H2020 – Secure societies - Protecting freedom and security of Europe and its citizens
SU-DRS02-2018-2019-2020– Technologies for first responders – Research and Innovation Action (RIA)

Emerging technologies for the Early location of Entrapped victims under Collapsed Structures & Advanced Wearables for risk assessment and First Responders Safety in SAR operations

D4.8 Data aggregation, V2

Workpackage:	WP4 – Data aggregation, Analysis and Decision Support
Authors:	NTUA
Status:	Final
Due Date:	30/06/2021
Version:	1.00
Submission Date:	30/06/2021
Dissemination Level:	PU

(*) Dissemination level.-PU: Public, fully open, e.g. web; CO: Confidential, restricted under conditions set out in Model Grant Agreement; CI: Classified, Int = Internal Working Document, information as referred to in Commission Decision 2001/844/EC.

Search and Rescue Project Profile

Grant Agreement No.:	882897
Acronym:	Search and Rescue
Title:	Emerging technologies for the Early location of Entrapped victims under Collapsed Structures & Advanced Wearables for risk assessment and First Responders Safety in SAR operations
URL:	www.search-and-rescue.eu
Start Date:	01/07/2020
Duration:	36 months

Partners

	NATIONAL TECHNICAL UNIVERSITY OF ATHENS (NTUA) Co-ordinator	Greece
	AIDEAS OÜ (AIDEAS)	Estonia
	SOFTWARE IMAGINATION & VISION S.R.L (SIMAVI)	Romania
	MAGGIOLI SPA (MAG)	Italy
	KONNEKT-ABLE TECHNOLOGIES LIMITED (KT)	Ireland
	THALES ITAIA Italia SPA (THALIT)	Italy
	ATOS IT SOLUTIONS AND SERVICES IBERIA SL (ATOS)	Spain
	ETHNIKO KENTRO EREVNAS KAI TECHNOLOGIKIS ANAPTYXIS (CERTH)	Greece
	UNIVERSITA DEGLI STUDI DI CAGLAIRI (UNICA)	Italy
	UKEMED GLOBAL LTD (UGL)	Cyprus
	PUBLIC SAFETY COMMUNICATION EUROPE FORUM AISBL (PSCE)	Belgium

	UNIVERSITA DEGLI STUDI DI FIRENZE (UNIFI)	Italy
	DEUTSCHES FORSCHUNGSZENTRUM FUR KUNSTLICHE INTELLIGENZ (DFKI)	Germany
	UNIVERSITA CATTOLICA DEL SACRO CUORE (UCSC)	Italy
	VRIJE UNIVERSITEIT BRUSSEL	Belgium
	SYNYO GmbH (SYNYO)	Austria
	UNIVERSITEIT HASSELT (UHASSELT)	Belgium
	SPOLECZNA AKADEMIA NAUK (SAN)	Poland
	GIOUMPITEK MELETI SCHEDIASMOS YLOPOIISI KAI POLISI ERGON PLIROFORIKIS ETAIREIA PERIORISMENIS EFTHYNIS (UBITECH)	Greece
Search and Rescue End-Users		
	ELLINIKI OMADA DIASOSIS SOMATEIO (HRT)	Greece
	ENOSI PTYCHIOYCHON AXIOMATIKON YPAXIOOMATIKON PYROSVESTIR OY SOMATEIO (EPAYPS)	Greece
	JOHANNITER-UNFALL-HILFE EV (JOHAN)	Germany
	JOHANNITER OSTERREICH AUSBLIDUNG UND FORSCHUNG GEMEINNUTZIGE GMBH (JOAFG)	Austria
	CONSIGLIO NAZIONALE DELLE RICERCHE	Italy

	POMPIERS DE L'URGENCE INTERNATIONALE (PUI)	France
	ASOCIATA CLUSTERUL ROMAN RENTRU PROTECTIE SI ECOLOGIE IN DOMENIUL MATERIALELOR CHIMICE, BIOLOGICE, RADIOLOGICE/NUCLEARE SI EXPLOZIVE (PROECO)	Romania
	SERVICIO MADRILENO DE SALUD (SERMAS)	Spain
	FUNDACIÓN PARA LA INVESTIGACIÓN E INNOVACIÓN BIOSANITARIA DE ATENCIÓN PRIMARIA (FIIBAP)	Spain
	ESCUELA ESPANOLA DE SALVAMENTO Y DETECCION CON PERROS (ESDP)	Spain

Document History

Version	Date	Author (Partner)	Remarks/Changes
0.10	17/05/2021	Christodoulos Santorinaios (NTUA), Iosif Sklavidis (KT)	ToC
0.20	19/05/2021	Christodoulos Santorinaios (NTUA)	Content for section 2.1
0.21	21/05/2021	Christodoulos Santorinaios (NTUA)	Content for section 2.2
0.21	24/05/2021	Iosif Sklavidis, Angelica Theodorou (KT)	Content for sections 3.1 – 3.2
0.22	25/05/2021	Christodoulos Santorinaios (NTUA)	Content for section 2.3
0.23	28/05/2021	Christodoulos Santorinaios (NTUA)	Content for sections 2.4 – 2.7
0.23	31/05/2021	Iosif Sklavidis, Angelica Theodorou (KT)	Content for section 3.3 – 3.5
0.30	01/06/2021	Christodoulos Santorinaios (NTUA)	Content for Section 1 and 4 and Executive Summary
0.30	10/06/2021	Jose Gato Luis (ATOS)	Internal Review
0.31	11/06/2021	Christodoulos Santorinaios (NTUA), Iosif Sklavidis (KT)	Corrections based on comments
0.31	14/06/2021	Giulia Sedda (UNICA)	Internal Review
0.32	14/06/2021	Christodoulos Santorinaios (NTUA), Iosif Sklavidis (KT)	Corrections based on comments
0.30	29/06/2021	Christodoulos Santorinaios (NTUA)	Quality Control
1.00	30/06/2021	Christos Ntanos (NTUA)	Final version to be submitted

Executive Summary

This deliverable provides information about the implementation of an API to serve aggregated data from resources relevant to past large-scale disasters and the aggregation of data coming from internal sources and devices, such as smart glasses, smart helmets, drones, and sensors.
As far as the past disasters are concerned, 7 relevant sources have been chosen that can be proven to be useful for crisis preparedness and search and rescue operations, after being evaluated according to specific criteria. Data aggregated and transformed from these sources, are then served through an API to be used as part of a collective access point for the execution of experimental knowledge discovery algorithms.
The second section of D4.8 aims to present the data aggregation mechanism. This section settles down the available components that enable the aggregation mechanisms that process the real-time data and combine them with rest data, such as historical data, data from COncORDE.

Table of Contents

List of Figures

List of Tables

1 Introduction

In the following sections, an overview of the purpose and scope of this document is presented. The structure and the relationship to other documents are described, as well as the list of abbreviations used in the current deliverable.

1.1 Purpose and Scope

The purpose of the present deliverable entitled "Data Aggregation, V2" is to implement the data aggregation mechanisms that will be utilised during the Search & Rescue operations. More specifically the present deliverable aims to fulfil the following main objectives:

- Provide an API that serves information aggregated from existing resources that contain data about past incidents and large-scale disasters for natural, technological, and natech hazards.
- Design and develop a data aggregation mechanism which will gather all heterogeneous data from the devices connected to the S&R platform. The objective is to homogenize all the incoming data, in order to be readable and used from the rest of the components and services of the S&R platform.

1.2 Structure of the Document

The structure of this document is the following:

- **Section 1** presents the purpose and scope of this document and explains the structure and the relationship of this deliverable with other documents of this project.
- **Section 2** presents the aggregation of data coming from external resources relevant to past large-scale disasters. It provides information about resources used, transformations made and the technologies used to blend everything together.
- **Section 3** provides information about the storage and processing of data coming from internal sources and devices, such as sensors and drones.
- **Section 4** concludes this deliverable.

1.3 Relationship with Other Documents

This deliverable is the second in a series of two reports (D4.1, D4.8) that documents the retrieval of data from external resources relevant to past large-scale disasters and the aggregation of data coming from internal heterogeneous sources, e.g., smart glasses, drones, and sensors. The results of these two documents will be utilized by tasks 4.4 – "Design of DSS components" and 4.5 – "Development of DSS components". The external and internal data that will be gathered will be used by the DSS component for the support of crisis preparedness and search & rescue activities. In particular, by utilising both past and real-time data, authorities will be able to leverage the DSS component to make the best possible decisions regarding search & rescue operations. The results of these two documents are also connected with tasks T2.5 – "Involvement of volunteer organisations / citizens", T4.3 – "Situational Analysis & Impact Assessment", T6.2 – "S&R Data Communication Interoperability framework", T6.3 – "S&R services interoperability framework", T6.4 – "S&R Design of interoperability framework", T6.5 – "Design of an aftermath knowledge capitalisation mechanism", T7.2 – "Architecture and Design Specifications of S&R", T7.3 – "Specification of interfaces based on the S&R interoperability framework", T7.4 – "Adaptation of systems and services", and T7.5 – "S&R platform component and service integration". In general, all these tasks will utilize either the data stored in external resources that are relevant to past large-scale disasters or the data fusion and

mediation system that is responsible for the aggregation of data from internal sources like drones and sensors.
As such, the present document is related to the following deliverables:

- D1.3 Definition, evaluation and refinement of the S&R CM governance model
- D1.4 Establishment of S&R Concept of operations
- D1.7 Definition, evaluation and refinement of the S&R CM governance model, V2
- D2.1 PIA report for the S&R design and development, pilots and platform
- D3.7 Requirements to knowledge management and SA Model, V2
- D3.8 Situation Awareness Model - specification, V2
- D4.2 Situational Analysis & Impact Assessment
- D4.3 Design of SOT DSS components
- D4.4 Design of PHYSIO DSS component
- D4.5 Development of SOT DSS components
- D4.6 Development of PHYSIO DSS component
- D4.7 DSS Validation
- D4.9 Design of SOT DSS components, V2
- D4.10 Design of PHYSIO DSS component, V2
- D4.11 Development of SOT DSS components, V2
- D4.12 Development of PHYSIO DSS component, V2
- D5.1 Design & development of the RESCUE MIMS
- D6.2 Voice, data and services interoperability frameworks
- D6.3 Presentation and analysis of the designed S&R interoperability framework
- D6.4 S&R lessons learnt mechanism
- D6.5 Establishment of technical components and legacy systems taxonomy
- D7.1 S&R extensive service catalogue
- D7.3 Component interface specifications for interoperability within S&R
- D7.4 Adapted S&R components and services
- D7.5 Integrated S&R platform 1st version
- D7.8 S&R Legal and Security infrastructure final
- D7.9 S&R platform Test Cases and overall system evaluation results 1st version
- D7.12 - Architecture and Design Specifications of S&R platform, V2List of Abbreviations

1.4 List of Abbreviations

The following table includes all the abbreviations used in the document.

Abbreviation	Explanation
S&R	Search & Rescue
CDD	Canadian Disaster Database
EM-DAT	Emergency Events Database
GLC	Global Landslide Catalog
HFS	Hellenic Fire Service
USGS	United States Geological Survey
GIDD	Global Internal Displacement Database
API	Application Programming Interface
HDFS	Hadoop Distributed File System
CDM	Canonical Data Model

ETL	Extract, Transform, Load
EIP	Enterprise Integration Patterns
PIA	Privacy Impact Assessment
RDD	Resilient Distributed Datasets
YARN	Yet Another Resource Negotiator
VM	Virtual Machine
DAG	Directed Acyclic Graph
BDE	Big Data Europe

Table 1-1: List of Abbreviations

2 Aggregation of data coming from external resources relevant to past large-scale disasters

In this chapter the aggregation of data coming from external resources relevant to past large-scale disasters is presented. The purpose of this chapter is to create an Application Programming Interface that will serve historical data to the appropriate partners.
Firstly, the used resources are shown and described. Then the appropriate transformations to get the data to the required format are presented. The API that serves the data is described along with the requirements that need to be taken into account to be able to run the service.

2.1 Used Resources

In this section the seven resources out of the twenty-three that have been chosen to aggregate data from, according to D4.1's requirements, are presented. These resources are the Canadian Disaster Database, the Emergency Events Database, the Global Landslide Catalog, the Hellenic Fire Service, the United States Geological Survey Earthquake Program and the Global Internal Displacement Database.

At the table below the various information included in the databases used is presented.

Resource	Details of the disaster	Costs	Payments	Equipment	Casualties
CDD	yes	yes	yes	yes	yes
EMDAT	yes	yes	yes	no	yes
GLC	yes	no	no	no	no
HFS(FOREST)	yes	no	no	yes	no
HFS(RESIDENTIAL)	yes	no	no	no	yes
USGS	yes	no	no	no	no
GIDD	yes	no	no	no	no

Table 2-1: Information available in databases

The following tables concern the detailed description, the extracted data format, the constraints imposed by each resource and the description of its fields.

Resource/Database	The Canadian Disaster Database [1]
Description	The Canadian Disaster Database (CDD) contains detailed disaster information on more than 1,000 natural, technological and conflict events (excluding war) that have happened since 1900

	at home or abroad and that have directly affected Canadians. The CDD tracks significant disaster events that meet one or more of the following criteria: - 10 or more people killed - 100 or more people affected/injured/infected/evacuated or homeless - an appeal for national/international assistance - historical significance - significant damage/interruption of normal processes such that the community affected cannot recover on its own The CDD describes: - where and when a disaster occurred - the number of injuries, evacuations, and fatalities - an estimate of the costs
Extracted data format	Data downloaded as tab-delimited text files
Constraints	Canada endeavours to provide the best information possible; however, the information contained in the Canadian Disaster Database (CDD) is based on information that is sourced from outside parties and may not be accurate. Canada makes no representations, warranties, or guarantees, express or implied, that the data contained in the CDD may be relied upon for any use whatsoever. Canada accepts no responsibility or liability for inaccuracies, errors or omissions in the data and any loss, damage or costs incurred as a result of using or relying on the data in any way. The CDD may contain material that is subject to licensing requirements or copyright restrictions and may not be reproduced, published, distributed or transferred in whole or in part without the consent of the author.

Table 2-2: Canadian Disaster Database Information

Field	Description
emergencyTypeEnum	Type of disaster
description	Description of disaster
database	Database name
openYear	Year disaster started
openMonth	Month disaster started
openDay	Day disaster started
openDate	Date disaster started
closeYear	Year disaster ended
closeMonth	Month disaster ended
closeDay	Day disaster ended
closeDate	Date disaster ended
country	Country the disaster happened at
locationType	Type of location of disaster
vehicleType	Type of vehicle the disaster happened at
magnitude	Magnitude of disaster in case of earthquake

casualties	Number of casualties
injuredPeopleNumber	Number of injured people
infectedPeopleNumber	Number of infected people
evacuatedPeopleNumber	Number of evacuated people
affectedPeopleNumber	Number of affected people
damages	Cost of damages
payments	Payment amount
insurancePayments	Insurance payment amount
currency	Currency format

Table 2-3: Canadian Disaster Database Fields

Resource/Database	EMDAT [2] [3]
Description	EM-DAT is a global database on natural and technological disasters, containing essential core data on the occurrence and effects of over 22,000 mass disasters in the world. EM-DAT includes all disasters from 1900 until the present, conforming to at least one of the following criteria: - 10 or more people dead; - 100 or more people affected; - The declaration of a state of emergency - A call for international assistance The database is compiled from various sources, including UN agencies, non-governmental organisations, insurance companies, research institutes and press agencies. The main objective of the database is to serve the purposes of humanitarian action at national and international levels. The initiative aims to rationalise decision making for disaster preparedness, as well as provide an objective base for vulnerability assessment and priority setting.
Extracted data format	Data downloaded as xls files
Constraints	If you are an **academic organization, a university, a non-profit research institution and/or an international public organization** (UN agencies, multi-lateral banks, other multi-lateral institution and national governments) with the intention to use the EM-DAT database (hereafter 'EM-DAT') for research, teaching or information purposes, you shall, conditional upon the acceptance of the present conditions of use, be granted free access to EM-DAT (also **'Authorized Use'**). Users of EM-DAT declare not to (also **Unauthorized use**) - Reproduce, copy, communicate, lend, or otherwise distribute EM-DAT or a substantial part of EM-DAT; - Reverse assemble, de-compile, de-compose, or disassemble EM-DAT;

	- Create substitute or derivative databases of EM-DAT; - Attempt to unlock or bypass any initialization or security systems used by EM-DAT; - Share, use and/or transmit any portion of EM-DAT via the Internet to unauthorized users; - Divulge the password to unauthorized users; - Remove, alter or obscure any proprietary legend, copyright, trademark or other intellectual property right notice, logo and image in or on EM-DAT; or - Perform acts that conflict with the normal exploitation of EM-DAT or unreasonably prejudices the interest of Licensor; - Make a commercial use of EM-DAT, except in the event such users enter into a separated Database License Agreement, described hereafter. If you are an entity (which may include a corporation, partnership, limited liability company, law firm or other business organization) that is not academic organization, university, non-profit research institution and/or an international public organization (UN agencies, multi-lateral banks, other multi-lateral institution and national governments), the use of EM-DAT shall be considered '**commercial use**'. More precisely, it means the use of EM-DAT for a commercial purpose rather than for a public, not for profit, or educational purpose. A commercial purpose is considered when: the use of EM-DAT for, amongst others, sale, lease and/or license or the use of EM-DAT to perform contract research, to produce or to manufacture products for sale purposes, or to conduct research activities that result in any sale, lease, license or transfer of EM-DAT to any organisation. Except if agreed upon in a separate Database License Agreement, EM-DAT (including its data and derivate products) cannot be used for any commercial purpose. In case of Commercial Use, the user shall send a prior written request to Ms Regina BELOW, EM-DAT data manager – *regina.below@uclouvain.be* (hereinafter "the Database Manager") describing the aim of such Commercial Use and the amount of users needing access to such data. Access shall be granted to EM-DAT upon proof of payment of

	the corresponding annual fee, as agreed upon in a separate Database License Agreement. **GENERAL TERMS AND CONDITIONS** - Access to EM-DAT shall be granted as of the moment the user agrees to it by clicking on the “I agree’ button. - All published papers and technical reports from the EM-DAT Database owner related to EM-DAT may be used free of charge by the user, provided the user uses Proper Citation. Proper Citation of both EM-DAT and data obtained through EM-DAT is the responsibility of the user alone. “*Proper Citation*” of EM-DAT means: “EM-DAT, CRED / UCLouvain, Brussels, Belgium – *www.emdat.be* (D. Guha-Sapir)” - Certain personal data may be collected and processed via our site, in order to provide access to EM-DAT, for example through the registration form to be completed. The collected personal data shall be used exclusively for the purpose indicated above, in accordance with the Belgian Law of 8 December 1992 on the protection of privacy in relation to the processing of personal data (as amended and the General Data Protection Regulation, Regulation (EU) 2016/679 of April 27, 2016 of the European Parliament and the Council Concerning the protection of individuals with regard to the processing of personal data and the free movement of such data and repealing Directive 95/46 / EC (General Data Protection Regulation)). - CRED makes every effort to ensure, but cannot and does not guarantee, and makes no warranties as to, the accuracy, accessibility, integrity and timeliness of this information. CRED assumes no liability or responsibility for any errors or omissions in the content of this site and further disclaims any liability of any nature for any loss howsoever caused in connection with using EM-DAT. CRED may make changes to EM-DAT at any time without notice. If you are undertaking in-depth analysis, we recommend you to consult our staff to obtain a good understanding of the weaknesses, limitations and peculiarities of our data; - Any use that causes unjustified damage to the legitimate interests of CRED – UCLouvain is strictly forbidden. The same applies to the use with the aim of creating a tool in competition with EM-DAT; - CRED reserves the right to restrict the access from any user who contravenes these conditions of use.

