FIWARE GLOBAL SUMMIT OVERVIEW (STATE-OF-ART and PROGRESS)

Smart Data Models

Open APIs for Open Minds

SMART DATA MODELS

Training Camp

13-6-2023

https://bit.ly/SDM1

Alberto Abella

Data modelling expert

FIWARE Foundation

alberto.abella@fiware.org

@aabella

@smartdatamodels

FIWARE

Contents and goals

- About:
  - During this part of the session you will get introduced to the Smart Data Models Initiative

- This session will:
  - Introduce the challenge of data interoperability in high-speed change rate environments
  - Explain basics about the Smart Data Models initiative and how it is governed
  - Introduce you to data models for specific verticals that are already available
  - Explain how you may contribute with new data model to to extend to existing data models under Smart Data Models

- Goals:
  - After this session you will be able to implement a service using a Smart Data Model
  - You will be able to explain how to become an active contributor to the Smart Data Model Program

Agenda

- Standardization in a digital market
  - Levels of standardization
  - Current situation
- Introduction to SDM:
  - What is in a data model
  - Board and what is Smart Data Models Program
  - Endorsements and adoptions
  - Mapped standards
  - Global structure
  - Current Status
- Facing the challenges of standardization in digital markets
  - SDM approach to solution
  - Differential factors
- Current status
  - Data models and domains
  - Subjects and data models
  - Contributors and dissemination
- Data Model Contribution
  - Sources of contribution
  - Adoption of open standards
  - Contribution workflow
  - Support
- Services to users and contributors
  - List of services
  - Incubated repository
  - pysmartdatamodels python package
  - SQL export
  - Linked data customization (mapping ontologies)
- Agile Standardization
  - Seven principles
- Life visit
  - smartdatamoels.org
  - https://github.com/smart-data-models

Smart Data Models

Open APIs for Open Minds

Standardization at Digital markets

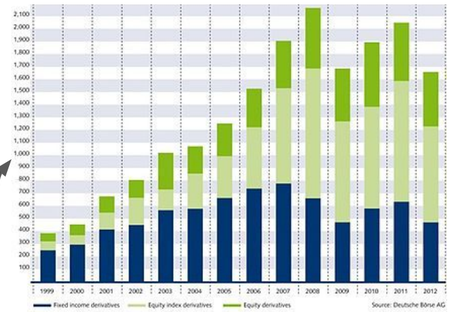
FIWARE

Interoperability levels

Source: Joinup

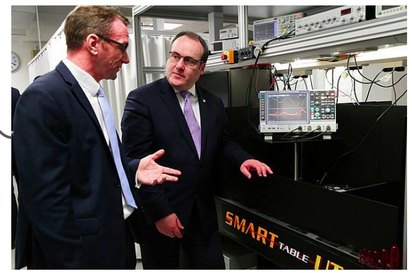
- Legal: out-of-scope
- Organizational : out-of-scope
- **Semantic interoperability:**
- Technical interoperability: NGSI LD

Standardization in a digital market



Emerging markets

months



Research

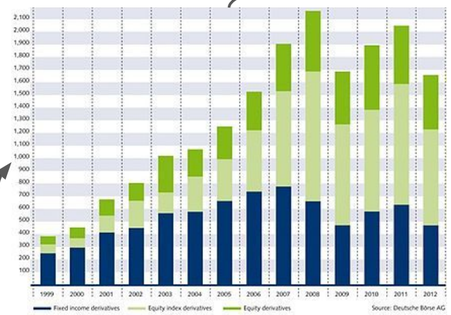
start

Smart Data Models

Standardization in a digital market

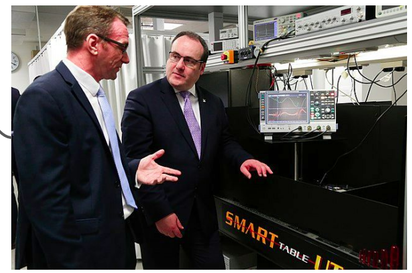
years

Standardization bodies



Emerging markets

months



Research

start

Smart Data Models

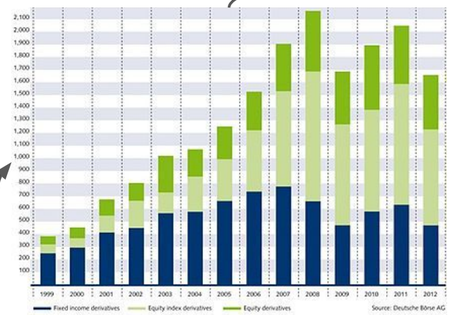
Standardization in a digital market

8 years

years

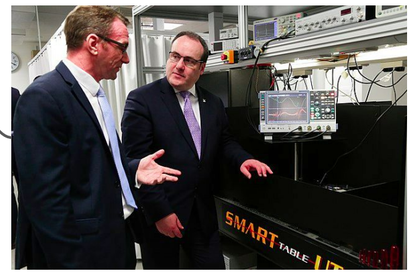
Standardization bodies

years



Emerging markets

months



Research

Current markets

start

end

Smart Data Models

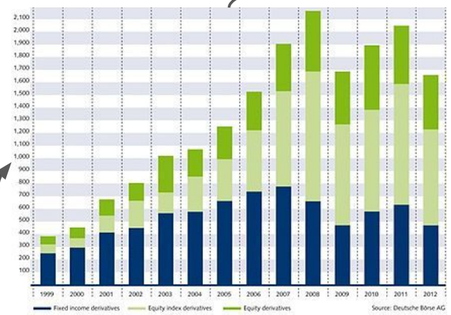
Standardization in a digital market

8 years

years

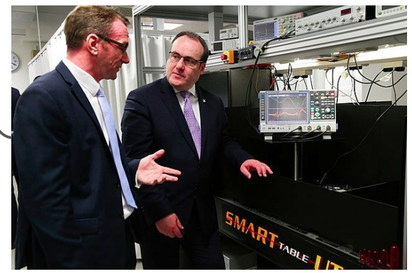
Standardization bodies

years



Emerging markets

months



Research

- Divergence of solutions
- Incompatible data models. Costly interconnection
- Higher cost (time, experience) of data model definitions
- Market for main players
- Difficult convergence when standards are available
- Standards does not cover ‘last mile’

Current markets

start

end

Smart Data Models

Standardization in a digital market

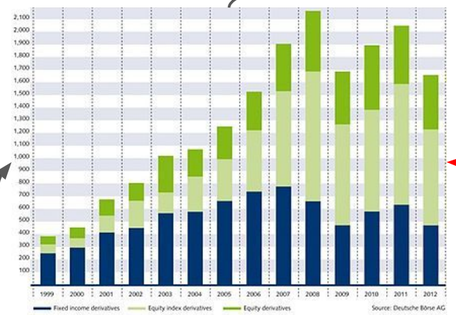
8 years

years

Standardization bodies

years

asap



Emerging markets

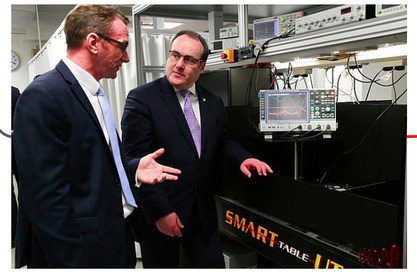
weeks

Agile standardization

days

Current markets

months



Research

weeks

start

end

Smart Data Models

Open APIs for Open Minds

Introduction

FIWARE

What is in Data Model

- **4 elements**
  The list of attributes (short names)
  Their data types
  Their definitions
  URI for linked data use

- **Examples** (mandatory*) in different formats for the
  Structure: json schema, yaml, SQL
  Payloads: json, jsonld, csv, DTDL, geojson features

- **Open licensed**
  Free use, modification and sharing

- **Always possible to contribute** to data models either with an example or based on open adopted standard

- **Services** for users
  Specification in 7 lang (EN, DE, FR, IT, SP, JP, ZH)
  Search for any element
  Map with existing ontologies
  Google sheet for creating a data model
  Generate random examples of a data model
  Python package pysmartdatamodels
  In progress export to RDF

- **Draft a data model 'on the fly'**
  Based on actual examples in json or csv with an online editor

- **Harmonization repository for standards without actual examples**
  In progress mapping of some standards (geojson, INSPIRE, etc)

FIWARE

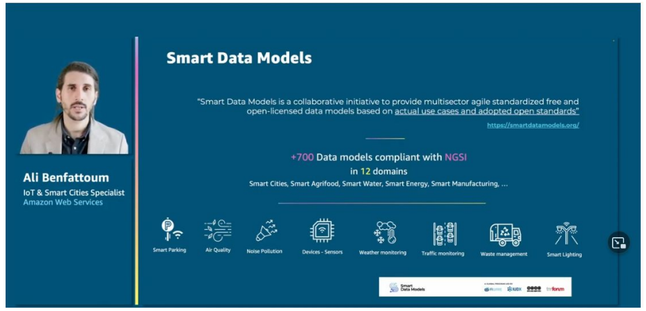
Board and what is the Smart Data Models Program

