

Private Public Partnership Project (PPP)

Large-scale Integrated Project (IP)



D.11.1.2: FIWARE Tour Guide

Project acronym: FI-Core

Project full title: Future Internet - Core

Contract No.: 632893

Strategic Objective: FI.ICT-2011.1.7 Technology foundation: Future Internet Core Platform

Project Document Number: ICT-2013-FI-632893-WP11-D.11.1.2

Project Document Date: 2016-05-15

Deliverable Type and Security: PU

Author: José Manuel Cantera (TID)

Contributors: Alberto Martín (Bitergia), Pablo Fernández (ULPGC)

Executive Summary

This report describes the activities performed in order to maintain and evolve the FIWARE Tour Guide for developers. The FIWARE Tour Guide is a tutorial document and accompanying application which allows developers to get started with FIWARE in a short timeframe.

During the period covered by the present document the activities which have been conducted are the following:

- Set up the infrastructure and guidelines to allow a collaborative and flexible approach for keeping the Tour Guide up to date. This infrastructure is based on <http://github.com> and <http://readthedocs.org>.
- Refine the contents of the Tour Guide by adding new sections or updating existing ones.
- Develop a “Tour Guide Application” to support all the examples described by the original Tour Guide content.
- Align the Tour Guide examples with the content and structure of the Tour Guide Application.
- Coordinate all the teams involved in contributing to the Tour Guide, including chapter leaders and GE development teams. For that purpose a call for contributions was made during the sprint held on March 2016.
- Define a roadmap for the Tour Guide so that it is always kept alive and aligned with the latest FIWARE platform advances.

Table of Contents

Table of Contents.....	3
1 Introduction.....	5
1.1 Introduction.....	5
1.2 Target audience.....	5
1.3 Related readings.....	5
2 Collaborative infrastructure for content.....	6
3 Introduction.....	6
3.1 Approach.....	6
3.2 Results.....	7
4 Tour Guide Application.....	11
4.1 Introduction.....	11
4.2 Approach.....	11
4.3 Results.....	12
4.3.1 Application functionalities.....	12
4.3.2 Architecture.....	12
4.3.3 Implementation approach.....	13
4.3.4 Development.....	14
4.3.5 Working with the Tour Guide Application.....	17
5 Conclusions and future work.....	21
Annex 1.....	22
Original Tour Guide Application Requirements.....	22
Introduction.....	22
Overall description of the application.....	22
Detailed description of functionalities (v1).....	23
User profiles.....	23
Customer Reservations and Reviews.....	23
Real time Management of Restaurants.....	24
Publication of data as Open Data.....	24
Big Data Analysis.....	25
Real time analysis of streaming sources (cameras).....	25

Non-Functional Requirements..... 25

1 Introduction

1.1 Introduction

This document describes all the activities targeted to maintain and evolve the FIWARE Tour Guide for developers. The FIWARE Tour Guide is continuously evolving, thus this document summarizes both the activities which have been completed at the time of writing, together with a roadmap of expected activities to be executed during next months. Next issues of this document will report on implementation results of the work items included in such roadmap.

1.2 Target audience

The target audience of the present document is anyone who wants to know more about the *FIWARE Tour Guide for developers*. In addition any member of the FIWARE Community will benefit from reading it, as it allows to understand the efforts made by FIWARE to lower the entry barriers.

1.3 Related readings

The document onon Friendliness Activities ([D11.9.1 Report on FIWARE Friendliness activities](#)) is closely related to this document, as it describes other complementary activities aimed at improving FIWARE adoption by developers. There is a certain degree of overlap between both and the present document. However, the latter provides a closer and more detailed view on a concrete aspect exclusively: the FIWARE Tour Guide.

2 Collaborative infrastructure for content

3 Introduction

Originally the FIWARE Tour Guide for developers was based on Wordpress and maintained by the FIWARE Press Office. This approach resulted to be rather appealing and professional but at the same time quite inflexible as it made it difficult to involve FIWARE development teams in content creation. In addition, it was creating a bottleneck in the edition process, which had to be performed by the Ogilvy Team.

To overcome such an issue it was decided to migrate the Tour Guide Content to a repository in <http://github.com/Fiware>. Such a repository together with the <http://readthedocs.org> infrastructure enables a collaborative and flexible approach in the development of contents. The following sections describe in more detail the approach and the results obtained.