Table 2-4: EMDAT Information

Field	Description
emergencyTypeEnum	Type of disaster
name	Name of disaster
database	Database name
openYear	Year disaster started
openMonth	Month disaster started
openDay	Day disaster started
openTime	Time disaster started
closeYear	Year disaster ended
closeMonth	Month disaster ended
closeDay	Day disaster ended
iso	ISO code of country
country	Country the disaster happened at
locationArea	Area the disaster occurred
longitude	Longitude of location
latitude	Latitude of location
locationType	Type of location
vehicleType	Type of vehicle the disaster happened at
cause	Cause of the disaster
linkedEmergencies	Emergencies linked to the disaster
magnitude	Magnitude of disaster in case of earthquake
magnitudeMeasurementUnit	Measurement unit of magnitude
riverBasin	River basin
casualties	Number of casualties
injuredPeopleNumber	Number of injured people
homelessPeopleNumber	Number of homeless people
affectedPeopleNumber	Number of affected people
payments	Payment amount
damages	Cost of damages
insuredDamages	Cost of insured damages
reconstructionCosts	Cost of reconstruction
currency	Currency format

Table 2-5: EMDAT Fields

Resource/Database	Global Landslide Catalog [4] [5] [6] [7]
Description	The Global Landslide Catalog (GLC) was developed with the goal of identifying rainfall-triggered landslide events around the world, regardless of size, impacts or location. The GLC considers all types of mass movements triggered by rainfall, which have been reported in the media, disaster databases, scientific reports, or other sources. The GLC has been compiled since 2007 at NASA Goddard Space Flight Center. This is a unique data set with the ID tag "GLC" in the landslide editor.
Extracted data format	Data downloaded as CSV files
Constraints	This dataset is intended for public access and use.

Table 2-6: Global Landslide Catalog Information

Field	Description
database	Database name
reportedDate	Day disaster was reported
title	Title of disaster
description	Description of disaster
cause	Cause of disaster
severity	Severity of disaster
openYear	Year disaster started
openMonth	Month disaster started
openDay	Day disaster started
openDate	Date disaster started
openTime	Time disaster started
country	Country the disaster happened at
locationType	Location type of the disaster
casualties	Number of casualties
injuredPeopleNumber	Number of injured people
emergencyTypeEnum	Type of disaster
latitude	Latitude of disaster location
longitude	Longitude of disaster location

Table 2-7: Global Landslide Catalog Fields

Resource/Database	Hellenic Fire Service (Forest) [8]
Description	Hellenic Fire Service (Forest) Resource provides data about past forest fires in Greece, what resources were used to extinguish the fire, and how many acres were burned.
Extracted data format	Data downloaded as .xlsx files
Constraints	Public data

Table 2-8: Hellenic Fire Service (Forest) Information

Field	Description
openYear	Year disaster started
openMonth	Month disaster started
openDay	Day disaster started
openDate	Date disaster started
closeYear	Year disaster ended
closeMonth	Month disaster ended
closeDay	Day disaster ended
closeDate	Date disaster ended
openTime	Time disaster started
closeTime	Time disaster ended
damagedLand	Damaged land in acres
fireFighters	Number of firefighters
volunteers	Number of volunteers
army	Number of army forces
otherForces	Number of other forces
fireTrucks	Number of firetrucks
helicopters	Number of helicopters
airplanes	Number of airplanes
country	Country the disaster happened at
emergencyTypeEnum	Type of disaster
locationType	Location type of the disaster
region	Region the disaster happened at

database	Database name

Table 2-9: Hellenic Fire Service (Forest) Fields

Resource/Database	Hellenic Fire Service (Residential) [8]
Description	Hellenic Fire Service (Residence) Resource provides data about past residential fires in Greece, what resources were used to extinguish the fire, and how many acres were burned.
Extracted data format	Data downloaded as .xlsx files
Constraints	Public data

Table 2-10: Hellenic Fire Service (Residential) Information

Field	Description
openYear	Year disaster started
openMonth	Month disaster started
openDay	Day disaster started
openDate	Date disaster started
closeYear	Year disaster ended
closeMonth	Month disaster ended
closeDay	Day disaster ended
closeDate	Date disaster ended
openTime	Time disaster started
closeTime	Time disaster ended
fireFighters	Number of firefighters
fireTrucks	Number of firetrucks
fireShips	Number of fireships
country	Country the disaster happened at
emergencyTypeEnum	Type of disaster
locationType	Location type of the disaster
region	Region the disaster happened at
involvedPeopleNumber	Number of involved people
injuredPeopleNumber	Number of injured people
casualties	Number of casualties
severity	Severity of disaster
database	Database name

Table 2-11: Hellenic Fire Service (Residential) Fields

Resource/Database	USGS Earthquake Program [9] [10]
Description	The USGS monitors and reports on earthquakes, assesses earthquake impacts and hazards, and conducts targeted research on the causes and effects of earthquakes. USGS undertakes these activities as part of the larger National Earthquake Hazards Reduction Program (NEHRP), a four-agency partnership established by the US Congress.
Extracted data format	Data downloaded as .csv files
Constraints	Public data – There is no need to ask for permission to export the data

Table 2-12: USGS Earthquake Program Information

Field	Description
database	Database name
emergencyTypeEnum	Type of disaster
openYear	Year disaster started
openMonth	Month disaster started
openDay	Day disaster started
openTime	Time disaster started
openDate	Date disaster started
latitude	Latitude of location
longitude	Longitude of location
depth	Depth of the event in meters
magnitude	The magnitude for the event
gap	The largest azimuthal gap between azimuthally adjacent stations (in degrees)
dmin	Horizontal distance from the epicentre to the nearest station (in degrees). 1 degree is approximately 111.2 kilometers
rms	The root-mean-square (RMS) travel time residual, in sec, using all weights
locationArea	Area the disaster occurred
horizontalError	The horizontal location error, in km, defined as the length of the largest projection of the three principal errors on a horizontal plane
depthError	The depth error, in km, defined as the largest projection of the three principal errors on a vertical line
magError	Uncertainty of reported magnitude of the event. The estimated standard error of the magnitude
country	Country the disaster happened at

Table 2-13: USGS Earthquake Program Fields

Resource/Database	Global Internal Displacement Database [11]
Description	This platform details the first results generated by the global disaster displacement risk model. It presents data on displacement risk associated with sudden-onset disasters. The main objective is to start presenting evidence on how to address internal displacement from a prospective point of view by assessing the likelihood of such population movements taking place in the future.
Extracted data format	Data downloaded as .xlsx files
Constraints	Public data - This interactive platform is designed for policy makers, NGOs, researchers, journalists and the general public. The GIDD enables users to explore, filter and sort the data to produce their own graphs and tables.

Table 2-14: Global Internal Displacement Database Information

Field	Description
database	Database name
country	Country the disaster happened at
openYear	Year disaster started
openMonth	Month disaster started

openDay	Day disaster started
openDate	Date disaster started
description	Description of disaster
emergencyTypeEnum	Type of disaster
evacuatedPeopleNumber	Number of evacuated people

Table 2-15: Global Internal Displacement Database Fields

2.2 Transformations

From the previous selected data sources, the following tables show how the initial data will be transformed. The transformation process is done by following the guidelines of the data model and transforming the fields into the appropriate format requested using a scripting language (for example we extract from the date field, the fields year, month and day as integers). An example of the data model and the fields required is shown in the following table.

Field	Type
emergencyTypeEnum	string
name	string
title	string
description	string
database	string
openYear	integer
openMonth	integer
openDay	integer
openDate	string
openTime	datetime.time
closeYear	integer
closeMonth	integer
closeDay	integer
closeDate	string
closeTime	datetime.time
iso	string
country	string
locationArea	string
longitude	float
latitude	float
locationType	string
vehicleType	string
cause	string
linkedEmergencies	string
magnitude	float
magnitudeMeasurementUnit	string
riverBasin	string
casualties	integer
injuredPeopleNumber	integer
homelessPeopleNumber	integer
affectedPeopleNumber	integer
payments	float
damages	float
insuredDamages	float
reconstructionCosts	float
currency	string

fireFighters	integer
fireTrucks	integer
fireShips	integer
volunteers	integer
army	integer
otherForces	integer
fireTrucks	integer
otherVehicles	integer
helicopters	integer
airplanes	integer
gap	float
dmin	float
rms	float
horizontalError	integer
depthError	integer
magError	float
evacuatedPeopleNumber	integer
insurancePayments	integer

Table 2-16: Fields in data model

Below the transformations made for every resource are presented.

Field of CDD	Field in JSON	Change
EVENT CATEGORY	-	
EVENT GROUP	-	
EVENT SUBGROUP	-	
EVENT TYPE	-	
EVENT START DATE	openDate, openYear, openMonth, openDay	Created fields for year, month, and day
COMMENTS	description	-
FATALITIES	casualties	-
INJURED / INFECTED	injuredPeopleNumber - infectedPeopleNumber	If emergencyTypeEnum == EPIDEMIC then use infectedPeopleNumber, else use injuredPeopleNumber
EVACUATED	evacuatedPeopleNumber	-
ESTIMATED TOTAL COST	damages	-
NORMALIZED TOTAL COST	-	
EVENT END DATE	closeDate, closeYear, closeMonth, closeDay	Created fields for year, month, and day
FEDERAL DFAA PAYMENTS	-	
PROVINCIAL DFAA PAYMENTS	-	
PROVINCIAL DEPARTMENT PAYMENTS	-	
MUNICIPAL COSTS	-	
OGD COSTS	-	
INSURANCE PAYMENTS	insurancePayments	
NGO PAYMENTS	-	
UTILITY - PEOPLE AFFECTED	affectedPeopleNumber	-
MAGNITUDE	magnitude	-

	payments	Payments = FEDERAL DFAA PAYMENTS + PROVINCIAL DFAA PAYMENTS + PROVINCIAL DEPARTMENT PAYMENTS + NGO PAYMENTS + INSURANCE PAYMENTS
	country	country = Canada
	currency	currency = USD
	locationType	locationType = EVENT TYPE if EVENT TYPE == Residential or Non-residential or Vehicle
	vehicleType	vehicleType = EVENT TYPE if EVENT TYPE == Air, Marine, rail, Road
	database	database = "Canadian Disaster Database (CDD)"
	emergencyTypeEnum	If locationType not NULL and vehicleType not NULL then emergencyTypeEnum = EVENT SUBGROUP else emergencyTypeEnum = EVENT TYPE also see following table

Table 2-17: Field Transformations in CDD

CDD	Final
Epidemic	EPIDEMY
Infestation	EPIDEMY
Pandemic	EPIDEMY
Drought	-
Wildfire	FIRE
Cold Event	-
Heat Event	-
Hurricane / Typhoon / Tropical Storm	STORM
Typhoon	STORM
Tropical Storm	STORM
Storm Surge	STORM
Storm – Unspecified / Other	STORM
Winter Storm	STORM
Storms and Sever Thunderstorms	STORM
Tornado	STORM
Geomagnetic Storm	STORM
Avalanche	AVALANCHE
Flood	FLOOD
Tsunami	TSUNAMI
Landslide	LANDSLIDE
Earthquake	EARTHQUAKE
Volcano	VOLCANO
Disturbances / Demonstrations	-
Rioting	-
Hijackings	TERRORISM

Biological	TERRORISM
Bomb Attacks	TERRORISM
Chemical	TERRORISM
False Alarm	-
Hoax	-
Kidnapping / Murder	-
Nuclear	TERRORISM
Shootings	TERRORISM
Radiological	TERRORISM
Arson	FIRE
Explosion	FIRE
Fire	FIRE
Leak / Spill Release	CHEMICAL
Derailment Release	CHEMICAL
Vehicle Release	CHEMICAL
Marine Release	CHEMICAL
Transportation Accident	TRANSPORTATION
Communications	-
Energy	-
Manufacturing / Industry	-
Transportation	TRANSPORTATION
Water	-
Space Debris	-
Space Launch	-

Table 2-18: Emergency Mapping in CDD

Field of EMDAT	Field in JSON	Change
Dis No	-	
Year	-	
Seq	-	
Disaster Group	-	
Disaster Subgroup	-	
Disaster Type	-	
Disaster Subtype	-	
Disaster Subsubtype	-	
Event Name	name	
Entry Criteria	-	
Country	country	
ISO	iso	
Region	-	
Continent	-	
Location	locationArea	
Origin	cause	
Associated Dis		
Associated Dis2		
OFDA Response	-	
Appeal	-	
Declaration	-	
Aid Contribution	Payments (x1000)	
Dis Mag Value	Magnitude	
Dis Mag Scale	magnitudeMeasurementUnit	
Latitude	latitude	
Longitude	longitude	
Local Time	openTime	

River Basin	riverBasin	
Start Year	openYear	
Start Month	openMonth	
Start Day	openDay	
End Year	enYear	
End Month	endMonth	
End Day	endDay	
Total Deaths	casualties	
No Injured	injuredPeopleNumber	
No Affected	affectedPeopleNumber	
No Homeless	homelessPeopleNumber	
Total Affected (sum of injured, affected and homeless)	-	
Reconstruction Costs ('000 US$)	ReconstructionCosts (x1000)	
Insured Damages ('000 US$)	insuredDamanges (x1000)	
Total Damages ('000 US$)	Damages (x1000)	
CPI	-	
	emergencyTypeEnum	See below
	vehicleType	See below
	locationType	See below
	database = "EMDAT"	
	linkedEmergencies = Associated Dis + Associated Dis2	

Table 2-19: Field Transformations in EMDAT

Disaster Group	Disaster Subgroup	Disaster Type	Disaster Subtype	Disaster Subsubtype	Category in SnR
Natural	Climatological	Drought	Drought		-
Technological	Technological	Industrial accident	Explosion		FIRE
Natural	Geophysical	Earthquake	Ground movement		EARTHQUAKE
Natural	Geophysical	Volcanic activity	Ash fall		VOLCANO
Natural	Geophysical	Mass movement (dry)	Rockfall		LANDSLIDE
Technological	Technological	Miscellaneous accident	Fire		FIRE
Technological	Technological	Miscellaneous accident	Explosion		FIRE
Natural	Meteorological	Storm	Tropical cyclone		STORM
Natural	Hydrological	Flood			FLOOD
Technological	Technological	Transport accident	Water		TRANSPORTATION
Technological	Technological	Transport accident	Rail		TRANSPORTATION

Technological	Technological	Miscellaneous accident	Collapse		COLLAPSE
Natural	Biological	Epidemic	Bacterial disease		EPIDEMY
Natural	Geophysical	Mass movement (dry)	Landslide		LANDSLIDE
Technological	Technological	Industrial accident	Other		-
Natural	Hydrological	Landslide	Avalanche		AVALANCHE
Natural	Climatological	Wildfire	Forest fire		FIRE
Natural	Hydrological	Flood	Riverine flood		FLOOD
Natural	Meteorological	Storm	Convective storm	Tornado	STORM
Technological	Technological	Transport accident	Air		TRANSPORTATION
Natural	Biological	Epidemic	Viral disease		EPIDEMY
Natural	Hydrological	Landslide	Landslide		LANDSLIDE
Natural	Hydrological	Landslide	Mudslide		LANDSLIDE
Natural	Geophysical	Earthquake	Tsunami		TSUNAMI
Natural	Meteorological	Storm	Convective storm	Hail	STORM
Natural	Meteorological	Storm			STORM
Natural	Hydrological	Landslide			LANDSLIDE
Natural	Meteorological	Extreme temperature	Heat wave		-
Natural	Climatological	Wildfire	Land fire (Brush, Bush, Pasture)		FIRE
Technological	Technological	Industrial accident	Gas leak		CHEMICAL
Technological	Technological	Miscellaneous accident	Other		-
Technological	Technological	Industrial accident	Fire		FIRE
Natural	Climatological	Wildfire			FIRE
Technological	Technological	Transport accident	Road		TRANSPORTATION
Natural	Meteorological	Fog			-
Natural	Hydrological	Flood	Coastal flood		FLOOD
Technological	Technological	Industrial accident	Collapse		COLLAPSE
Natural	Meteorological	Storm	Convective storm	Severe storm	STORM
Natural	Meteorological	Storm	Convective storm	Winter storm/Blizzard	STORM
Natural	Meteorological	Extreme temperature	Cold wave		-
Natural	Hydrological	Flood	Flash flood		FLODD
Natural	Biological	Epidemic	Parasitic disease		EPIDEMY
Natural	Meteorological	Storm	Convective storm	Lightning/Thunderstorms	STORM

