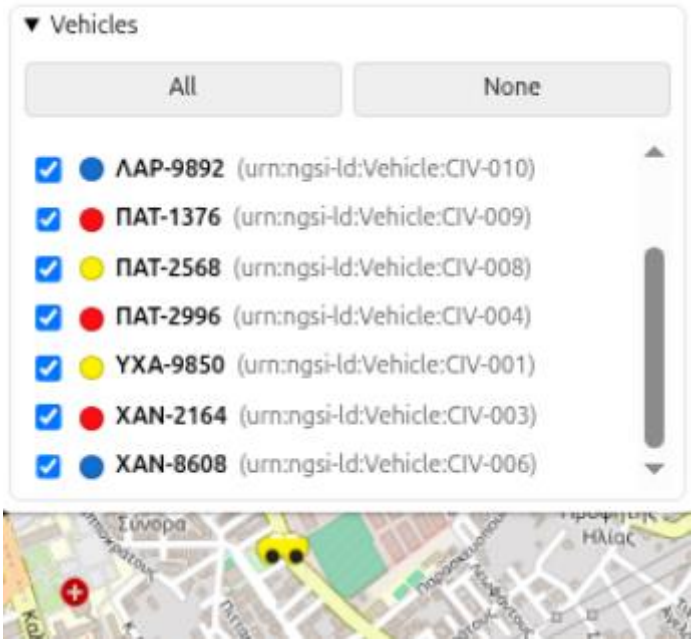


Figure 89 Selecting a vehicle

After selecting (or deselecting) the desired vehicles, the operator can click on the “Plan pickups” button to create an evacuation plan for the people inside the to-be-evacuated areas.

After that, the Google CFR API assigns targets to each vehicle, according to their position and available seats. Then the ORS or OSRM APIs provide a route for each vehicle to follow according to the target assignment. That route is then displayed on the map, as well as each pickup location.