3.1 Approach

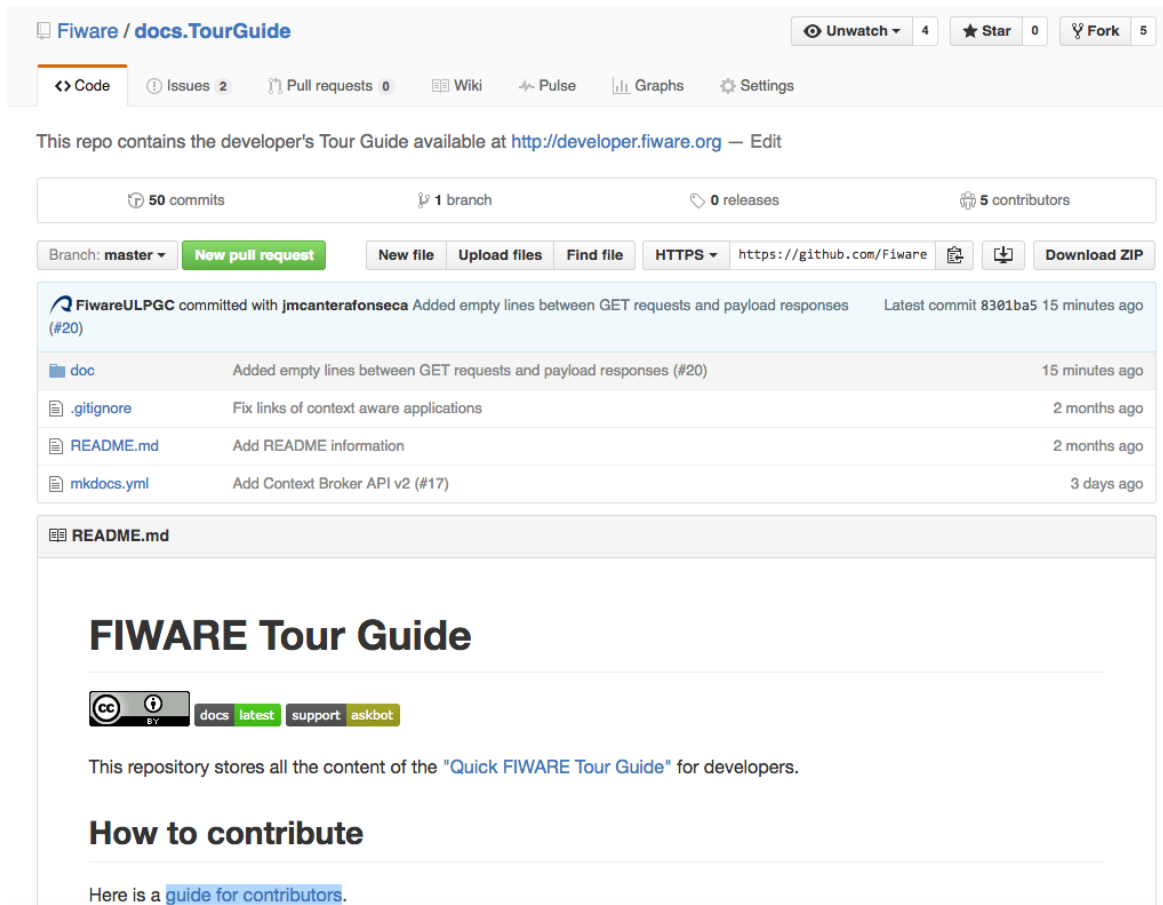
The strategy has consisted of keeping the [developers landing page](#) hosted by Wordpress and maintained by Ogilvy, but at the same time migrating all the specific contents to a different schema. To achieve this objective the following tasks have been performed:

- Creation of a [Github repository](#) to hold all the Tour Guide Contents.
- Migration of all the Tour Guide contents to markdown format (MD). Markdown format is text-based and enables a quick and easy transformation (using the [mkdocs](#) tool) to HTML content for the Web.
- Specification of the target index structure of the Tour Guide content through a [mkdocs.yml](#) file.
- Creation of a [readthedocs project](#) bound to the Github repository, enabling the automatic publication of new contents once they land in such repository.
- Definition of [contribution guidelines](#) to enable a process to manage all the lifecycle of new contributions, including their review and final publication. Such contribution guidelines are based on typical git workflows, so that a contributor has to make a pull request when he wants to suggest new content for the Tour Guide. Such pull request will be reviewed by the owners of the Tour Guide, and, if approved, it will land on the main code base. Afterwards, all those new contents will be automatically published on the [readthedocs site](#).
- Launch of a call for contributions to the Tour Guide in order to update it according to the latest FIWARE advances. Such call for contributions has been instrumented using the agile methodology and Jira.

Last but not least, links between the Wordpress developer's [landing page](#) and the [readthedocs site](#) have been created, thus all the original functionality is preserved.

3.2 Results

In this section the results obtained from the approach described above are illustrated. The figures below show the [github repository](#) (under the Github FIWARE organization) which keeps all the contents of the Tour Guide. It can be observed the structure of the Github repository which contains one folder per section of the Tour Guide. Inside each folder there are markdown documents which include all the content which is displayed by the readthedocs site.



