Fusion Engine V0.3

DEVELOPER'S GUIDE

TABLE OF CONTENTS

1 - INTRODUCTION

1.1 - Enablers. FICONTENT and FIWARE

In order to strengthen a powerful European industry market for the upcoming years in the context of the Future Internet (FI), an FI-PPP called FIWARE was created (http://www.fiware.org/). The main outputs of this project are:

- To provide a reference platform for developing smart applications in an easy and efficient way.
- To provide a set of core (horizontal) components that build the core of this platform, upong which new (vertical) components can be built.

In order to further spread the usage of FIWARE in the European industry, new actions and use case projects were created, one of which was called FICONTENT (http://mediafi.org/) , focussed on topics such as social connected TV, smart city guides and pervasive games.

The components developed in FIWARE are called Generic Enablers (GEs), as they are envisioned for a general (horizontal) market. On the other side, components developed in FIWARE use case projects such as FICONTENT are called Specific Enablers (SEs), as they are primarily intended for special purposes (e.g. multimedia, TVs, databases, etc.).

All components (enablers) are just middleware that should facilitate big and small companies develop next generation Internet applications in order to be more competitive and/or productive. Some GEs and SEs are open source, and other can be used under specific terms of use defined by each project partner, but all of them have been built with the intention of providing helpful tools for SMEs.

1.2 - Fusion Engine Definition and Contextualization

The Fusion Engine (FE) is an open source specific enabler (SEs) developed in the framework of the FICONTENT2 project. The specific area of interest of this enabler is Smart Cities, as it provides a tool to fusion data from different data providers to be used for further smart city applications.

The Fusion Engine is about data fusion for smart cities. In this context, the most useful piece of data for general purposes is called a POI (Point of Interest). Though POIs are mostly used for navigational GPS and touristic applications, it can be easily extended to any smart city environment you can imagine. Basically, a POI is something located at a certain position with certain properties that might be of interest while developing your application.

It is important to clarify what fusion means in the context of FE and why it might be helpful. With the appearance of the smart city concept, many cities (town council) are showing strong interest in providing local open data for their citizens. Other companies are also providing more and more open data repositories (e.g. for touristic and environmental purposes). There are also other global data repositories (e.g. OSM and DBPedia) that can act as additional data sources. So many data sources might be apparently helpful, but it typically results in POI replication and multiple connections to different data sources. Why not making this process offline before? One may wish to:

- Select only those data sources that are of interest
- Select only those categories from the sources that are of interest
- Get only one single merged POI from multiple ones
- Get only one single access interface to retrieve the resulting POIs

The result of the fusion generated by the FE is a georeferenced database of POIs. We called this Open City Database (OCD) as the input data (POIs) might typically come from open data. Any way you can also use proprietary data if you comply with the corresponding license. This is much more a legal issue than a technological issue. In fact, the FE supports in its data model the multiple usage of licenses. It is up to the user to correctly (legally) exploit the data.

1.3 - Fusion Challenges

There are 3 technological challenges targeted by data fusion in general and FE in particular, as depicted in the following Figure 1 (see C in red).