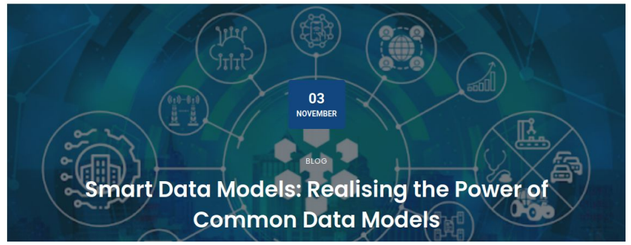
- All non-profit or public organizations
- Smart Data Models is a collaborative program to **provide data models**
  a. Free and open-licensed data models (0€ cost)
  b. Multisector
  c. Based on actual use cases and adopted open standards. Collaborative.
  d. At market speed
  e. Customizable to local needs
  f. Compatible with linked data

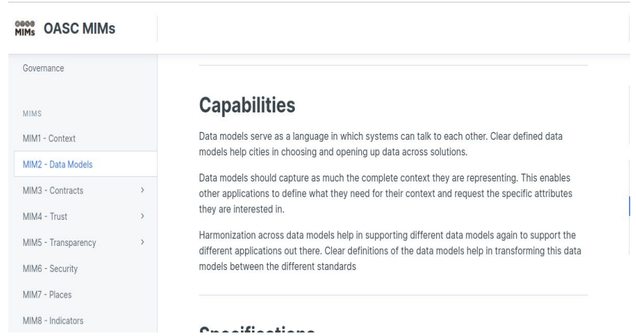
Smart Data Models

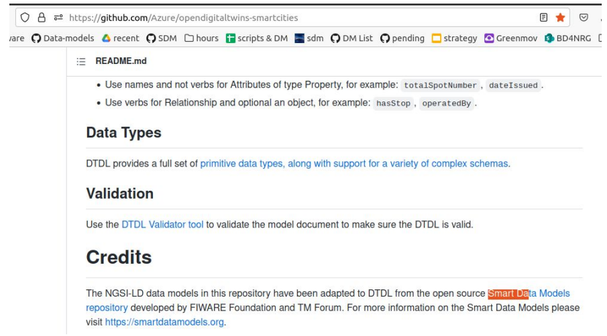
Endorsements and adoption

- **India**: IUDX adopts the smart data models
- **OASC** (Association of Smart cities): Define Minimum Interoperability Mechanisms (MIM 2 = Smart Data Models)
- **Atos**: Urban Data Platform
- **AWS**: Smart Territory Framework use smart data models
- **Microsoft**: Some of their DTDL classes founded on Smart Data models for cities









Mapped standards

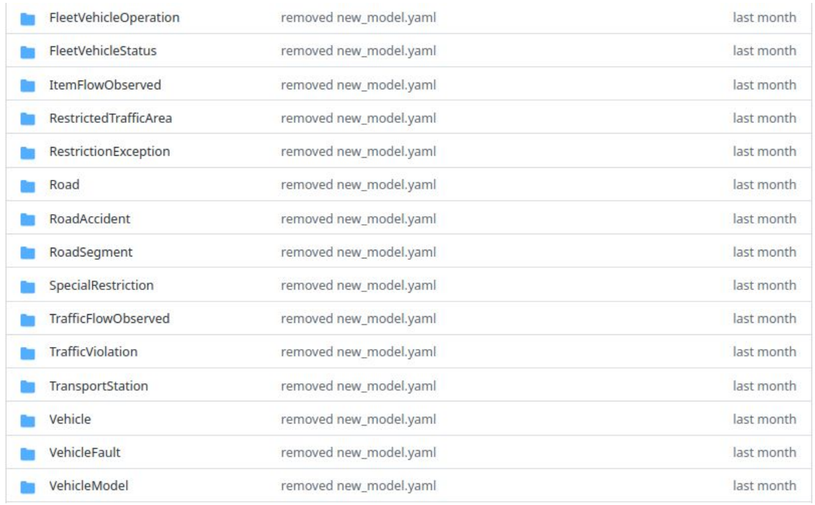
- **Smart data models** does not create dat models but adopt other proved at market. Some of them come from open and adopted standards
  - CPSV-AP: Mapping of the core vocabulary of public services 2.2.1
  - DCAT-AP: Definition of open data datasets. Mapping DCAT-AP 2.1.0
  - STAT-DCAT-AP: Definition of open data datasets. Mapping STAT-DCAT-AP 1.0.1 (in progress)
  - **Urban mobility: Maps GTFS for Urban Mobility**
  - **GBFS: Mobility for bicycles and scooters GBFS**
  - EnergyCIM: Mapping Common Information Model (CIM) specified by the IEC61970
  - EPANET: Mapping the interaction with the open source tool EPANET for water distribution
  - Frictionslessdata: Mapping frictionless data standard for interaction with open source tool OpenSDG for sustainable development goals
  - Issuetracking: Mapping elements of open311 standard
  - OPCUA: Mapping the standard OPCUA (in progress)
  - OCF: Mapping standardization by Open Connectivity Foundation
  - **OSLO: Mapping OSLO ontology for mobility**
  - HL7: Mapping version 4.3 of HL7 standard for smart health (just started)
  - **S4BLDG: SAREF for buildings ontology**
  - S4SYST: SAREF ontology for typology of systems and their inter-connections

FIWARE

Global Structure

DATA MODELS	- README with examples' generators - README pointing to specifications - Specifications in 6 languages - Schema for validation (single-source-of-truth) - Payloads Examples - Adopters	PublicService, Evidence, Cost, BusinessEvent,

Global Structure



FleetVehicleOperation	removed new_model.yaml	last month
FleetVehicleStatus	removed new_model.yaml	last month
ItemFlowObserved	removed new_model.yaml	last month
RestrictedTrafficArea	removed new_model.yaml	last month
RestrictionException	removed new_model.yaml	last month
Road	removed new_model.yaml	last month
RoadAccident	removed new_model.yaml	last month
RoadSegment	removed new_model.yaml	last month
SpecialRestriction	removed new_model.yaml	last month
TrafficFlowObserved	removed new_model.yaml	last month
TrafficViolation	removed new_model.yaml	last month
TransportStation	removed new_model.yaml	last month
Vehicle	removed new_model.yaml	last month
VehicleFault	removed new_model.yaml	last month
VehicleModel	removed new_model.yaml	last month

github repo

SUBJECT

- README with list of data models
- Shared elements of the subject
- Context.jsonld for linked data
- Contributors

Weather, CPSV-AP, STAT-DCAT-AP

DATA MODELS

- README with examples' generators
- README pointing to specifications
- Specifications in 6 languages
- Schema for validation (single-source-of-truth)
- Payloads Examples
- Adopters

PublicService, Evidence, Cost, BusinessEvent,

FIWARE

Global Structure

Domains

Smart Cities	Smart Agrifood	Smart Water	Smart Energy	Smart Logistics
Smart Robotics	Smart Sensoring	Cross sector	Smart Health	Smart Destination
Smart Environment	Smart Aeronautics	Smart Manufacturing	Incubated	Harmonization

DOMAIN

- Shared elements of the domain

Smart AgriFood, Smart cities

SUBJECT

- README with list of data models
- Shared elements of the subject
- Context.jsonld for linked data
- Contributors

Weather, CPSV-AP, STAT-DCAT-AP

DATA MODELS

- README with examples' generators
- README pointing to specifications
- Specifications in 6 languages
- Schema for validation (single-source-of-truth)
- Payloads Examples
- Adopters

PublicService, Evidence, Cost, BusinessEvent,

Global Structure

SMART DATA MODELS
Umbrella Repo

- Guides for coding new data models
- Template for new data models and examples
- Directory for scripting tools to check data models
- Official list of data models, domains and subjects

DOMAIN

- Shared elements of the domain

Smart Cities, Smart Agrifood

SUBJECT

- README with list of data models
- Shared elements of the subject
- Context.jsonld for linked data
- Contributors

Weather, CPSV-AP, STAT-DCAT-AP

DATA MODELS

- README with examples' generators
- README pointing to specifications
- Specifications in 6 languages
- Schema for validation (single-source-of-truth)
- Payloads Examples
- Adopters

PublicService, Evidence, Cost, BusinessEvent,

FIWARE

Smart Data Models: domains and subjects

DATA-MODELS

- Guides for coding new data models
- Template for new data models and examples
- Directory for scripting tools to check data models
- Inventory of domains and data models
- Inventory of attributes and terms
- @Context for json-ld

data-models
Umbrella repo

DOMAINS REPOSITORIES

Readme pointing to the list of subjects
General info or shared resources

Smart Water | Smart Cities | Smart Environment | Cross Sector | Smart Destinations | Smart Agrifood | Smart Energy | Smart Manufacturing | Smart Aeronautics | Smart Robotics

Subject 1 (sewage) | Subject 2 (parking) | Subject 3 (weather) | Subject 4 (Power Transformer)

SUBJECTS' REPOSITORIES

Readme pointing to the list of data models for the objects
Contributors.md
subject-schema.json

DATA MODELS

README.md
/doc/spec.md
/examples
schema.json
Adopters
LICENSE

LIFECYCLE MANAGEMENT REPOSITORIES

Incubated | Harmonization

FIWARE

Open APIs for Open Minds

Facing the challenges of standardization in a data market

FIWARE

Standardization in a digital market

Standardization issue	SDM approach to solution
• **Divergence of solutions**	Availability in days / weeks, you can align from the very first moment