The screenshot shows the Github repository page for 'Fiware / docs.TourGuide'. The repository has 4 unwatchers, 0 stars, and 5 forks. It contains 50 commits, 1 branch, 0 releases, and 5 contributors. The current branch is 'master'. A recent commit by 'FiwareULPGC' with the message 'Added empty lines between GET requests and payload responses' is shown, along with a list of files: 'doc', '.gitignore', 'README.md', and 'mkdocs.yml'. The 'README.md' file is expanded, showing the title 'FIWARE Tour Guide', a Creative Commons license, and a link to a 'guide for contributors'.

Figure 1.- Github repository for the Tour Guide Content

FiwareULPGC committed with jmcanterafonseca Added empty lines between GET requests and payload responses Latest commit 8301ba5 19 minutes ago (#20)		
..		
big-data-analysis-of-historic-context-inf...	Automatic style links refactored to inline style (#15)	22 days ago
connection-to-the-internet-of-things	Automatic style links refactored to inline style (#15)	22 days ago
creating-application-dashboards	Automatic style links refactored to inline style (#15)	22 days ago
development-context-aware-applications	Added empty lines between GET requests and payload responses (#20)	19 minutes ago
fiware-tour-guide-application-a-tutorial-...	Rename introductory files for every chapter to introduction.md	2 months ago
handling-authorization-and-access-con...	Automatic style links refactored to inline style (#15)	22 days ago
hosting-your-application-on-a-fiware-clo..	Rename introductory files for every chapter to introduction.md	2 months ago
providing-an-advanced-user-experienc...	Automatic style links refactored to inline style (#15)	22 days ago
publishing-open-data-in-fiware	Automatic style links refactored to inline style (#15)	22 days ago
real-time-processing-of-context-events	Automatic style links refactored to inline style (#15)	22 days ago
real-time-processing-of-media-streams	Automatic style links refactored to inline style (#15)	22 days ago
index.md	Update index.mdd with new description	3 months ago

Figure 2.- Folder structure of the Tour Guide Content

The following figures show:

- the configuration of the readthedocs project associated to the Tour Guide. Readthedocs makes use of the [mkdocs](#) package to automatically create a Web site associated to the markdown content present in the Github repository referenced by the configuration.
- The final pages generated, ready to be consumed by final users

Figure 3.- Readthedocs project bound to the Tour Guide content

The screenshot shows the 'QUICK FIWARE TOUR GUIDE' website. On the left is a navigation menu with sections like 'Home', 'Development of context-aware applications', 'Introduction', 'Using NGSi v1', 'Using NGSi v2', and 'Connection to the Internet of Things'. The main content area has a breadcrumb trail: 'Docs » Development of context-aware applications » Introduction'. Below this, there is introductory text about context-aware applications and a diagram titled 'The Context Broker GE'. The diagram shows an 'Application/Service' connected to a 'Context Broker' via the 'NGSI API'. The 'Context Broker' is represented as a cloud containing three entities: 'Bus' (with attributes: Location, N° Passengers, Driver, Licence plate), 'Citizen' (with attributes: Name-Surname, Birthday, Preferences, Location, ToDo list), and 'Shop' (with attributes: Location, Business name, Franchise, Offerings). Below the diagram, text states 'Context information may come from many different sources:' followed by a bulleted list: 'Already existing systems', 'Users, through mobile apps', and 'Sensor networks'. At the bottom, it notes 'One of the most important features of the Context Broker GE is that it allows to model and gain acc...'.

Figure 4.- Tour guide published at <http://fiwaretourguide.readthedocs.io>

The above figure shows the Tour Guide content, ready to be used by final users. The left menu is automatically generated from the configuration defined by the [mkdocs.yml](#) descriptor file. The FIWARE look and feel is provided thanks to a [custom CSS](#) developed by Ogilvy. Such custom CSS has been used by GERis documentation sites as well, in order to show a unified aspect.

The last figures allow to illustrate how the new contribution workflow is working. The first figure shows a Jira task, devoted to Tour Guide updates, automatically created and scheduled for Sprint 5.2.3. These Tour Guide update tasks were created at chapter level, so that chapter leaders have been able to dispatch those activities to each GE. Second figure shows a pull request made by a contributor and which includes content updated, in this case for the WebUI chapter sections. Such pull request was reviewed by owners of the Tour Guide and acknowledged (*LGTM*), which stands for *looks good to me*. Finally, it was merged, and as a result, all the content in the Website of the Tour Guide was properly updated.