Technological	Technological	Industrial accident	Chemical spill		CHEMICAL
Natural	Geophysical	Mass movement (dry)	Landslide	Mudslide	LANDSLIDE
Technological	Technological	Industrial accident	Poisoning		CHEMICAL
Complex Disasters	Complex Disasters	Complex Disasters	Famine		-
Technological	Technological	Industrial accident	Oil spill		CHEMICAL
Natural	Biological	Epidemic			EPIDEMY
Natural	Biological	Insect infestation	Locust		-
Complex Disasters	Complex Disasters	Complex Disasters			-
Technological	Technological	Industrial accident	Radiation		-
Natural	Biological	Insect infestation			-
Technological	Technological	Miscellaneous accident			-
Natural	Meteorological	Storm	Convective storm		STORM
Natural	Meteorological	Storm	Convective storm	Sand/Dust storm	STORM
Natural	Meteorological	Extreme temperature	Severe winter conditions		-
Natural	Biological	Insect infestation	Grasshopper		-
Natural	Geophysical	Mass movement (dry)	Avalanche		AVALANCHE
Natural	Geophysical	Mass movement (dry)	Subsidence	Sudden subsidence	COLLAPSE
Natural	Meteorological	Storm	Extra-tropical storm		STORM
Natural	Hydrological	Landslide	Subsidence	Sudden subsidence	LANDSLIDE
Natural	Hydrological	Landslide	Avalanche	Winter storm/Blizzard	AVALANCHE
Natural	Meteorological	Extreme temperature	Severe winter conditions	Snow/Ice	-
Natural	Hydrological	Landslide	Rockfall		LANDSLIDE
Natural	Geophysical	Mass movement (dry)			EARTHQUAKE
Technological	Technological	Industrial accident			-
Natural	Extra-terrestrial	Impact			-
Natural	Meteorological	Storm	Convective storm	Rain	STORM

Natural	Geophysical	Volcanic activity	Lava flow		VOLCANO
Natural	Geophysical	Volcanic activity			VOLCANO
Natural	Biological	Animal accident			-
Natural	Geophysical	Earthquake			EARTHQUAKE
Natural	Meteorological	Storm	Convective storm	Storm/Surge	STORM
Natural	Meteorological	Storm	Convective storm	Derecho	STORM
Natural	Geophysical	Volcanic activity	Pyroclastic flow		VOLCANO
Natural	Climatological	Drought			-
Natural	Climatological	Glacial lake outburst			-
Technological	Technological	Transport accident			TRANSPORTATION

Table 2-20: Emergency Mapping in EMDAT

Disaster Type	Disaster Subtype	Category in SnR	EMERGENCY TYPE	Sequence	VEHICLE TYPE	LOCATION TYPE
Landslide	Avalanche	AVALANCHE	disaster subtype == avalanche	1		
Mass movement (dry)	Avalanche	AVALANCHE				
Landslide	Avalanche	AVALANCHE				
Industrial accident	Gas leak	CHEMICAL	disaster subtype in [gas leak, chemical spill, poisoning, oil spill]	2		Industrial
Industrial accident	Chemical spill	CHEMICAL				Industrial
Industrial accident	Poisoning	CHEMICAL				Industrial
Industrial accident	Oil spill	CHEMICAL				Industrial
Miscellaneous accident	Collapse	COLLAPSE	disaster subtype == collapse	3		
Industrial accident	Collapse	COLLAPSE				Industrial

Mass movement (dry)	Subsidence	EARTHQUAKE	disaster type in [mass movement (dry), earthquake]	12		
Earthquake	Ground movement	EARTHQUAKE				
Mass movement (dry)		EARTHQUAKE				
Earthquake		EARTHQUAKE				
Epidemic	Bacterial disease	EPIDEMY	disaster type == epidemic	4		
Epidemic	Viral disease	EPIDEMY				
Epidemic	Parasitic disease	EPIDEMY				
Epidemic		EPIDEMY				
Industrial accident	Explosion	FIRE	disaster type == wildfire or disaster subtype in [explosion, fire]	10		Industrial
Miscellaneous accident	Fire	FIRE				
Miscellaneous accident	Explosion	FIRE				
Wildfire	Forest fire	FIRE				Natural Open Space
Wildfire	Land fire (Brush, Bush, Pasture)	FIRE				Natural Open Space
Industrial accident	Fire	FIRE				Industrial
Wildfire		FIRE				
Flood	Flash flood	FLODD	disaster type == flood	5		
Flood		FLOOD				
Flood	Riverine flood	FLOOD				
Flood	Coastal flood	FLOOD				

Mass movement (dry)	Rockfall	LANDSLIDE	disaster type == landslide or disaster subtype in [rockfall, landslide]	11		
Mass movement (dry)	Landslide	LANDSLIDE				
Landslide	Landslide	LANDSLIDE				
Landslide	Mudslide	LANDSLIDE				
Landslide		LANDSLIDE				
Mass movement (dry)	Landslide	LANDSLIDE				
Landslide	Subsidence	LANDSLIDE				
Landslide	Rockfall	LANDSLIDE				
Storm	Tropical cyclone	STORM	disaster type == storm	6		
Storm	Convective storm	STORM				
Storm	Convective storm	STORM				
Storm		STORM				
Storm	Convective storm	STORM				
Storm	Convective storm	STORM				
Storm	Convective storm	STORM				
Storm	Convective storm	STORM				
Storm	Convective storm	STORM				
Storm	Extra-tropical storm	STORM				
Storm	Convective storm	STORM				
Storm	Convective storm	STORM				

Storm	Convective storm	STORM				
Transport accident	Water	TRANSPORTATION	disaster type == transport accident	7	marine	vehicle
Transport accident	Rail	TRANSPORTATION			rail	vehicle
Transport accident	Air	TRANSPORTATION			air	vehicle
Transport accident	Road	TRANSPORTATION			road	vehicle
Transport accident		TRANSPORTATION				vehicle
Earthquake	Tsunami	TSUNAMI	disaster subtype == tsunami	8		
Volcanic activity	Ash fall	VOLCANO	disaster type == volcanic activity	9		
Volcanic activity	Lava flow	VOLCANO				
Volcanic activity		VOLCANO				
Volcanic activity	Pyroclastic flow	VOLCANO				
Drought	Drought					
Industrial accident	Other					
Extreme temperature	Heat wave					
Miscellaneous accident	Other					
Fog						
Extreme temperature	Cold wave					
Complex Disasters	Famine					
Insect infestation	Locust					
Complex Disasters						

Industrial accident	Radiation					
Insect infestation						
Miscellaneous accident						
Extreme temperature	Severe winter conditions					
Insect infestation	Grasshopper					
Extreme temperature	Severe winter conditions					
Industrial accident						
Impact						
Animal accident						
Drought						
Glacial lake outburst						

Table 2-21: Vehicle and Location Type Mapping in EMDAT

Field in GLC	FIeld in JSON	Change
	database	database = "Global Landslide Catalog"
submitted_date	reportedDate	string -> datetime.date
event_tltle	title	
event_description	description	
landslide_trigger	cause	
landslide_size	severity	Mapped fields based on below severity mapping
event_date	openDate openYear openMonth openDay	Created fields for open year, month, day as int from string
event_date	openTime	string -> datetime.time
longitude	longitude	
latitude	latitude	
country_name	country	
landslide_setting	locationType	Mapped fields based on below locationType mapping
fatality_count	casualties	float -> int
injured_count	injuredPeopleNumber	float -> int

	emergencyTypeEnum	EmergencyTypeEnum="LANDSLIDE"

Table 2-22: Field Transformations in Global Landslide Catalog

landslide_size	severity
small	MINOR
medium	MEDIUM
large	MAJOR
very_large	CRITICAL
catastrophic	CRITICAL
unknown	

Table 2-23: Severity Mapping in Global Landslide Catalog

landslide_setting	locationType
mine	Industrial
unknown	
above_road	Infrastructure
urban	Residential
natural_slope	Non-residential
engineered_slope	Man-made open space
below_road	Infrastructure
above_river	Non-residential
retaining_wall	Infrastructure
other	
above_coast	Non-residential
bluff	Non-residential
burned_area	Non-residential
deforested_slope	Non-residential

Table 2-24: Location Type Mapping in Global Landslide Catalog

Field in HFS	Field in JSON	Change
Α/Α ΕΓΓΡΑΦΗΣ	-	
Υπηρεσία	-	
Νομός	region	Map "Νομός" to region using region_mapping dictionary
Ημερ/νία Έναρξης	openDate openYear openMonth openDay	Created fields for open year, month, day as int from pandas._libs.tslibs.timestamps.Timestamp
Ώρα Έναρξης	openTime	string -> datetime.time
Ημερ/νία Κατασβεσης	closeDate closeYear closeMonth closeDay	Created fields for close year, month, day as int from pandas._libs.tslibs.timestamps.Timestamp
Ώρα Κατάσβεσης	closeTime	string -> datetime.time
Δασαρχείο	-	
Δήμος	-	
Περιοχή	-	
Διεύθυνση	-	
Δάση	damagedLand	damagedLand = Δάση + Δασική Έκταση + Άλση + Χορτ/κές Εκτάσεις + Καλάμια – Βάλτοι + Γεωργικές Εκτάσεις +
Δασική Έκταση		
Άλση		

Χορτ/κές Εκτάσεις Καλάμια – Βάλτοι Γεωργικές Εκτάσεις Υπολλέιματα Καλλιεργειών Σκουπιδότοποι		Υπολλέιματα Καλλιεργειών + Σκουπιδότοποι float -> int
ΠΥΡΟΣ. ΣΩΜΑ ΠΕΖΟΠΟΡΑ ΤΜΗΜΑΤΑ	fireFighters	fireFighters = ΠΥΡΟΣ. ΣΩΜΑ + ΠΕΖΟΠΟΡΑ ΤΜΗΜΑΤΑ float -> int
ΕΘΕΛΟΝΤΕΣ	volunteers	float -> int
ΣΤΡΑΤΟΣ	army	float -> int
ΑΛΛΕΣ ΔΥΝΑΜΕΙΣ	otherForces	float -> int
ΠΥΡΟΣ. ΟΧΗΜ.	fireTrucks	float -> int
ΟΧΗΜ. ΟΤΑ ΒΥΤΙΟΦΟΡΑ ΜΗΧΑΝΗΜΑΤΑ	otherVehicles	otherVehicles = ΟΧΗΜ. ΟΤΑ + ΒΥΤΙΟΦΟΡΑ + ΜΗΧΑΝΗΜΑΤΑ float -> int
ΕΛΙΚΟΠΤΕΡΑ	helicopters	float -> int
Α/Φ CL415 Α/Φ CL215 Α/Φ PZL Α/Φ GRU	airplanes	Airplanes = Α/Φ CL415 + Α/Φ CL215 + Α/Φ PZL + Α/Φ GRU float -> int
	country	country = Greece
	emergencyTypeEnum	emergencyTypeEnum = "FIRE"
	locationType	locationType = "Non-residential"

Table 2-25: Field Transformations in Hellenic Fire Service (Forest)

Field in HFS	Field in JSON	Changes
Α/Α ΕΓΓΡΑΦΗΣ	-	
Υπηρεσία	-	
Νομός	region	Map "Νομός" το region using region_mapping dictionary
Είδος Συμβάντος	-	
Ημερ. Έναρξης Συμβάντος	openDate openYear openMonth openDay	Created fields for open year, month, day as int from pandas._libs.tslibs.timestamps.Timestamp
Ώρα Έναρξης	openTime	string -> datetime.time
Ημερ. Κατασβεσης	closeDate closeYear closeMonth closeDay	Created fields for close year, month, day as int from pandas._libs.tslibs.timestamps.Timestamp
Ώρα Κατάσβεσης	closeTime	string -> datetime.time
Δήμος	-	
Χωριό	-	
Περιγραφή Χώρου	-	
Χαρακτηρισμός Συμβάντος	severity	Map "Χαρακτηρισμός Συμβάντος" to severity
Σύνολο Πυρ. Οχημάτων	fireTrucks	float -> int
Σύνολο Πυρ. Δυνάμεων (σε άνδρες και γυναίκες)	fireFighters	float -> int

Σύνολο Πυροσβ. Πλοιαρίων	fireShips	float -> int
Τύπος Ατυχήματος	-	
Αριθμός εμπλεκομένων ανά τύπο	involvedPeopleNumber	
Τραυματίες	injuredPeopleNumber	InjuredPeopleNumber = Τραυματίες + Εγκαύματα
Εγκαύματα		
Θάνατοι	casualties	
Καταστροφές	-	
	country	country = “Greece”
	locationType	locationType = “Residential”
	emergencyTypeEnum	emergencyTypeEnum = “FIRE”

Table 2-26: Field Transformations in Hellenic Fire Service (Residential)

Field in USGS	Field in JSON	Changes
	database	database = “USGS Earthquake Program”
	emergencyTypeEnum	emergencyTypeEnum = “EARTHQUAKE”
time	openYear openMonth openDay openDate openTime	string -> datetime.datetime object Extract year, month, day as int Extract date as string Extract time as datetime.time()
latitude	latitude	
longitude	longitude	
depth (measured in km)	depth (measured in meters)	return depth * 1000 if exists else “”
mag	magnitude	return mag if exists else “”
gap	gap	return gap if exists else “”
dmin (measured in degrees)	dmin	return dmin if exists else “”
rms	rms	return rms if exists else “”
place	locationArea	return place if exists else “”
horizontalError (measured in km)	horizontalError (measured in meters)	return horizontalError * 1000 if exists else “”
depthError (measured in km)	depthError (measured in meters)	return depthError * 1000 if exists else “”
magError	magError	return magError if exists else “”

	country	create country from latitude,longitude
		if country is found:
		    return country
		else:
		    return “”

Table 2-27: Field Transformations in USGS Earthquake Program

Field in GIDD	Field in JSON	Changes
	database	database = “Global Internal Displacement Database”
Name	country	
Year	openYear	
Start Date	openMonth, openDay	Created fields for open month, day as int from string

Event Name	description	
Hazard Type	emergencyTypeEnum	Mapped fields based on below emergency mapping
New Displacements	evacuatedPeopleNumber	

Table 2-28: Field Transformations in Global Internal Displacement Database

Hazard Type	emergencyTypeEnum
Flood	FLOOD
Extreme Temperature	
Earthquake	EARTHQUAKE
Wet Mass Movement	FLOOD
Dry Mass Movement	LANDSLIDE
Storm	STORM
Drought	
Volcanic eruption	VOLCANO
Wildfire	FIRE
Mass Movement	LANDSLIDE
Volcanic Activity	VOLCANO
Severe Winter condition	AVALANCHE

Table 2-29: Emergency Mapping in Global Internal Displacement Database

2.3 Component's Functional Specification

In this section the tools used to implement the API, how the system should be configured and how the errors are handled by the application are presented.

2.3.1 Development Tools

The technologies and tools used to implement the API service are shown in the below table.

Tool	Description
Python [12]	Python is an interpreted high-level general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.
FastAPI [13]	FastAPI is a modern, fast (high-performance), web framework for building APIs with Python.
Selenium [14]	Selenium Python bindings provides a simple API to write functional/acceptance tests using Selenium WebDriver. Through Selenium Python API you can access all functionalities of Selenium WebDriver in an intuitive way.
Docker [15]	Docker is a set of platform as a service (PaaS) products that use OS-level virtualization to deliver software in packages called containers. Containers are isolated from one another and bundle their own software, libraries and configuration files; they can communicate with each other through well-defined channels. Because all of the containers share the services

	of a single operating system kernel, they use fewer resources than virtual machines.

Python was chosen as the programming language to implement the API as it has great documentation, a lot of libraries and can speed up development.

FastAPI was used as the framework to create the API because it is a modern web framework that specializes in building APIs in a fast way, is pretty robust and compatible with open standards such as OpenAPI.

Selenium was used to automate navigation in some of the web sites in order to download specific resources.

Docker was used to containerize the application so it is easier to test and distribute to relevant partners.

2.3.2 System Configuration

The code repository resides at the Decision and Support Systems' internal Gitlab server. To run the service the system is configured with the following.

- Ubuntu 20
- Docker 20.10.6
- Git 2.31.1

2.3.3 Error Reporting and Exception Handling

It is important to correctly report errors and handle exceptions in all cases. So, when doing an API call, there are various exceptions being handled and the user is given a reason in case the call could not return a valid response.

```
[
    "<urlopen error [Errno -2] Name or service not known>",
    400
]
```

Figure 2-1: Example of handled exception

2.4 API Specification

The following table shows the endpoints and provide the API specification for each one.