Smart Data Models

Standardization in a digital market

Standardization issue	SDM approach to solution
• **Divergence of solutions**	Availability in days / weeks, you can align from the very first moment
• **Incompatible data models**	Data models can be used in a very flexible way the part you need (customizable to local needs). Also easy extension. Besides SDM does best effort to make every model compatible with others but also provide some tools for detecting unavoidable conflicts

Standardization in a digital market

Standardization issue	SDM approach to solution
• **Divergence of solutions**	Availability in days / weeks, you can align from the very first moment
• **Incompatible data models**	Data models can be used in a very flexible way the part you need (customizable to local needs). Also easy extension. Besides SDM does best effort to make every model compatible with others but also provide some tools for detecting unavoidable conflicts
• **Costly interconnection**	0 € accessing the data models and free training

Standardization in a digital market

Standardization issue	SDM approach to solution
• **Divergence of solutions**	Availability in days / weeks, you can align from the very first moment
• **Incompatible data models**	Data models can be used in a very flexible way the part you need (customizable to local needs). Also easy extension. Besides SDM does best effort to make every model compatible with others but also provide some tools for detecting unavoidable conflicts
• **Costly interconnection**	0 € accessing the data models and free training
• **Higher cost. experience of data model definitions**	Experience is aggregated as long as data models can be easily extended and additional notes can be added

Standardization in a digital market

Standardization issue	SDM approach to solution
• **Divergence of solutions**	Availability in days / weeks, you can align from the very first moment
• **Incompatible data models**	Data models can be used in a very flexible way the part you need (customizable to local needs). Also easy extension. Besides SDM does best effort to make every model compatible with others but also provide some tools for detecting unavoidable conflicts
• **Costly interconnection**	0 € accessing the data models and free training
• **Higher cost. experience of data model definitions**	Experience is aggregated as long as data models can be easily extended and additional notes can be added
• **Higher cost, time, of data model definitions**	Data models are publicly available at GitHub.com

Standardization in a digital market

Standardization issue	SDM approach to solution
• **Divergence of solutions**	Availability in days / weeks, you can align from the very first moment
• **Incompatible data models**	Data models can be used in a very flexible way the part you need (customizable to local needs). Also easy extension. Besides SDM does best effort to make every model compatible with others but also provide some tools for detecting unavoidable conflicts
• **Costly interconnection**	0 € accessing the data models and free training
• **Higher cost. experience of data model definitions**	Experience is aggregated as long as data models can be easily extended and additional notes can be added
• **Higher cost, time, of data model definitions**	Data models are publicly available at GitHub.com
• **Market for main players**	Board members are either non profit or public entities

Standardization in a digital market

Standardization issue	SDM approach to solution
• **Divergence of solutions**	Availability in days / weeks, you can align from the very first moment
• **Incompatible data models**	Data models can be used in a very flexible way the part you need (customizable to local needs). Also easy extension. Besides SDM does best effort to make every model compatible with others but also provide some tools for detecting unavoidable conflicts
• **Costly interconnection**	0 € accessing the data models and free training
• **Higher cost. experience of data model definitions**	Experience is aggregated as long as data models can be easily extended and additional notes can be added
• **Higher cost, time, of data model definitions**	Data models are publicly available at GitHub.com
• **Market for main players**	Board members are either non profit or public entities
• **Difficult convergence when standards are available**	The specification are open license and standardization bodies can use them (translated in 7 languages)

Standardization in a digital market

Standardization issue	SDM approach to solution
• **Divergence of solutions**	Availability in days / weeks, you can align from the very first moment
• **Incompatible data models**	Data models can be used in a very flexible way the part you need (customizable to local needs). Also easy extension. Besides SDM does best effort to make every model compatible with others but also provide some tools for detecting unavoidable conflicts
• **Costly interconnection**	0 € accessing the data models and free training
• **Higher cost. experience of data model definitions**	Experience is aggregated as long as data models can be easily extended and additional notes can be added
• **Higher cost, time, of data model definitions**	Data models are publicly available at GitHub.com
• **Market for main players**	Board members are either non profit or public entities
• **Difficult convergence when standards are available**	The specification are open license and standardization bodies can use them (translated in 7 languages)
• **Standards does not cover ‘last mile’**	Some standards does not have final details (SDM provides an implementation)

Smart Data Models

Smart Data Models: Differential factors

Open Licensed

Free Use

Free Modification

Free Sharing

Credit to Authors

Agile Standardization

Collaborative

Complementary to classical standardization

Practical use of the 7 principles

Quick evolution

We can meet market speed

Publication time: 5-8 minutes

Backwards compatible

Linked data compatible

We provide immediate long URL for all the terms

Mappable with existing ontologies

Real use

Mandatory use in real systems

Examples json, jsonld included

Exports other formats csv, yaml, SQL, DTDL, geojson features, etc

Automation is the base for being sustainable and agile

FIWARE

Open APIs for Open Minds

Current Status

FIWARE

Current status (domains, subjects & data models)

1	Smart Energy	424
2	Smart Cities	145
3	Smart Sensoring*	138
4	Cross Sector	95
5	Smart Water	38
6	Smart Environment	31

7	Smart Agrifood	28
8	Smart Destination	15
9	Smart Aeronautics	13
10	Smart Robotics	12
11	Smart Health	11
12	Smart Logistics	4
13	Smart Manufacturing	3

Updated 9-6-23

* Many sensors are specific from other domains but not counted there
Smart cities 27, Health 19, Environment 12, Energy 5, Water 4, Agrifood 1, Robotics 1

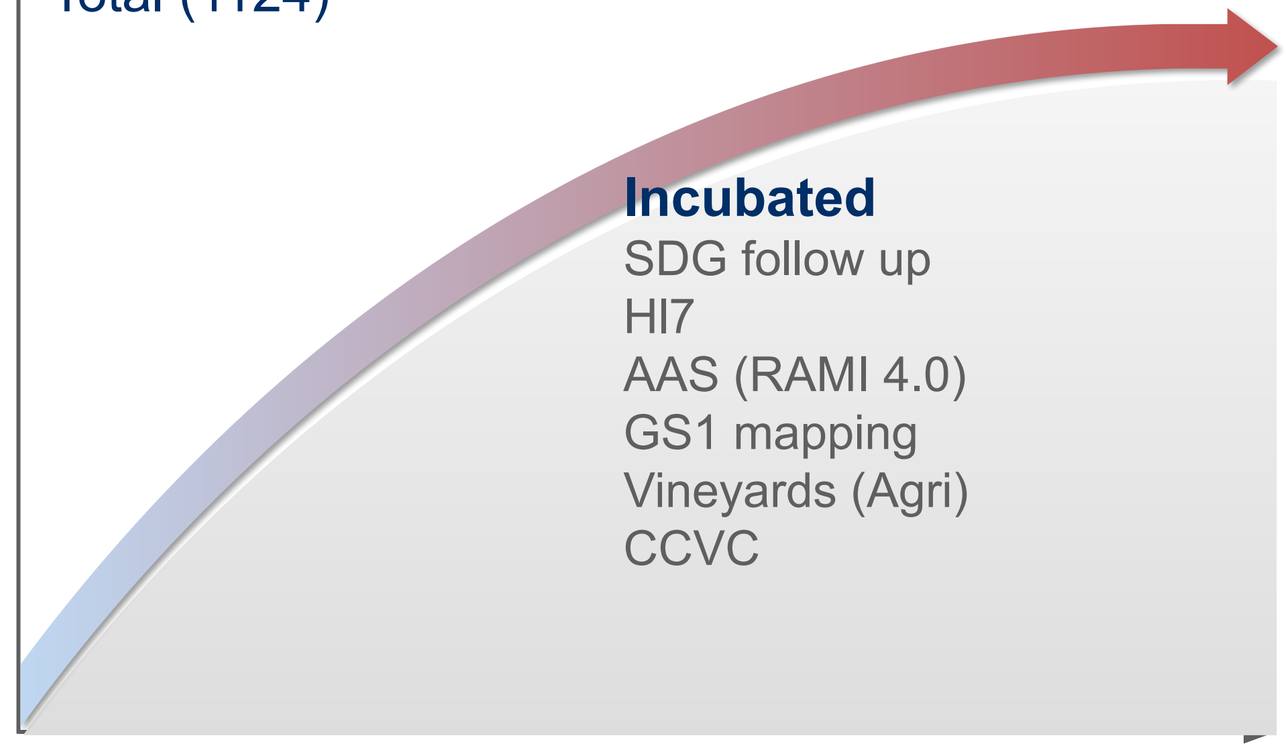
FIWARE

Current status: New subjects new data models

13 domains (Groups of subjects)
67 subjects (Groups of data models)

1	SAREF
2	S4SYST
3	S4BLDG
4	VerifiableCredentials
5	WaterDistribution
6	DataQuality

905 Official Smart Data Models
219 on the queue to be accepted*
Total (1124)



* Not all of them will become official

Updated 9-6-23

FIWARE

Contributors and dissemination

- 150 active contributors
- 278 contribution in data models
- 24 services to contributors in data models

- Contributors belong to 100 different organizations
- Terms available for search 20.960
- Documented adopters 282

- Every term in data models has an associated page https://smartdatamodels.org/*Subject/term*
- Google finds 669 pages in smartdatamodels.org

