D2.2 Database structure

Deliverable ID:	**D2.2**
Dissemination Level:	**PU**
Project Acronym:	**Modus**
Grant:	**891166**
Call:	**H2020-SESAR-2019-2 SESAR-ERA-10-2019**
Topic:	**ATM Role in Intermodal Transport**
Consortium Coordinator:	**BHL**
Edition date:	**03 May 2022**
Edition:	**1.0**
Template Edition:	**02.00.05**

Authoring & Approval

Authors of the document

Name / Beneficiary	Position / Title	Date
Damir Valput / INX	Project team	23/04/2022
Ernesto Gregori / INX	Project team	23/04/2022
Inés Gomez / INX	Project team	23/04/2022
Antonio Correas / SKY	Project team	23/04/2022
Ignacio Correas / SKY	Project team	23/04/2022
Pierre Arich / ENAC	Project team	23/04/2022

Reviewers internal to the project

Name / Beneficiary	Position / Title	Date
Annika Paul / BHL	Project coordinator	03/05/2022

Reviewers external to the project

Name / Beneficiary	Position / Title	Date

Approved for submission to the SJU By - Representatives of all beneficiaries involved in the project

Name / Beneficiary	Position / Title	Date
Annika Paul / BHL	Project coordinator	03/05/2022

Rejected By - Representatives of beneficiaries involved in the project

Name and/or Beneficiary	Position / Title	Date

Document History

Edition	Date	Status	Name / Beneficiary	Justification
1.0	03/05/2022	Release	Modus Consortium	Document delivered

Modus

MODELLING AND ASSESSING THE ROLE OF AIR TRANSPORT IN AN INTEGRATED, INTERMODAL TRANSPORT SYSTEM

This deliverable is part of a project that has received funding from the SESAR Joint Undertaking under grant agreement No 891166 under European Union's Horizon 2020 research and innovation programme.

Abstract

This deliverable details the structure of the data lake, providing as well the complete information on the data sources, their relationships to each other, and all the data management techniques applied.

Table of Contents

List of Tables

List of Figures

1 Introduction

This deliverable complements the Data Management Plan (DMP) of the Modus project by describing in more detail the infrastructure used to implement the Modus data lake (used for storing the data in Modus and sharing the data between the partners of the project), the structure of the data lake implemented and the data management and preprocessing techniques applied to the data so that it can be "model-ready".

We describe the cleaning and preparing (and sometimes acquiring) techniques that were performed so far in order to get the data ready for the technical activities in subsequent work packages (3-5). All of the data we work with in Modus are structured data, for which purpose tabular format is used. A data infrastructure relying on the BeSt platform, as described in the Modus Data Management Plan [1] has been set up for all the data needs of the project, and required accessibility was provided to the members of the consortium. Therefore, the data are organised in a structured called *data lake* provided by the BeSt platform by Databeacon (a spin-off company of Innaxis), and it is protected using password-encrypted access. The reader can find more details on the infrastructure used to implement the data lake in Section 2 of this deliverable.

The processing and transforming of the data so far have been performed using the languages Python and to lesser extent SQL.

Since it has been decided that the original DMP will be updated twice during the project (the first update was provided halfway through the project, and the second will be delivered at the end of the project), the final version of the DMP will include a full overview of the data sources used in Modus as well as the output data produced by the project. At the time of writing this deliverable, there are still a lot of activities being performed on the datasets, and therefore this deliverable only captures the current status of the performed data management techniques which can still change until the end of the project. All those changes will be captured in the final update of the DMP.

EUROPEAN PARTNERSHIP

Co-funded by the European Union

2 Database: Modus data lake

Modus operates using the artificial intelligence/data storage and processing platform **BeSt operated by DataBeacon**. BeSt stands for "**Beacon Stack**". BeSt is a scalable, secure, on-demand multi-side computing and data storage platform that allows fast deployment of AI applications in aviation as it securely fuses datasets and runs computations over private, confidential data that are isolated from the rest of the platform.

To ensure confidentiality, privacy and non-disclosure of the data, data owners and consortium members will follow **BeSt's global governance model**, the model consist on a data protection agreement (DPA) with general terms and a series of annexes describing particular usages of such data, e.g., scenarios. More specifically, the platform maintains the confidentiality of any information that may, in any manner, violate the commercial secrecy of any particular data owner before any use by the consortium, leaving only such data necessary for the analyses and modelling.

2.1 BeSt: Technical description

BeSt (Beacon Stack) is a **multi-sided platform (MSP)** for artificial intelligence (AI) applications specifically designed for the **aviation** domain. In a MSP participants are usually both **data providers and consumers of data analysis services.** Participants interact through the MSP using **secure common exploitation of data** to improve their performance among various aspects of their business. This interactions are funded over an **open, collaborative IT infrastructure that operates under a global governance model**. The goal is to consummate matches among users and facilitate the exchange of data and applications, thereby enabling value creation for all participants.

Figure 1 below shows the overall structure of the platform BeSt and how the different blocks connect and interplay. BeSt uses a data de-coupling architecture, which means a data 'broker' sits in between data sources and data analysts. No analyst can directly access the data and only the data broker has access to the data repositories. This adds an additional layer of security. In any data leak, the amount of data leaked will be limited and mitigated as previously de-identified by the SDF (Secure Data Fusion). To learn more about how privacy and security are handled in BeSt, refer to Section 6 on data security.

The analysts work in the block "App / Analytics Environment" (see Figure 1), where Jupyter notebooks are available for performing various analytical tasks over datasets stored in the cloud in an elegant and relatively simple way, relying on the most common programming languages and libraries used in Data Science. The admin functions of de-coding, formatting and de-identification enable data preprocessing that can ensure that the data privacy is respected, giving at the output the data that the analyst can work with without compromising data privacy.