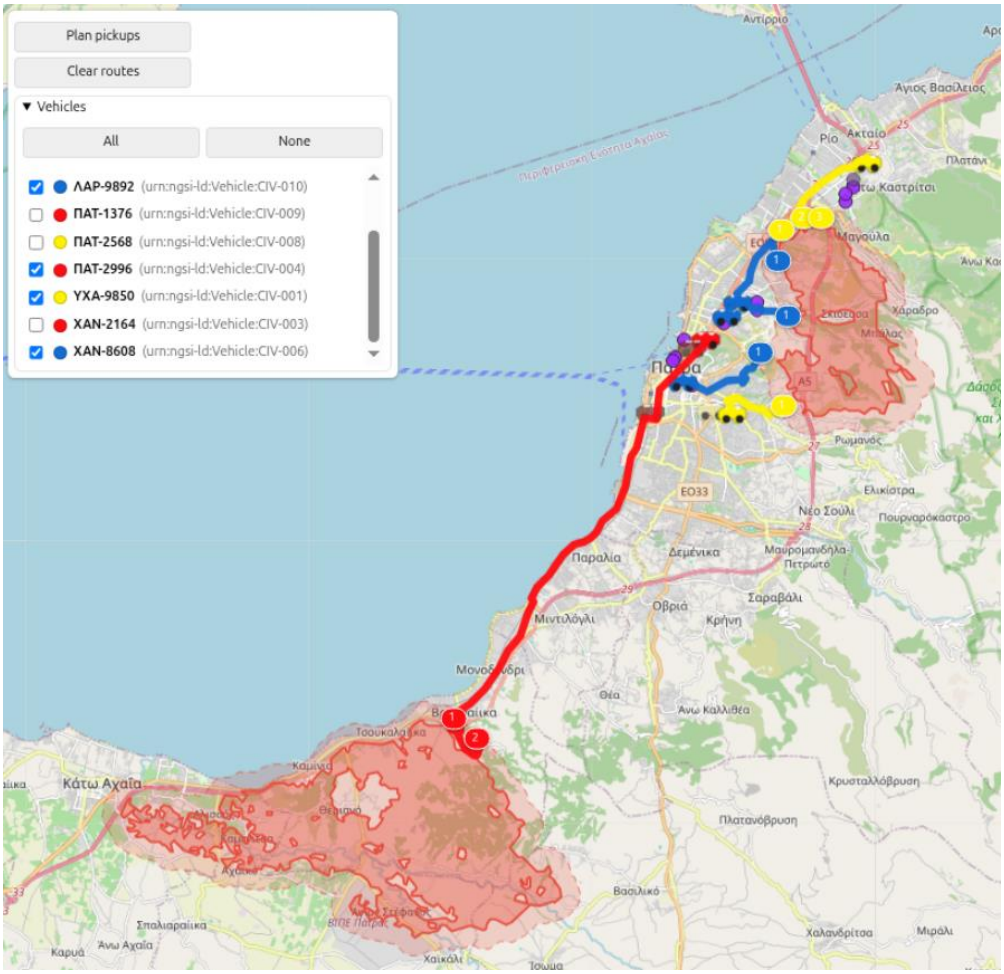


Figure 90 Evacuation plan

Clicking on the pickup location shows the vehicle's plate, and the name of the person being picked up.

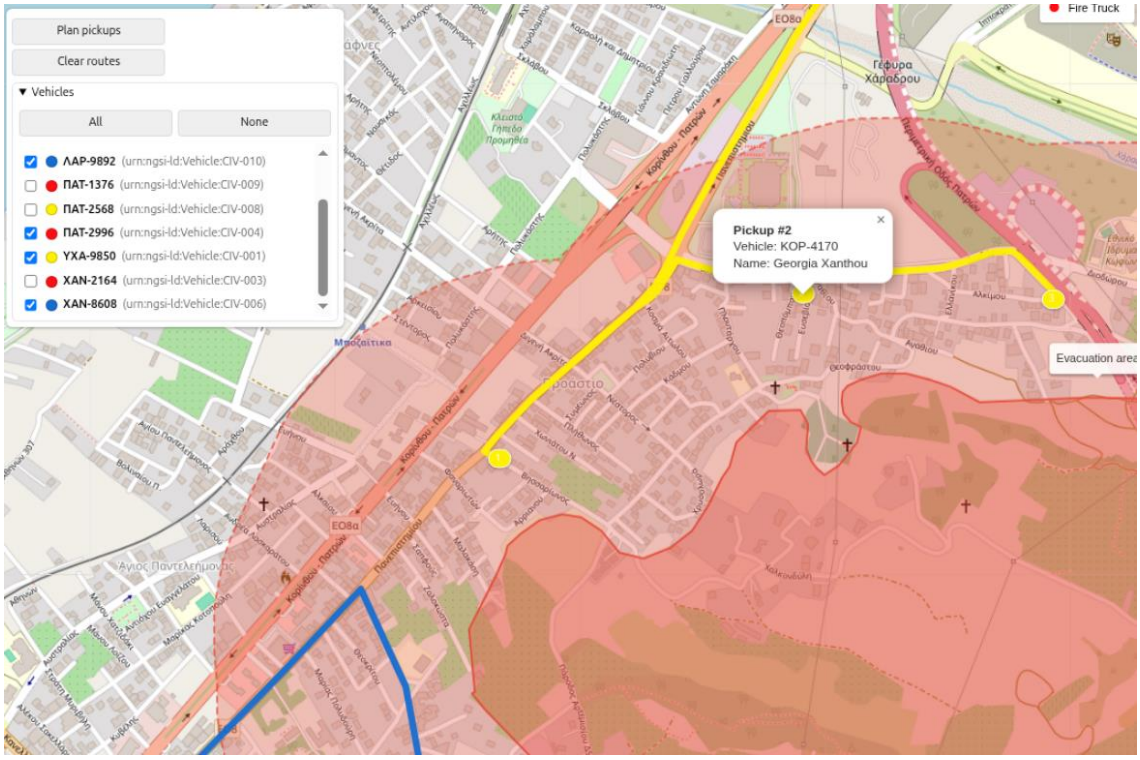


Figure 91 Pickup locations

7.3.4 Providing information to vehicle operators

The central operator now has a comprehensive evacuation plan, but that plan needs to be communicated to the vehicle operators: the people that will carry it out. For this reason, each vehicle in the system is required to provide a point of contact through which their role in the evacuation can be communicated. In a real-world scenario, the most reliable and mobile option would almost certainly be a phone number, so that instructions could be delivered instantly by SMS. Messages of this kind are difficult to miss and can be received even when data coverage is limited. However, given the limited scope of this thesis and the fact that no real emergency infrastructure was connected, email was selected as the channel of communication. Each vehicle therefore registers an email address during setup, and that address becomes the point where its instructions are delivered. This approach is simpler to implement while still showing how targeted communication can be achieved.