Resource	Endpoint	Method	Response
Canadian Disaster Database	/canadian_disaster_database	GET	{ "emergencyTypeEnum" : string, "description": string, "database": string, "openYear": int, "openMonth": int, "openDay": int, "openDate": string, "closeYear": int, "closeMonth": int, "closeDay": int, "closeDate": string,

			"country": string, "locationType": string, "vehicleType": string, "magnitude": float, "casualties": int, "injuredPeopleNumber" : int, "infectedPeopleNumber ": int, "evacuatedPeopleNum ber": int, "affectedPeopleNumbe r": int, "damages": int, "payments": int, "insurancePayments": int, "currency": string }

EM-DAT	/emdat	POST	{
			emergencyTypeEnum:
			string,
			name: string,
			database string,
			openYear: int,
			openMonth: int,
			openDay: int,
			openTime: datetime.time,
			closeYear: int,
			closeMonth: int,
			closeDay: int,
			iso: string,
			country: string
			locationArea: string
			longitude: string
			latitude: string
			locationType: string
			vehicleType: string
			cause: string
			linkedEmergencies: string
			magnitude: float
			magnitudeMeasurementU
			nit: string
			riverBasin: string
			casualties: int,
			injuredPeopleNumber: int,
			homelessPeopleNumber:
			int,
			affectedPeopleNumber:
			int,
			payments: float,
			damages: float,
			insuredDamages: float,
			reconstructionCosts: float,
			currency: string
			}

Global Landslide Catalog	/global_landslide_catalog	GET	{
			database: string,
			reportedDate: datetime.date,
			title: string,
			description: string,
			cause: string,
			severity: string,
			openYear: int,
			openMonth: int,
			openYear: int,
			openDate: string
			openTime: datetime.time,
			country: string,
			locationType: string,
			casualties: int,
			injuredPeopleNumber: int,
			emergencyTypeEnum: string,
			latitude: float,
			longitude: float
			}

Hellenic Fire Service (Forest)	/hellenic_fire_service_forest	GET	{
			openYear: int,
			openMonth: int,
			openDay: int,
			openDate: string
			closeYear: int,
			closeMonth: int,
			closeDay: int,
			closeDate: string,
			openTime: datetime.time,
			closeTime: datetime.time,
			damagedLand: float,
			fireFighters: int,
			volunteers: int,
			army: int,
			otherForces: int,
			fireTrucks: int,
			otherVehicles: int,
			helicopters: int,
			airplanes: int,
			country: string,
			emergencyTypeEnum: string,
			locationType: string,
			region: string,
			database: string
			}

Hellenic Fire Service (Residential)	/hellenic_fire_service_residential	GET	{ openYear: int, openMonth: int, openDay: int, openDate: string closeYear: int, closeMonth: int,

			closeDay: int, closeDate: string, openTime: datetime.time, closeTime: datetime.time, fireFighters: int, fireTrucks: int, fireShips: int, country: string, emergencyTypeEnum: string, locationType: string, region: string, involvedPeopleNumber: int, injuredPeopleNumber: int, casualties: int, severity: string, database: string }

USGS Earthquake Program	/usgs_earthquake_program	GET	{
			database: string,
			emergencyTypeEnum: string,
			openYear: int,
			openMonth: int,
			openDay: int,
			openTime: datetime.time,
			openDate: string,
			latitude: float,
			longitude: float,
			depth: int,
			magnitude: float,
			gap: float,
			dmin: float,
			rms: float,
			locationArea: string,
			horizontalError: int,
			depthError: int,
			magError: float,
			country: string
			}

Global Internal Displacement Database	/global_internal_displacement_database	GET	{ database: string, country: string, openYear: int, openMonth: int, openDay: int, openDate: string description: string, emergencyTypeEnum: string, evacuatedPeopleNumber: int}

Table 2-30: API Specification

2.5 Technical Evaluation

To evaluate the responses unit tests are used.

The unit tests evaluate the following:

- The response code
- The response content type
- The response schema


```
▼ object {1}
  ▼ testsuites {1}
    ▼ testsuite {9}
      ▼ testcase {3}
          _classname : test_app
          _name : test_cdd
          _time : 2.254
        _errors : 0
        _failures : 0
        _hostname : snr_databases
        _name : pytest
        _skipped : 0
        _tests : 1
        _time : 3.092
        _timestamp : 2021-06-02T13:32:46.663253
```


Figure 2-2: Example Unit test for the Canadian Disaster Database

2.6 Development and Implementation Environments

The development was done on local machines using the above mentioned technologies and the code repository resides at the Decision and Support Systems' internal Gitlab server.
The component is being deployed for testing purposes on a shared physical Hyper-V server hosted at the Decision Support Systems Laboratory, NTUA. The deployment environment is an Ubuntu 20 Virtual Machine with 4GB of provisioned RAM and 50GB of storage. Deployment is done by synching the GitLab repository and deploying the component as a docker image.

2.7 Indicative Output

The following figure showcases the indicative output of a request to the Canadian Disaster Database.

```
[
  { … }, // 24 items
  { … }, // 24 items
  { … }, // 24 items
  { … }, // 24 items
  { … }, // 24 items
  { … }, // 24 items
  { … }, // 24 items
  { … }, // 24 items
  {
    "emergencyTypeEnum": "STORM",
    "description": "[redacted]",
    "database": "Canadian Disaster Database (CDD)",
    "openYear": 1992,
    "openMonth": 9,
    "openDay": 1,
    "openDate": "1992-09-01",
    "closeYear": 1992,
    "closeMonth": 9,
    "closeDay": 1,
    "closeDate": "1992-09-01",
    "country": "Canada",
    "locationType": "",
    "vehicleType": "",
    "magnitude": 0,
    "casualties": 0,
    "injuredPeopleNumber": 0,
    "infectedPeopleNumber": 0,
    "evacuatedPeopleNumber": 0,
    "affectedPeopleNumber": 0,
    "damages": [redacted],
    "payments": [redacted],
    "insurancePayments": [redacted],
    "currency": "USD"
  },
  { … }, // 24 items
  { … }, // 24 items
```


Figure 2-3: Indicative output for CDD

3 Aggregation of data coming from internal sources and devices

3.1 Scope

This chapter describes the data management related service, in order to achieve the data aggregation of the incoming data of the sensors and devices used at the field and every other data parameter affects the SnR operation. A demonstration of D4.8 will be released on the upcoming S&R's review in M15.
D4.8 focuses on the components that will enable the data aggregation, the storage and the transformation of:

- structured data
- semi-structured data
- raw data

These types of data have different formats, values and measurements. A main approach must be adapted on all these types to turn them into a homogeneous and useful knowledge for the S&R operation.

This section presents:

- The Big Data approach
- The Data Lake Definition
- The Suitable Components that will enable the data aggregation mechanism
- A data aggregation instance using dummy data (instead of real-time data) and historical data

All the services that are going to use the aggregated and processed data in their environments must give specific requirements to guide the outcomes of the required knowledge. For instance, SOT DSS component needs historical data from T4.1 with specific fields such as country, type of emergency, location type, but they also need specific values from the field devices, such as smartwatches that generate the heart rate, the blood oxygen measurement and the blood pressure. Moreover, the SOT components need the average of the smartwatch's outcome. As a result, the aggregation mechanism will be adjusted on the requirements above and it will process the data, in order to have an outcome such as the below:

Country: { Greece }
Type of emergency: { Fire }
Location Type: { Residential }
Sensor: { Smartwatch }
Heart Rate: { 73,45 bpm }
Blood Oxygen Measurement: { 95,23 % }
Blood Pressure: { 138/88 mmHg }

On the one hand, this outcome will be sent to the SOT component, in order to train its models/algorithms, and on the other hand the requested average values of the smartwatch will be sent to COncORDE dashboard as a notification. This is a simple instance, in order to make the scope, the requirements and the outcome of this technical deliverable, understandable to the S&R consortium. The data aggregation mechanism is responsible to aggregate all the incoming data, process them, create homogeneous knowledge and spread it to the rest of the services.

3.2 Data Platform

3.2.1 Big Data Approach

Search and Rescue is characterized by multiple different and difficult concepts, such as complexity, multiple edge devices, services and more. A big and crucial concern of the S&R project is the data generated by multiple sources on the field, with scope to feed the system with information on the evolving incident. All these data are heterogeneous with each other, have different formats and as a result, in their initial form, they cannot communicate with the system. There are technologies such as smartwatches, giving the heart rate of a patient, chemical sensors, detecting gas leaks, UAV imagery, robots and more. There are also software components such as detection algorithms, responsible for object detection (human, obstacles) applied on UAV imagery and robots, in order to enable the situation awareness on the field. Moreover, there are services, such as ConCORDE, the 3D command center and more, responsible for incident management and visualizations. Last but not least, S&R will contain decision making services, providing useful information and knowledge to first responders via dedicated services. In order to make the system produce valid information and decision making, there are some actions and steps that must be taken at strategic and technological level.
Search & Rescue operations will handle Big Data, a huge volume of data that cannot be stored and processed using the traditional computing approach within a given time frame. Big Data is categorized by three important characteristics [16]:

- Volume: refers to the amount of data that is getting generated
- Velocity: refers to the speed at which the data is getting generated
- Variety: refers to the different types of data that is getting generated

In a traditional approach, usually the data that is being generated is given as an input to an ETL (Extract, Transform and Load) System. An ETL System would extract, transform (convert the data to a proper format) and finally load data into the database. Once this process is completed, the end users would be able to perform various operations, such as generate reports and perform analytics by querying this data. But as this data grows, it becomes a challenging task to manage and process this data using only this traditional approach. This is one of the main reasons for not using the traditional approach for storing and processing Big Data, since S&R is a project with multiple data sources with different data formats, which have to be processed and generated in real-time.
Moreover, another important characteristic of Big Data, is the classification of the data into different categories, such as:

- Structured Data: Data that has a proper structure, such as data from databases with csv and xml format
- Semi-Structured Data: Data that does not have a proper structure, such as emails, log files, word documents, json files with json fields and more
- Unstructured Data: Data that does not have any structure associated with it at all, known as raw data, such as image, audio and video files

All the above categories of data will be produced by the technologies mentioned in the "D4.1 Data aggregation" and other deliverables such as "D7.1 S&R extensive service catalogue" and T4.2 is responsible for finding the most suitable mechanism to combine and get common and useful knowledge from all. The previous reasons led to the most suitable solution for Big Data projects, the Data Lake adaptation.

3.2.2 Data Lake Definition

A data lake is a centralized repository that allows the storage of structured, semi-structured and unstructured data at any scale. In general, data can be stored as-is, without having to first structure the data, and run different types of analytics, from dashboards and visualizations to big data processing, real-time analytics, and machine learning to guide better decisions.

Data lakes can store any type of data format, including images, videos, texts and files without any schema. Therefore, data lakes are efficient for various data sources and don't require ETL or transformations on them at first sight. This is another benefit that S&R Project can take advantage of. Data lakes can store Machine Learning model artifacts, real-time data, and analytics outputs, making them suitable to store data for numerous applications. In this case, the schema is defined on reading while data is getting processed on export and finally used outside of its ecosystem.
A data lake constitutes a base on Big Data architecture in which many components can run simultaneously in the same ecosystem for different main purposes. For example, a data lake ecosystem is capable of hosting or working with components for:

- Data storage: Apache Hadoop is most superior and commonly used for large data storage at scale
- Data replication: Replication is a smart way to ensure data are backed up and data loss is avoided
- Data (event) streaming: Spark Streaming receives live input data streams from an integration platform and divides the data into batches
- Data Distribution: All technologies have a default distributed function

This leads to the conclusion that Data Lakes run like an ecosystem and a collection of different technology architectures. Since Search and Rescue will run as a multi-software system with heterogeneous and real-time data, a Data Lake is the best solution to be adopted as a base of the data aggregation mechanism.

3.2.3 Data Lake Architecture

As it was mentioned in "D4.1 Data aggregation" in chapter 2.2, S&R will contain data fusion servers and a mediation system. At this point, it is essential to mention that Data fusion is the process of integrating from multiple data sources. The data aggregation mechanism, as part of the S&R Mediation System, is responsible for retrieving data from the data fusion system, and then handling every complex and critical event (data from multiple sources and data streams), in order to create valuable knowledge.
Before demystifying the data aggregation mechanism for the S&R project, it is essential to present the S&R's Data Lake architecture into layers, divided into three parts for better understanding, as it is shown in the figure below.

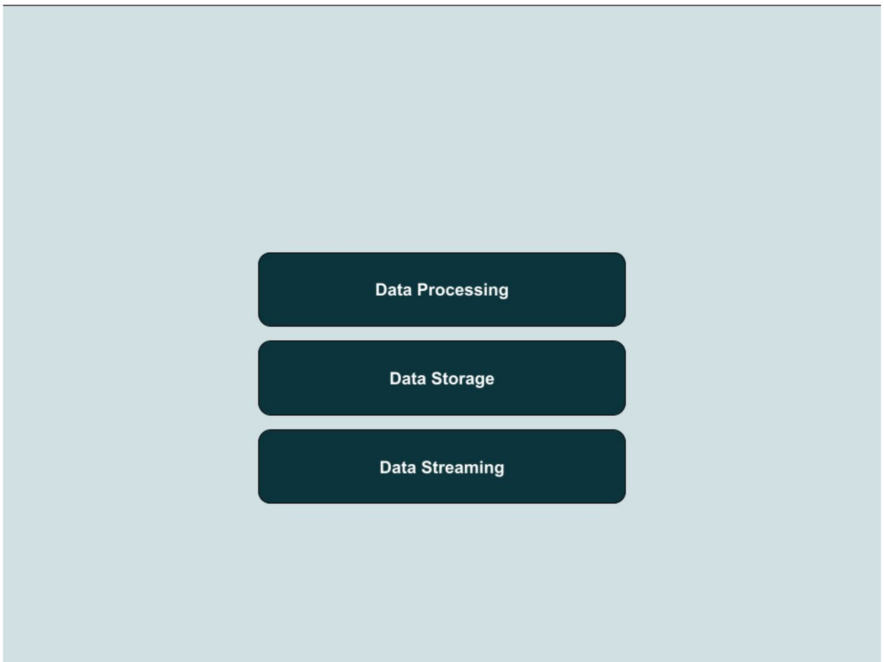


Figure 3-1: High Level Architecture, divided into layers

* This division serves the current document, and it is built around the Data Lake Ecosystem, in order to illustrate the components that will enable the data aggregation of S&R (in the next sub-chapter). The layers' order starts from the data streaming layer.

Dependencies

These layers of the data lake will be connected to other procedures outside the data lake ecosystem. More specifically, field data providers are responsible for feeding the system with data coming from the emergency location. Technologies such as sensors on first responders, drones, rescue robots and wearables have the role of the field data providers (as were described in the "D4.1 Data aggregation", "D1.1 Report on user requirements, existing tools and infrastructure", "D7.1 S&R extensive service catalogue"). All these sources are heterogeneous and have different communication systems, such as GPS, A-GPS, WiFi and bluetooth technologies and in their original form, they cannot provide useful information to the S&R platform. A standardization on the data and information from the data sources must be carried out, starting with the data ingestion.
Data ingestion or data collection layer is a process that collects data from various data sources in an unstructured format and stores it somewhere to analyze that data. This data can be real-time or integrated in batches. Real-time data is ingested as soon as it arrives, while the data in batches can be ingested at a periodical interval of time. To make this ingestion process work properly, it is essential to use different tools at different layers which will help to build the data pipelines. The data ingestion layer is a combination of "D4.8 Data aggregation" with "T7.5-S&R platform component and service integration", "T6.2 S&R Data Communication Interoperability framework", "T6.3 S&R services interoperability framework" and "T6.4 S&R Design of interoperability framework".
More specifically, the first layer comes after:

S&R Data Model

A designed data model coming from the T6.2 and THALIT, will guide the data ingestion, satisfying the requirements for data exchange, in order to build a schema-registry.
Apache Avro is the preferable tool which constitutes a remote procedure call and data serialization framework which uses JSON for defining data types and protocols, and serialized data in a compact binary format. The objective of T6.2 "Data Communication Interoperability Framework" is to define a

framework to acquire all the data produced on the field by the various devices, sensors, drones, rovers thus allowing the operators/rescuers to monitor the crisis evolution.
A study over the most common Enterprise Integration Patterns (EIP) has been done and, among all the patterns, the Canonical Data Model (CDM) has been chosen to design the Data Communication Interoperability Framework. In fact, the CDM is particularly suitable for contexts where many different and heterogeneous systems/sources produce data that shall be moved to one (or more) receiver(s). The CDM pattern allows encapsulation of all these different data into a unique common representation which proves to be very useful for integration problems.
The data model has been designed using UML notation, implemented in JAVA as relational POJO and released in two versions, i.e., as an XML Schema file representation and as a set of AVRO files. This allows the use of the data model both with the classical approach based on web services interfaces or with the most recent rest service approach based on data bus and topics.

Data Bus

D7.5 will mainly involve the integration of the software components and sensors that constitute the S&R system. Moreover, back-end integration with databases, other systems or data sources will be taken into consideration. The integration of software components and sensors must serve the integration of sensors and field data provided to the data lake for the data ingestion layer.
A preferable tool to allow the data bus and topics can be structured by Apache Kafka, an open-source event streaming platform. It is used for a distributed streaming platform that is used to build data pipelines. All Kafka records are organized into topics. A Topic is a category in which records are stored and published. Apache Kafka also contains in its ecosystem the kafka producers (generate data) and the kafka consumers (get data). For instance, in the publish-subscribe or pub-sub system, the messages are contained in a topic. Contrary to the point-to-point system, the consumers in this system can subscribe to more than one topic and consume all the messages in that topic. Producer applications write data to topics and consumer applications read from topics. Moreover, the ingestion process will act as a sink, eventually collecting the incoming data into a read-optimised database. When Kafka gets data from a producer (field data provider) via a topic, it takes bytes as input and publishes them without parsing or even reading the data. This zero-copy principle reduces the consumption of CPU and memory bandwidth, improving performance especially when the data volumes are huge.
In conclusion, Apache Kafka would be a preferable solution to constitute the data integration platform of Search and Rescue, since it can be used in S&R by all the technologies (field devices and application services), in order to produce and consume data.
There are always alternatives, such as Apache Flume, Apache Nifi, Apache Storm, Apache Sqoop sustaining substitutes to Kafka, which match and work in harmony with the data lake ecosystem. More details for the final tool will be provided in D7.5 in M16.

The above components can be a part of a data lake ecosystem consisting of the data ingestion layer. However, since these components are prerequisites for enabling the data aggregation mechanism, only the layers developing the aggregation mechanism are shown in D4.8.