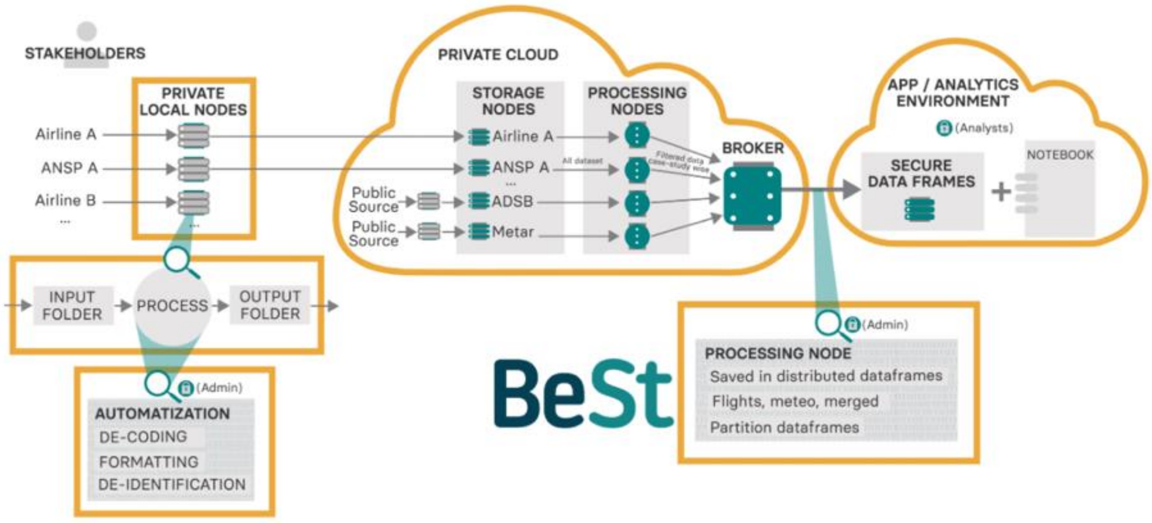


Figure 1: BeSt: Overview

In Modus we have implemented a data lake which we use as a data storage repository (called *Modus data lake* from now in). Unlike more traditional data storing solutions, a data lake is a centralised repository that allows you to store all your **structured and unstructured** data in its natural/raw format (usually object blobs or files), at any scale. Compared to a hierarchical data warehouse, which stores data in files or folders, a data lake uses a flat architecture and object storage to store the data. This way, through leveraging inexpensive object storage and open formats, data lakes make it easier to retrieve data across regions, improve performance and take greater advantage of the data. Data lakes arose as a response to the limitations of data warehouses and data marts, such as information siloing.

A data lake has a number of advantages over more traditional data storages:

Growing data volume. A data lake can help to adapt easily to a growing volume of data, as storage is elastic rather than pre-allocated and the capacity scales with need. The cost of using a data lake is a function of usage, so the users do not pay more than what they actually utilise.
Data variety. Data lakes are designed to naturally handle various data sets and formats, i.e. they are agnostic to data format and one can co-locate any type of data in the same data lake. That way, we can progressively refine data without changing the platform itself and access the data in a uniform way across the lake. Since the Modus data lake is the storage of all the data that is used, it is always up to date and contains any new data sets acquired or updated.
Self-service tools. Data lakes enable users to use different tools and languages to perform different analytics tasks all at once.
Data velocity. We can store data to a data lake without having to define its structure, which means we can store data of any size and even valid and invalid data. This is very flexible if one wishes to perform any type of post-processing, and thus data lakes naturally adapt well to handling high velocity of generating and migrating data that is a common aspect of big data. This is a great advantage for the Modus project as it allows partners to store their data ad-hoc as their data needs arise, with no need to plan ahead the structure of the repository or to satisfy a particular format.
Centralised data catalogue. A data lake that is centralised eliminates problems with data silos (like data duplication, multiple security policies and difficulty with collaboration), offering downstream users a single place to look for all sources of data.

Due to the fact that research data is usually varies, a data lake lends itself naturally to the data needs of a research project such as Modus. In Modus, we rely on Amazon Web Services (AWS) to build our data lake and to do that we will follow a standard procedure to creating a data lake in Amazon S3 (you can see some examples of open data lakes on the following link: https://registry.opendata.aws/):

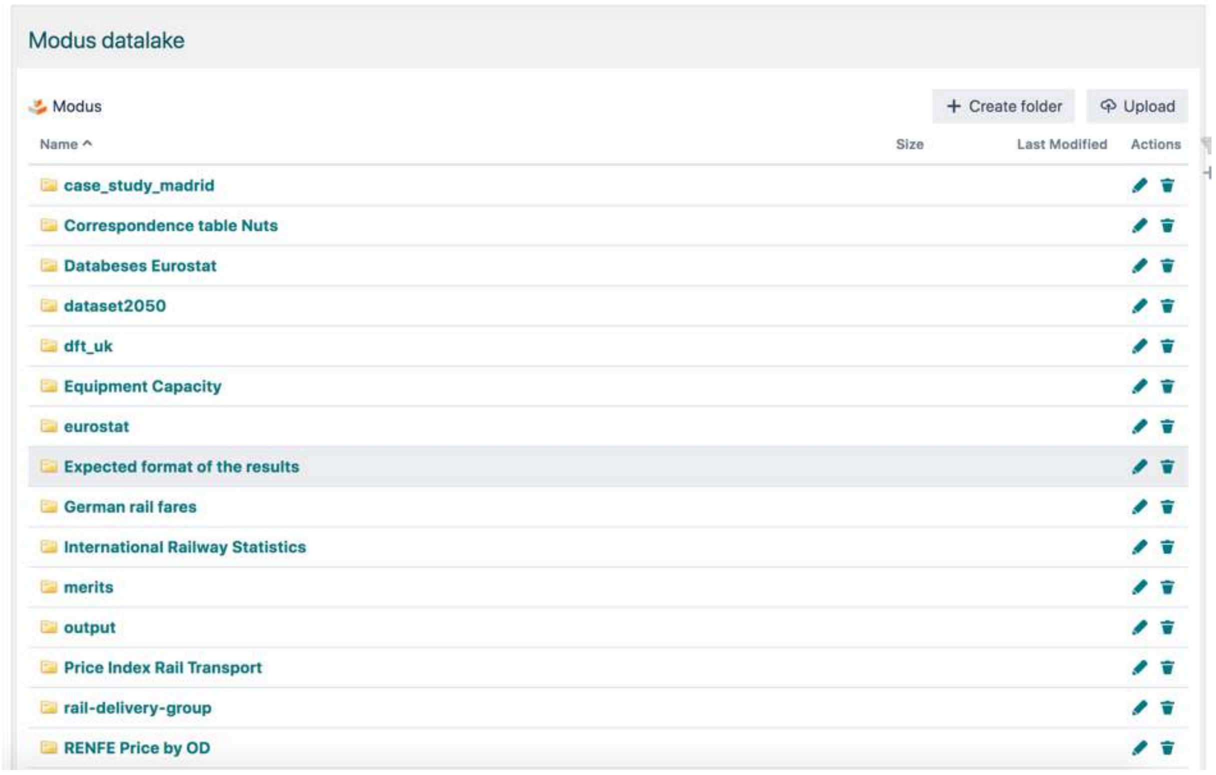
1 **Set up an Amazon S3 storage**. In our case, we create an S3 repository in BeSt by Databeacon '**databeacon-modus**'.
2 **Move the data collected to the created data lake.** This process is repeated as newly acquired data for the project needs to be ingested in the data lake.
3 **Cleanse, prepare and catalogue stored data.** As raw data is ingested, it needs to be cleaned and prepared for usage in predictive modelling and analytics. Such **clean data sets are then moved to other folders in the data lake**, designed to store pre-processed data. The architecture and all changes to the data sets need to be reflected in the **documentation**, which we maintain up to date in **InGrid**.
4 Configure and enforce **security mechanisms to protect confidential data**, if and as needed. This is a very straightforward and easy process enabled by the already provided mechanisms in AWS.