The email sent to operators contains a Google Maps link with the exact directions the vehicle should follow to reach all its assigned targets. By opening the link on a smartphone or in-car device, the operator can immediately see the route laid out step by step. Other than the directions link, the message also includes a clear list of notes for each stop. These notes describe each person who needs to be picked up, together with details that will be relevant on the field, for example, whether the person uses a wheelchair, whether they require medical equipment, or if they should be approached

with special care due to anxiety or other conditions. Having such information in advance helps the operators to be prepared and reduces the risk of delays or confusion at the pickup point.

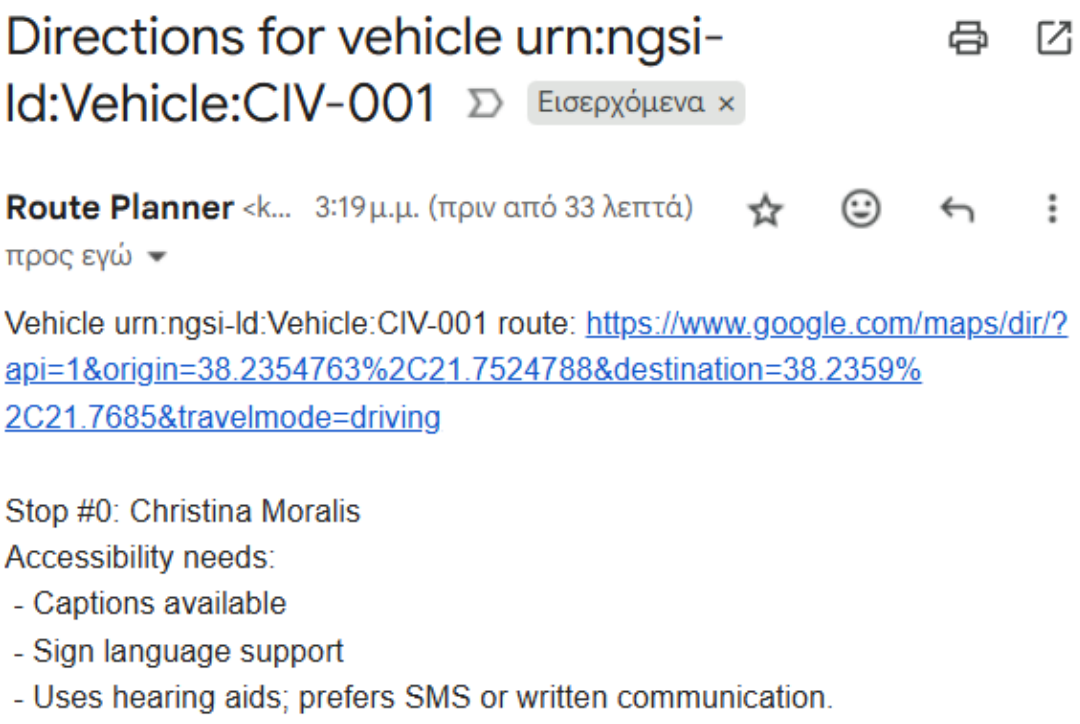


Figure 92 Directions and information sent to each vehicle

8 Conclusion

8.1 Summary

In this summary section we will restate the goals of the thesis and how they were achieved.

The project set out to leverage a Data Space infrastructure to facilitate and organize the secure exchange of data and use that framework to create an application that provides efficient and coordinated evacuation planning for the people in need, in cases of natural disasters. To accomplish this, a functional Data Space (According to the IDS standards) was deployed, and a prototype application was built to integrate information from diverse sources -including emergency services, disability registries and transportation fleets. The successful creation of this ecosystem allowed for secure data sharing and real-time data discovery, showing how this example can lead the way for more widespread adoption of the Data Space archetype.