Layer 1. Data Streaming

In this part of the deliverable the actual data aggregation mechanism will be introduced. Considering the figure "High Level Architecture, divided into layers", the first layer consists of the data streaming layer. A data stream is an unbounded sequence of data arriving continuously. In Search and Rescue, field devices will generate data in a data bus such as Kafka, as described in the previous chapter. Spark Streaming enables scalable and fault tolerant ETL operations that allow continuously aggregating data with low latency.
This component performs the operations related to the normalization of data and helps in converting the different keys and values received from various sources to respective associated forms, pointed by THALIT's data model.

More specifically, an entry point to the Spark streaming functionality is the creation of a Dstream from various input sources. In the S&R case, Dstream which stands for "direct stream", will fetch data from the data bus.

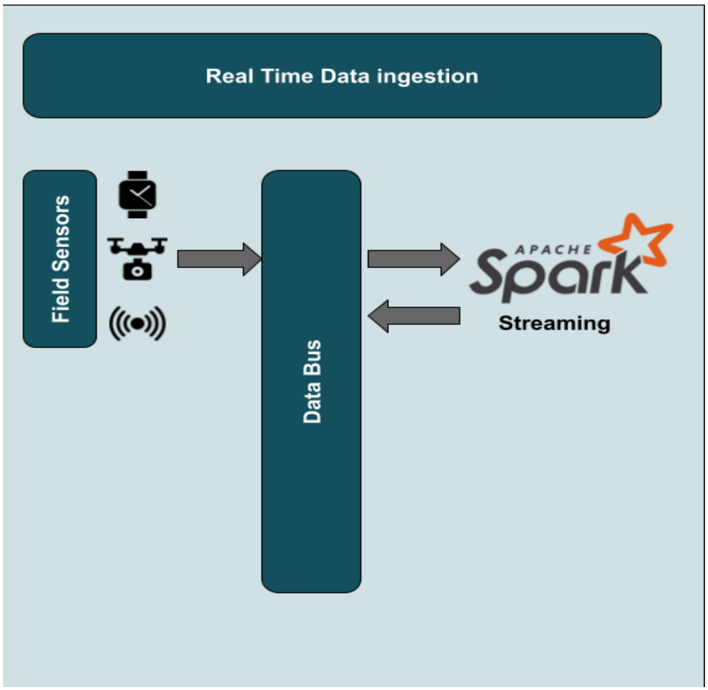


Figure 3-2: Real-time Data Ingestion

Spark streaming is a micro-batch based streaming library. This ensures that the streaming data is divided into batches based on time slice. Every batch gets converted into RDD and the continuous stream of RDD is called Dstream. From this point on, multiple functions, such as filter, foreach, foreachRDD can be used for data transformation. A DStream can be converted to a data frame for further aggregation too (more details about aggregation and data frames are described in layer 3 of the architecture and in the aggregation mechanism chapter). There is also the possibility of managing the pipeline by starting and terminating when the required aggregation is made [17].
Moreover, this module is capable of sending the transformed data immediately into external relational databases, such as postgreSQL(CONCORDE's DB), however this cannot consist a solution to the whole system, since for example, there are components with deep learning algorithms, which are responsible to create a further knowledge to the final system. For this reason, Apache streaming will be responsible for sending the transformed data to the storage layer (layer 2).
As a result, spark streaming is a way to export data from a Kafka topic to HDFS [17]. However, this document will not analyze these ways, since this layer has dependencies from other partners and their dedicated tasks in the coming future, as described above.

Layer 2. Data Storage

The second layer is the center of the actual Data Lake ecosystem, starting with the data storage and the first part of the aggregation mechanism. As described in the previous chapters, a data lake is a central repository that allows the storage of structured, semi-structured and unstructured data at any scale. Search and Rescue will use a cloud-based data lake in Digital Ocean, in order to store the field data and further process them. This repository will be provided by Konnektable for the needs of the S&R project.
For data storage, Hadoop is the selected component which is an open-source software framework for storage and large-scale processing of data sets on clusters of commodity hardware. The software licensed under Apache License 2.0 [18]. Hadoop allows the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines.

Hadoop Distributed File System (HDFS) is the storage system of Hadoop which splits big data and distributes across many nodes in a cluster allowing local computation and storage [16]. This also replicates data in a cluster thus providing high availability and backup in case of data loss. An HDFS cluster follows a master-slave architecture, with one (or more) NameNode as the master that manages the file system namespace and regulates access to files [17]. HDFS also has a number of DataNodes, usually one per node, acting as slaves that manage storage attached to their corresponding nodes. It supports a traditional hierarchical file organization and reliably stores very large files across machines in a large cluster by using replication for fault tolerance [19].
HDFS splits the data into small pieces (called blocks) and further distributes it to all the nodes in any typical Hadoop cluster. Moreover, it creates the replication of these small pieces of data as well as it stores them to ensure that the data is available from another node if any node is down.

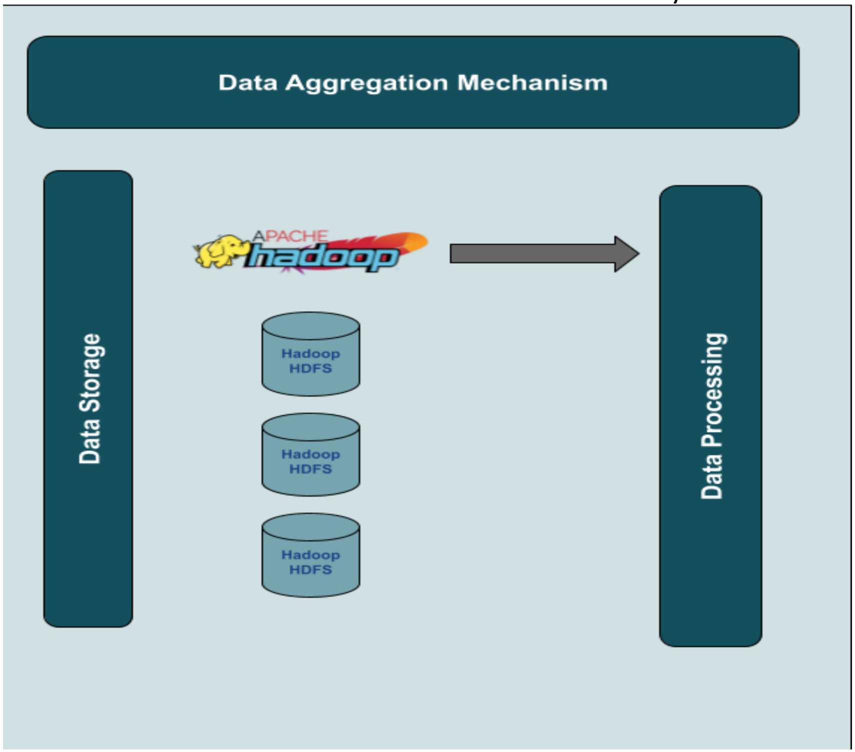


Figure 3-3: Data Storage Layer

One of the most important advantages of Hadoop and HDFS is that all data can be stored in its raw native format without any transformation or preprocessing. The data can be structured or unstructured and this makes HDFS an unnecessary component for a project such as S&R, which handles data types such as sensor and video data among the others. This feature provides further safety, in case there is incoming data that needs to be further processed (e.g with ML algorithms).
As mentioned above, HDFS uses Master-Slave architecture to distribute, store and retrieve the data efficiently. Hadoop distributed file System (HDFS) splits the large data files into parts which are managed by different machines/components in the cluster. An HDFS cluster consists of:

- HDFS Client: On user behalf, HDFS client interacts with NameNode and Datanode to fulfill user requests.
- NameNode: It is the centerpiece of an HDFS file system. It keeps the directory tree of all files in the file system and tracks where across the cluster the file data is kept. It does not store the data of these files itself, but it stores data about those files or metadata.
- DataNode: It stores data in the [HadoopFileSystem]. A functional filesystem has more than one DataNode, with data replicated across them.
- Files and blocks: The file are the data which is requested to be stored into HDFS. To achieve this, it is broken into blocks, the default size of each one is 128/256 MB in Hadoop 2.x.

- Block Replication: Each block is replicated for providing fault tolerance and availability, the default number of replicants is 3.

WIth this division, Hadoop achieves to store any data type on its ecosystem. Moreover, Apache YARN (Yet Another Resource Negotiator) is one of the major components of Apache Hadoop and it can be used [18], in order to plan the use of cluster and node resources as well as the treatments applied to the data, taking advantage of its own components:

- The Resource Manager: It controls the resource management of the cluster, also makes allocation decisions. The resource manager has two main components: Scheduler, allowing different policies for managing constraints such as capacity, fairness, and service level agreements and the Applications Manager, which is responsible for maintaining a list of submitted applications.
- The Node Manager: It is responsible for launching and managing containers on a node. Containers execute tasks as specified by the AppMaster which is an instance of a framework-specific library that negotiates resources from the Resource Manager and works with the NodeManager to execute and monitor the granted resources (bundled as containers) for a given application. An application can be a mapreduce job, or a spark executor and more.

Finally, the incoming field data is stored, splitted in nodes and ready to be further processed in the next layer.

Layer 3. Data Processing

Moving on, there is the data processing layer. Processes, such as classification, ML algorithms, curation, indexing, mining and more can be made in this layer. In past years, MapReduce was responsible for this procedure, but for sake of speed and faster processing, Search and Rescue will use Apache Spark.

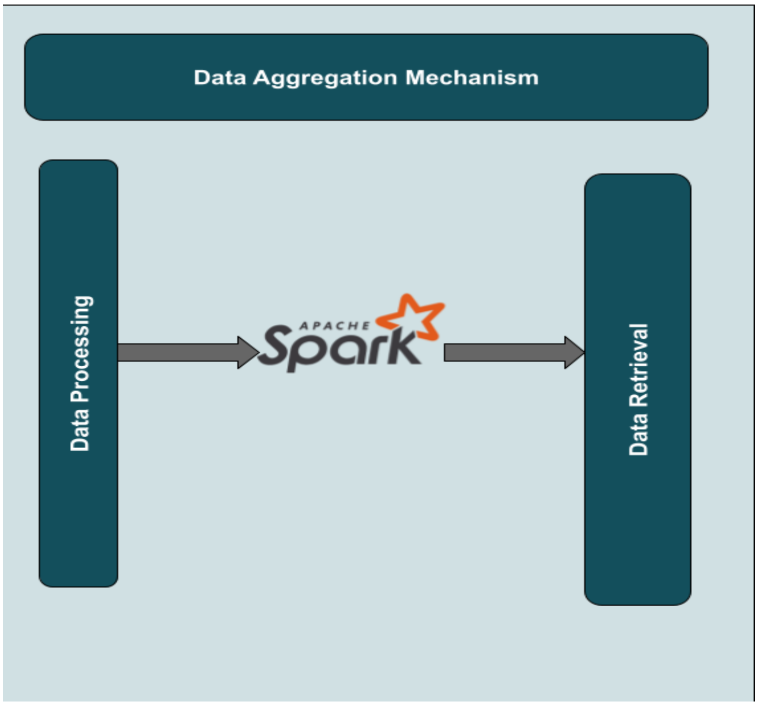


Figure 3-4: Data Processing Layer

Spark is a fast and general engine for large-scale data processing [17]. It empowers the system with flexibility, where scripts on Java, Scala and Python can be written, in order to do complex manipulations, transformations and analysis of the data. Spark has a rich ecosystem consisting of complicated things such as machine learning, data mining, graph analysis and streaming data.
As any Hadoop based technology, Spark follows the same pattern where there is a driver program which is a script that controls what is going to happen in the jobs. This is going through the cluster manager where it is used by Yarn (as described above), where the job will be distributed across the entire cluster, in order to start the parallel data processing. Each executor process has a sort of cache and task, responsible for how the task will be distributed to the system. For this reason, Spark is scalable and faster than Hadoop MapReduce. Moreover, Spark is based on the DAG engine (directed acyclic graph) which optimizes workflows [17].

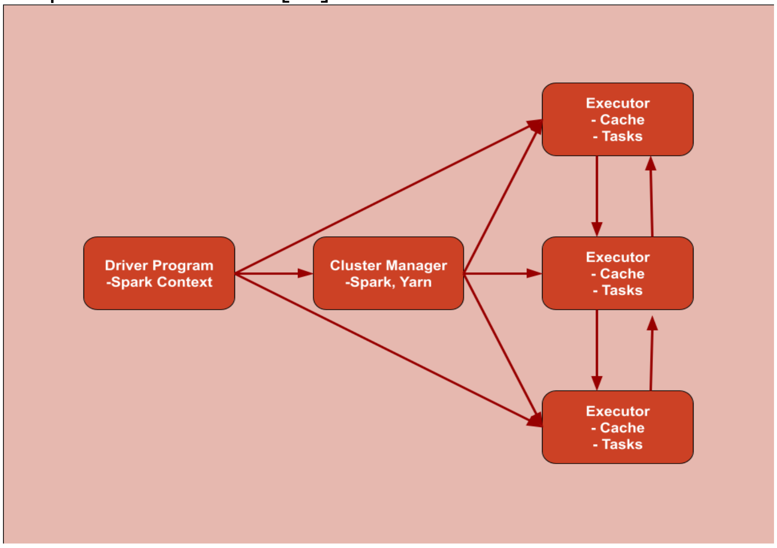


Figure 3-5: Apache Spark

There are three main abstractions in Apache Spark that can be used:

- Resilient Distributed Datasets (RDD): This is a fault-tolerant collection of elements that can be operated on in parallel. RDD is an immutable distributed collection of elements of the incoming data, partitioned across nodes in the cluster that can be operated in parallel with a low-level API that offers transformations and actions. There are 2 ways of creating RDD, either by parallelizing an existing collection in the driver program or by reading data from an external storage system, such as in S&R case, a shared file system, HDFS.
- Data frame: It is a distributed collection of data organized into named columns. It is conceptually equivalent to a table in a relational database or a data frame. Data frame allows to impose a structure onto a distributed collection of data, allowing higher-level abstractions. Moreover, the Data frames API was merged with Dataset, and as a result a Data frame is a Dataset organized into named columns. In addition, Data Frames can use schemas, which are giving shape to the data types (e.g integerType, etc) when needed. Data frames can be created from RDDs (which can be converted from Dstreams, as it is mentioned in the first layer). A Data frame can be read from a variety of data sources such as csv, jdbc, json,parquet, s3, and many others.
- Datasets: It is a collection of strongly typed domain-specific objects that can be transformed in parallel using functional or relational operations. A logical plan can be created for each transformation.

3.3 Installation Details

The data lake is installed in a Konnektable' s server in the current phase of the S&R project for development and testing purposes.

- Environment: DigitalOcean Droplet (VM) with 154.90 GB and 8 GB RAM capacity
- Deployment: Docker. Docker-images from BDE, which stands for Big Data Europe, a European Union's Horizon 2020 [7]. These docker images are under MIT license and were used only for the installation of the data lake's components (since Hadoop does not have any official docker images) that enable the aggregation mechanism:
- Components:
  - 1 HDFS namenode: This is the main node, which doesn't store data, but is responsible for the coordination of the rest of the nodes.
  - 1 HDFS datanode: This node is responsible for the actual storage of the data.
  - 1 YARN nodemanager: This manager is installed on every data node, and it executes the tasks on each one of them.
  - 1 YARN resource manager: This is the main node for processing. It passes parts of the requests to the corresponding node managers.
  - 1 MapReduce history server: The history server REST API allows the user to get the status on finished applications.
  - 1 Apache Spark spark-master: This creates a Spark standalone node, meaning a compiled version of Spark on each node on the cluster. Put simply, it creates an environment for Spark scripts to be executed on.
  - 1 Apache Spark spark-worker: This node executes the Spark scripts.
- Customizations:
  - docker-compose.yml adjustments on environment and volumes
  - pySpark files for the aggregation mechanisms
  - Shell scripts for the automation of the pySpark applications' execution
  - Shell scripts for setting up the HDFS repository structure for the first time
  - Shell scripts for copying the dummy data to the HDFS

- Testing:
  - Dummy data
    - Python scripts were created to generate a series of dummy data in the format of JSON (chemical data) and AVRO (smartwatch data) files.
    - Weather data in CSV format was found from online existing datasets.
    - Drone images and videos in PNG and MP4 format were found online.
    - All this data was then stored directly to the HDFS using the shell scripts mentioned above.

  - Spark provides a pySpark terminal to execute individual commands but the spark-submit.sh shell script was used to test Spark processing applications on the data.
    - Individual pySpark scripts were created for each category of the data, meaning historical, chemical, smartwatch, weather and ConCORDE data.

- These scripts were used and tested through the spark-submit.sh while also using the necessary packages for each one of them. (e.g. AVRO files need the "org.apache.spark:spark-avro_2.12:3.1." to be processed with Spark).

3.4 Data Aggregation Mechanism

Data aggregation is any process in which information is gathered and expressed in a summary form for purposes such as statistical analysis. A common **aggregation** purpose is to get the heterogeneous information from the field and convert it to homogeneous knowledge, in order to be used in the S&R operation through the system and services. The fundamental purpose of the data aggregation strategy is to aggregate and collect the data packets in an effective manner in order to improve the energy consumption, network lifetime, traffic bottleneck, and data accuracy. Data aggregation efficiency depends on the network design and the size of sensing data [20].

Apache Spark is used as the main aggregation mechanism for the processing stage and Hadoop for the storage stage. Spark processes data and stores them temporarily in memory, however this is not persistent storage. Hadoop is.

Hadoop already has a very powerful and efficient distributed parallel processing environment. On the other hand, the traditional Hadoop ecosystem with MapReduce programming paradigm is only suitable for batch processing. Therefore, Apache Spark was used for real-time processing while working with the powerful Hadoop environment. Spark has two main components; Spark streaming, that is responsible for taking real-time data in chunks and Spark engine, that immediately processes each data chunk at its generation by Spark streaming [21].