2.1.1 Data interfaces

In BeSt, there are several interfaces available for accessing data: Data Catalogue (AWS Glue), Athena intended principally for relational databases and users familiar with SQL queries, and data lakes intended to work with non-relational databases (AWS data wrangler, parquet and partitioning).

However, in order to facilitate the data storage and sharing between the consortium to the fullest, we rely on the simple and elegant implementation of the interface in our collaborative platform InGrid, that allows everyone with access to upload and download various types of files. We implemented this visual user interface (UI) that is integrated and accessible from a protected InGrid page and allows easy upload and download of the data to/from the underlying data lake without the need to directly access the storage in BeSt, i.e. simply using the provided UI. On Figure 2 below the reader can see what the interface looks like (the state is captured on February 25, 2022).

As one can observe, the interface resembles a simple file storage system as one can find on Windows or MacOS, with which every computer user nowadays is familiar. That makes it extremely simple to use and navigate. Nevertheless, the maximum file size allowed to be uploaded via this interface is 50Mb, which is set for in order to prevent abuse. Therefore, in case of larger files, one should either partition it into smaller units or upload the file directly in the BeSt storage implemented in S3 AWS. Since Innaxis is managing the data lake, their data engineers and analysts who have access to the BeSt platform can easily execute those uploads at the request of any member from the Modus team.



Name ^	Size	Last Modified	Actions
case_study_madrid			
Correspondence table Nuts			
Databeses Eurostat			
dataset2050			
dft_uk			
Equipment Capacity			
eurostat			
Expected format of the results			
German rail fares			
International Railway Statistics			
merits			
output			
Price Index Rail Transport			
rail-delivery-group			
RENFE Price by OD			

Figure 2: Interface to the Modus data lake implemented in InGrid (status on February 25, 2022)

3 Data Sources

3.1 Overview of data sources

Table 1 provides an overview of the data sources collected so far in Modus, indicating whether or not they are confidential.

While new data sources might still be acquired, once a data set is acquired, Modus does not plan to update it during the project execution, due to the fact that all the data are collected for a certain historical time period and geographical scope according to the defined case studies. The case studies in Modus have been defined, and thus the data are acquired, from chosen data providers, in line with the defined time period and geographical scope. Nonetheless, no changes to the collected datasets are foreseen at this point.

Produced data sets on the other hand are subject to change (e.g. new results obtained upon improving some model). Those changes will be adequately tracked through the standard processes of data versioning. When Modus produces new versions of some files, it will record what changes are being made to the files and give the new files a unique name, commonly by appending a version number to the end of the file (e.g. new simulation results produced by updating the model version form "v2.0" to "v2.1"). For all the software and data version control needs, Modus relies on:

- the data storage platform BeSt, where the Modus data lake is implemented (see Section 2) and all shareable data sources are ingested, and
- a distributed version control system git (using git repositories on GitHub).

Table 1: Data Sources in Modus

	Data source (provider)	Description of the data source	Purpose in Modus	WPs	Time resolution (if applicable); geographical scope	Status	Open Data status (PU - public, CO - confidential)
1	Official Airline Guide (OAG)	The OAG data is available for each year. The dataset contains planned global flight schedules, such as departure and arrival times, ASKs, carrier name, and type of service. The schedule contains also cargo aviation services. OAG data shows also the utilisation of airports. More information available at https://www.oag.com/.	Modal choice modelling, door to door models. It's not shared between the consortium members due to confidentiality restrictions, available only to ENAC.	WP3, WP4	Monthly or daily data (2018, 2016, 2014, 2012, 2010, 2008 etc.); world	acquired	CO

2	Air transport data database (FRACS)	The Air Transport Database is owned by FRACS (*France Aviation Civile Services*) funded by the French Civil aviation (DGAC) and ENAC. It contains data on airline companies, airports and traffic between countries and cities.	Air traffic network models. It's not shared between the consortium members due to confidentiality restrictions, available only to ENAC.	WP3, WP4	Yearly; world	acquired	CO
3	MERITS	MERITS (Multiple East-West Railways Integrated Timetable Storage) is a database, owned by UIC, containing the integrated timetable data of many European and some non-European countries (Russia, Turkey, Belarus), comprising a few hundred railway undertakings (RUs), which are published twice a week.	Modal choice modelling, door to door models	WP3, WP4	2016, 2017, 2019; Europe	acquired	CO
4	UK DfT data (UkGov)	The data published annually by the UK's Department for Transport on rail passenger numbers, crowding and capacity through a series of tables ('RAI02').	Modal choice modelling, door to door models	WP3, WP4	Yearly; UK only	acquired	PU
5	SKY synthetic data	Synthetic dataset representing hypothetical routes e.g. in intermodal connections, last-mile mobility, or long-distance road, rail and air routes, under specific geographic constraints.	Route and route cost modelling	WP3, WP4	N/A, any geographical region within EU	Generated using Skymantics Routing Engine	PU
6	SNCF	Statistics on the French national rail operator. Provides information about average time per line, different levels of delay, number of train anticipated, number of canceled trains, loss leader price per O-D according to the class, etc.	Modal choice modelling	WP3	Monthly, France-Europe (domestic and international services)	acquired	PU

7	Mercury database	The data that defined the operational environment in which the simulations run by the Mercury simulator are performed.	Gate-to-gate modelling of air traffic. It's not shared between the consortium members due to confidentiality restrictions, available to UoW.	WP4	A full day of operations, ECAC	acquired, part of Mercury	CO
8	Eurostat	The data contains various socio-economic variables.	Modal choice modelling	WP3	2009-2019, Europe	acquired	PU
9	UK Rail Delivery Group	UK train fares data.	Modal choice modelling	WP3	2016-September 2021, UK	acquired	PU
10	Renfe fare data	Spain train fares data.	Modal choice modelling	WP3	April 2019 - December 2020, Spain	acquired	PU
11	Deutsche Bahn	Germany train fares data.	Modal choice modelling	WP3	2020-2021, Germany	acquired	CO

Detailed description of all the data sets listed in the Table 1 as well as the data produced by Modus (output data sets), can be found in the Modus Data Management Plan (DMP) and its respective updates.