8.2 Contributions

This section outlines the key contributions of this thesis.

The first contribution lays in the extensive research in the field of Data Space. By providing a thorough examination of current Data Space technologies and standards, we can see the mechanisms that facilitate secure, trustworthy and reliable data discovery and exchange. By examining those mechanisms, we can see how this emerging technology has already seen use in real-world applications and proven itself to be a much-needed part of modern data-driven societies. Additionally, we covered several of those real-world examples of how employing Data Spaces can improve the lives of the people, as well as the industry that chooses them.

The second contribution is the documentation around the deployment of a working Data Space, according to the standards set forth by the IDS. The IDS Testbed is an ideal framework for new developers coming into the field of Data Spaces that want to experiment or develop this technology. However, some parts of that process can be hard to get down, while official instructions do not cover some essential processes. Having a comprehensive guide on how to approach this new technology (on a theoretical and practical level) is important for future developers attempting to join the Data Space ecosystem.

The third contribution revolves around the creation of a prototype application that consumes data from the Data Space and uses it to coordinate evacuation of people with disabilities in areas close to natural disasters. This prototype not only proves the usability and security of Data Space, but also sets an example for real-world

applications that take advantage of the huge volumes of data in today's societies, in order to help or improve the lives of people.

8.3 Evaluation

In this qualitative evaluation, we reflect on the performance of the deployed framework; whether it met the requirements we set forth in the early stages of development or not.

The first, and most essential, requirement has been the secure and confidential exchange of data inside our environment. Data sovereignty and integrity must be ensured, while traceability is needed to ensure accountability. We have seen that a Data Space that upholds the standards of the IDS does offer all these assurances. The certification process is strict and doesn't allow unauthorized access to the ecosystem. The sovereignty of the data was never compromised, since the data never left the control of the providers, while their usage is restricted according to the offer they provide. The confidentiality, integrity and overall transport security was upheld by employing modern encryption algorithms and token-based access to the APIs.

The second requirement has been the enabling of data discovery even in a dense environment of sources. We saw how the metadata Broker offers such capabilities by allowing the providers to register their offers, and by responding to the consumers' queries by providing the relevant resources according to the queries' metadata requirements.

Another functional requirement is interoperability. Data discovery is pointless to the consumers if they can't understand the structure and use the data. For that reason, documentation for each source and for the relationship of entities in each data source is necessary. That has been a very important aspect of the implementation.

The prototype app itself also had some requirements. Most notably the visualization of the incoming data, the planning and dispatching of forces in an optimal way and providing necessary information to the field agents. All those requirements have been met as we have seen in the showcase of the demo app in subchapter 7.3.

Lastly, the requirement for scalability has also been met. The platform chosen (IDS Testbed) not only complies with the international standards but also has a modular architecture that allows for the extension of its capabilities according to the needs of the ecosystem.

Even though it was not a "strict" requirement, the general motivation for this thesis has been to demonstrate the value of Data Spaces in modern societies. The combination of meeting all the previously mentioned goals proves that it is a technology that needs to be considered for any data sharing initiative.

8.4 Limitations

The scope of any thesis inevitably places boundaries on what can be achieved, especially when the subject touches on large-scale initiatives like a platform for sensitive, multi-sector data and an application that coordinates evacuation operations under demanding conditions. In this section we reflect on some of the main limitations that we faced.

The most obvious limitation is the lack of real data in most cases. For understandable reasons, a student project cannot access personal information about people with disabilities or operational details from government fleets. This restriction ensures privacy and security, but it also reduces the impact of the results. Working with simulated datasets is useful for showing the technical feasibility of the approach, yet it cannot capture all the complexity, messiness, and unpredictability of real-world information. In a genuine deployment, one of the first steps would be negotiating access to actual registries, live vehicle fleets, and official disaster reports.

Another clear limitation is the scale of the ecosystem. Real Data Spaces often bring together hundreds of organizations and a large number of data offerings from different sectors. Applications then emerge on top of this ecosystem, benefiting from the richness and diversity of available information. By contrast, this thesis works with a very small number of participants and sources. Even though those participants serve as examples of each type of source, the ecosystem would only benefit from the inclusion of more providers.