It is important to note that Spark engine was chosen as the processing tool of this stage of the S&R project due to its speed and because it allows the simultaneous performance of all its applications. It is known to achieve high performance for both Batch and Streaming data, which is crucial to this project since all the real-time data will be collected by streaming.

3.4.1 Parameters

In order to achieve the data aggregation, different types of parameters must be taken into account.

3.4.1.1 COncORDE

The Search and Rescue operation starts with CONCORDE which is a cloud-based platform with the role of sharing common knowledge and orchestrating the operation with its EMS features.

At an operational level, when an incident occurs, the Command Center will feed the CONCORDE system with initial details from the "caller", who is the person who detected the incident (e.g fire in a forest). The High Commander who is responsible for the Command Center will log into the CONCORDE system and fill the information report form with the details received from the caller. This information report has information about the incident, the type of the hazard, when the incident was created, the area type and more. This kind of information is needed, in order to trigger the aggregation mechanism with the scope to get all the relevant data from the sources and manipulate the historical data.

At a technological level, this information will be retrieved from the Apache Spark component through the CONCORDE API. The information form's format is in json as it is shown in the figure below.

```
{
    "data": {
        "id": 7,
        "hazardtype_set":"FIRE",
        "sitrep_set": [
            {
                "id": 1,
                "hazardtype_set":  "FIRE",
                "created": "2021-03-29T08:48:12.030583Z",
                "lastModified": "2021-03-29T08:48:12.030612Z",
                "incidentTimeOfAnnouncement": "2021-03-29T08:48:12.030621Z",
                "incidentStatusCode": "MAJOR_INCIDENT_ALERT_STANDBY",
                "typeOfEmergency": "VEHICLE_FIRE",
                "priorityDispatchCard": "ABDOMINAL_PAIN_PROBLEMS",
                "priorityDispatchCode": "IMMEDIATE",
                "description": "Vehicle fire close to forest",
                "additionalInformation": "Vehicle fire caused by a car crash accident",
                "hazardBackground": "Big danger of vehicle's explosion, entrapped person, fire spread to the forest",
                "numberOfAdultPatients": 2,
                "numberOfChildren": 1,
                "numberOfFatalities": 0,
                "name": "Vehicle Fire",
                "areaType": "WOODS",
                "location": "Thrakomakedones",
                "incidentId": 7
            }
        ],
```


Figure 3-6: Concorde's Incident Management Feature in json format

Spark retrieves the information from http://***.**.**.**:****/api/incidents/incident/{id} in a unified entry point of a Spark Session.

This information is temporarily saved in the cluster(spark-master). Then, the application reads the json file as a data frame and extracts the most useful information from it. Basically, this is the initial processing part which aims to present the data in such a way that makes it easier to get the most important information and provide it to the services that need it.
In the Figures below general information regarding the emergency and the patients' status is extracted from the ConCORDE form.

Summarized information from ConCORDE incident form about the emergency:

incident_id	hazard_type	description	status_code	hazard_background	area_type	location	timestamp
7	FIRE	Vehicle fire clos...	MAJOR_INCIDENT_AL...	Big danger of veh...	WOODS	Thrakomakedones	2021-03-29T08:48:...

Figure 3-7: ConCORDE incident form as a data frame

Summarized information from ConCORDE incident form about the patients:

incident_id	timestamp	location	adults	children	fatalities	notified_hospitals
7	2021-03-29T08:48:...	Thrakomakedones	2	1	0	[Agia Olga Hospit...

Figure 3-8: ConCORDE incident form's information about the patients

Notified hospitals: 2

notified_hospitals
Agia Olga Hospital
KAT Attica General Hospital

Figure 3-9: ConCORDE incident form' s information about hospitals

Since, this data frame includes structured data coming from the CONCORDE, there is no need for extra customizations, such as cleaning, indexing and more. This data frame will be renewed every time a CONCORDE user interacts with the information report and the situation report, which stands for a renewed information report form coming from the first responders on the field.

3.4.1.2 Real-time data from field data providers

Since S&R is a crisis management project it is only natural for all the data that concerns it to be delivered immediately after it is collected. That is also the definition of real time data, which is what this project relies on to produce the best results. Data from all the sensors on the field will be gathered, while the use cases are in action and they will be stored in the Data Lake or HDFS, as mentioned above. It is crucial to note that HDFS handles only the storage of the data whereas any kind of processing in this stage will be implemented by the Apache Spark framework for Python, also known as "pySpark".
Real-time data should be provided by the sensors on field in the time of action, which is not available until samples from field data providers are given for development and testing purposes and the use cases are implemented. Hence, for the purposes of this deliverable dummy data were created to support the presentation of the aggregation tools' functionality. It is important to note that this will not be the final form or format of the data that will be collected.
The data lake storage layer (Hadoop) will be accessed by a URL, such as https://XXX.XX.XX.XX:9870. From this URL one can find all the data concerning the project, by selecting the "Utilities" tab and then the "Browse the file system" button on the dropdown list.

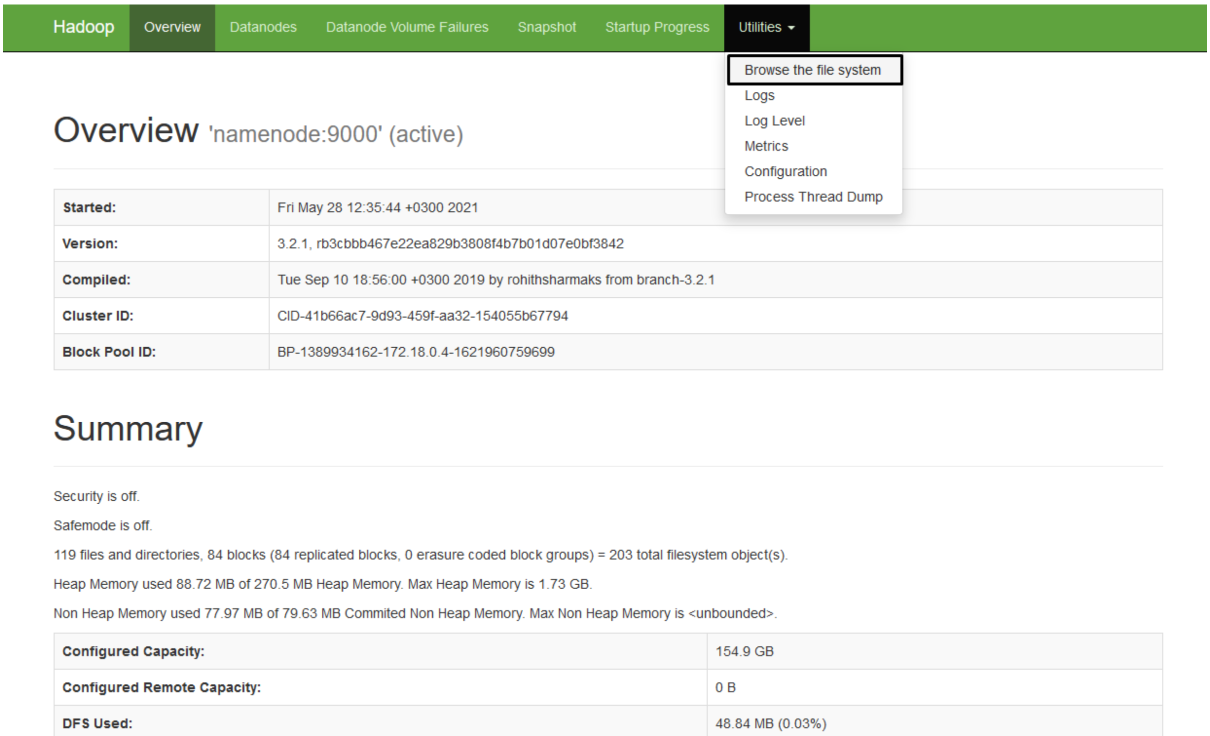
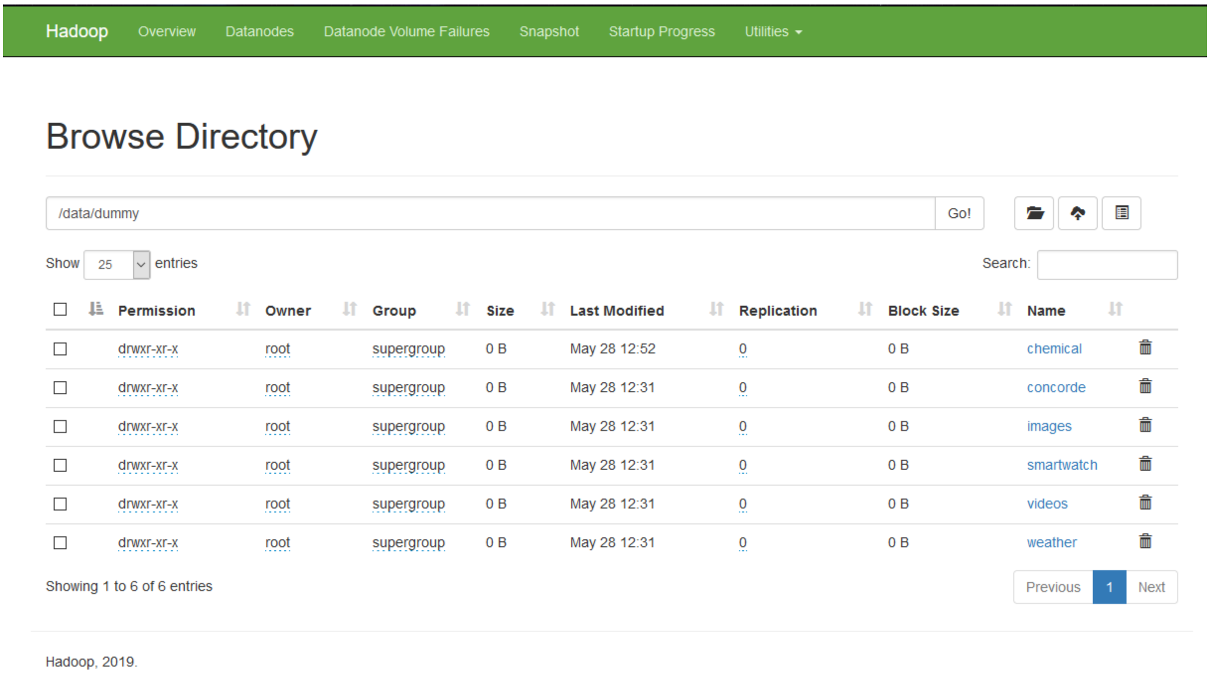


Figure 3-10: HDFS UI

The data is then displayed to the user archived in dedicated directories based on the source of information.



Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	root	supergroup	0 B	May 28 12:52	0	0 B	chemical
drwxr-xr-x	root	supergroup	0 B	May 28 12:31	0	0 B	concorde
drwxr-xr-x	root	supergroup	0 B	May 28 12:31	0	0 B	images
drwxr-xr-x	root	supergroup	0 B	May 28 12:31	0	0 B	smartwatch
drwxr-xr-x	root	supergroup	0 B	May 28 12:31	0	0 B	videos
drwxr-xr-x	root	supergroup	0 B	May 28 12:31	0	0 B	weather

Figure 3-11: HDFS directory structure

Having ConCORDE's information report (Figure x. Concorde's Incident Management Feature in json format) as a reference, we suppose that there are various sensors on the field. More specifically, dummy data were created to represent the chemical measurements, the patient state measurements, weather data, images and videos. Each of this data comes in their own format and individual pySpark applications are made to process it. Fundamentally, all pySpark applications will perform the same actions on the data but taking into consideration the volume of the data that will be sent in real time and the variety of the formats it will be sent in, it is better to divide the processing into many worker nodes and applications. These instances are also made, in case the T6.2 S&R model cannot handle all the data types (e.g videos).

The steps of the Spark processing are five in the current instance:

1. Read the data from the Data Lake (HDFS)
2. Convert the data into data frames
3. Aggregate all data frames into a single one
4. Process the data frame accordingly (extract specific information, perform metrics/functions)
5. Save the processed data in a file (preferably avro or json)

Let's suppose that there are n sensors on the field, and they all send information in a unique format.

<u>Chemical data from the chemical sensors</u>

A .json format was chosen for the chemical data. We suppose there are 15 chemical sensors on the field and an instance of the data they send could be the following:

```json
{
    "sensor_type": "chemical",
    "sensor_id": 1,
    "geolocation": {
        "x": 56.94,
        "y": -89.09,
        "z": 138.16
    },
    "drh": 27.63,
    "pm_25": 316.04,
    "temp": 24.6,
    "time": 1494787099,
    "hpa": 716.4,
    "dtemp": 25.2,
    "rh": 25.4,
    "pm_10": 66.0
}
```


Figure 3-12: Chemical data as json

The "chemical_data" pySpark application first converts it into a data frame, in order that all its values can be easily accessed and read. The form is the following:

```
The initial chemical data
+-----+-----+--------------------+-----+-----+------+----+---------+-----------+----+----------+
|  drh|dtemp|         geolocation|  hpa|pm_10| pm_25|  rh|sensor_id|sensor_type|temp|      time|
+-----+-----+--------------------+-----+-----+------+----+---------+-----------+----+----------+
|27.63| 25.2|{56.94, -89.09, 1...|716.4| 66.0|316.04|25.4|        1|   chemical|24.6|1494787099|
+-----+-----+--------------------+-----+-----+------+----+---------+-----------+----+----------+
```

Figure 3-13: Chemical data as a data frame

The second and most important function of Spark applications in this stage is the data aggregation. Apart from converting the data into a uniform agile format (data frame), they also unite all the data into a single data frame and by extension into a single file. In this example the data from all the chemical sensors is aggregated into one data frame and then stored in a json file. So, instead of having to access n different files to gather information about the data, S&R services can have access to one unified file where all the chemical data is stored.

The aggregated chemical data

sensor_id	sensor_type	geoloc_x	geoloc_y	geoloc_z	dtemp	temp	drh	rh	hpa	pm_10	pm_25	time
1	chemical	56.94	-89.09	138.16	25.2	24.6	27.63	25.4	716.4	66.0	316.04	1494787099
2	chemical	52.74	-106.32	134.4	30.0	29.8	29.32	26.4	716.2	68.3	319.71	1494856945
3	chemical	62.93	-100.62	136.23	29.3	28.8	27.25	24.0	714.1	61.6	312.85	1494878862
4	chemical	47.05	-107.52	141.28	25.5	25.2	29.48	28.7	715.3	69.1	312.65	1494859541
5	chemical	50.84	-96.27	140.55	26.1	25.6	27.64	27.2	719.3	69.4	310.96	1494806808
6	chemical	56.43	-91.94	142.33	28.8	28.1	28.29	26.6	713.3	61.5	312.97	1494826439
7	chemical	48.67	-105.0	138.14	26.5	26.0	28.05	28.4	719.0	60.3	313.07	1494874725
8	chemical	63.52	-100.59	151.82	26.2	25.9	27.24	29.2	710.2	69.5	317.16	1494870739
9	chemical	54.86	-101.35	143.52	27.5	27.0	29.07	28.4	716.4	66.0	318.84	1494801697
10	chemical	52.23	-106.24	143.37	28.9	27.9	29.15	28.2	718.2	66.3	316.43	1494814452
11	chemical	61.59	-99.19	143.75	25.7	24.9	29.03	28.6	714.4	61.1	314.82	1494862258
12	chemical	49.66	-99.85	142.0	26.1	25.5	29.74	26.0	712.3	62.5	318.67	1494875315
13	chemical	60.55	-103.39	134.84	26.3	25.5	29.66	27.0	716.9	65.0	310.72	1494791949
14	chemical	62.88	-91.78	145.13	27.5	26.7	27.47	26.0	714.3	64.0	318.93	1494866720
15	chemical	49.14	-98.78	146.42	25.2	24.5	29.41	25.0	712.8	65.9	314.64	1494802526

Figure 3-14: Aggregation of chemical data in a data frame

It is also possible to perform some initial processing of the data, after it has been converted into data frames. For this example, the average of each chemical component on a temperature greater than 27 degrees was calculated.