Co-funded by the European Union

4 Data management techniques

This sections details the data management and preprocessing techniques applied to various datasets used in Modus.

4.1 MERITS: preprocessing

MERITS, the Multiple European Railway Integrated Timetable Storage, is a database owned by UIC (International Union of Railways) containing the integrated timetable data of many European and some non-European countries (Russia, Turkey, Belarus). It comprises a few hundred railway undertakings (RUs) which are published twice a week. It is designed so that is capable to exchange timetable data in a unique format (EDIFACT).

The MERITS database collects and integrates European railway timetable data and provides all DSU with access to its contents. The format it relies on, EDIFACT, is a message oriented data format, standard for data exchange that relies on two different message types: one for the transfer of "schedule data" which (in general) are train data (SKDUPD), and another for the transfer of "static data" which are location data (TSDUPD). EDIFACT is the single format used for importing and exporting the data (SKDUPD and TSDUPD messages).

This data format is not suitable for the analysis purposes in Modus, therefore the data had to be preprocessed (transformed into a CSV format and cleaned) so that it could be used for analysis and modelling purposes in Modus. The data in the EDIFACT format was parsed relying on the parser provided by UIC (code property of UIC, shared with Modus team for the purposes of the project - limited usage) which extracts the meaningful parameters from MERITS and delivers them in a tabular format (CSV). The process and the output obtained is described below.

4.1.1 MERITS data parser

The MERITS data parser and converter was provided by UIC, a partner in the Modus projects, and with their permission it has been used to extract meaningful data fields from EDIFACT messages in Merits and store them as CSV files. The code is written in Python and takes as input MERITS SKDUPD files (TRAIN, ODI, POR and RELATION files) and outputs the information contained in those files to CSV files, which can then be easily read and analysed by common data analytics techniques.

The data conversion was done on all of the acquired data in Modus, covering years (fully or partially) 2016, 2017 and 2019. The **converted data in CSV files has been stored in Modus data lake**.

In continuation we present the variables that can be found in three different SKDUPD files in MERITS: ODI, POR and TRAIN. Upon consultation with the data owner UIC, it was decided that the data contained in the RELATION files is not relevant for Modus and thus it was not processed.

4.1.2 MERITS: tabular output

In this section we present the list of all the parameters extracted from MERITS, partitioned by file type: ODI, POR and TRAIN. For the full description of the data fields, please take a look at the technical manual of MERITS (MERITS-guide).

Table 2: SKDUPD_ODI

Variable	Description
ID	ID of the train. It is reset for each year.
start	Stop from which the characteristics apply
end	Stop at which the characteristics cease to apply
value	Not relevant for Modus
reservation	Not relevant for Modus
equipment	Not relevant for Modus
tariff	Fare information

Table 3. SKDUPD_POR

Variable	Description
ID	ID of the train. It is reset for each year.
pos	The position of the stop: first, second, third, etc.
UIC	UIC code of the stop
arrival	Arrival time
offsetA	Indicates if it is past midnight
departure	Departure time
offsetD	Indicates if it is past midnight
quay1	Arrival dock
quay2	Departure dock
detail	Not relevant for Modus
boarding	Not relevant for Modus
message	Not relevant for Modus
load	Not relevant for Modus
unload	Not relevant for Modus

Table 4: SKDUPD_TRAIN

Variable	Description
ID	ID of the train. It is reset for each year.
service_number	Train service number
service_characteristic	Reservation type
pricing_category	Product characteristics identification code

EUROPEAN PARTNERSHIP

Co-funded by the European Union

service_mode	Mode of service: rail, road, etc. Examples: 'ICE INTERNATIONAL', 'SOUTHSIDE-EXPRESS'
service_name	Name of the service
service_provider	Provider of the service. Examples: SNCF, SNCB, DB, etc.
information_provider	Empty for all data observations. We can discard this parameter.
reservation_company	Not relevant for Modus
beginning_date	Timestamp of the first date of circulation
end_date	Timestamp of the last date of circulation
circulation_days	String containing a digit for each day between beginning date and end date. The corresponding digit is 1 if the train circulated that day and 0 if it didn't.
RFR_number	Not relevant for Modus.

4.1.3 MERITS for the landside model

The data provided by MERITS is of use for the landside part of the Mercury model. As the airside of Mercury will run on the data from one day in September 2018, we analyse two full days of rail traffic, as provided in MERITS, from September 2017 and 2019:

- September 2017, 15 → 266,891 rail services.
- September 2019, 6 → 510,152 services.

In order to prepare the data for the use in the landside model, we filter the relevant data from the cleaned MERITS dataset and join all the parameters we deem useful for this model into one dataset of tabular format, stored in the data lake. Final produced CSV files, with all the extracted parameters as listed in Table 5 below, can be found under the following paths in the data lake:

- September 2017: *merits/merits-for-rail-model/merits_final_rail_model_sep17.csv*
- September 2019: *merits/merits-for-rail-model/merits_final_rail_model_sep19.csv*

Table 5: Metadata of the Merits data subset produced for the landside model

Variable name	Source in MERITS	Description	Data example
service_name	SKDUPD_TRAIN	Type of train	'ICE International'
pricing_characteristic	SKDUPD_TRAIN	Pricing category	1
service_provider	SKDUPD_TRAIN	Provider of the service	X113
beginning_date	SKDUPD_TRAIN	porFirst day of circulation	2017-07-08
end_date	SKDUPD_TRAIN	Last day of circulation	2017-07-28

circulation_days	SKDUPD_TRAIN	String representing the days of circulation	1000000100000 010000001
id	**SKDUPD_POR**	**Unique identifier of the service**	**Primary key**
pos	SKDUPD_POR	The position of the stop: first, second, third...	1
UIC	SKDUPD_POR	UIC code of the stop.	008759400
arrival	SKDUPD_POR	Time of the arrival to the station	08:01:00
offsetA	SKDUPD_POR	Flag to know if the train arrived after midnight	1
departure	SKDUPD_POR	Time of the departure to the station	16:38:00
offsetD	SKDUPD_POR	Flag to know if the train left after midnight	1
value	SKDUPD_ODI	N/A	This is NaN in most cases, to be ignored.
equipment	SKDUPD_ODI	Equipment used	91
tariff	SKDUPD_ODI	Tariff information	NaN

4.2 UK Rail Delivery Group: data preprocessing

The Rail Delivery Group (RDG) brings together the companies that run Britain's railway into a single team. All the passenger and freight rail companies are members of the RDG, as well as Network Rail and HS2. The data obtained from the RDG contain information on train fares. Each year in the UK, train fares can be subject to revision in January, May and September, with the main revision taking place in January each year and the May and September revisions being available to make seasonal alterations to fares. The fare data download available on the RDG website is released three times a year in January, May and September. These three releases are known as 'full releases' of fares data and are made available free of charge under the terms of a Creative Commons licence. The data containing fares information are extracted from the Data Transformation and Distribution Service (DTD).