The screenshot shows a Jira issue page for 'Chp - WebUI / WEB-923' with the title 'FIWARE.WorkItem.WebUI.Coordination.TourGuide.Update'. The issue is in a 'CLOSED' status. The details section shows the type is 'WorkItem', priority is 'Major', and component is '_Coordination'. The description includes a link to contribution guidelines and a deadline of '31-03-2016 at 18:00'. The activity section shows a comment from Torsten Spieldenner dated 31/Mar/16 1:58 PM, mentioning a pull request on GitHub.

Figure 5.- A jira task devoted to update the Tour Guide

Webui contribution #13

The screenshot shows a GitHub pull request conversation. It starts with a 'Merged' notification from 'jmcanterafonseca' merging 1 commit into 'fiware:master' from 'tospie:webui-contribution' on 31 Mar. The conversation includes three messages: 1) A comment from 'tospie' explaining that the Tour Guide content was updated for WebUI, mentioning a Jira issue and updates to FI-Lab and architecture diagrams. 2) A comment from 'jmcanterafonseca' (FIWARE member) asking to squash commits into a single one, with a link to a guide on git-squash. 3) A comment from 'tospie' confirming that all commits are now squashed into a single one.

Figure 6.- Workflow of Tour Guide Contributions (pull request)

4 Tour Guide Application

4.1 Introduction

When the first version of the FIWARE Tour Guide was created, it was based on some fictitious use cases that facilitated the description of the main functionalities offered by the different GEs or Chapters in FIWARE. Soon it was detected the necessity of enabling direct experimentation with the technologies to improve learning processes. As a result, it was decided to build a tutorial application aimed at supporting the Tour Guide use cases, enabling developers to follow a “try and tweak” learning process.

The [FIWARE Tour Guide Application](#) is a reference application aimed at teaching and demonstrating how to combine different Generic Enablers (GEs), in order to create a smart context-aware application. It exploits the capabilities offered by [Docker Containers](#) provided by FIWARE GERis. Furthermore, the application allows an incremental instantiation and linkage of different FIWARE GEs, as it is based on [docker compose](#).

4.2 Approach

The first step taken to build the FIWARE Tour Guide application was to think up rich use case: the smart management of a big restaurant franchise, including reservations, reviews and all the parameters which have to do with managing each restaurant on a day by day basis. Afterwards a complete specification of requirements was drafted (included in this document as an annex) to be used as a reference for the development team.

Once the requirements were ready, a decision on the supporting technologies was made. At this point it was clear that Docker technology was perfect for this purpose, as it simplifies the deployment and configuration processes. Furthermore, it enables a modular composition (docker compose), which is crucial for separating the different chapters and GEs in FIWARE. This choice was reinforced by the fact that the FIWARE Technical Committee agreed on making the provision of docker containers mandatory for each GERi.

Another important point was to define a bootstrap process that would enable to load initial data into the application. For this purpose, data coming from the [Open Euskadi](#) (*Basque Country Government*) open data portal was used. Data from restaurants located at the Basque Country Region in Spain were loaded. All this bootstrapping process was automated for the benefit of the end user who will be able to try and tweak with mock data.

Finally, it was decided to give priority to the integration of those GEs which have to do with Internet of Things (IoT) scenarios, i.e. Orion, IDAS and CKAN, together with those GEs which implement transversal security layers.

The application has been developed following the [FIWARE Developer Guidelines](#), including peer reviews and controlled landing of new features. The development of the application has been integrated into the FIWARE sprint plannings and Jira. As any other FIWARE software artefact, it

includes unit tests and automated build procedures so that to ensure that no new landing code breaks existing functionalities.

4.3 Results

A blog post which summarizes the Tour Guide Application was published at <https://www.fiware.org/2016/01/08/fiware-tour-guide-application-now-ready/>. It was intended to spread the word among the FIWARE Developer's Community.

A more detailed description of the Tour Guide Application is continuously updated at <http://fiwaretourguide.readthedocs.io/en/latest/fiware-tour-guide-application-a-tutorial-on-how-to-integrate-the-main-fiware-ges/introduction/>.

Additionally, the following subsections describe in more detail the results achieved, and provides a good technical overview of the work undertaken.

4.3.1 Application functionalities

The list below summarizes the main functionalities currently offered by the smart restaurant (Tour Guide) application:

- Different user profiles (customer, restaurant manager, franchise manager) and functionalities per profile
- Admit Customer reservations in accordance with current occupation and reservations made.
- Register customer reviews according to different criteria (service, food, etc.).
- Real-time control of different parameters at each restaurant location (occupation, temperature, etc.).
- Short time historic data of the different parameters monitored. Publication of open data concerning the most relevant information about the different restaurant locations.
- Web user interface to monitor information about restaurants.