Updated 9-6-23

FIWARE

Open APIs for Open Minds

Data Model Contribution

FIWARE

Contribution to SDM

Open and adopted

Actual examples

Adoption of open standards

Mapping standards and ontologies

- **Conditions:**
  a. **Original Standard/Ontology has to be open:**
     i. Free use
     ii. Free modification
     iii. Free sharing of the modifications
     iv. Only requesting credits to the authors

- Standard/ontology has to be adopted by the market. So capable to provide examples or organizations using it and examples of payloads

- Standard/ontology has to be documented and their data types defined*

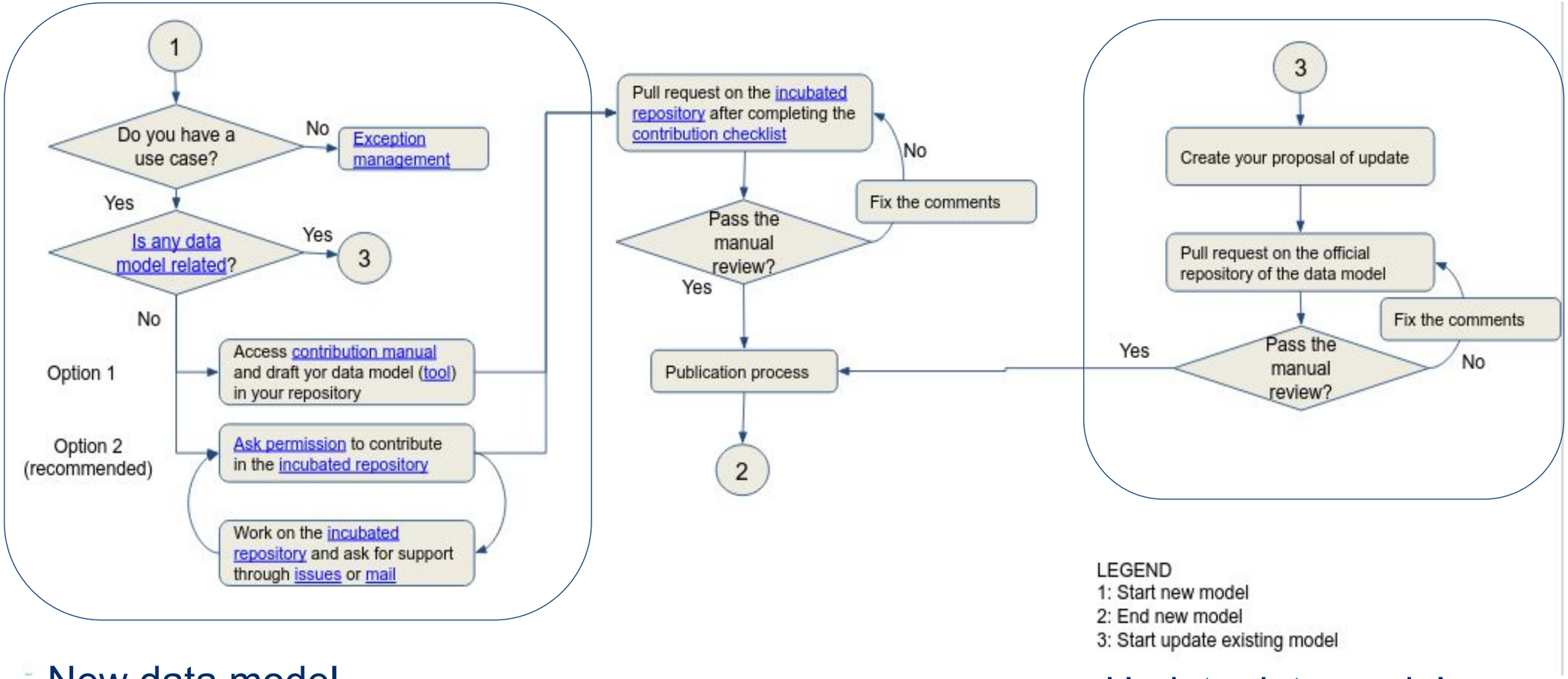
*Sometimes SDM complete the standard for the undefined sections

FIWARE

Contribution to SDM

- **Contribution manual:** https://bit.ly/contribution_manual
- **New Data Model**: Contribution can be done in the incubated repository or on your own
- **Extend existing Data Model**: Contribution can be done by PR on the official repositories
- Mandatory examples.

Contribution to SDM



New data model

Update data model

Support

Follow us on

Web	https://www.smartdatamodels.org
Mail	info@smartdatamodels.org
Twitter	@smartdatamodels
Slack	smart-data-models.slack.com
LinkedIn	https://www.linkedin.com/company/smart-data-models
GitHub	https://github.com/smart-data-models
Calendly	https://calendly.com/smartdatamodels

Open APIs for Open Minds

Services to users and contributors

FIWARE

Services (some)

Create example from keyvalues example

Input your example and it returns a drafted data model.

- Search existing attributes
- Complete all descriptions
- Draft new ones

Map data models with other ontologies and vocabularies

Automatic @context generated with IRI from SDM, pointing to actual web pages resolving

- Map user ontologies
- Notes_context.jsonld allows official IRI

Generate examples from existing data models

Chose the schema of the data model and it returns payloads compliant with the data models

- Values meet data types
- Generated randomly
- Constantly generated

Database of attributes

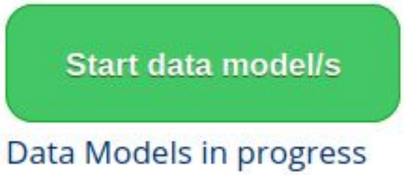
Every time a data model is versioned it is updated

- Also available in python
- pysmartdatamodels
- Searchable
- Consistency

Incubated repository

Located at https://github.com/smart-data-models/incubated

- Look for the green button at the front page
- Check the list ‘data models in progress’

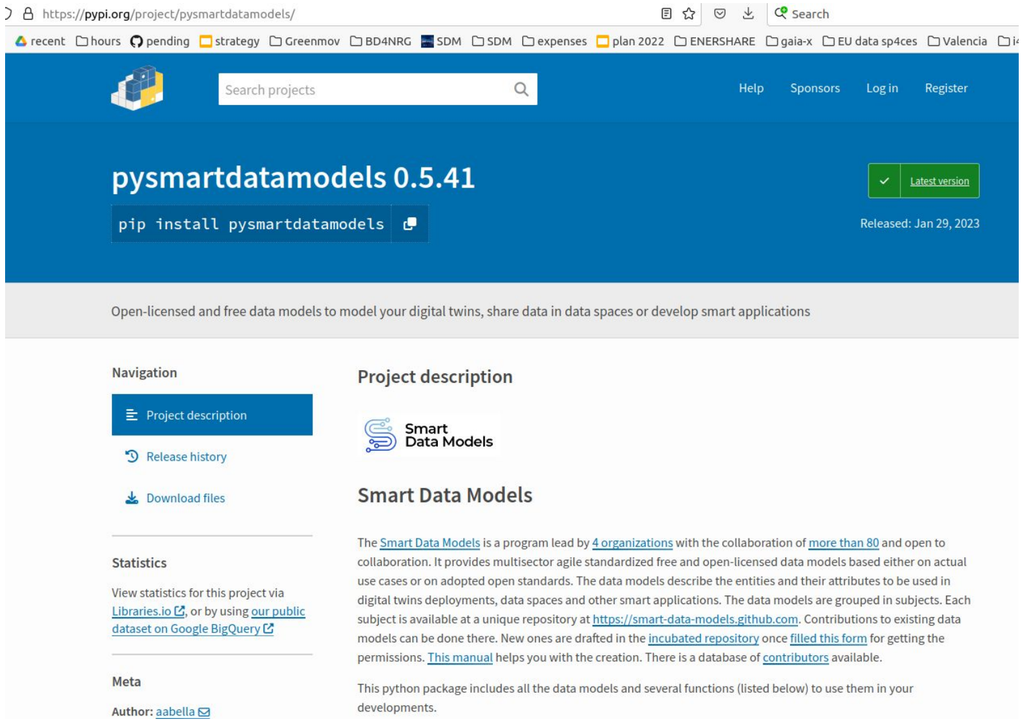


Smart Data Models

Python package: pysmartdatamodels

- https://pypi.org/project/pysmartdatamodels/
- 15 functions now
  - Load all data models
  - Load all attributes (> 19.500)
  - List all data models
  - List all subjects.
  - List the data models of a subject.
  - List description of an attribute.
  - List data-type of an attribute.
  - Give reference model for an attribute.
  - Give reference units for an attribute.
  - List the attributes of a data model.
  - List the NGSI type (Property, Relationship or Geoproperty) of the attribute.
  - Print formatted data models' attributes
  - Return the repository of a subject
  - Return the repository/ies of a data model
  - **Update the list of data model to the last version**