4.2.1 Raw data format and preprocessing

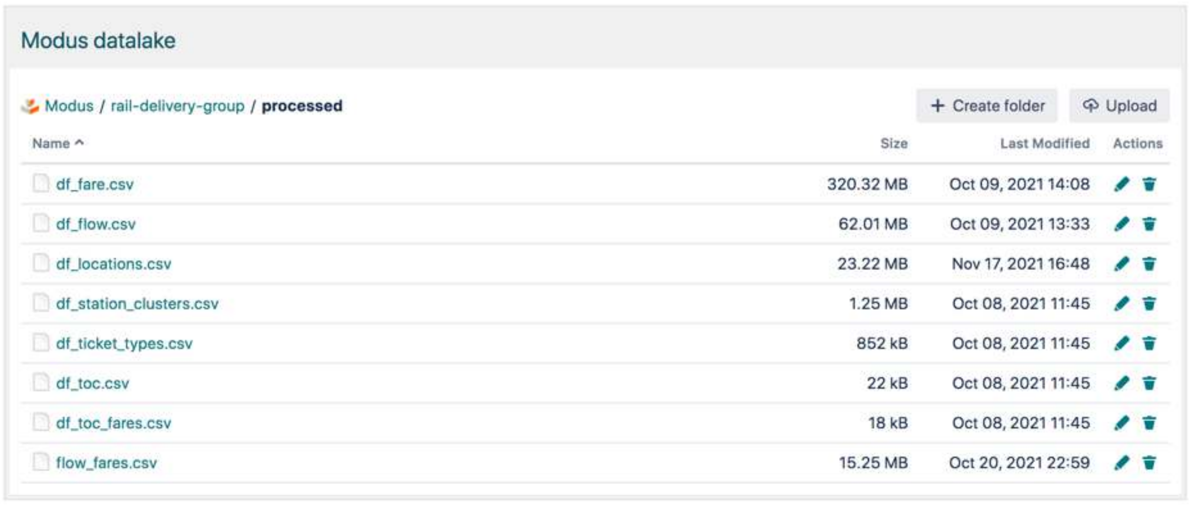
Data feeds are delivered in **fixed format flat text files**. Some files contain several record types. The downloaded data set contains 24 export files typed defined for the Fares feed, out of which Modus has selected several of them that contained information relevant for the project. The following files (fixed format text files) have been selected for their use in Modus, and in each of those files **one line contains a record**. Some files have several record types.

These files had to be preprocessed so to extract the fields of interest. The extracted variables were organised in a tabular format and stored as CSV files in the Modus data lake, under "rail-delivery-group/processed" (see the screenshot of the data lake on Figure 3). The data covers UK train routes from 2016 onwards.

Co-funded by the European Union

Since the parameters are stored in a fixed format, the preprocessing is fairly straightforward and consists of:

- identifying the variables of interest according to the Rail Delivery Group technical manual (RDG-manual) and their location in the record
- parsing the records to extract the identified variables


Modus datalake

Modus / rail-delivery-group / **processed**

Name	Size	Last Modified
df_fare.csv	320.32 MB	Oct 09, 2021 14:08
df_flow.csv	62.01 MB	Oct 09, 2021 13:33
df_locations.csv	23.22 MB	Nov 17, 2021 16:48
df_station_clusters.csv	1.25 MB	Oct 08, 2021 11:45
df_ticket_types.csv	852 kB	Oct 08, 2021 11:45
df_toc.csv	22 kB	Oct 08, 2021 11:45
df_toc_fares.csv	18 kB	Oct 08, 2021 11:45
flow_fares.csv	15.25 MB	Oct 20, 2021 22:59

Figure 3: The output files after preprocessing UK Rail Delivery Group data

Each text file has a fixed header and terminator which need to be removed before parsing the data, i.e. the file consists of three parts:

- informational header - lines start with '/'
- ordered sequence of records - what we are interested in
- terminator - lines start with '/'

The header and the terminator look as follows:

Header:

/!! Start of file
/!! Content type: type (as in table above)
/!! Sequence: nnn
/!! Records: nnnnnnn
/!! Generated: dd/mm/yyyy
/!! Exporter: DTD
_module version

Terminator:

/!! End of file (dd/mm/yyyy)

4.2.2 UK Rail Delivery Group: output of preprocessing

After the preprocessing stage, the files shown on Figure 3 were obtained. Their description is summarised in Table 6.

Table 6: Description of the output files containing data extracted from UK RDG

File name	Description of the content	Raw record type from which the data was extracted	Extracted parameters
df_fare.csv	Point to point adult fares in a clustered format	FLOW record	flow-id: Uniquely identifies the flow to which the fare pertains ticket-code: 3-character ticket code for the fare fare: Fare in pence restriction-code: restriction code associated with this fare
df_flow.csv	Point to point flows in the network.	FLOW record	flow-id: Uniquely identifies the flow to which the fare pertains origin-code: A code representing the flow origin (4 digit NLC code, county code, zone code). This may be a cluster NLC, in which case this flow applies to all stations in the cluster. Where DIRECTION = 'R' then this flow may also be used for fares in the reverse direction, in which case ORIGIN-CODE should be used as DESTINATION-CODE in the reverse direction destination-code: A code representing the flow destination (4 digit NLC code or county code). This may be a cluster NLC, in which case this flow applies to all stations in the cluster. Where DIRECTION = 'R' then this flow may also be used for fares in the reverse direction, in which case DESTINATION-CODE should be used as ORIGIN - CODE in the reverse direction route-code: Route code direction: S - fare applies in a single direction; R - fare applies in both directions (it is reversible end-date: Last date for which this record can be used. Format is ddmmyyyy. A high date (31122999) is used to indicate records which have no defined end date start-date: First date for which this record can be used. Format is ddmmyyyy toc: The Fare TOC code of the TOC setting the fares on the flow