4.3.2 Architecture

The figure below describes the architecture of the Smart Restaurant application, which integrates a number of GEs:

- **IoT GEs:** *Backend Device Management - IDAS*. It is in charge of connecting IoT devices (temperature & humidity) using the UL20 client. This component translates UL20 client requests into NGSI context entities, enabling querying and subscribing to sensor data.
- **Data GEs:** *Orion Context Broker*. It is responsible for managing all the application context information modelled as NGSI entities (Restaurant, Reservation, Review, ...). *Cygnus*, part of the Cosmos ecosystem, is responsible for persisting historical context data in a target backend (MySQL or Hadoop) or as open data (*CKAN*). *Cygnus* is connected to *Orion Context Broker* through the subscription/notification interface.

- **Security GEs:**
 - *Authorization PDP - AuthZForce*, provides an API to get authorization decisions based on authorization policies.
 - *IDM KeyRock* covers a number of aspects involving user profile management, OAuth authentication, authorization & trust management, Single Sign-On (SSO) to service domains and identify federation towards applications. It interacts with *AuthZForce*.

Finally, through a front-end application, managers get access to restaurants under his duty and restaurant customers can make and browse reviews, or even ask for reservations.

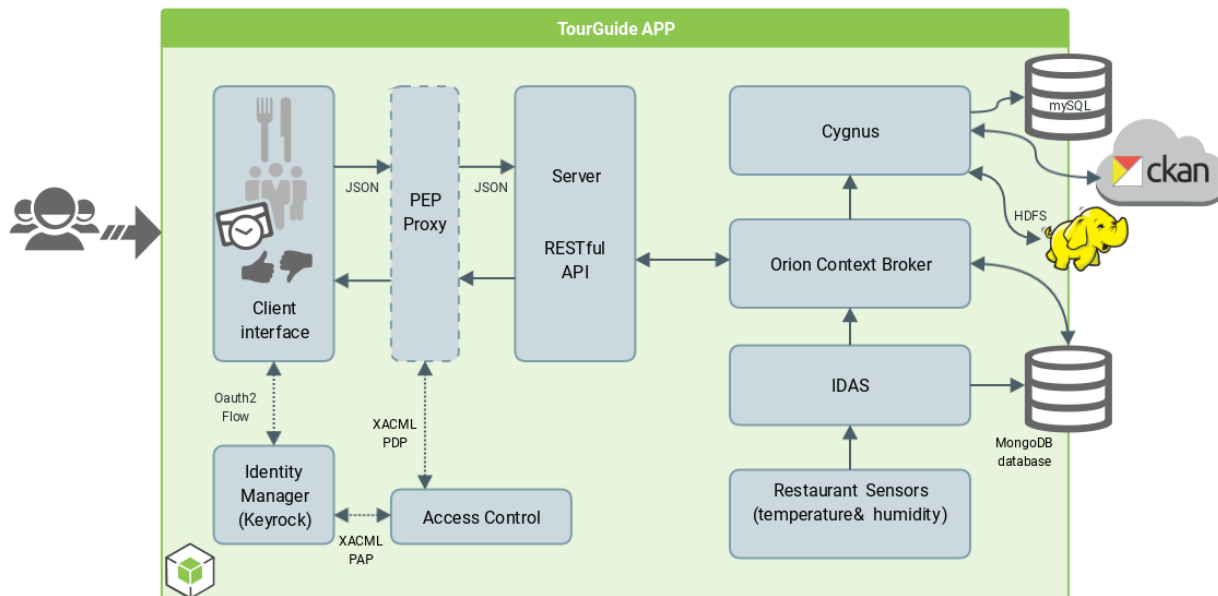


Figure 7 Architecture of the Tour Guide Application

4.3.3 Implementation approach

To this aim we have connected, using Docker-compose, different GEs deployed on different containers. Docker-compose enables the creation of a complete environment just by writing a .yml (or .yaml) schema file. In that file several parameters are defined: containers to be created, volumes to share data, how to link containers, ports exposed and environment variables that will be used to configure the Generic Enablers. The [docker-compose](#) file created for the Tour Guide Application defines the scenario described by the image below. This configuration has all the advantages of using isolated environments and exposing each generic enabler from a desired port.

The Tour Guide Application uses the official Docker from the GERis and has contributed to debugging or polishing them.