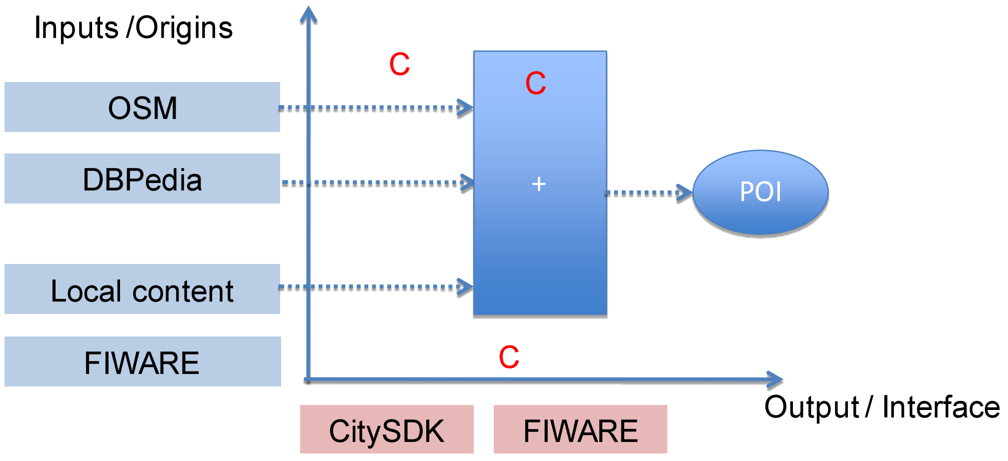


Figure 1. Fusion challenges in FiContent.

The first challenge relates to the input data. There are many data sources to be used:

- Global data sources, such as OSM and DBPedia
- Local data sources, such as the data portals of many cities (e.g. Valencia datos abiertos, opendatabcn, opendatacanarias, etc.)
- Particular data sources, such as FIWARE GEs

Note here that each data source has its own way (format and API) for providing the data. Besides, data categories are not necessarily preserved between data sources and are typically different.

The second challenge is how we are going to perform the fusion, the merging of POIs:

- How do we define that one POI from data source A and another POI from data source B refer to the same POI?
- When two or more POIs match, how do we create one single POI (e.g. which name and location do we preserve?

The third challenge is about the output of the fusion. Even if it is a georeferenced POI repository, there should be a single API access. One would think in a standard way of defining POIs. Unfortunately, there is no such. Several years ago there was a first POI draft specification by the W3C (http://www.w3.org/2010/POI/), which later moved to OCG (http://www.opengeospatial.org/projects/groups/poiswg). However, the activity within this last group is frozen for the last two years. Thus, there were basically two alternatives:

- Define a new API format and API interface. This has been the approach within FIWARE
- Use a current POI specification that most resembles the POI draft, called CitySDK (http://www.citysdk.eu/).

2 - Architecture of the Fusion Engine

The FE has two main components: (i) the frontend and (i) the service, as depicted in Figure 2.

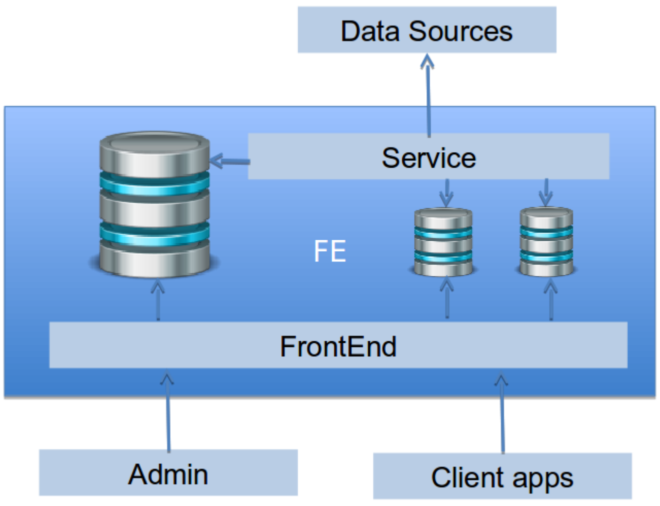


Figure 2. FE general architecture.

2.1 - FE Frontend

The FE frontend acts as interface with the outside world:

- Administrators access the frontend to set up the fusion engine and generate OCD instances
- Users and client applications access the frontend to request for POI data of a particular OCD instance

Note in Figure 2 that the FE has many different databases.

- There is a main database, called OCD_BASE, which indexes and configures all OCD instances. The data stored in this database is entered by the administrator.
- There are multiple databases, corresponding to each of the OCDs that are being built. This allows separating the information about POIs and securing data access. It also facilitates exporting operations of an OCD. Client apps access one of these OCD (small) databases through the frontend.

For client applications, there is a tiny difference with CitySDK interface. As the FE is able to generate as many OCDs as set up by the administrator, you also have to provide in the access API and 'ocdName' parameter, in order to select the city (OCD) you are interested in. The management of different databases allows introducing different access models and therefore business models. For example, an admin (SME) can offer free access to a basic OCD but pay-per-access to an enriched OCD.