df_station_clusters.csv	Lists the station clusters, and the locations included in each of the clusters	Station clusters record: FSC file type	cluster-id: 4-character alphanumeric NLC code at which the cluster fares are set cluster-nlc: NLC code of a location which is a member of the cluster (it may also be a zone code or a county code). The fares for this location may be set using the Cluster NLC instead of this NLC. A member may exist in several clusters end-date: Last date for which this record can be used. Format is ddmmyyyy. A high date (31122999) is used to indicate records which have no defined end date start-date: First date for which this record can be used. Format is ddmmyyyy.
df_ticket_types.csv	Ticket codes, their type, class and other ticketing information	Ticket types record: TTY file type	tkt-code: alphanumeric ticket code end-date: Last date for which this record can be used. Format is ddmmyyyy. A high date (31122999) is used to indicate records which have no defined end date start-date: First date for which this record can be used. Format is ddmmyyyy quote-date: First date for which this record can be queried. Format is ddmmyyyy tkt-description: Ticket description tkt-class: Ticket class, currently '1', '2' or '9' tkt-type: Ticket type, single ('S'), return ('R'), or season ('N') tkt-group: Ticket group. First, Standard, Promotion or Euro: 'F', 'S', 'P', 'E' max-pax: Defines the maximum number of passengers who can travel on this ticket min-pax: Defines the minimum number of passengers who can travel on this ticket.
df_locations.csv	Locations, including group locations	Locations record: LOC file type	uic-code: a unique code which identifies this location end-date: Last date for which this record can be used. Format is ddmmyyyy. A high date (31122999) is used to indicate records which have no defined end date start-date: Earliest date for which this record can be used. Format is ddmmyyyy. A high date (31122999) is used to indicate records which have no defined end date nlc-code: National location code, for British locations only. No value is output in this field for non-GB locations description: location description

			county: Used to decide if a location is in Scotland, England & Wales or elsewhere. County codes on the mainland are all numeric values. Other values are 'NI' (Northern Ireland), 'IR' (Ireland), 'CI' (Channel Islands) assoc-uic-code: UIC Code of associated station
df_toc.csv	TOC codes and descriptions	TOC record: TOC file type	toc-id: TOC identifier. Used in CIF to identify the trains of a particular TOC toc-name: TOC name active-indicator: indicates whether there is an active entry: 'Y' or 'N'
df_toc_fares.csv	TOC codes and fares	Fare TOC record: TOC file type	toc-id: TOC identifier. Used in CIF to identify the trains of a particular TOC fare-toc-id: TOC identifier. Used in Flow file to identify which TOC is responsible for the fares on this flow fare-toc-name: TOC name.
flow_fares.csv	Obtained by merging: df_flow, df_fare, df_ticket_types, df_toc, df_toc_fares, df_locations; i.e. everything excect df_station_clusters	N/A	

4.3 Synthetic data generation: city archetypes

Cities are spaces containing hubs of more than one transportation mode. Models developed in WP4 account for long-distance air and rail travel between European cities. In addition, to have a complete view of the end-to-end journey, it is necessary to also account for passenger connectivity within the city, both as last-mile (airport to home/office, home/office to airport) and intermodal changes (airport-rail, rail-airport). The time taken when changing transport modes gives information of the frictions created at the local level, and allows to compare performance of modal choices.

Synthetic datasets have been generated using Skymantics routing engine to simulate the fastest/shortest routes between airports and other locations in an urban environment. These locations include railway stations and city centres, but also any other point in the city and surrounding region that is relevant as an origin or destination of travel. The resulting datasets represent catchment areas for required travel time, defined concentrically around city airports of different configurations.

4.3.1 Data generation method

The routing engine supports the calculation of route metrics between pairs of locations in a map. Routes are calculated as optimal results of routing algorithms following constraints defined in time and space (e.g. roads, schedules). Catchment areas are generated as collections of generated routes terminating at the location of choice. The data set for Modus has been generated to represent the catchment area to the city airport of choice based on shortest time metric. For cities with multiple airports, catchment areas are generated for each airport separately.

Per airport, two separate catchment area layers are generated representing different modal choices and making use of different datasets:

Private vehicle (incl. taxi or ride-sharing) routes use Open Street Map (Geofabrik download - https://download.geofabrik.de/). Open Street Map is an open data project supported by a global community which contains global data about roads, classification of roads, and features in maps. Geofabrik has data extracts from Open Street Map which are updated daily. Road grids in the bounding box of the selected region are selected for the route calculations. Open Street Map data is free, complete, and fairly accurate. It provides information on traffic elements (lights, stop signals, crossings) and some speed limit information. Those roads without speed limit information have been inferred as a stochastic model based on neighbours and proximity. Although Open Street Map does not provide information on real traffic speed, estimations have been based on street/road hierarchy, speed limit and traffic elements.

Public transit routes use combinations of walking plus public transportation network present in the city (bus, metro, or short-distance rail). Walking routes are extracted from Open Street Map Geofabrik. Public transportation is consumed from local, regional or national GTFS open data projects maintained by the corresponding governments:

- Madrid (airport MAD): https://data-crtm.opendata.arcgis.com/
- Paris/Ile de France (airports CDG, ORY, BVA): https://data.iledefrance-mobilites.fr/ https://transport.data.gouv.fr/datasets/ Transport networks of the regions Ile de France , Hauts-de-France (Aisne, Oise, Somme), Bourgogne, Normandy, Grand-Est, Centre-Val de Loire. It includes shuttle service from Paris - Porte Maillot to Beauvais (https://www.aeroportparisbeauvais.com).
- Stockholm/Sweden (airport ARN): https://www.trafiklab.se/api/trafiklab-apis/gtfs-sverige-2/
- Brussels/Benelux (airports BRU, CRL): https://hello.irail.be/gtfs/ https://transport.data.gouv.fr/datasets/ https://data.public.lu/ https://gtfs.dehttps://gtfs.ovapi.nl/ Transport networks of the regions Brussels, Wallonie, Flanders, Nord and Grandest (France), Luxembourg, the Netherlands and Germany. It includes shuttle service from Charleroi airport to a series of cities in the region (https://www.flibco.com)