Average values of the chemicals when the Temperature is over 27 degrees

avg(drh)	avg(dtemp)	avg(hpa)	avg(pm_10)	avg(pm_25)	avg(rh)	avg(time)
28.502499999999998	29.25	715.45	64.425	315.49	26.299999999999997	1.4948441745E9

Figure 3-15: Averaging methods on a data frame

<u>Patient State data from the smartwatches</u>

An .avro format was chosen for the smartwatch data. We suppose there are 20 smartwatches on the field and the schema used to produce the avro files they send is shown below.

```
smartwatch_schema = """
{
    "name": "smartwatch",
    "type": "record",
    "fields": [
        {"name": "sensor_type", "type": "string"},
        {"name": "sensor_id", "type": "int"},
        {"name": "heart_rate", "type": "string"},
        {"name": "blood_pressure", "type": "string"},
        {"name": "Sp02", "type": "string"},
        {
            "name": "geolocation",
            "type": {
                        "type" : "record",
                        "name" : "geolocation",
                        "fields" : [
                            {"name": "x", "type": "double"},
                            {"name": "y", "type": "double"},
                            {"name": "z", "type": "double"}
                        ]
                    }
        }
    ]
}"""
```

Figure 3-16: Avro Schema for smartwatch dummy data

The dedicated "smartwatch_data" pySpark-application converts each avro file into a data frame and then aggregates all the data into a single data frame.
Initially, the avro files are read, as following:

sensor_type	sensor_id	heart_rate	blood_pressure	Sp02	geoloc_x	geoloc_y	geoloc_z
smartwatch	10	92bpm	134/74mmHg	87%	70.6	-79.65	158.36

Figure 3-17: Smartwatch data as a data frame (initially avro)

And then an aggregation process is performed on all smartwatch data:

```
The aggregated smartwatch data
+---------+-----------+--------+--------+--------+----------+--------------+----+
|sensor_id|sensor_type|geoloc_x|geoloc_y|geoloc_z|heart_rate|blood_pressure|Sp02|
+---------+-----------+--------+--------+--------+----------+--------------+----+
|        1| smartwatch|    70.6|  -79.65|  158.36|     95bpm|     125/84mmHg| 89%|
|        2| smartwatch|    70.6|  -79.65|  158.36|     82bpm|     139/74mmHg| 91%|
|        3| smartwatch|    70.6|  -79.65|  158.36|     77bpm|     136/85mmHg| 89%|
|        4| smartwatch|    70.6|  -79.65|  158.36|    108bpm|     121/67mmHg| 92%|
|        5| smartwatch|    70.6|  -79.65|  158.36|     74bpm|     128/76mmHg| 83%|
|        6| smartwatch|    70.6|  -79.65|  158.36|    113bpm|     137/77mmHg| 85%|
|        7| smartwatch|    70.6|  -79.65|  158.36|    114bpm|     123/77mmHg| 94%|
|        8| smartwatch|    70.6|  -79.65|  158.36|     93bpm|     126/67mmHg| 83%|
|        9| smartwatch|    70.6|  -79.65|  158.36|     79bpm|     133/85mmHg| 96%|
|       10| smartwatch|    70.6|  -79.65|  158.36|     92bpm|     134/74mmHg| 87%|
|       11| smartwatch|    70.6|  -79.65|  158.36|    118bpm|     125/72mmHg| 96%|
|       12| smartwatch|    70.6|  -79.65|  158.36|     92bpm|     123/88mmHg| 97%|
|       13| smartwatch|    70.6|  -79.65|  158.36|     80bpm|     136/90mmHg| 93%|
|       14| smartwatch|    70.6|  -79.65|  158.36|    108bpm|     120/78mmHg| 97%|
|       15| smartwatch|    70.6|  -79.65|  158.36|     79bpm|     129/82mmHg| 82%|
|       16| smartwatch|    70.6|  -79.65|  158.36|     84bpm|     137/85mmHg| 87%|
|       17| smartwatch|    70.6|  -79.65|  158.36|     98bpm|     136/78mmHg| 85%|
|       18| smartwatch|    70.6|  -79.65|  158.36|     78bpm|     131/75mmHg| 82%|
|       19| smartwatch|    70.6|  -79.65|  158.36|    107bpm|     127/80mmHg| 87%|
|       20| smartwatch|    70.6|  -79.65|  158.36|     82bpm|     122/80mmHg| 87%|
+---------+-----------+--------+--------+--------+----------+--------------+----+
```

Figure 3-18: Aggregation of smartwatch data in a data frame

The aggregated smartwatch data frame is then stored as a new processed avro file.

<u>Weather data</u>

A .csv format was chosen for the weather data. Weather data does not come from a sensor so only one file is enough to provide enough information.
An instance of the data could be this.

	A	B	C	D	E	F	G	H
1	Station,GEO Location,Local Time,Conditions,Temperature,Pressure,Humidity							
2	BEN,33.9435602229-117.641409054,2012-09-27 08:04:55,Snow,-5.2,949.5,59							
3	HOB,33.9435602245-117.640761244,2015-11-02 14:18:00,Sunny,39.6,859.5,56							
4	CAN,33.9435602262-117.640113434,2009-12-07 16:31:53,Rain,21.9,1023.9,56							
5	DAR,33.9435602278-117.639465625,2011-06-29 08:31:23,Rain,18.2,959.3,57							
6	SYD,33.9435602295-117.638817815,2011-12-18 14:27:15,Sunny,13.3,1168.3,61							
7	MEL,33.9435602312-117.638170006,2016-12-11 18:38:29,Snow,-5.4,944.9,61							
8	ADL,33.9435602328-117.637522196,2011-04-21 11:37:51,Sunny,36.6,960.9,69							
9	ALB,33.9435602345-117.636874387,2008-06-10 20:11:15,Snow,-5.9,1008.1,65							
10	BRB,33.9435602362-117.636226577,2014-03-27 01:45:01,Sunny,37.1,1160.8,57							

Figure 3-19: Weather data csv

The dedicated “weather_data” pySpark-application converts the csv file into a data frame first. A declaration of the CSV file’ s separators and the existence of headers has to be made for the file to be read correctly.

The initial weather data

Station	GEO Location	Local Time	Conditions	Temperature	Pressure	Humidity
BEN	33.9435602229-117...	2012-09-27 08:04:55	Snow	-5.2	949.5	59.0
HOB	33.9435602245-117...	2015-11-02 14:18:00	Sunny	39.6	859.5	56.0
CAN	33.9435602262-117...	2009-12-07 16:31:53	Rain	21.9	1023.9	56.0
DAR	33.9435602278-117...	2011-06-29 08:31:23	Rain	18.2	959.3	57.0
SYD	33.9435602295-117...	2011-12-18 14:27:15	Sunny	13.3	1168.3	61.0
MEL	33.9435602312-117...	2016-12-11 18:38:29	Snow	-5.4	944.9	61.0
ADL	33.9435602328-117...	2011-04-21 11:37:51	Sunny	36.6	960.9	69.0
ALB	33.9435602345-117...	2008-06-10 20:11:15	Snow	-5.9	1008.1	65.0
BRB	33.9435602362-117...	2014-03-27 01:45:01	Sunny	37.1	1160.8	57.0
PER	33.9435602378-117...	2015-02-24 02:28:01	Sunny	36.0	1192.1	59.0
BEN	33.9435602395-117...	2010-06-05 01:59:31	Snow	-3.8	1080.8	61.0
HOB	33.9435602412-117...	2008-09-01 02:16:26	Snow	-6.5	787.9	69.0
CAN	33.9435602428-117...	2014-06-18 20:06:27	Snow	-3.6	941.9	60.0
DAR	33.9435602445-117...	2013-12-24 08:46:39	Sunny	39.3	1175.8	64.0
SYD	33.9435602461-117...	2016-05-27 20:31:18	Rain	16.0	753.9	66.0
MEL	33.9435602478-117...	2014-03-16 14:05:30	Rain	22.6	713.7	60.0
ADL	33.9435602495-117...	2012-11-28 05:07:28	Sunny	16.6	1069.4	56.0
ALB	33.9435602511-117...	2010-05-22 17:21:08	Snow	-5.8	1140.0	66.0
BRB	33.9435602528-117...	2014-07-01 00:23:03	Sunny	10.1	1028.2	60.0
PER	33.9435602545-117...	2010-08-22 22:10:39	Snow	-4.7	833.4	63.0

only showing top 20 rows

Figure 3-20: Weather data as a data frame

Now it can perform some aggregating functions on it to extract some knowledge from it.

Average values of Humidity and Pressure when it's raining

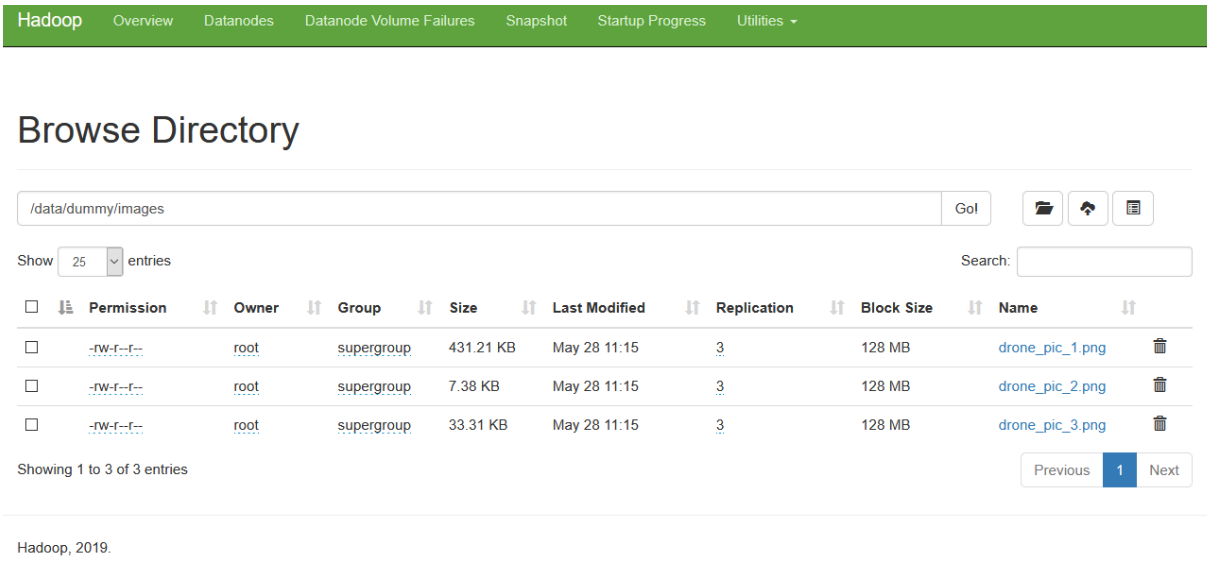
avg(Pressure)	avg(Humidity)
950.6460147174118	63.006656761467994

Figure 3-21: Averaging method on data frame

This data frame is then stored as a new processed csv file.

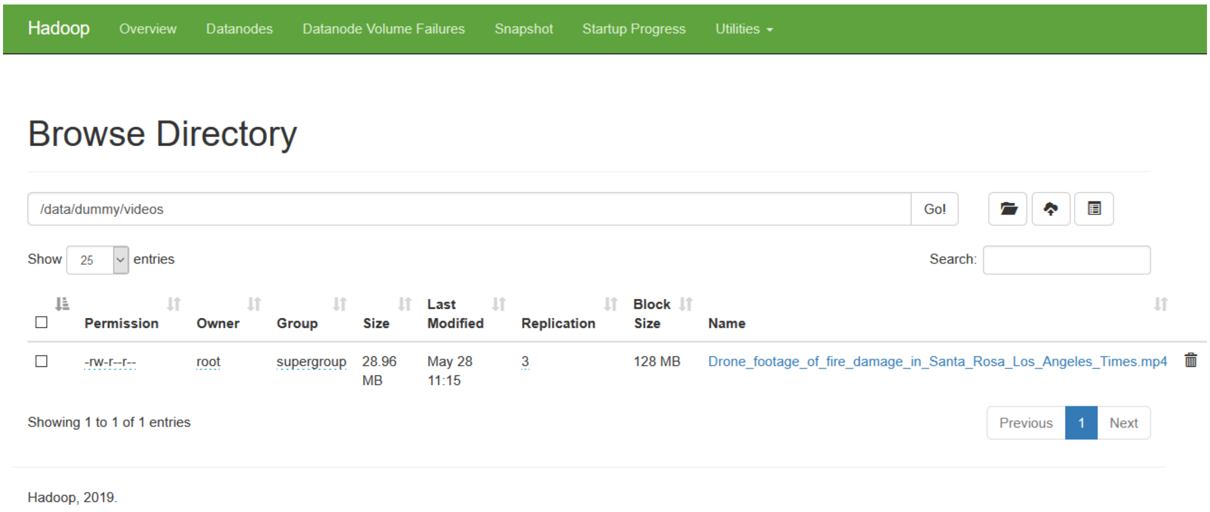
<u>Images and Videos</u>

HDFS can store both structured and unstructured data, meaning images and videos. Image and video storage are crucial to the S&R project since there are tasks that perform Deep Learning algorithms for object(obstacle) detection for in-disaster scene situation awareness on them. For that reason, a dedicated node will be made in HDFS so that services in need of images or videos from the field can have access to it and process them and give the requested detection output back to the data lake ecosystem. More details on the interconnection will be released on the specific deliverables of the dedicated tasks (T3.3 and T3.4), as an outcome of WP3.



Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	root	supergroup	431.21 KB	May 28 11:15	3	128 MB	drone_pic_1.png
-rw-r--r--	root	supergroup	7.38 KB	May 28 11:15	3	128 MB	drone_pic_2.png
-rw-r--r--	root	supergroup	33.31 KB	May 28 11:15	3	128 MB	drone_pic_3.png

Figure 3-22: HDFS images directory



Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	root	supergroup	28.96 MB	May 28 11:15	3	128 MB	Drone_footage_of_fire_damage_in_Santa_Rosa_Los_Angeles_Times.mp4

Figure 3-23: HDFS videos directory

3.4.1.3 Historical Data from existing DBs

Meanwhile, there are services such as SOT and Physio DSS, where historical data from existing databases are needed to train their algorithms and models.These historical data can be called through urls, as it was mentioned in the previous chapters of D4.8. For the aggregation instance, the dockerized databases from NTUA were running in the Digital Ocean Droplet. Apache Spark will be used to process and store temporarily the data in memory, in order to request specific knowledge from these existing databases.
More specifically, continuing the aggregation instance from chapter x.2.1 CONCORDE, existing databases where the emergency type enum equals "fire" will be triggered when concorde.json gives to the system the hazard_type of "fire". A dedicated spark application has been developed for this reason.
Since, the existing databases have too many rows with irrelevant data for the emergency type of fire, a reduce method must be developed. This approach will save valuable time by facilitating the system to run only the relevant rows to the incident.

emergencyTypeEnum	description	database	openYear	openMonth	openDate	magnitude	vehicleTYpe	locationType	country	injuredPeopleNumber	damages
FIRE	Ottawa ON and Hul...	Canadian Disaster...	1900	4	1900-04-26	0.0		Non-Residential	Canada	0	0.0
FIRE	Grand Forks BC, 1...	Canadian Disaster...	1901	1	1901-01-01	0.0		Non-Residential	Canada	0	0.0
FIRE	Cranberry BC, 190...	Canadian Disaster...	1902	1	1902-01-01	0.0		Non-Residential	Canada	0	0.0
FLOOD	Saint John River ...	Canadian Disaster...	1902	3	1902-03-01	0.0			Canada	0	0.0
FIRE	Fernie BC, April ...	Canadian Disaster...	1902	5	1902-05-22	0.0		Non-Residential	Canada	0	0.0
FIRE	Grand Forks BC, 1...	Canadian Disaster...	1903	1	1903-01-01	0.0		Non-Residential	Canada	0	0.0
LANDSLIDE	Frank AB, April 2...	Canadian Disaster...	1903	4	1903-04-29	0.0			Canada	23	0.0
FIRE	Carbonado BC, 190...	Canadian Disaster...	1904	1	1904-01-01	0.0		Non-Residential	Canada	0	0.0
STORM	Edmonton AB, Sept...	Canadian Disaster...	1992	9	1992-09-01	0.0			Canada	0	2.2522E7
STORM	Calgary AB, July ...	Canadian Disaster...	1995	7	1995-07-17	0.0			Canada	0	7.4559612E7
LANDSLIDE	Upper Arrow Lake ...	Canadian Disaster...	1903	2	1903-02-28	0.0			Canada	0	0.0
FIRE	Toronto ON, April...	Canadian Disaster...	1904	4	1904-04-19	0.0		Non-Residential	Canada	0	0.0
FLOOD	Red River MB, Apr...	Canadian Disaster...	1904	4	1904-04-24	0.0			Canada	0	0.0
STORM	Nova Scotia, Febr...	Canadian Disaster...	1905	2	1905-02-15	0.0			Canada	0	0.0
LANDSLIDE	Spences Bridge BC...	Canadian Disaster...	1905	8	1905-08-13	0.0			Canada	0	0.0
TRANSPORTATION	British Columbia,...	Canadian Disaster...	1906	1	1906-01-01	0.0	Marine		Canada	0	0.0
TRANSPORTATION	Quebec City QC, A...	Canadian Disaster...	1907	8	1907-08-29	0.0			Canada	0	0.0
LANDSLIDE	Notre-Dame-de-la-...	Canadian Disaster...	1908	4	1908-04-26	0.0			Canada	0	0.0
FIRE	Fernie BC, August...	Canadian Disaster...	1908	8	1908-08-01	0.0		Residential	Canada	0	0.0
FLOOD	Chester NB, Janua...	Canadian Disaster...	1909	1	1909-01-04	0.0			Canada	0	0.0

Figure 3-24: Initial outcomes of the canadian_disaster_database

As it is shown in the figure above, these historical data refer to emergency types such as landslide, flood, fire, transportation. Since, the requested type of emergency is fire, it is crucial to only collect the information relevant to the system.

database	emergencyTypeEnum	locationType	country	injuredPeopleNumber	casualties
Canadian Disaster...	FIRE	Non-Residential	Canada	0	7
Canadian Disaster...	FIRE	Non-Residential	Canada	0	64
Canadian Disaster...	FIRE	Non-Residential	Canada	0	32
Canadian Disaster...	FIRE	Non-Residential	Canada	0	125
Canadian Disaster...	FIRE	Non-Residential	Canada	0	16
Canadian Disaster...	FIRE	Non-Residential	Canada	0	14
Canadian Disaster...	FIRE	Non-Residential	Canada	0	0
Canadian Disaster...	FIRE	Residential	Canada	0	100
Canadian Disaster...	FIRE		Canada	0	73
Canadian Disaster...	FIRE	Non-Residential	Canada	48	189
Canadian Disaster...	FIRE	Non-Residential	Canada	0	22
Canadian Disaster...	FIRE	Residential	Canada	0	7
Canadian Disaster...	FIRE		Canada	0	233
Canadian Disaster...	FIRE	Non-Residential	Canada	0	34
Canadian Disaster...	FIRE	Non-Residential	Canada	0	65
Canadian Disaster...	FIRE	Non-Residential	Canada	11	88
Canadian Disaster...	FIRE	Non-Residential	Canada	0	0
Canadian Disaster...	FIRE		Canada	0	0
Canadian Disaster...	FIRE		Canada	0	0
Canadian Disaster...	FIRE		Canada	0	0

Figure 3-25: Filtering EmergencyTypeEnum "fire" on canadian_disaster_database

The figure above is what the data frame is reading by conditioning the fire emergency. In this instance, the table demonstrates specific columns such as: databases, emergencyTypeEnum, locationType, country, injuredPeopleNumber and casualties. Additional adjustments can be made, such as returning this data frame with all the casualties that are not equal to zero.

database	emergencyTypeEnum	locationType	country	injuredPeopleNumber	casualties
Canadian Disaster...	FIRE	Non-Residential	Canada	0	7
Canadian Disaster...	FIRE	Non-Residential	Canada	0	64
Canadian Disaster...	FIRE	Non-Residential	Canada	0	32
Canadian Disaster...	FIRE	Non-Residential	Canada	0	125
Canadian Disaster...	FIRE	Non-Residential	Canada	0	16
Canadian Disaster...	FIRE	Non-Residential	Canada	0	14
Canadian Disaster...	FIRE	Residential	Canada	0	100
Canadian Disaster...	FIRE		Canada	0	73
Canadian Disaster...	FIRE	Non-Residential	Canada	48	189
Canadian Disaster...	FIRE	Non-Residential	Canada	0	22
Canadian Disaster...	FIRE	Residential	Canada	0	7
Canadian Disaster...	FIRE		Canada	0	233
Canadian Disaster...	FIRE	Non-Residential	Canada	0	34
Canadian Disaster...	FIRE	Non-Residential	Canada	0	65
Canadian Disaster...	FIRE	Non-Residential	Canada	11	88
Canadian Disaster...	FIRE	Non-Residential	Canada	0	26
Canadian Disaster...	FIRE	Non-Residential	Canada	0	2
Canadian Disaster...	FIRE	Non-Residential	Canada	5	2
Canadian Disaster...	FIRE	Non-Residential	Canada	58	2
Canadian Disaster...	FIRE		Canada	0	1

Figure 3-26: Filtering emergencyTypeEnum "fire" and no casualties on canadian_disaster_database

The same procedure happens with the rest of the databases. In the figures below, there are results of the Hellenic Fire Service (Forest) and Hellenic Fire Service (Residential).