pysmartdatamodels 0.5.43

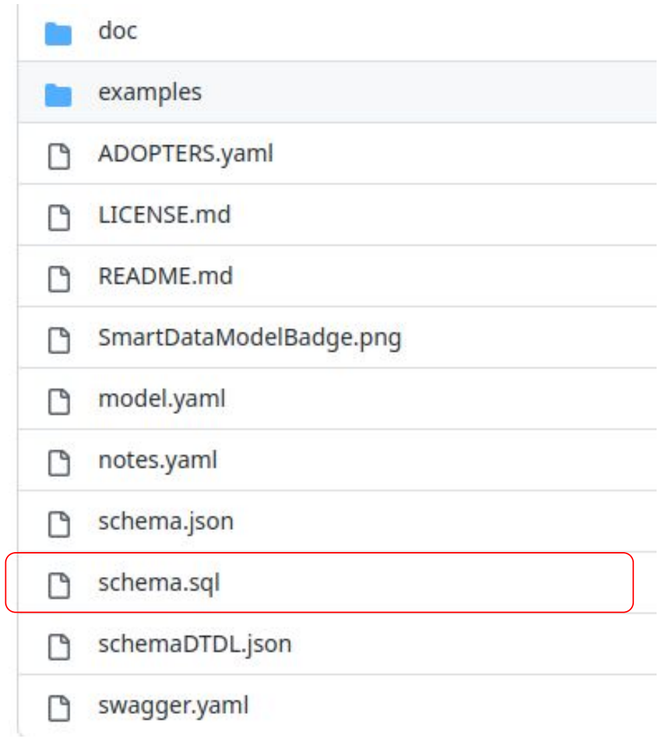


SQL export of structure PostgreSQL

- **SQL export**
  - New file schema.sql
  - Compatible with PostgreSQL
  - Some limitations for multitype attributes (few ones)
  - Automatically generated
  - Objects and arrays are coded as json types for postgreSQL

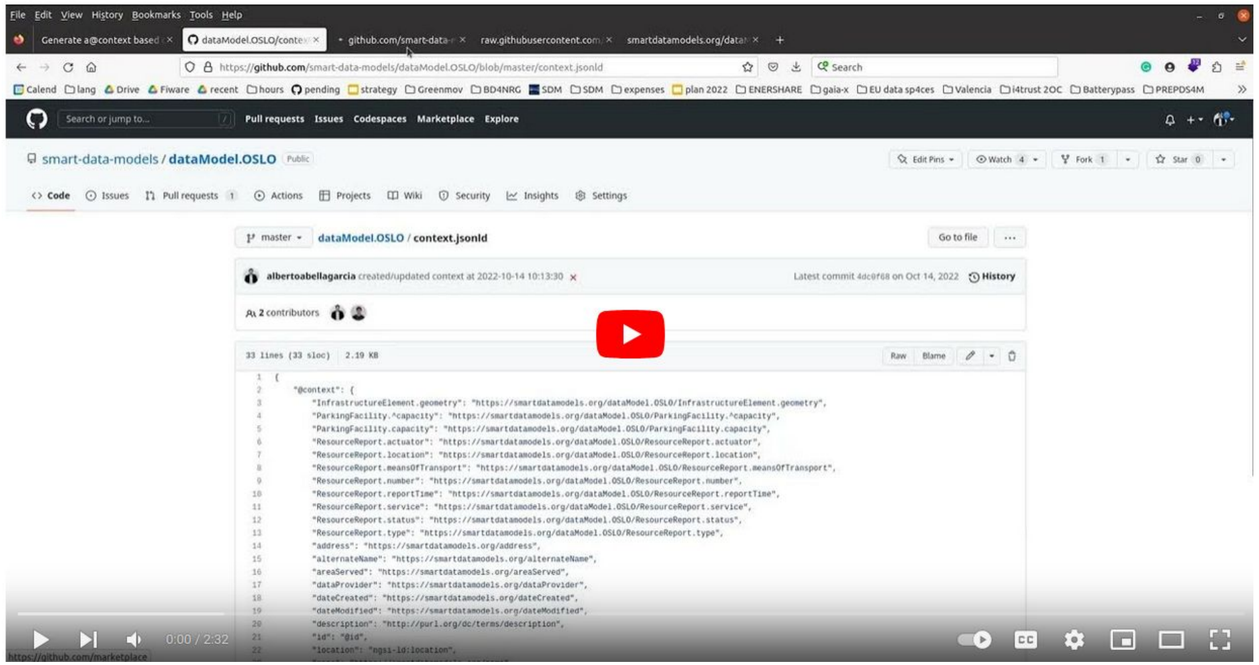
3 lines (3 sloc) | 684 Bytes — Raw | Blame

```
1  /* (Beta) Export of data model AircraftModel of the subject dataModel.Aeronautics for a postgreSQL database. Pending translation of enumerations and multityped
2  CREATE TYPE AircraftModel_type AS ENUM ('AircraftModel');
3  CREATE TABLE AircraftModel (address json, alternateName text, areaServed text, capacity integer, ceiling text, codeIATA text, codeICAO text, dataProvider text,
```



New data services. Linked data customization

- **Customization service for officially derived subjects (linked data)**
  - New file notes_context.jsonld
  - Allows to use the original IRI instead of the SDM automatically generated ones
  - Can be customized with the service for mapping external ontologies
  - Video https://www.youtube.com/watch?v=Za9fChSn9yg



Open APIs for Open Minds

Agile standardization

FIWARE

MAS: Manifesto for Agile Standardization

- Principles extracted from the lessons learnt by doing
- Official manifesto available

PRINCIPLES

0. Don't just standardize, be agile and standardize

1. Do not reinvent the wheel

2. Normalize real cases

3. Be open

4. Don't be overly specific

5. Flat not Deep

6. Sustainability is key

Smart Data Models

Open APIs for Open Minds

Life visit

FIWARE

Contents of a data model

- Visit https://smartdatamodels.org and chose one data model

Schema	• Name: schema.json. Location: Root of the subject. / • File in json schema format containing the attributes of the data model, its data types and the description of every attribute. See next sections of this manual. • Template: schema.json
Examples	• Names: (NGSIv2) example.json and example-normalized.json, (NGSI-LD) example.jsonld, example-normalized.jsonld. Location: /examples • 1 examples of this data model. Either NGSIv2 (key-values and normalized), named 2 for NGSI-LD (key-values and normalized) • Templates: example.json, example-normalized.json, example.jsonld, example-normalized.jsonld.
notes.yaml	• Name: notes.yaml. Location: / • Customization messages for specification. Four zones, header, middle and footer and README. Possible empty. • Template: notes.yaml
ADOPTERS.yaml	• Name: ADOPTERS.yaml. Location: / • Use cases of the data model. See the template. Mandatory existence but possible empty (discouraged). • Template: ADOPTERS.yaml
CONTRIBUTORS.yaml	• Name: CONTRIBUTORS.yaml. Location: Root of the subject. / • Authors of any of the data models. Possible not adding author. Mandatory sign of contribution agreement. • Template: CONTRIBUTORS.yaml
notes_context.yaml	• Name: notes_context.yaml. Location: Root of the subject. ./ • Optional IRIs for the elements defined in the data models when inherited from existing standards. Optional existence. • Template: notes_context.yaml

Schema

- Identify schema.json file
- Choose one attribute
  - What data type is it?
  - Identify if it is a property / relationship / geoproperty
  - Does it have a model? (Model:”)
  - Does it have an enumeration? (Enum:”)
  - Does it have recommended units? (Units:”)
- Have it a schemaVersion
- Have it modeTags?

Additional documents

In your chosen data model

- Visit examples directory.
- notes.yaml. (at data model level). Sections.
- notes.yaml. (at subject level). Sections.
- Notes_context.json (only at OSLO subject)
- Find ADOPTERS.yaml (who is in it?)
- Find CONTRIBUTORS.yaml at the subject’s root

Additional documents

In your chosen data model

- README at data model level
- README at subject level
- Model.yaml
- Swagger.yaml
- Csv examples
- DTDL.json
- Context.jsonld at subject level

Services for generating schemas

Create a key values and a csv payload with:

- An attribute ‘id’
- Another attribute ‘type’
- A couple of other attributes

```
{
    “id”: “ngsi-ld:ABDC:001”
    “type”: “test”,
    “attrib1”: “my value”
    “attrib2”: 3
}
```

```
id, type, attrib1, attrib2
“ngsi-ld:ABDC:001”, “test”, “my value” 3
}
```

Home -> Tools menu ->

- Generator of data model from json example
- Create data model from csv payload

Services for generating schemas

Take the raw url of one schema.json

- Check with the service

Sections:
- documentationStatusOfProperties
- schemaDiagnose
- alreadyUsedProperties
- availableProperties

Home -> Tools menu ->
- Data model documentation checker

Smart Data Models

Open APIs for Open Minds

Annexes

FIWARE

Open APIs for Open Minds

Data models moderation

FIWARE

Data model moderation

1. Provide help
2. Check required conditions
   a. Real use case. We don’t care about ontologies if there is not an actual use case
   b. Have examples
   c. Have definitions of the attributes
   d. Willing to share with others (open license)
3. Minimum knowledge about json schema
   a. Service of editor home -> tools -> Data Model Editor
   b. Life session (calendly)
4. Check other issues
   a. Is it a general payload (applicable to other use cases). Modifications
   b. Is it inherited from an existing, **open** and adopted regulation / standardization
   c. Validate schema with examples
5. Launch publication
   a. Procedure for new data model
   b. Procedure for new subject