2.2 - FE service

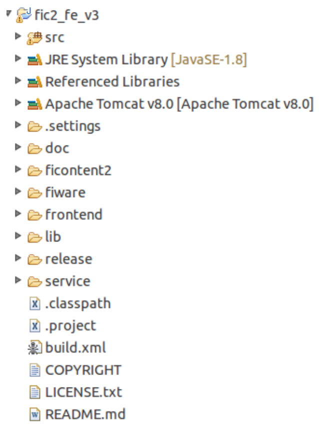
The FE service is a standalone process that reads the set up information in the OCD_BASE database and builds the different OCDs taking information from the corresponding (selected) data sources. This is an offline process that can take several minutes depending on the configuration (number of sources, number of categories, city size, etc.)

3 - DEVELOPER'S GUIDE

This guide should be useful for developers that want to go further with the FE and enhance or adapt the source code to their own needs.

3.1 - Importing the code for Eclipse. Quick View to Folders and classes

All the Fusion Engine code is available in Github:
https://github.com/satrd/poi_fusion_engine3
You can select to download the whole project (Download ZIP in the right-middle area of the page), use the git clone command to obtain the data, or just import it from Eclipse. Be sure you have git support in Eclipse. Then simply do: File -> Import -> Git -> Projects From Git > URI (Enter the Github repository url)
After all this process, you will get the following structure (see Figure 3).



```
fic2_fe_v3
  src
  JRE System Library [JavaSE-1.8]
  Referenced Libraries
  Apache Tomcat v8.0 [Apache Tomcat v8.0]
  .settings
  doc
  ficontent2
  fiware
  frontend
  lib
  release
  service
  .classpath
  .project
  build.xml
  COPYRIGHT
  LICENSE.txt
  README.md
```

Figure 3. Directory structure in Eclipse

You can browse the folder structure to find the source code (src), the documentation (doc), libraries (lib), the Frontend service (frontend) and the fusion service (service). The folders ficontent2 and fiware relates to integration and further work within both projects, as will be described later.
The file build.xml corresponds to the basic ant file for performing the build and releases (release folder). You can have a look at this file and all its 'targets', but I basically generates (target build-all) one WAR file for the Frontend service and one JAR file for the fusion service, which are placed in the release directory.

3.2 - Data sources

The FE can natively use any data source that supports the CitySDK interface, thus you need to map any input data source with the CitySDK interface. We have used it for:

- OSM: OpenStreetMaps allows obtaining data from various world zones (continents, countries, etc.)
- DBPedia : this is the structured format of (part of) Wikipedia
- Local sources : valencia datos abiertos, opendatacanarias
- POIProxy (http://mediafi.org/?portfolio=poiproxy): this enabler from FIContent2 allows accessing to geolocated social network information. FE uses it for retrieving photos from Flickr.

For the local sources, we have also used POIProxy, as it easily treat local data sources similar to social networks.
The FE data model allows using any other interface, but then you will need to develop your own connector and add it to the source code. Connectors should be written as org.upv.satrd.fic2.fe.connectors.citySDK

3.3 - Data Model

The data model represents the underlying data all Java classes are based on. This has an impact on what the code is able to do, thus if you need something different for your project you might require either changing some piece of data or even adding new tables.
The FE enabler uses several databases:

- OCD_BASE : this is the base database which is always available and keeps track of all other databases
- OCD_<ocdName> : this refers to multiple databases, each one corresponding to a new OCD

3.3.1 - OCD_BASE

The OCD_BASE database is the basis for both the Fusion service and the Frontend service. At startup, only this database is necessary to start building new OCDs. The data model is represented in Figure 4. You can find:

- The SQL file under <base_dir>/frontend/Webcontent/config/dbScripts/fe_base_reset_postgres.sql
- The PNG file under <base_dir>/doc/fe_base.png

The item ‘ocd.status’ represents the communication flag between the Fusion service and the FrontEnd service:

- When the administrator starts the fusion through the FrontEnd, the ‘ocd.status’ is set to STATUS_RUNNING_START
- The Fusion service reads periodicaly the ocd table in OCD_BASE. For those marked as STATUS_RUNNING_START, it performs the fusion and changes the value to STATUS_RUNNING

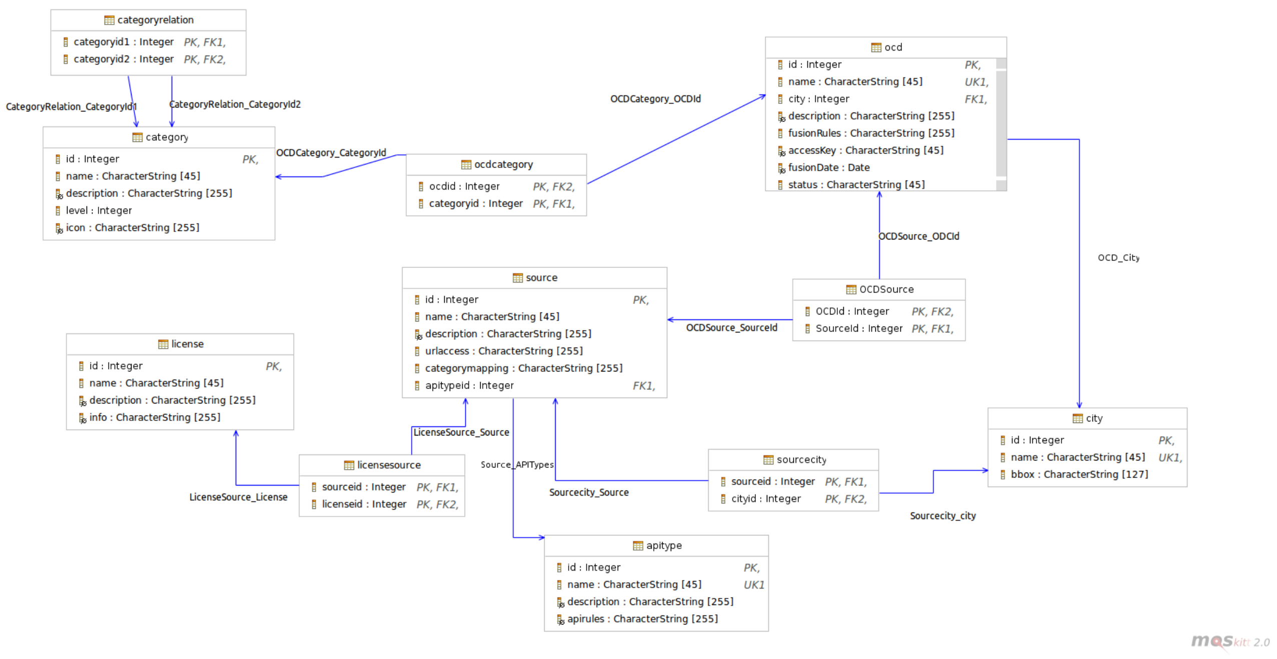

categoryrelation
categoryid1 : Integer PK, FK1,
categoryid2 : Integer PK, FK2,
CategoryRelation_CategoryId1
CategoryRelation_CategoryId2
category
id : Integer PK,
name : CharacterString [45]
description : CharacterString [255]
level : Integer
icon : CharacterString [255]
OCDCategory_CategoryId
ocdcategory
ocdid : Integer PK, FK2,
categoryid : Integer PK, FK1,
OCDCategory_OCDId
ocd
id : Integer PK,
name : CharacterString [45] UK1,
city : Integer FK1,
description : CharacterString [255]
fusionRules : CharacterString [255]
accessKey : CharacterString [45]
fusionDate : Date
status : CharacterString [45]
OCDSource_ODCId
OCD_City
OCDSource
OCDId : Integer PK, FK2,
SourceId : Integer PK, FK1,
OCDSource_SourceId
source
id : Integer PK,
name : CharacterString [45]
description : CharacterString [255]
urlaccess : CharacterString [255]
categorymapping : CharacterString [255]
apitypeid : Integer FK1,
license
id : Integer PK,
name : CharacterString [45]
description : CharacterString [255]
info : CharacterString [255]
LicenseSource_Source
licensesource
sourceid : Integer PK, FK1,
licenseid : Integer PK, FK2,
LicenseSource_License
Source_APITypes
Sourcecity_Source
sourcecity
sourceid : Integer PK, FK1,
cityid : Integer PK, FK2,
Sourcecity_city
city
id : Integer PK,
name : CharacterString [45] UK1,
bbox : CharacterString [127]
apitype
id : Integer PK,
name : CharacterString [45] UK1
description : CharacterString [255]
apirules : CharacterString [255]
maskit 2.0