Finally, it is important to underline that the goal of the thesis was to build a functioning prototype, not a production-ready platform. The application that was developed demonstrates the basic flow of discovering data, planning an evacuation, and sending instructions, but it does not attempt to cover the entire range of tasks involved with disaster management. Resource allocation, communication between different agencies, integration with existing emergency systems: all these aspects fall outside of the scope of this project. For this reason, the system is described as a demo: something that works and can be tested.

8.5 Future work

Scaling up is an important part of future work, especially for a platform that is designed to combine diverse sources of data. At the current stage, the system shows what is possible with a limited number of providers, but a natural next step would be to include a wider range of organizations and agencies. Weather stations could contribute real-time environmental measurements, traffic management systems could share live road conditions, hospitals and health facilities could provide information on available resources, and even agencies from neighboring countries could participate when disasters become too extreme for a single country to manage on its own.

The growth of the ecosystem would also create opportunities for new types of applications. Beyond the evacuation planner demonstrated here, other apps could be built around specific themes. For example, accessibility-focused apps could help people with disabilities in their everyday life or in difficult situations by requesting assistance or reporting issues through a service that can verify them. Another avenue is the use of AI: with enough data, models could be trained to predict how a wildfire might develop in certain conditions and areas, or to recommend evacuation strategies that minimize risk, considering a multitude of factors. These predictive insights could then feed back into operational tools, giving planners better information the moment they need it.

Future development of the application itself should not be overlooked. Additional functionality such as dedicated pages for registering, editing incoming data, more flexible options for planning evacuations, and tools to visualize different “what-if” scenarios would make it more practical in real deployments. A dedicated mobile app for vehicle operators would also close the loop, ensuring instructions can be received, acknowledged and updated in real time from the field. On the user side, structured UI/UX testing with larger audiences could provide feedback on how the interface feels under pressure, and what design changes would make it easier for non-technical users to operate during an actual emergency.

In short, this implementation serves as a starting point. It shows that Data Spaces can provide a secure and sovereign framework for sharing sensitive but critical information, and it shows one concrete way of applying these ideas to disaster response. Future work can take this proof-of-concept and grow it into pilot projects that involve real agencies, more complex data flows, and richer applications, ultimately moving closer to a system that could be deployed in real emergencies.

9 References

[1] International Data Spaces Association (IDSA). (2025, September) IDSA Website. [Online]. https://internationaldataspaces.org/

[2] Fraunhofer ISST, Sebastian Steinbuß, International Data Spaces Association, Andreas Teuscher, SICK, Dr.-Ing. Steffen Lohmann, Fraunhofer IAIS Prof. Dr.-Ing. Boris Otto. (2019) International Data Spaces Reference Architecture Model 3.0. [Online]. https://internationaldataspaces.org/wp-content/uploads/IDS-Reference-Architecture-Model-3.0-2019.pdf

[3] Mobility Data Space (MDS). (2025, September) Mobility Data Space. [Online]. https://mobility-dataspace.eu/

[4] Gaia-X. (2025, January) Gaia-X EU. [Online]. https://gaia-x.eu/wp-content/uploads/2025/01/Lighthouse-Projects-Overview_January-2025.pdf

[5] Mobility Data Space (MDS). (2025, September) Mobility Data Space EU data catalogue. [Online]. https://mobility-dataspace.eu/data-catalogue

[6] Manufacturing Data Space. (2024, May) Mobility Data Space Report. [Online]. https://manufacturingdataspace-csa.eu/wp-content/uploads/2024/05/MDS-Mobility-Data-Space.pdf

[7] Mobility Data Space (MDS). (2025, September) Mobility Data Space EU use cases. [Online]. https://mobility-dataspace.eu/use-cases

[8] Yunex Traffic. (2025, March) Priobike Project. [Online]. https://www.yunextraffic.com/newsroom/priobike-projekt-hamburg-enhances-cyclist-safety/

[9] Yunex Traffic. (2022, December) PrioBike bicycle totem. [Online]. https://www.yunextraffic.com/newsroom/bicycle-totem-hamburg/