Open APIs for Open Minds

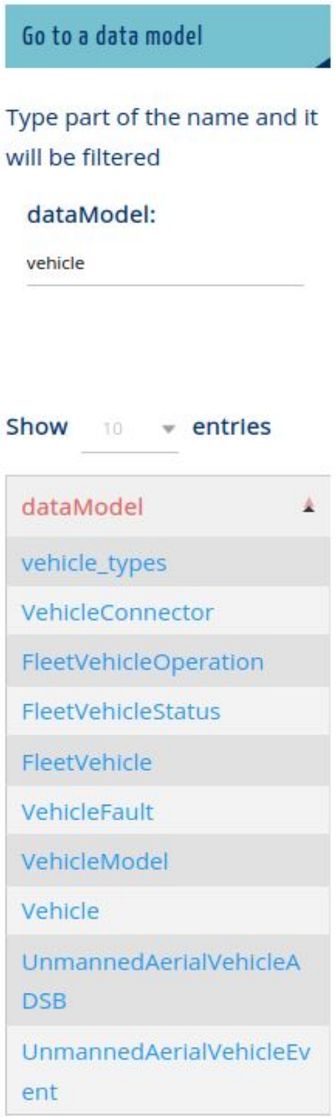
One example: Vehicle

FIWARE

One example: Vehicle

- Specification readable by humans (markdown)
- Located in a github repository
- Searchable also from smartdatamodels.org
- But also a json schema (single-source-of-truth)

```
{
    "$schema": "http://json-schema.org/schema#",
    "$schemaVersion": "0.0.1",
    "modelTags": "",
    "$id": "https://smart-data-models.github.io/dataModel.Environment/SeaConditions/schema.json",
    "title": "Sea Conditions schema",
    "description": "This entity contains a harmonised geographic description of sea conditions",
    "type": "object",
    "allOf": [
        {
            "$ref": "https://smart-data-models.github.io/data-models/common-schema.json#/definitions/GSMA-Commons"
        },
        {
            "$ref": "https://smart-data-models.github.io/data-models/common-schema.json#/definitions/Location-Commons"
        },
        {
            "properties": {
                "type": {
                    "type": "string",
                    "enum": [
                        "SeaConditions"
                    ],
                    "description": "Property. NGSI-LD Entity Type. It has to be SeaConditions"
                },
                "waveLevel": {
                    "type": "number",
                    "minimum": 0,
                    "maximum": 9,
                    "description": "Property. Model:'https://schema.org/Number'. Units:'Douglas sea scale'. It indicate
                },
                "surfaceTemperature": {
```



One example: Vehicle (1/6)

- `address[object]` : The mailing address . Model: https://schema.org/address
- `alternateName[string]` : An alternative name for this item
- `annotations[array]` : Annotations about the item . Model: https://schema.org/Text
- `areaServed[string]` : The geographic area where a service or offered item is provided . Model: https://schema.org/Text
- `bearing[number]` : Gives the vehicle GPS angle measured in a clockwise direction from the True North. SameAs 'bearing' field from GTFS Realtime message-Position(https://developers.google.com/transit/gtfs-realtime/reference#message-position) . Model: https://schema.org/Number
- `cargoWeight[number]` : Current weight of the vehicle's cargo . Model: https://schema.org/Number
- `category[array]` : Vehicle category(ies) from an external point of view. This is different than the vehicle type (car, lorry, etc.) represented by the `vehicleType` property. Enum:'municipalServices, nonTracked, private, public, specialUsage, tracked'. Tracked vehicles are those vehicles which position is permanently tracked by a remote system. Or any other needed by an application They incorporate a GPS receiver together with a network connection to periodically update a reported position (location, speed, heading ...). . Model: https://schema.org/Text
- `color[string]` : The color of the product . Model: https://schema.org/color
- `currentTripCount[number]` : The current count of trips made by the vehicle corresponding to this observation on the given day of operation. . Model: https://schema.org/Number
- `dataProvider[string]` : A sequence of characters identifying the provider of the harmonised data entity.
- `dateCreated[string]` : Entity creation timestamp. This will usually be allocated by the storage platform.
- `dateFirstUsed[string]` : Timestamp which denotes when the vehicle was first used . Model: https://schema.org/DateTime.
- `dateModified[string]` : Timestamp of the last modification of the entity. This will usually be allocated by the storage platform.
- `dateVehicleFirstRegistered[string]` : The date of the first registration of the vehicle with the respective public authorities . Model: https://schema.org/dateVehicleFirstRegistered.

Name of the attribute

Data type

Description

Model

FIWARE

One example: Vehicle (2/6)

- `description[string]` : A description of this item
- `deviceBatteryStatus[string]` : Gives the Battery charging status of the reporting device. Enum:'connected, disconnected'. . Model: https://schema.org/Text
- `deviceSimNumber[string]` : Gives the SIM number of the device in the vehicle. . Model: https://schema.org/Text
- `emergencyVehicleType[string]` : Type of emergency vehicle corresponding to this observation. Enum:'policeCar, policeMotorcycle, policeVan, policeSWAT, fireEngine, waterTender, airAmbulance, ambulance, motorcycleAmbulance, rescueVehicle, hazardousMaterialsApparatus, towTruck . Model: https://schema.org/Text
- `feature[array]` : Feature(s) incorporated by the vehicle. Enum:' abs, airbag, alarm, backCamera, disabledRamp, gps, internetConnection, overspeed, proximitySensor, wifi'. Or any other needed by the application. In order to represent multiple instances of a feature it can be used the following syntax: `<feature>,<occurences>`. For example, a car with 4 airbags will be represented by `airbag,4` . . Model: https://schema.org/Text
- `fleetVehicleId[string]` : The identifier of the vehicle in the context of the fleet of vehicles to which it belongs . Model: https://schema.org/Text.
- `fuelEfficiency[number]` : The distance traveled per unit of fuel used, commonly in kilometers per liter (km/L). . Model: https://schema.org/Number
- `fuelFilled[number]` : Amount of fuel filled in liters to the vehicle corresponding to this observation. . Model: https://schema.org/Number
- `fuelType[string]` : The type of fuel suitable for the engine or engines of the vehicle corresponding to this observation. . Model: https://schema.org/Text
- `heading[*]` : Denotes the direction of travel of the vehicle and is specified in decimal degrees, where 0 <= `heading` < 360, counting clockwise relative to the true north. If the vehicle is stationary (i.e. the value of the `speed` attribute is `0`), then the value of the heading attribute must be equal to `-1` . Model: https://schema.org/Number
- `id[*]` : Unique identifier of the entity
- `ignitionStatus[boolean]` : Gives the ignition status of the vehicle. True means ignited . Model: https://schema.org/Boolean
- `image[string]` : An image of the item . Model: https://schema.org/URL

FIWARE

One example: Vehicle (3/6)

- `license_plate[string]` : Gives the License Plate number of the vehicle. SameAs: license_plate field from GTFS Realtime message-VehicleDescriptor (https://developers.google.com/transit/gtfs-realtime/reference#message-vehicledescriptor)' . Model: https://schema.org/Text
- `location[*]` : Geojson reference to the item. It can be Point, LineString, Polygon, MultiPoint, MultiLineString or MultiPolygon
- `mileageFromOdometer[number]` : The total distance travelled by the particular vehicle since its initial production, as read from its odometer . Model: https://schema.org/mileageFromOdometer.
- `municipalityInfo[object]` : Municipality information corresponding to this observation. . Model: https://schema.org/Text
- `name[string]` : The name of this item.
- `observationDateTime[string]` : Last reported time of observation . Model: https://schema.org/DateTime
- `owner[array]` : A List containing a JSON encoded sequence of characters referencing the unique Ids of the owner(s)
- `previousLocation[*]` : Geojson reference to the item. It can be Point, LineString, Polygon, MultiPoint, MultiLineString or MultiPolygon
- `purchaseDate[string]` : The date the item e.g. vehicle was purchased by the current owner . Model: https://schema.org/purchaseDate.
- `refVehicleModel[*]` : Reference to a VehicleModel . Model: https://schema.org/URL
- `reportId[string]` : Unique Id assigned for the issue or report or feedback or transaction corresponding to this observation. . Model: https://schema.org/Text
- `seeAlso[*]` : list of uri pointing to additional resources about the item
- `serviceOnDuty[string]` : Nature of service provided by emergency vehicle corresponding to this observation. True indicates the emergency vehicle corresponding to this observation is attending to/ servicing to an emergency call of duty and is False otherwise. . Model: https://schema.org/Boolean
- `serviceProvided[array]` : Service(s) the vehicle is capable of providing or it is assigned to. Enum:'auxiliaryServices, cargoTransport, construction, fairground, garbageCollection, goodsSelling, maintenance, parksAndGardens, roadSignalling, specialTransport, streetCleaning, streetLighting, urbanTransit, wasteContainerCleaning'. Or any other value needed by an specific application. . Model: https://schema.org/Text

FIWARE

One example: Vehicle (4/6)