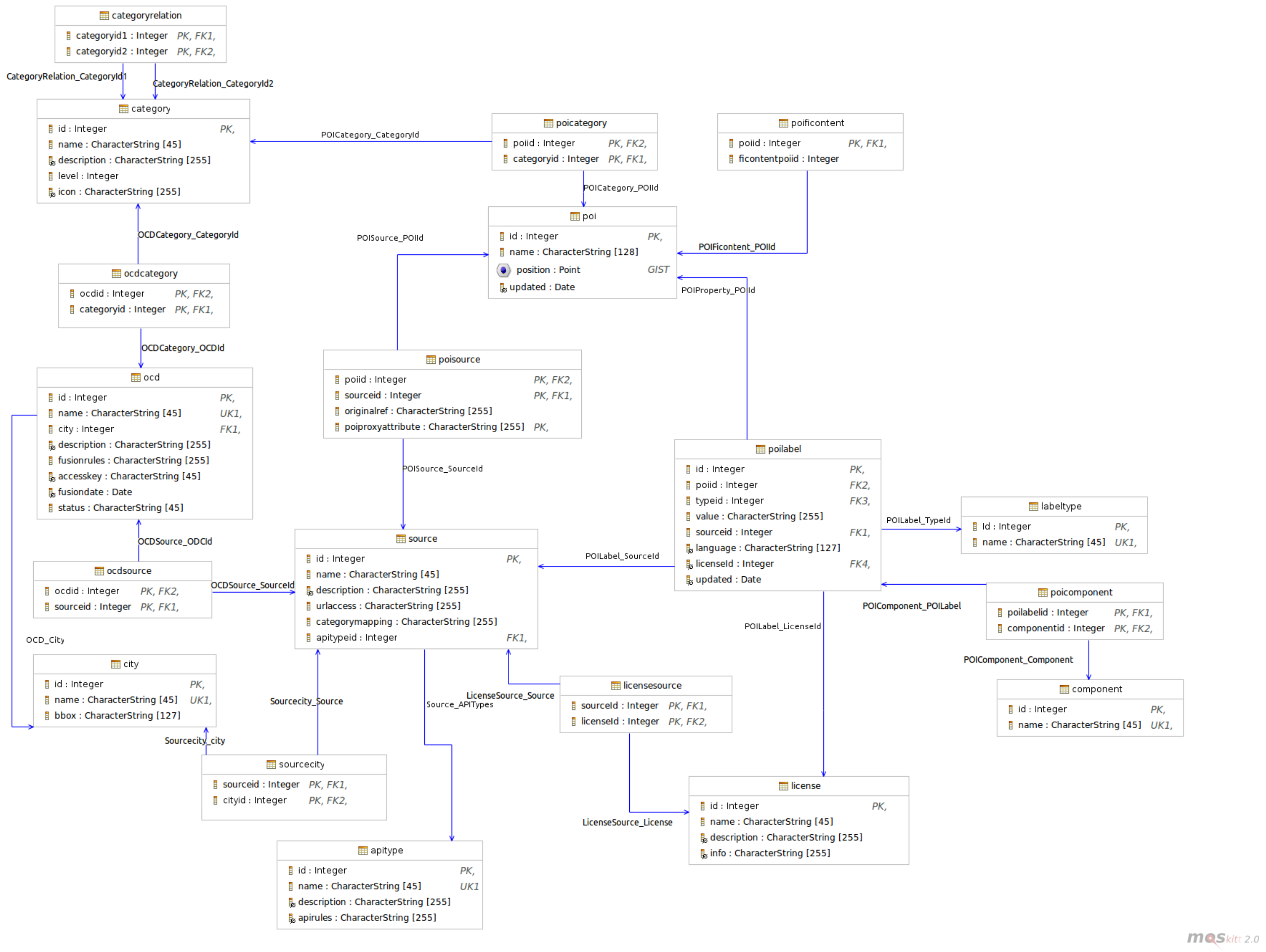
Figure 4. OCD_BASE data model

The different status values can be found in the OCD source class (see table below).

```
public static final String STATUS_NEW = "1";
public static final String STATUS_INITIALIZED = "2";
public static final String STATUS_RUNNING_START = "3"; //transitional state to mark to the service it has to start
(Frontend marks to service)
public static final String STATUS_RUNNING = "4";
public static final String STATUS_RUNNING_END_OK = "5";  //transitional state to mark to the service it has to end (Fe
Thread marks to service)
public static final String STATUS_RUNNING_END_ERR = "6";
public static final String STATUS_FINISHED_OK = "7";
public static final String STATUS_FINISHED_ERR = "8";
```

3.3.2 - OCD_<ocdName>

The OCD_<ocdName> database corresponds to a specific OCD built by the administrator. The data model replicates the same structure as OCD_BASE, but additionally adds new tables (e.g. poi)


categoryrelation
categoryid1 : Integer PK, FK1,
categoryid2 : Integer PK, FK2,
CategoryRelation_CategoryId1
CategoryRelation_CategoryId2
category
id : Integer PK,
name : CharacterString [45]
description : CharacterString [255]
level : Integer
icon : CharacterString [255]
POICategory_CategoryId
poicategory
poiid : Integer PK, FK2,
categoryid : Integer PK, FK1,
poificontent
poiid : Integer PK, FK1,
ficontentpoiid : Integer
POICategory_POIId
poi
id : Integer PK,
name : CharacterString [128]
position : Point GIST
updated : Date
POISource_POIId
POIFicontent_POIId
POIProperty_POIId
OCDCategory_CategoryId
ocdcategory
ocdid : Integer PK, FK2,
categoryid : Integer PK, FK1,
OCDCategory_OCDId
ocd
id : Integer PK,