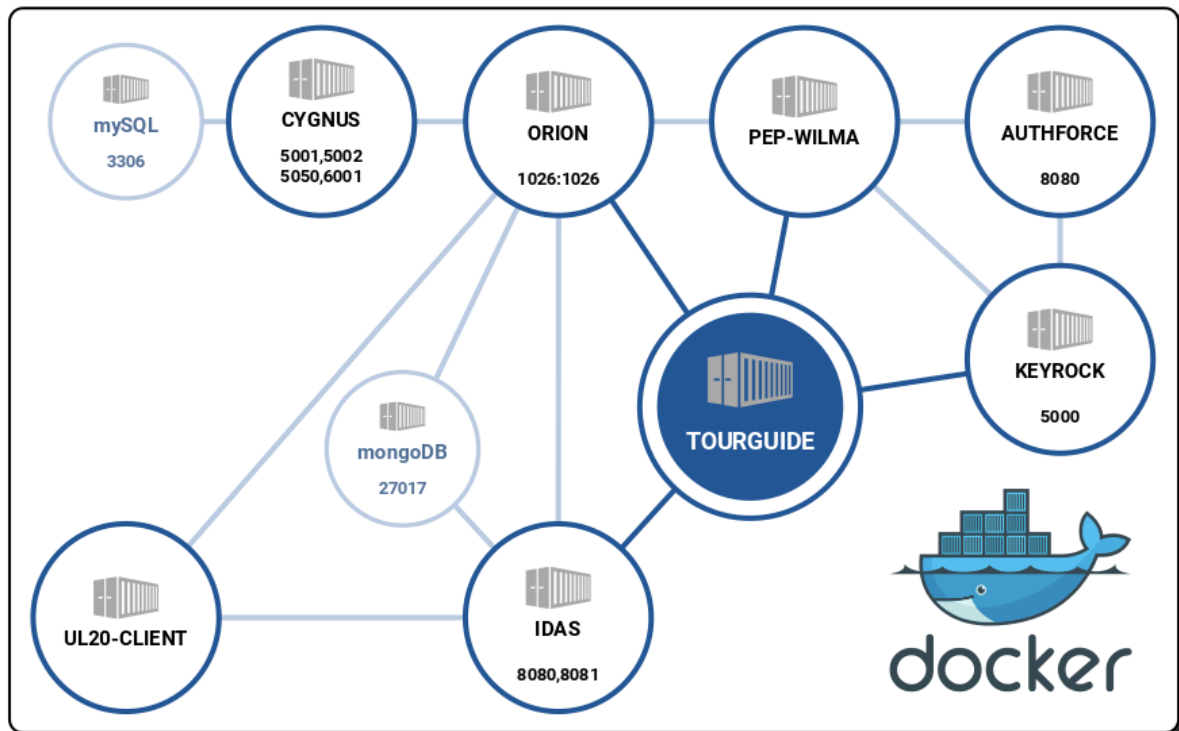


Figure 8 .- Docker container structure

4.3.4 Development

The figure below shows the repository which holds all the code of the Tour Guide Application. It has been released using the MIT License. Contributions from external developers are welcome. The development process follows usual Github workflows with pull requests, reviews and code landing. The application code has been developed using [Node.js](#) and [HTML5](#) technologies. The [Travis CI tool](#) is used for running all tests and ensuring quality.

Additionally, the development has been integrated into the usual agile process followed by FIWARE, performing sprint plannings and management of tasks through Jira.