[10] Reset org. (2022, February) Priobike application. [Online]. https://reset.org/priobike-mit-dieser-app-haben-radfahrende-eine-gruene-welle/

[11] Esri. (2025, September) Esri digital twin overview. [Online]. https://www.esri.com/en-us/digital-twin/overview

[12] Mobility Data Space (MDS). (2025, September) Esri use case. [Online]. https://mobility-dataspace.eu/use-cases/esri

[13] Mobility Data Space (MDS). (2025, September) DeepVolt use case. [Online]. https://mobility-dataspace.eu/use-cases/deepvolt

[14] Smart Connexted Supplier Network (SCSN). (2025, September) SCSN overview. [Online]. https://smart-connected.nl/en

[15] Smart Connected Supplier Network (SCSN). (2025, September) SCSN how it works. [Online]. https://smart-connected.nl/en/about-scsn/how-it-works

[16] Health-X dataLOFT. (2025, September) Health-x home. [Online]. https://www.health-x.org/en/home

[17] Health-X dataLOFT. (2025, September) Health-x platform. [Online]. https://www.health-x.org/en/platform

[18] Docker. (2025, September) Docker home page. [Online]. https://www.docker.com/

[19] Docker. (2025, September) Docker compose docs. [Online]. https://docs.docker.com/compose/

[20] FIWARE. (2025, September) FIWARE home page. [Online]. https://www.fiware.org/

[21] FIWARE. (2025, September) FIWARE Orion Context Broker. [Online]. https://fiware-orion.readthedocs.io/en/master/

[22] Koufopavlou Odysseas (Professor), Denazis Spyros (A. Professor). (2024) Network Architectures and Management Group. [Online]. https://nam.ece.upatras.gr/

[23] Python Software Foundation. (2025, September) Python home page. [Online]. https://www.python.org/

[24] Konstantinos Tsampras. (2025, September) Thesis repository. [Online]. https://github.com/ultrongr/IDS-Civil-Protection

[25] Greek Government. (2025, September) Registry of people with disabilities. [Online]. https://epan.gov.gr/mitroo_anapirias

[26] Microsoft. (2025, June) Data Encryption at Rest. [Online]. https://learn.microsoft.com/en-us/dynamics365/business-central/dev-itpro/security/transparent-data-encryption

[27] npm contributor Victor Parmar. (2023) Npm packages: mongoose-field-encryption. [Online]. https://www.npmjs.com/package/mongoose-field-encryption/v/2.3.0

[28] Pravallika Yakkala. (2024, August) Understanding AES Encryption and AES/GCM Mode: An In-Depth Exploration using Java. [Online]. https://medium.com/@pravallikayakkala123/understanding-aes-encryption-and-aes-gcm-mode-an-in-depth-exploration-using-java-e03be85a3faa

[29] SMARTBEAR. (2025, September) Swagger home page. [Online]. https://swagger.io/

[30] Οργανισμός Προνοιακων Επιδομάτων και Κοινωνικής Αλληλεγγύης (ΟΠΕΚΑ). (2025, September) Financial aid for people with disabilities. [Online]. https://opeka.gr/atoma-me-anapiria/anapirika-epidomata/programmata-oikonomikis-enischysis-atomon-me-varia-anapiria/

[31] SQLite Consortium. (2025, September) SQLite home page. [Online]. https://sqlite.org/

[32] FIWARE. (2018) FIWARE data models: Transportation Vehicle example. [Online]. https://github.com/FIWARE/data-models/blob/master/specs/Transportation/Vehicle/Vehicle/example.json

[33] Greek Civil Protection. (2025, September) Details about the 112 warning system. [Online]. https://civilprotection.gov.gr/112/pote-pos-me-eidopoiei

[34] Civil Protection Twitter/X account. (2025, September) 112 Warning service twitter account. [Online]. https://x.com/112Greece

[35] Greek Civil Protection Facebook page. (2025) Facebook page of 112 service. [Online]. https://www.facebook.com/112Greece/

[36] Copernicus. (2025, September) Copernicus homepage. [Online]. https://www.copernicus.eu/en