name : CharacterString [45] UK1,
city : Integer FK1,
description : CharacterString [255]
fusionrules : CharacterString [255]
accesskey : CharacterString [45]
fusiondate : Date
status : CharacterString [45]
poisource
poiid : Integer PK, FK2,
sourceid : Integer PK, FK1,
originalref : CharacterString [255]
poiproxyattribute : CharacterString [255] PK,
POISource_SourceId
poilabel
id : Integer PK,
poiid : Integer FK2,
typeid : Integer FK3,
value : CharacterString [255]
sourceid : Integer FK1,
language : CharacterString [127]
licenseId : Integer FK4,
updated : Date
POILabel_TypeId
labeltype
Id : Integer PK,
name : CharacterString [45] UK1,
OCDSource_ODCId
ocdsource
ocdid : Integer PK, FK2,
sourceid : Integer PK, FK1,
OCDSource_SourceId
source
id : Integer PK,
name : CharacterString [45]
description : CharacterString [255]
urlaccess : CharacterString [255]
categorymapping : CharacterString [255]
apitypeid : Integer FK1,
POILabel_SourceId
poicomponent
poilabelid : Integer PK, FK1,
componentid : Integer PK, FK2,
POIComponent_POILabel
POILabel_LicenseId
POIComponent_Component
OCD_City
city
id : Integer PK,
name : CharacterString [45] UK1,
bbox : CharacterString [127]
Sourcecity_Source
Source_APITypes
LicenseSource_Source
licensesource
sourceId : Integer PK, FK1,
licenseId : Integer PK, FK2,
component
id : Integer PK,
name : CharacterString [45] UK1,
Sourcecity_city
sourcecity
sourceid : Integer PK, FK1,
cityid : Integer PK, FK2,
license
id : Integer PK,
name : CharacterString [45]
description : CharacterString [255]
info : CharacterString [255]
LicenseSource_License
apitype
id : Integer PK,
name : CharacterString [45] UK1
description : CharacterString [255]
apirules : CharacterString [255]
mosKit 2.0