This is the first data frame created by the fire_service_forest database.

database	emergencyTypeEnum	country	locationType	damagedLand	fireFighters
Hellenic Fire Ser...	FIRE	Greece	Non-residential	6.0	0
Hellenic Fire Ser...	FIRE	Greece	Non-residential	8.0	0
Hellenic Fire Ser...	FIRE	Greece	Non-residential	4.0	0
Hellenic Fire Ser...	FIRE	Greece	Non-residential	2.0	0
Hellenic Fire Ser...	FIRE	Greece	Non-residential	8.0	0
Hellenic Fire Ser...	FIRE	Greece	Non-residential	3.0	0
Hellenic Fire Ser...	FIRE	Greece	Non-residential	2.0	0
Hellenic Fire Ser...	FIRE	Greece	Non-residential	4.0	0
Hellenic Fire Ser...	FIRE	Greece	Non-residential	1000.0	0
Hellenic Fire Ser...	FIRE	Greece	Non-residential	1.0	0
Hellenic Fire Ser...	FIRE	Greece	Non-residential	11.0	0
Hellenic Fire Ser...	FIRE	Greece	Non-residential	1400.0	0
Hellenic Fire Ser...	FIRE	Greece	Non-residential	5.0	0
Hellenic Fire Ser...	FIRE	Greece	Non-residential	10.0	0
Hellenic Fire Ser...	FIRE	Greece	Non-residential	6.0	0
Hellenic Fire Ser...	FIRE	Greece	Non-residential	4.0	0
Hellenic Fire Ser...	FIRE	Greece	Non-residential	10.0	0
Hellenic Fire Ser...	FIRE	Greece	Non-residential	5.0	0
Hellenic Fire Ser...	FIRE	Greece	Non-residential	2.0	0
Hellenic Fire Ser...	FIRE	Greece	Non-residential	2.0	0

Figure 3-27: Example from hellenic fire_service_forest

And this is a more specific data frame that returns the fireFighters who were involved in the past incidents.

database	emergencyTypeEnum	country	locationType	damagedLand	fireFighters
Hellenic Fire Ser...	FIRE	Greece	Non-residential	1.0	4
Hellenic Fire Ser...	FIRE	Greece	Non-residential	1.5	10
Hellenic Fire Ser...	FIRE	Greece	Non-residential	5.0	8
Hellenic Fire Ser...	FIRE	Greece	Non-residential	4.0	37
Hellenic Fire Ser...	FIRE	Greece	Non-residential	0.3	2
Hellenic Fire Ser...	FIRE	Greece	Non-residential	0.0	10
Hellenic Fire Ser...	FIRE	Greece	Non-residential	0.5	15
Hellenic Fire Ser...	FIRE	Greece	Non-residential	20.0	38
Hellenic Fire Ser...	FIRE	Greece	Non-residential	450.0	95
Hellenic Fire Ser...	FIRE	Greece	Non-residential	0.8	16
Hellenic Fire Ser...	FIRE	Greece	Non-residential	2.0	8
Hellenic Fire Ser...	FIRE	Greece	Non-residential	0.5	4
Hellenic Fire Ser...	FIRE	Greece	Non-residential	5.0	17
Hellenic Fire Ser...	FIRE	Greece	Non-residential	0.0	2
Hellenic Fire Ser...	FIRE	Greece	Non-residential	0.2	9
Hellenic Fire Ser...	FIRE	Greece	Non-residential	0.0	6
Hellenic Fire Ser...	FIRE	Greece	Non-residential	30.0	7
Hellenic Fire Ser...	FIRE	Greece	Non-residential	5.0	24
Hellenic Fire Ser...	FIRE	Greece	Non-residential	0.0	9
Hellenic Fire Ser...	FIRE	Greece	Non-residential	0.1	10

Figure 3-28: Example from hellenic fire_service_forest with the involved fireFighters resources on the incidents

Moreover, the selected hellenic_fire_service_residential database instance gives the outputs below.

database	emergencyTypeEnum	locationType	severity	casualties	injuredPeopleNumber	involvedPeopleNumber
Hellenic Fire Ser...	FIRE	Residential	MINOR	0	0	1
Hellenic Fire Ser...	FIRE	Residential	MINOR	0	0	1
Hellenic Fire Ser...	FIRE	Residential	MINOR	0	0	1
Hellenic Fire Ser...	FIRE	Residential	MINOR	0	0	1
Hellenic Fire Ser...	FIRE	Residential	MINOR	0	0	1
Hellenic Fire Ser...	FIRE	Residential	MINOR	0	0	1
Hellenic Fire Ser...	FIRE	Residential	MINOR	0	0	1
Hellenic Fire Ser...	FIRE	Residential	MINOR	0	0	1
Hellenic Fire Ser...	FIRE	Residential	MINOR	0	0	1
Hellenic Fire Ser...	FIRE	Residential	MINOR	0	0	1
Hellenic Fire Ser...	FIRE	Residential	MINOR	0	0	1
Hellenic Fire Ser...	FIRE	Residential	MINOR	0	0	1
Hellenic Fire Ser...	FIRE	Residential	MINOR	0	0	1
Hellenic Fire Ser...	FIRE	Residential	MINOR	0	0	1
Hellenic Fire Ser...	FIRE	Residential	MINOR	0	0	1
Hellenic Fire Ser...	FIRE	Residential	MINOR	0	0	1
Hellenic Fire Ser...	FIRE	Residential	MINOR	0	0	1
Hellenic Fire Ser...	FIRE	Residential	MINOR	0	0	1
Hellenic Fire Ser...	FIRE	Residential	MINOR	0	0	1
Hellenic Fire Ser...	FIRE	Residential	MINOR	0	0	1

Figure 3-29: Example from hellenic fire_service_residential

And the below figure reads the data frame with casualties different than zero on the incident.

database	emergencyTypeEnum	locationType	severity	casualties	injuredPeopleNumber	involvedPeopleNumber
Hellenic Fire Ser...	FIRE	Residential	MAJOR	2	0	2
Hellenic Fire Ser...	FIRE	Residential	MINOR	1	0	1
Hellenic Fire Ser...	FIRE	Residential	MAJOR	1	0	1
Hellenic Fire Ser...	FIRE	Residential	MINOR	1	0	1
Hellenic Fire Ser...	FIRE	Residential	MEDIUM	1	0	1
Hellenic Fire Ser...	FIRE	Residential	MINOR	1	0	1
Hellenic Fire Ser...	FIRE	Residential	MINOR	1	0	1
Hellenic Fire Ser...	FIRE	Residential	MINOR	1	1	1
Hellenic Fire Ser...	FIRE	Residential	MINOR	1	1	1
Hellenic Fire Ser...	FIRE	Residential	MINOR	1	0	1
Hellenic Fire Ser...	FIRE	Residential	MAJOR	3	0	3
Hellenic Fire Ser...	FIRE	Residential	MAJOR	3	0	3
Hellenic Fire Ser...	FIRE	Residential	MEDIUM	1	0	1
Hellenic Fire Ser...	FIRE	Residential	MEDIUM	1	0	1
Hellenic Fire Ser...	FIRE	Residential	MEDIUM	1	0	1
Hellenic Fire Ser...	FIRE	Residential	MEDIUM	1	0	1
Hellenic Fire Ser...	FIRE	Residential	MINOR	1	0	1
Hellenic Fire Ser...	FIRE	Residential	MINOR	1	0	1
Hellenic Fire Ser...	FIRE	Residential	MAJOR	1	1	2
Hellenic Fire Ser...	FIRE	Residential	MINOR	1	0	1

Figure 3-30: Example from hellenic fire_service_residential with casualties on the incidents

At this point, it is essential to aggregate the previous data frames into a common one. The figure below demonstrates the aggregation of the first two data frames.

database	emergencyTypeEnum	locationType	country	injuredPeopleNumber	casualties	damagedLand

Figure 3-31: Aggregation of the first two databases (Columns)

Canadian Disaster...	FIRE	Residential	Canada	15	32	null
Canadian Disaster...	FIRE		Canada	0	1	null
Canadian Disaster...	FIRE	Residential	Canada	0	9	null
Canadian Disaster...	FIRE		Canada	0	2	null
Hellenic Fire Ser...	FIRE	Non-residential	Greece	null	null	6.0
Hellenic Fire Ser...	FIRE	Non-residential	Greece	null	null	8.0
Hellenic Fire Ser...	FIRE	Non-residential	Greece	null	null	4.0
Hellenic Fire Ser...	FIRE	Non-residential	Greece	null	null	2.0
Hellenic Fire Ser...	FIRE	Non-residential	Greece	null	null	8.0
Hellenic Fire Ser...	FIRE	Non-residential	Greece	null	null	3.0

Figure 3-32: Aggregation of the first two databases (values)

After this, the aggregated data frame will finally aggregate with the third database.

database	emergencyTypeEnum	locationType	country	injuredPeopleNumber	casualties	damagedLand	severity	involvedPeopleNumber

Figure 3-33: Aggregation of the first two databases with the third database

The final common data frame will consist of all the columns and given values of the 3 databases. This aggregation mechanism will constitute a common data frame that will be used from the DSS services, in order to train their models. This approach is faster than sending every data frame separately to the requested services. The "null" values mean that the selected database does not have that type of values in its initial form. They can easily transform into "0".

These figures show a simple instance of the aggregating mechanism for S&R's historical data. The responsible partners for the DSS development (CNR and Konnektable) will indicate the requested fields ('columns' of the existing databases and other aggregations, such as average, filtered, max, min values) that they are going to use for their models training. This will be an outcome of "T4.4 Design of DSS components" in M14. The selection of the field will save time and it will be crucial to eliminate every not related information to emergency types of S&R's Use Cases. However, there is always the option to send all the fields of the existing databases.

The figure below shows that all the applications made for this deliverable run at the same time.

▾ **Running Applications (5)**

Application ID		Name	Cores	Memory per Executor
app-20210602085246-0007	(kill)	smartwatch_data	0	1024.0 MiB
app-20210602085213-0006	(kill)	weather_data	0	1024.0 MiB
app-20210602085151-0005	(kill)	historical_data	0	1024.0 MiB
app-20210602085110-0003	(kill)	concorde_data	0	1024.0 MiB
app-20210602085024-0002	(kill)	chemical_data	8	1024.0 MiB

Figure 3-34: Spark UI: Running Applications

This figure reveals that the data aggregation mechanism run simultaneously, in order to give the outputs that were shown in the figures of the data aggregation mechanism chapter.

3.5 Next Steps

The data aggregation mechanism for the Search and Rescue project is settled and ready to be further developed depending on the data that the other components want to use.

The next technical steps are divided into:

- Data retrieval from COncORDE platform of all the data used in the instance
- Demonstration of the dummy data on the COncORDE Dashboard as notifications
- Sample data from the real-time data providers
- Collection of the required knowledge (outcomes) from the technical partners who want to further use the data in their applications
- Give access to the system for partners with detections algorithms
- Connection with data bus and S&R data model for real-time data ingestion and processing
- Security on the system
- Testing with real time data or Mock API Server Online Testing

4 Conclusions

To conclude, this deliverable describes the data aggregation mechanisms for both historical and real-time data. These are crucial for the knowledge and information elements of the Search and Rescue Project.

In the first section of D4.8 deliverable, seven resources have been chosen to aggregate data from, according to D4.1’s requirements. The aggregated data were then transformed and served through an API to be used as part of a collective access point for the execution of experimental knowledge discovery algorithms. More specifically, the output of T4.1 can be used by SnR’s Decision Making components and lessons learned mechanisms.

Moreover, T4.2 Data aggregation is a technical task that has to closely follow the related technical and operational tasks. Therefore, it will be updated regularly, in order to give the final and accurate data aggregation mechanism to the Search and Rescue Project. More specifically, in order to be homogeneous different parameters must be taken into account from the data sources, data models and integration mechanisms, to operational requirements derived by the dedicated Work Packages. Finally, the aggregated knowledge will be consumed by the whole system.

Annex I: References

[1] "The Canadian Disaster Database," [Online]. Available: https://www.publicsafety.gc.ca/cnt/rsrcs/cndn-dsstr-dtbs/index-en.aspx. [Accessed 31 05 2021].

[2] "EM-DAT | The international disasters database," [Online]. Available: https://www.emdat.be/. [Accessed 31 05 2021].

[3] "The International Disaster - Emergency Events Database (EM-DAT) | Knowledge for policy," [Online]. Available: https://knowledge4policy.ec.europa.eu/dataset/ds00107_en. [Accessed 31 05 2021].

[4] "Global Landslide Catalog Export | NASA Open Data Portal," [Online]. Available: https://data.nasa.gov/Earth-Science/Global-Landslide-Catalog-Export/dd9e-wu2v. [Accessed 31 05 2021].

[5] "Global Landslide Catalog | NASA Open Data Portal," [Online]. Available: https://data.nasa.gov/Earth-Science/Global-Landslide-Catalog/h9d8-neg4. [Accessed 31 05 2021].

[6] D. B. Kirschbaum, R. Adler, Y. Hong, S. Hill, and A. Lerner-Lam, "A global landslide catalog for hazard applications: Method, results, and limitations," *Nat. Hazards,* 2010.

[7] D. Kirschbaum, T. Stanley, and Y. Zhou, "Spatial and temporal analysis of a global landslide catalog," 2015.

[8] "Hellenic Fire Service," [Online]. Available: https://www.fireservice.gr/el_GR/synola-dedomenon. [Accessed 01 06 2021].

[9] "USGS Earthquake Hazards Program," [Online]. Available: https://earthquake.usgs.gov/. [Accessed 01 06 2021].

[10] "Spreadsheet Format," [Online]. Available: https://earthquake.usgs.gov/earthquakes/feed/v1.0/csv.php. [Accessed 01 06 2021].

[11] "Global Internal Displacement Database | IDMC," [Online]. Available: https://www.internal-displacement.org/database/global-displacement-risk-model. [Accessed 01 06 2021].

[12] "Welcome to Python.org," [Online]. Available: https://www.python.org/. [Accessed 02 06 2021].

[13] "FastAPI," [Online]. Available: https://fastapi.tiangolo.com/. [Accessed 02 06 2021].

[14] "Selenium with Python - Selenium Python Bindings 2 documentation," [Online]. Available: https://selenium-python.readthedocs.io/. [Accessed 02 06 2021].

[15] "Empowering App Development for Developers | Docker," [Online]. Available: https://www.docker.com/. [Accessed 02 06 2021].

[16] Singh K. and Kaur R., "Hadoop: addressing challenges of big data," in *IEEE International Advance Computing Conference (IACC) (pp. 686-689)*, 2014.

[17] "Spark Release 3.1.1 | Apache Spark," [Online]. Available: https://spark.apache.org/releases/spark-release-3-1-1.html. [Accessed 04 06 2021].

[18] "Hadoop - Apache Hadoop 3.2.1," [Online]. Available: https://hadoop.apache.org/docs/r3.2.1/. [Accessed 04 06 2021].

[19] Nusrat S.I., Md. Wasi-ur-Rahman, Xiaoyi Lu, Dipti S., and Dhabaleswar K., "Performance Characterization and Acceleration of In-Memory File Systems for Hadoop and Spark Applications on HPC Clusters," in *IEEE International Conference on Big Data*, 2015.

[20] Behrouz Pourghebleh and Nima Jafari Navimipour, "Data Aggregation mechanisms in the Internet of things: A Systematic Review of the Literature and Recommendations for Future Research, Journal of Network and Computer Applications," 20 08 2017.

[21] M. Mazhar R., Hojae S., Awais A., Anand P., Gwanggil J., "Real-Time Big Data Stream Processing Using GPUwith Spark Over Hadoop Ecosystem," 15 06 2017.

[22] "big-data-europe/README: General README for the Big Data Europe project," [Online]. Available: https://github.com/big-data-europe/README. [Accessed 04 06 2021].