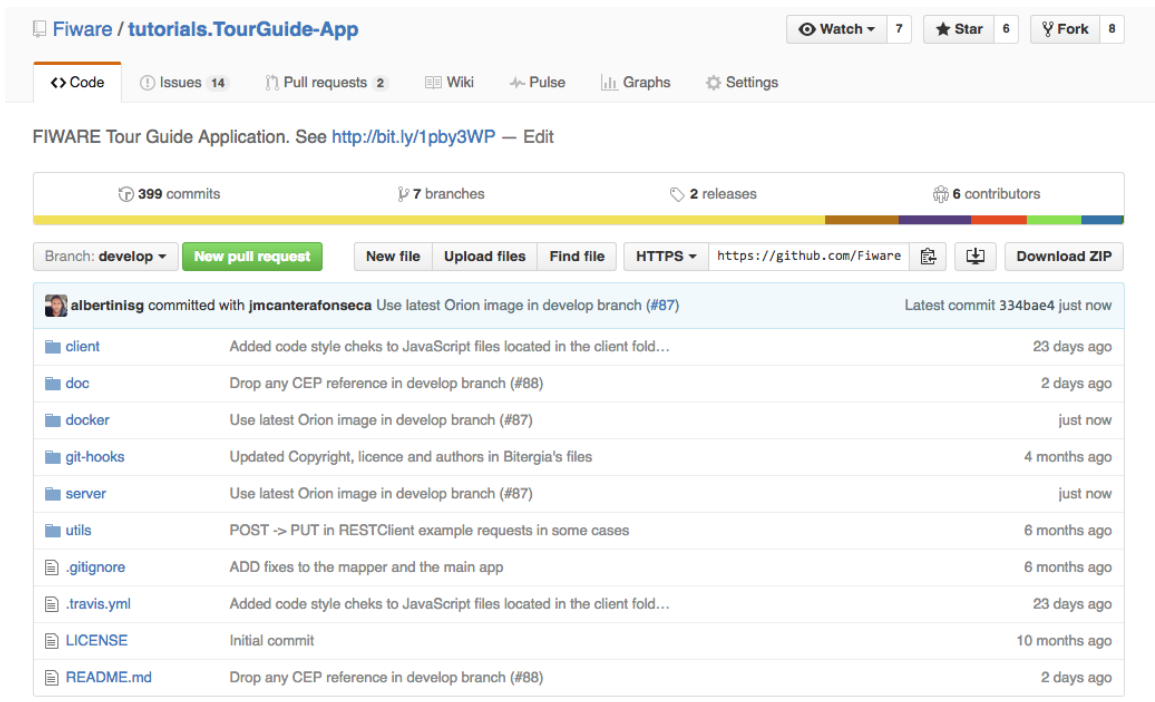


Figure 9 .- Tour Guide Application Github Repository

In addition, the application repository contains all the badges that provide information and direct links to the relevant parts of the application, including its docker container. See figure below. It is noteworthy that FIWARE offers support to this application through the ask.fware.org platform.

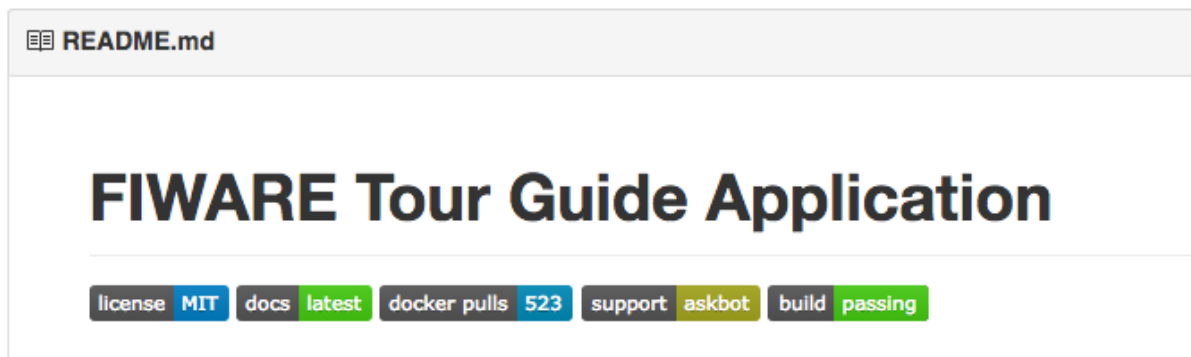


Figure 10 .- Badges associated to the Tour Guide Application Repository

The screenshot shows a Jira task page for 'WG - Sustainability / SUS-159'. The task title is 'FIWARE.WorkItem.Sustainability.DevelopersCommunity.TourGuide-App'. Below the title are buttons for 'Edit', 'Comment', 'Assign', 'More', and 'Reopen'. The 'Details' section includes: Type: WorkItem, Priority: Major, Component/s: Developers Community, Labels: None, Status: CLOSED (View Workflow), Resolution: Done, and Fix Version/s: Sprint 5.3.1. The 'Description' section contains a list of GitHub issue links. The 'Activity' section has tabs for 'All', 'Comments', 'History', 'Activity', 'Emails', and 'Transitions'.

Figure 11 .- Jira task to track work items for the Tour Guide Application

Last but not least it is important to remark that FIWARE has been able to create a real community of developers around this Tour Guide Application. Developers from Bitergia, TID and the University of Las Palmas (ULPGC) are contributing to the code on a daily basis. The figure below shows a pull request made which contains review comments and iterations until final code landing. It is our aspiration that contributions coming from other members of the FIWARE Community happen, particularly in the context of the upcoming [FIWARE Bounty Programme](#).

The screenshot shows a GitHub pull request titled 'Fix/patch aggregate rating #91'. It is an 'Open' pull request by 'albertinisp' merging 5 commits into 'Fiware:develop' from 'albertinisp:fix/patch-aggregate-rating'. The PR description states: 'This PR fixes #90 and updates the data image with the error fixed'. The PR includes several commits: 'Fix element sent to aggregateRating function to update values', 'added bug server labels', 'added this to the Sprint 5.3.2 milestone', and 'added some commits' which includes 'Add single quotes for strings in conditions', 'Update tourguide image data', and 'Fix filter element in reservations function'. The right sidebar shows labels 'bug' and 'server', a milestone 'Sprint 5.3.2', and a notification to 'Unsubscribe'. There are 2 participants in the conversation.