Three tables are important in this data model:

- Poi : stores all pois
- Poilabel : stores all metadata related to a POI
- Poisource : allows finding out from which source(s) the POI has been built. The same aplies for each poilabel.

Note that this structure is quite general and is in line with the drafted data model described by W3C and OGC, used in CitySDK. However, there are other tables (poicomponent, component) that allows assigning poilables to components (fw_core, fw_media, fw_contact) as described by the FIWARE POI spec. This is currently not used but you should use it if you plan to add this *apitype* to your project.

3.4 - Integration con FiContent2

The FE enabler has been used in two scenarios within the FIC2 project: in the city of Tenerife and the city of Valencia. We will describe the Valencia scenario, as it integrates several enablers from FIContent2.

3.4.1 - Live Fallas

In order to test and make a large scale experiment using FICONTENT2 technology, several enablers have been integrated in a backend to offer services to a mobile app, which was used by users (citizens of Valencia) during Las Fallas Festival (the biggest festival in the city of Valencia).
The architecture is depicted in Figure 5 where the different enablers are in green:

- The Fusion Engine is able to fusion data from multiple data sources and generates one single POI repository.
- POIProxy obtains data from social networks and local RSS news
- The OCDB is able to provide user interaction, so that users through their mobile app are able to comment, rate and check in POIs.
- The Context-Aware Recommendation is able to provide real-time recommendations to users showing the next POI (Falla) to visit.

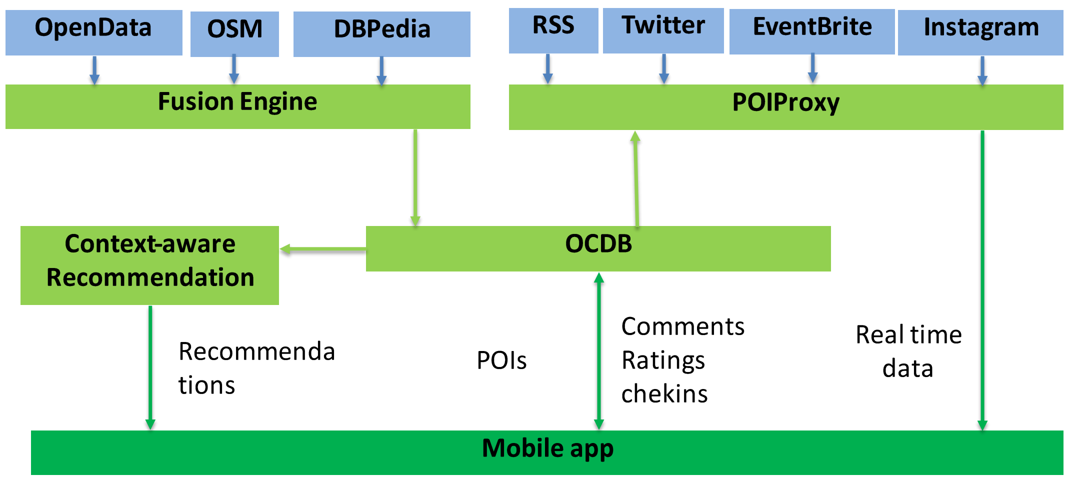


Figure 5. Live Fallas General architecture

Though the Fusion Engine is able to provide a POI access interface, it does not provide user interaction support, thus the OCDB was an optimal candidate for thus.
Under the folder ficontent2/ocdb you can find a README file that describes how the data generated by the Fusion Engine can be exported to the OCDB.