- `license_plate[string]` : Gives the License Plate number of the vehicle. SameAs: license_plate field from GTFS Realtime message-VehicleDescriptor (https://developers.google.com/transit/gtfs-realtime/reference#message-vehicledescriptor)' . Model: https://schema.org/Text
- `location[*]` : Geojson reference to the item. It can be Point, LineString, Polygon, MultiPoint, MultiLineString or MultiPolygon
- `mileageFromOdometer[number]` : The total distance travelled by the particular vehicle since its initial production, as read from its odometer . Model: https://schema.org/mileageFromOdometer.
- `municipalityInfo[object]` : Municipality information corresponding to this observation. . Model: https://schema.org/Text
- `name[string]` : The name of this item.
- `observationDateTime[string]` : Last reported time of observation . Model: https://schema.org/DateTime
- `owner[array]` : A List containing a JSON encoded sequence of characters referencing the unique Ids of the owner(s)
- `previousLocation[*]` : Geojson reference to the item. It can be Point, LineString, Polygon, MultiPoint, MultiLineString or MultiPolygon
- `purchaseDate[string]` : The date the item e.g. vehicle was purchased by the current owner . Model: https://schema.org/purchaseDate.
- `refVehicleModel[*]` : Reference to a VehicleModel . Model: https://schema.org/URL
- `reportId[string]` : Unique Id assigned for the issue or report or feedback or transaction corresponding to this observation. . Model: https://schema.org/Text
- `seeAlso[*]` : list of uri pointing to additional resources about the item
- `serviceOnDuty[string]` : Nature of service provided by emergency vehicle corresponding to this observation. True indicates the emergency vehicle corresponding to this observation is attending to/ servicing to an emergency call of duty and is False otherwise. . Model: https://schema.org/Boolean
- `serviceProvided[array]` : Service(s) the vehicle is capable of providing or it is assigned to. Enum:'auxiliaryServices, cargoTransport, construction, fairground, garbageCollection, goodsSelling, maintenance, parksAndGardens, roadSignalling, specialTransport, streetCleaning, streetLighting, urbanTransit, wasteContainerCleaning'. Or any other value needed by an specific application. . Model: https://schema.org/Text

One example: Vehicle (5/6)

- `serviceStatus[string]` : Vehicle status (from the point of view of the service provided, so it could not apply to private vehicles). `parked` : Vehicle is parked and not providing any service at the moment. `onRoute` : Vehicle is performing a mission. A comma-separated modifier(s) can be added to indicate what mission is currently delivering the vehicle. For instance `onRoute,garbageCollection` can be used to denote that the vehicle is on route and in a garbage collection mission. 'broken' : Vehicle is suffering a temporary breakdown. `outOfService` : Vehicle is on the road but not performing any mission, probably going to its parking area. Enum:'broken, onRoute, outOfService, parked' . Model: https://schema.org/DateTime
- `source[string]` : A sequence of characters giving the original source of the entity data as a URL. Recommended to be the fully qualified domain name of the source provider, or the URL to the source object.
- `speed[*]` : Denotes the magnitude of the horizontal component of the vehicle's current velocity and is specified in Kilometers per Hour. If provided, the value of the speed attribute must be a non-negative real number. `-1` MAY be used if speed is transiently unknown for some reason . Model: https://schema.org/Number
- `tripNetWeightCollected[number]` : The net weight collected by the vehicle corresponding to this observation at the end of the trip. . Model: https://schema.org/Number
- `type[string]` : NGSI Entity type. It has to be Vehicle
- `vehicleAltitude[string]` : Gives the current altitude of the vehicle using GPS . Model: https://schema.org/Text
- `vehicleConfiguration[string]` : A short text indicating the configuration of the vehicle, e.g. '5dr hatchback ST 2.5 MT 225 hp' or 'limited edition' . Model: https://schema.org/vehicleConfiguration.
- `vehicleIdentificationNumber[string]` : The Vehicle Identification Number (VIN) is a unique serial number used by the automotive industry to identify individual motor vehicles . Model: https://schema.org/vehicleIdentificationNumber.
- `vehiclePlateIdentifier[string]` : An identifier or code displayed on a vehicle registration plate attached to the vehicle used for official identification purposes. The registration identifier is numeric or alphanumeric and is unique within the issuing authority's region. Normative References: DATEXII `vehicleRegistrationPlateIdentifier` . Model: https://schema.org/Text
- `vehicleRunningStatus[string]` : Gives the Battery charging status of the reporting device. Enum:'running, waiting, stopped'. . Model: https://schema.org/Text
- `vehicleSpecialUsage[string]` : Indicates whether the vehicle is been used for special purposes, like commercial rental, driving school, or as a taxi. The legislation in many countries requires this information to be revealed when offering a car for sale. Enum:'ambulance, fireBrigade, military, police, schoolTransportation, taxi, trashManagement' . Model: https://schema.org/vehicleSpecialUsage
- `vehicleTrackerDevice[string]` : Installation status of the GPS device or the tracking device fitted to the vehicle corresponding to this observation. . Model: https://schema.org/Text

FIWARE

One example: Vehicle (6/6)

- `vehicleType[string]` : Type of vehicle from the point of view of its structural characteristics. This is different than the vehicle category . Enum:'agriculturalVehicle, anyVehicle, articulatedVehicle, bicycle, binTrolley, bus, car, caravan, carOrLightVehicle, carWithCaravan, carWithTrailer, cleaningTrolley, constructionOrMaintenanceVehicle, fourWheelDrive, highSidedVehicle, lorry, minibus, moped, motorcycle, motorcycleWithSideCar, motorscooter, sweepingMachine, tanker, threeWheeledVehicle, trailer, tram, twoWheeledVehicle, trolley, van, vehicleWithoutCatalyticConverter, vehicleWithCaravan, vehicleWithTrailer, withEvenNumberedRegistrationPlates, withOddNumberedRegistrationPlates, other'. The following values defined by *VehicleTypeEnum* and *VehicleTypeEnum2*, DATEX 2 version 2.3 and extended for other uses . Model: https://schema.org/Text
- `wardId[string]` : Ward ID of the entity corresponding to this observation. . Model: https://schema.org/Text
- `wardName[string]` : Ward name of the entity corresponding to this observation. . Model: https://schema.org/Text
- `zoneName[string]` : Zone name of the entity corresponding to this observation . Model: https://schema.org/Text

One example: Vehicle in other languages

`address[object]` : Die Postanschrift . Model: https://schema.org/a

`alternateName[string]` : Ein alternativer Name für diesen Artikel

`annotations[array]` : Anmerkungen zum Artikel . Model: https://schema.org/Tex

`areaServed[string]` : Das geografische Gebiet, in dem eine Dienstleistung oder e
https://schema.org/Text

`bearing[number]` : Gibt den GPS-Winkel des Fahrzeugs, gemessen im Uhrzeigersi
"Peilung" der GTFS-Echtzeitnachricht-Position (https://developers.google.com/tra
Model: https://schema.org/Number

`cargoWeight[number]` : Aktuelles Gewicht der Ladung des Fahrzeugs . Model: http

`category[array]` : Fahrzeugkategorie(n) aus externer Sicht. Dies ist etwas andere
die Eigenschaft "vehicleType" dargestellt wird. Enum:'municipalServices, nonTracl
Verfolgte Fahrzeuge sind Fahrzeuge, deren Position permanent von einem entfer
einen GPS-Empfänger und eine Netzverbindung, um die gemeldete Position (Star

`address[object]` : La dirección postal . Model: https://schema.org

`alternateName[string]` : Un nombre alternativo para este artículo

`annotations[array]` : Anotaciones sobre el artículo . Model: https://schema.o

`areaServed[string]` : La zona geográfica en la que se presta un servicio o se

`bearing[number]` : Indica el ángulo GPS del vehículo medido en el sentido de
que el campo "bearing" del mensaje GTFS Realtime-Position(https://develope
/reference#message-position) . Model: https://schema.org/Number

`cargoWeight[number]` : Peso actual de la carga del vehículo . Model: https://sc

`category[array]` : Categoría de vehículo(s) desde un punto de vista externo.
representado por la propiedad `vehicleType` . Enum:'municipalServices, nonTr
vehículos rastreados son aquellos cuya posición está permanentemente rastr
necesite una aplicación Incorporan un receptor GPS junto con una conexión c
reportada (ubicación, velocidad, rumbo...). . Model: https://schema.org/Text

`address[object]` : L'adresse postale . Model: https://schema.org/addres

`alternateName[string]` : Un nom alternatif pour cet élément

`annotations[array]` : Annotations sur l'élément . Model: https://schema.org/Text

`areaServed[string]` : La zone géographique où un service ou un article offert est fou

`bearing[number]` : Donne l'angle GPS du véhicule, mesuré dans le sens des aiguilles d
Identique au champ "bearing" du message GTFS Realtime-Position (https://developer
/reference#message-position). . Model: https://schema.org/Number

`cargoWeight[number]` : Poids actuel du chargement du véhicule . Model: https://schei

`category[array]` : Catégorie(s) de véhicule(s) d'un point de vue externe. Elle est diffé
etc.) représenté par la propriété `vehicleType` . Enum : 'municipalServices, nonTracke
véhicules suivis sont les véhicules dont la position est suivie en permanence par un s
ainsi qu'une connexion réseau pour mettre à jour périodiquement une position rapp

`address[object]` : L'indirizzo postale . Model: https://schema.org/ad

`alternateName[string]` : Un nome alternativo per questa voce

`annotations[array]` : Annotazioni sull'elemento . Model: https://schema.org/Text

`areaServed[string]` : L'area geografica in cui viene fornito il servizio o l'articolo of

`bearing[number]` : Indica l'angolo GPS del veicolo misurato in senso orario rispetto
messaggio di posizione in tempo reale GTFS (https://developers.google.com/trans
Model: https://schema.org/Number

`cargoWeight[number]` : Peso attuale del carico del veicolo . Model: https://schema.