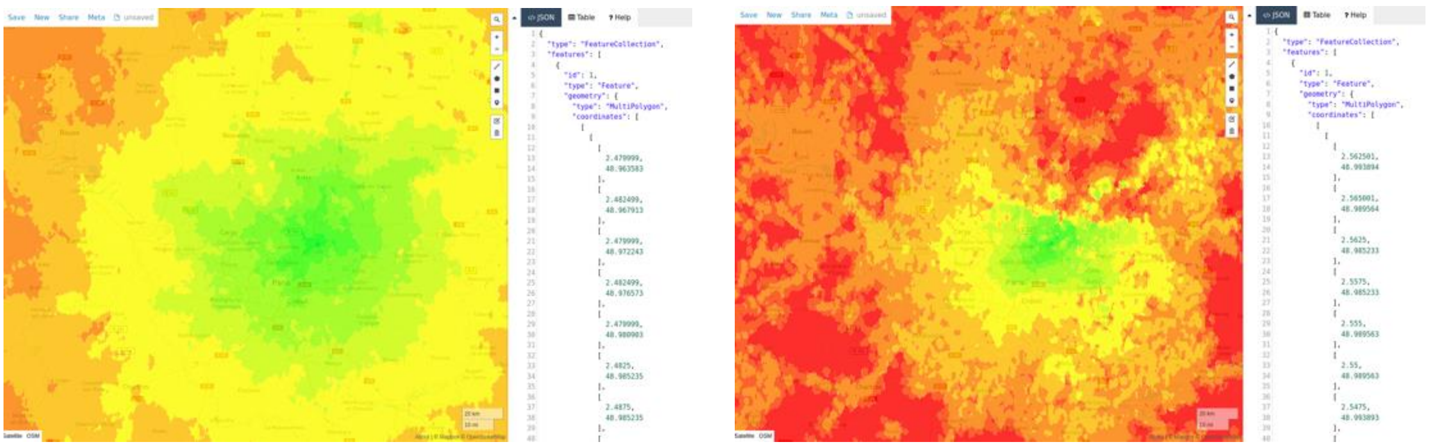
GTFS data contains stations and accesses, stop, routes, timetables and frequencies, although not all sources offer the same type of information or level of accuracy. For each public transport dataset, the routing engine has been adapted to build a grid of connected stations, including stops and applying time costs between them. Accesses to pedestrian traffic have been added to allow for the transfer among stations, stops or transport means. In addition, waiting times have been generated at stops based on average frequencies between 6am and midnight during labour days.

For the case of public transit routes to CRL, two catchment area maps have been generated in order to assess the effects of creating a route network of shuttle service to improve an airport's accessibility.

4.3.2 Output data

The synthetic datasets are generated in GeoJSON format with the following data structure:

{	
"type": "FeatureCollection",	
"name",	*Name of the catchment area (e.g. "ARN catchment area with public transportation")*
"features": [	*Array of overlaid polygons*
"id",	
"type": "Feature",	*Each polygon defined as a feature*
"geometry": {	
"type":"MultiPolygon",	
"coordinates": [lon, lat] },	*Collection of polygon points and coordinates*
"properties": {	
"area-type": 1,	
"description",	*Describes the polygon (e.g. "10-minutes boundary")*
"stroke", "stroke-width", "stroke-opacity", "fill", "fill-opacity"	*For visualization*
}] }	



```
{
  "type": "FeatureCollection",
  "features": [
    {
      "id": 1,
      "type": "Feature",
      "geometry": {
        "type": "MultiPolygon",
        "coordinates": [
          [
            [
              [
                2.479999,
                48.963583
              ],
              [
                2.482499,
                48.967913
              ],
              [
                2.479999,
                48.972243
              ],
              [
                2.482499,
                48.976573
              ],
              [
                2.479999,
                48.980903
              ],
              [
                2.4825,
                48.985235
              ],
              [
                2.4875,
                48.985235
              ],
              [
```