3.4.2 - Docker

Docker is an open platform for developers and sysadmins to build, ship, and run distributed applications (https://www.docker.com/). The FIContent2 project is using Docker as a tool for its FIC2LAB (experimentation environment for FIContent2 enablers).
However, you can also use Docker for testing purposes.
Under the folder ficontent2/docker you can find a README file that describes how you can use docker to generate a FE Docker image; it includes the Dockerfile and the scripts

3.5 - Integration with FIWARE

The FE enabler integrates within FIWARE in two ways: it exports data to POI DP GE and it can interact with any GE that uses the FIWARE POI spec

3.5.1 - Export to POI DP GE

The FE has been correctly integrated in the OCDB (FIContent2 enabler) as explained in section 3.4.1 -. For the FIWARE project, the GE that stores POI information is the POI DP GE. So we have developed a script able to export data to this GE
Under the folder ../fiware/poi_dp you can find a README file that describes how data is exported to POI DP. Note that stored data in the FE has to be converted into another data format (POI spec) which consists in multiple components. We have only focussed on the '*fw_core'* component. For metadata, the POI DP GE recommends to use your own custom component.

3.5.2 - Integration with other POI GE

By exporting data to the POI DP GE you should be able to interoperate with any other SE or GE using this POI interface, at least at 'fw_core' level.
However, similar as we provide an output CitySDK interface for the FE, we have also developed an output FIWARE POI interface, supporting the 'fw_core' component. Obviously, similar to the CitySDK interface, it is only a read-only interface, so one can only request for POIs, but it not possible to add or update POIs.
Following operations are possible:

- Look for POIs (providing categories, radial search or bounding box)
- Look for a particular POI (providing a UUID)

Several notes:

- As the FE is able to give POIs for several cities, the <ocdName> should be given a part of the URL. This means that the FIWARE POI interface is provided OCD by OCD.
- Note that the FE data model uses the *Integer* datatype for POIs, whereas the FIWARE POI spec uses UUID. Thus we need to covert each Integer in its corresponding UUID and vice versa. As different POIs from different OCDs may have the same Integer value, this is the main reason for the previous note.

The FIWARE API is implemented in the FrontEnd Service, under the *'fiware'* folder. Just supposing that you have correctly installed the FE in your *localhost*, following queries are available for the '*valencia_demo'* OCD:

- Get POIs around a certain position (lat, lon are mandatory parameters ; if no radius is given, 10 km is used as default value)

http://localhost:8080/fic2_fe_v3_frontend/fiware/search/valencia_demo/radial_search?lat=39.4666667&lon=-0.3666667

- Get POIs around a certain position with a given radius (specified in meters)

http://localhost:8080/fic2_fe_v3_frontend/fiware/search/valencia_demo/radial_search?lat=39.4666667&lon=-0.3666667&radius=100

- Get POIs around a certain position from various categories (accommodation,museum)

http://localhost:8080/fic2_fe_v3_frontend/fiware/search/valencia_demo/radial_search?lat=39.4666667&lon=-0.3666667&category=accommodation,museum

- Get POIs around a certain position, but only the *fw_core* component. Currently only this component is implemented http://localhost:8080/fic2_fe_v3_frontend/fiware/search/valencia_demo/radial_search?lat=39.4666667&lon=-0.3666667&component=fw_core

- Get POIs around a certain position, but only the *fw_core* component with a maximum number of results (provided by the max_results parameter)

http://localhost:8080/fic2_fe_v3_frontend/fiware/search/valencia_demo/radial_search?lat=39.4666667&lon=-0.3666667&component=fw_core&max_results=3

- Get POIs within a bounding box

http://localhost:8080/fic2_fe_v3_frontend/fiware/search/valencia_demo/bbox_search?north=39.482084&west=-0.352649&south=39.471020&east=-0.335612

- Get POIs specified by their UUIDs (all UUIDs should exists, otherwise i twill throw an error)

http://localhost:8080/fic2_fe_v3_frontend/fiware/search/valencia_demo/get_pois?poi_id=00000000-0000-0000-0000-00000000013c,00000000-0000-0000-0000-00000000014c