`category[array]` : Categoria/e del veicolo da un punto di vista esterno. È diversa d
rappresentato dalla proprietà `vehicleType` . Enum:'municipalServices, nonTracked
tracciati sono quei veicoli la cui posizione è permanentemente tracciata da un sist
necessario a un'applicazione. Essi incorporano un ricevitore GPS e una connession

`address[object]` : 郵送先住所 . Model: https://schema.org/address

`alternateName[string]` : この項目の別称

`annotations[array]` : アイテムに関するアノテーション . Model: https://schema.org/Text

`areaServed[string]` : サービスまたは提供品が提供される地理的な地域 . Model: https://sche

`bearing[number]` : 真北から時計回りで測定した車両の GPS 角度を与える。GTFS Realtime m
Position(https://developers.google.com/transit/gtfs-realtime/reference#message-position
Model: https://schema.org/Number

`cargoWeight[number]` : 現在の車両積載重量 . Model: https://schema.org/Number

`category[array]` : 外部から見た車両カテゴリー。これは、`vehicleType` プロパティで表され
とは異なる。Enum:'municipalServices, nonTracked, private, public, specialUsage, tracked'
特別利用、追跡）。追跡車両とは、リモートシステムによって位置が常時追跡されている車両
接続を備え、報告された位置（位置、速度、方向...）を定期的に更新する。 . Model: https://s

- `address[object]` : 邮寄地址 . Model: https://schema.org/address
- `alternateName[string]` : 这个项目的一个替代名称
- `annotations[array]` : 关于该项目的注释 . Model: https://schema.org/Text
- `areaServed[string]` : 提供服务或提供项目的地理区域 . Model: https://schema.org/T
- `bearing[number]` : 提供车辆GPS的角度，以顺时针方向从真北测量。与GTFS实时信息-
realtime/reference#message-position)的 "方位 "字段相同。 . Model: https://schema
- `cargoWeight[number]` : 车辆货物的当前重量 . Model: https://schema.org/Number
- `category[array]` : 从外部角度看车辆类别。这与 "vehicleType "属性所代表的车辆类型
踪，私人，公共，特殊用途，跟踪'。追踪的车辆是那些由远程系统永久追踪位置的车辆。
收器和一个网络连接，以定期更新报告的位置（位置、速度、方向...）。 . Model: https
- `color[string]` : 产品的颜色 . Model: https://schema.org/color

FIWARE

Specification in 7 languages

- List of specifications

Specification in English

Specification in German

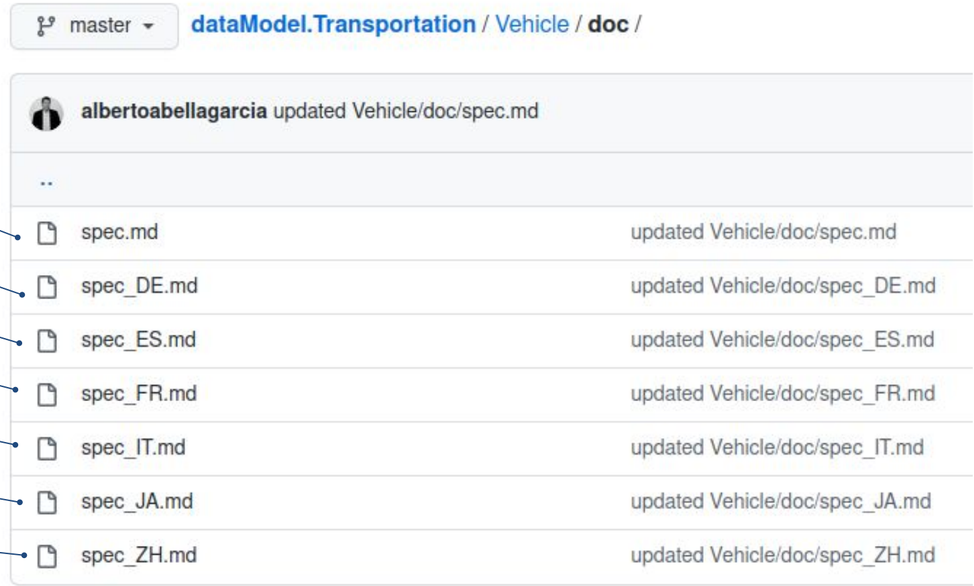
Specification in Spanish

Specification in French

Specification in Italian

Specification in Japanese

Specification in Chinese



Structure of the data model's files

- Independently of the tool used it would lead us to a repository in github with this layout

Specifications (documents in 7 languages)

Examples in various formats (json, jsonld, csv, etc)

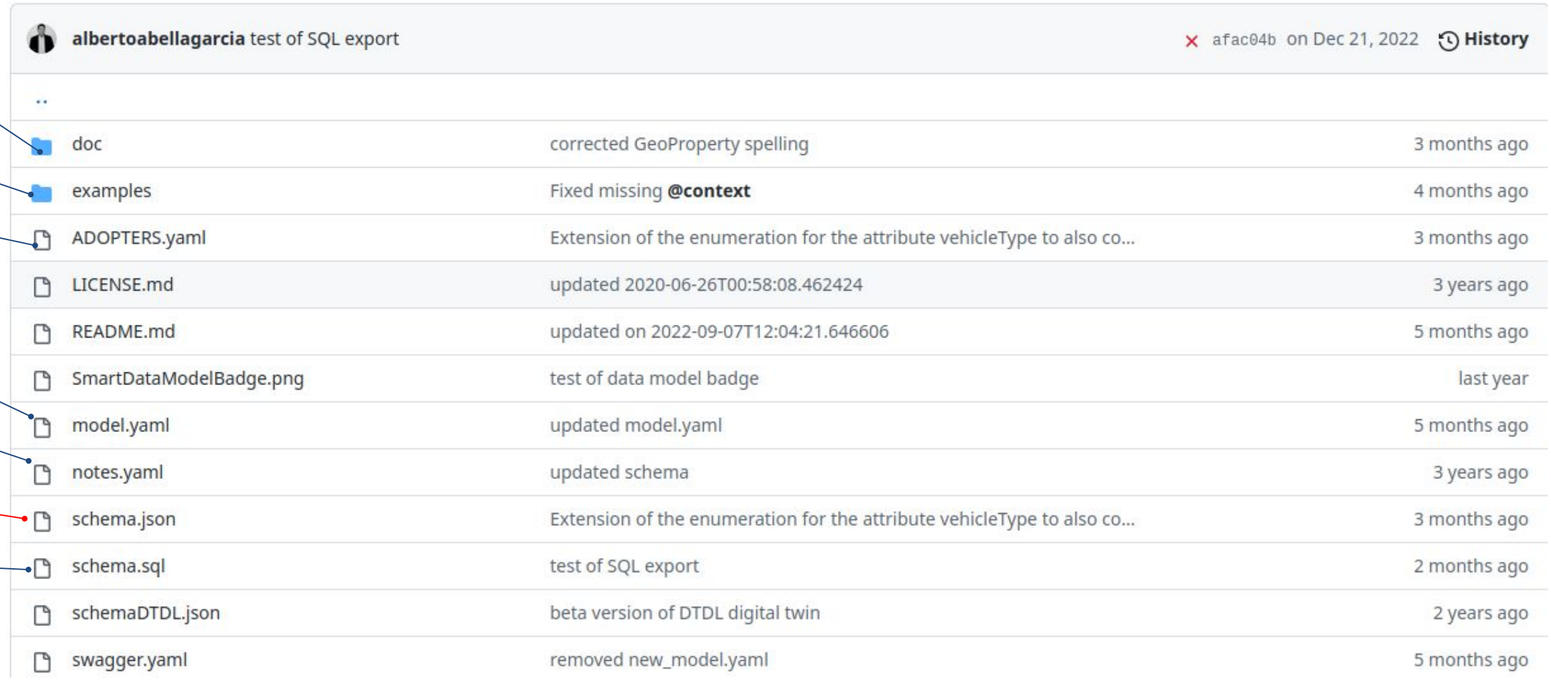
Where this data model is being used

Structure in yaml format

Texts to customize specifications

Single-source-of-truth. Schema of the data model

Schema in SQL (PostgreSQL)



FIWARE Global Summit

Find Us On

Stay up to date

JOIN OUR NEWSLETTER

Be certified and featured

FIWAREMarketplace

Hosting Partner

City of Vienna

vienna business agency

Keystone Sponsors

aws

ms.gis

Raseel

Red Hat

Media Partners

bee smart city

CITIES FORUM

CitiesToday

Our Future Water

RM RENEWABLE MATTER

Smart Cities World

the smartcity journal

Vienna, 12-13 June, 2023 | #FIWARESummit

FIWARE www.fiware.org

FIWARE
Global
Summit

Thanks!

Vienna, Austria
12-13 June, 2023
#FIWARESummit

FIWARE
Open APIs for Open Minds