[37] European Space Agency. (2014, April) Copernicus sentinel 1. [Online]. https://sentinels.copernicus.eu/copernicus/sentinel-1

[38] European Space Agency. (2015, June) Copernicus sentinel 2 program. [Online]. https://sentiwiki.copernicus.eu/web/sentinel-2

[39] Copernicus. (2025, September) Copernicus Emergency Management Service. [Online]. https://emergency.copernicus.eu/

[40] Copernicus. (2025, September) European Forest Fire Information System (EFFIS). [Online]. https://forest-fire.emergency.copernicus.eu/

[41] Turf org. (2025, September) Turf.js homepage. [Online]. https://turfjs.org/

[42] International Data Spaces Association (IDSA). (2025) IDSA Reference Testbed information. [Online]. https://internationaldataspaces.org/offers/reference-testbed/

[43] International Data Spaces Association (IDSA). (2025, September) IDS Testbed repository. [Online]. https://github.com/International-Data-Spaces-Association/IDS-testbed

[44] International Data Spaces Association (IDSA). (2023) IDS Testbed's role as Minimum Viable Dataspace. [Online]. https://github.com/International-Data-Spaces-Association/IDS-testbed/blob/master/minimum-viable-data-space/MVDS.md

[45] Git. (2025, September) Git homepage. [Online]. https://git-scm.com/

[46] CloudFlare. (2014, July) Introduction of CFSSL - CloudFlare's PKI toolkit. [Online]. https://blog.cloudflare.com/introducing-cfssl/

[47] OpenSSL organization. (2025, September) OpenSSL homepage. [Online]. https://www.openssl.org/

[48] Postman. (2025, September) Postman homepage. [Online]. https://www.postman.com/

[49] Microsoft. (2025, September) Visual Studio Code homepage. [Online]. https://code.visualstudio.com/

[50] International Data Spaces Association (IDSA). (2025, July) Postman collection for the IDS Testbed. [Online]. https://github.com/International-Data-Spaces-Association/IDS-testbed/blob/master/Testsuite/Testsuite.postman_collection.json

[51] Wikipedia. X.509 Certificates. [Online]. https://en.wikipedia.org/wiki/X.509

[52] PKIglobe. AKI and SKI. [Online]. https://www.pkiglobe.org/aki_ski.html

[53] International Data Spaces Association (IDSA). (2023) Testbed User Guide. [Online]. https://docs.internationaldataspaces.org/ids-knowledgebase/ids-reference-testbed/getting-started-with-ids-reference-testbed/testbeduserguide

[54] jfernandezsqs Github Account. (2022) Github Issues on missing pki.py file. [Online]. https://github.com/International-Data-Spaces-Association/IDS-testbed/issues/105

[55] Computer Hope. (2025, July) Chmod command usage. [Online]. https://www.computerhope.com/unix/uchmod.htm

[56] Mozzila. (2025, September) MDN HTML docs. [Online]. https://developer.mozilla.org/en-US/docs/Web/HTML

[57] Mozilla. (2025, September) MDN docs CSS. [Online]. https://developer.mozilla.org/en-US/docs/Web/CSS

[58] Vlodymyr Agafonkin. Leafletjs home page. [Online]. https://leafletjs.com/

[59] OpenStreetMap. (2025, September) OpenStreetMap homepage. [Online]. https://www.openstreetmap.org/#map=7/38.359/23.810

[60] Abiola Farounbi. (2024, February) Comparing Javascript mapping APIs. [Online]. https://blog.logrocket.com/javascript-mapping-apis-compared/

[61] Google. (2022, April) Google Cloud Fleet Routing API. [Online]. https://cloud.google.com/blog/products/ai-machine-learning/google-cloud-optimization-ai-cloud-fleet-routing-api

[62] Open Route Service. (2025, September) Open Route Service homepage. [Online]. https://openrouteservice.org/

[63] Open Source Routing Machine. (2025, September) Open Source Routing Machine homepage. [Online]. https://project-osrm.org/

[64] Andris Reinman. (2025, September) Nodemailer homepage. [Online]. https://nodemailer.com/

[65] OpenJS Foundation. (2025, September) NodeJS home page. [Online]. https://nodejs.org/en