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "id": 1,
      "type": "Feature",
      "geometry": {
        "type": "MultiPolygon",
        "coordinates": [
          [
            [
              [
                2.562501,
                48.993894
              ],
              [
                2.565001,
                48.989564
              ],
              [
                2.5625,
                48.985233
              ],
              [
                2.5575,
                48.985233
              ],
              [
                2.555,
                48.989563
              ],
              [
                2.55,
                48.989563
              ],
              [
                2.5475,
                48.993893
              ],
              [
```

Figure 4: Example of visualization of a) Private vehicle transport, b) Public transport to Charles de Gaulle airport (CDG) in Paris region

4.3.3 Data usage

The catchment areas are used to generate some of the time distributions in the door-to-door model. They are the base of the door-to-kerb and kerb-to-door stages, which represent the route from the home address to the departure airport and from the arrival airport to the final destination.

In order to generate these distributions, the catchment areas are exploited through a spatial sampling method. This technique allows to estimate the time used by a set of passengers with different home addresses within the airport's catchment area to reach the airport.

Once this is done for a set of model airports, the results can be extrapolated to the set of airports with the same archetype as the ones processed with the data. This hypothesis allows to simplify the computations, since only a set of template airports have to be studied in detail to obtain the catchment areas.

4.4 Renfe Spain data: preprocessing and acquisition

The Renfe fare data was collected on Kaggle and concerns the period between 12/04/2019 and 05/12/2020. Kaggle (available at: https://www.kaggle.com/) is a platform for machine learning practitioners to share their work and compete in various machine learning challenges, and often one can find many data sets freely available, such as was the case with this dataset.

The original downloaded data set is very large with a volume of about 8GB. This is due to the method of automated data collection in which data samples were ingested repeatedly at regular and very fine intervals. It was therefore essential to preprocess the data in order to clean it and keep only the information that we will use later in the modelling phase.

The preprocessing of the Renfe data set was carried out in two stages, with an additional overhead step before in which we split the dataset into ten partitions in order to be able to read such a large database. Dividing the parent dataset into subsets helped us overcome the difficulty of reading all of the data at once.

Once this was done, the first step in the preprocessing was to remove the duplicates contained in the dataset. We defined a duplicate is a row in the dataset that is completely identical to another row in the dataset. The method of data collection described above in which data samples were ingested with a very high frequency resulted in a large number of duplicates in the dataset. Once this step was completed, we continued the data cleaning by deleting the variables not necessary for our study, such as the date and time of collection, ID, the number of seats available as this was a variable with 97% missing data, or travel company as it was composed of a single mention (Renfe).

All the adjustments described above are applied to all ten data subsets in the same way. Once the whole process was completed, the cleaned version of those subsets were once again merged into a single dataset. The final dataset contains following variables:

- Origin;
- Destination;
- Departure;
- Arrival;
- Duration;
- Vehicle_type;
- Vehicle_class;
- Price.

4.5 Modal choice model: data preparation

The data acquisition phase was followed by an important preparation phase of the acquired datasets. This was particularly long and complex due to the large volume of data acquired on the one hand, and the significant differences between air and rail transport on the other. We therefore had to harmonise the structures of the different databases, particularly OAG and MERITS, so that they all adopted the same structure.

For those purposes, we chose to adopt the structure proposed by the OAG database, i.e., a line is equal to an origin-destination (OD), an operator, a date, a departure, and arrival time and all the associated available characteristics. Having made this choice to have the **OAG data schema as the referent one**, the next logical step was to **transform the MERITS data** so that it fits the desired data schema.

4.5.1 Transformation of MERITS data to fit the modal choice model data scheme

Initially, MERITS had a completely different structure with aggregated information. Moreover, the MERITS database is originally composed of three sub-databases, each with a readable ID. The different information about a train[1] is divided into three separate databases. The first sub-base is called "SKDUPD_TRAIN" and contains information on the provider, the type of equipment used, the date, day of departure and the train number. The second sub-base, named "SKDUPD_POR", contains information on the route such as intermediate stops. This second sub-base is completed by the third sub-base 'SKDUPD_ODI' which provides information on the station from which all characteristics of the two previous bases apply and the station at which these characteristics end. In addition to this information, this last sub-base specifies the presence of additional services such as the presence of a dining car.

The other particularity of the MERITS database is that one row of data does not correspond to one concrete rail trip, but to a set of train services sharing the same characteristics except for the date of travel. As a result, we had to perform a lot of restructuring and disaggregation work to obtain a data schema as the one of the OAG data. Subsequently, we merged the data from the three sub-databases into a single dataset containing the information and characteristics of each train. Although the structure of the databases obtained after preprocessing is identical between OAG and MERITS, in the next step we had to work on the differences related to the terminology and codes used by each mode.

Concretely, air and rail transport are two different transport modes and as such use a language and codification specific to their world. In the case of rail, stations are coded with a 7-digit "UIC" code. In the case of air transport, airports use a three letters "IATA" code. In order to adopt a common language and to group stations and airports in the same geographical area, we have adopted the language and coding used by Eurostat, i.e., NUTS[2]. Beforehand, this required the creation of a table allowing the correspondence between the UIC/IATA-NUTS codes. These correspondence tables are essential for correctly joining the data between the OAG and MERITS databases. In other words, NUTS codes becomes the common denominator of the databases, i.e. the key that can be used to join data examples from the two databases.

4.5.2 Preprocessing fare data

With respect to processing **fare data**, the format of those differed depending on the country. Some countries, such as France where the data came directly from an SNCF document referencing all passenger fares by OD and class and Spain where we obtained the data via a massive and automated collection available in open data on Kaggle, did not require much pre-processing of the data as it was already provided grouped by OD pairs and travel class. In contrast, in the case of Germany, we had prices per kilometre and per class provided by Professorship of Travel Behavior (Dr. Moeckel) at the Technical University of Munich, so we simply calculated the price for each DO and class via the distances.

For more details on the sources of these data, please refer to the section "Data sources" in Modus Data Management Plan [1].

[1]A train equals, an origin-destination, an operator, a date, a departure and arrival time and a rolling stock.

[2]The classification of territorial units for statistics, abbreviated NUTS (from the French version Nomenclature des Unités territoriales statistiques) is a geographical classification subdividing the economic territory of the European Union (EU) into regions at three different levels.

EUROPEAN PARTNERSHIP

Co-funded by the European Union

5 Final remarks

This deliverable describes the structure of the data lake implemented for storing and sharing data in Modus, as it was on April 2022. Moreover, it describes the data processing activities that were performed so far in the project.

The data preprocessing tasks in Modus are done with the primary goal of cleaning and preparing the data for the model development (part of WP4) and simulation and analysis (part of WP5), as in most cases the raw data (data as received and ingested) is not directly suitable to be fed to models. Normally substantial efforts need to be invested to transform the data from its raw format to one that can be utilised in modelling and analysis with high certainty that the data used are accurate and reliable. For those purposes, we performed the tasks of data cleaning, transformation, correction of erroneous data points, outlier removal, and similar, described in this deliverable.

As the data processing and modelling are still ongoing activities in Modus, we can expect further data processing tasks to be performed in the following months, as well as some possible minor changes to the data lake structure. Those will be reported in the subsequent deliverables, such as the final version of the Data Management Plan (to be delivered) at the end of the project and the deliverables on modelling and analysis that will describe in details how the data are used in those tasks.

EUROPEAN PARTNERSHIP

Co-funded by the European Union

6 References

[1] (Modus DMP) Modus Data Management Plan (DMP), Deliverable ID: D2.1, November 2020. Edition: 1.1. Updated version 1.2 submitted in December 2021.

[2] (UkGov) Statistical datasets by GOV.UK, Available at: https://www.gov.uk/government/statistical-data-sets/

[3] (OAG) Schedules Analyser: User guide, OAG Analytics. Available at: https://www.oag.com/schedules-analyser-user-guide

[4] (MERITS-guide) MERITS User Guide, Version 17.1. Date: 2017-02-28. International Union of Railways (UIC)

[5] (VISTA) Final Project results Report, Deliverable D1.2. H2020 project "Vista" coordinated by the University of Westminster. Edition date: 09 November 2018.

[6] (POEM) Passenger-Oriented Enhanced Metric (project POEM), coordinated by the University of Westminster. Project webpage: https://devengagewiki.com/POEM

[7] (AWSGlue) AWS Glue: Simple, scalable, and serverless data preparation. Available at: https://aws.amazon.com/glue/?whats-new-cards.sort-by=item.additionalFields.postDateTime&whats-new-cards.sort-order=desc

[8] (SURE) Building a routing engine. Part 2: Skymantics Universal Routing Engine (SURE). Available at: http://skymantics.com/2020/05/19/building-a-routing-engine-part-1-about-ogcs-open-routing-api-2/

[9] (Git) Data Management Planning: Version Control. Available at: https://guides.nyu.edu/data_management/version-control

[10] (RDG-manual) The Rail Delivery Group. Available at: https://www.raildeliverygroup.com/about-us.html. Accessed: 14 September 14 2021

[11] (RDG Fares) Fares Data, The Rail Delivery Group. Available at: http://data.atoc.org/fares-data. Accessed: 14 September 14 2021

[12] (Renfe Kaggle) Spanish Rail Tickets Pricing - Renfe. Publicly available dataset, under GPL 2 license. Available at: https://www.kaggle.com/thegurusteam/spanish-high-speed-rail-system-ticket-pricing/metadata. Accessed: 20 September 2021.