Figure 12 .- Pull request to fix open bugs in the Tour Guide Application

4.3.5 Working with the Tour Guide Application

Before starting it is important to install Docker and docker-compose, as they are the basic infrastructure for the Tour Guide Application to run.

The first step for getting started with the Tour Guide Application consists of cloning its Github Repository:

```
git clone https://github.com/Fiware/tutorials.TourGuide-App
```

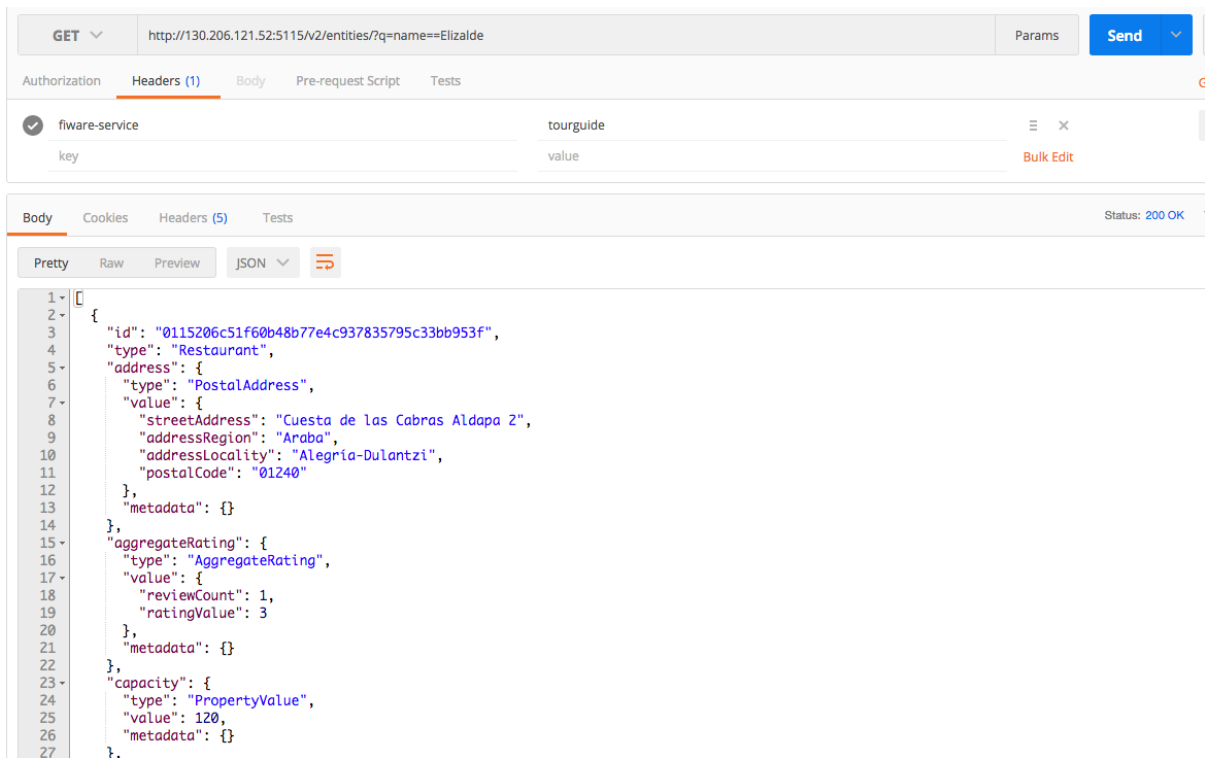
Once cloned, the branch *'develop'* should be selected. The code is structured around different branches, being *'develop'* the one which contains the latest version developed. There are other branches which contain previous releases of the application and which can be useful in certain contexts, for instance when *'develop'* has not been stabilized yet.

Then, experimentation can start. For instance, it can be run an instance of the *Context Management GE, Orion*, by executing the following command:

```
docker-compose up orion
```

Using this instruction a new instance, populated with example data, of the Orion Context Broker GErI will be available for trying and tweaking. Then, experimentation can be performed by running the generic REST client [Postman](#) or a curl command line like:

```
curl -H fiware-service:tourguide http://host:1026/v2/entities/?q=name==Elizalde
```



The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** http://130.206.121.52:5115/v2/entities/?q=name==Elizalde
- Headers:**
 - fiware-service: tourguide
 - key: value
- Status:** 200 OK
- Body (JSON):**

```

1- [
2- {
3-   "id": "0115206c51f60b48b77e4c937835795c33bb953f",
4-   "type": "Restaurant",
5-   "address": {
6-     "type": "PostalAddress",
7-     "value": {
8-       "streetAddress": "Cuesta de las Cabras Aldapa 2",
9-       "addressRegion": "Araba",
10-      "addressLocality": "Alegria-Dulantzi",
11-      "postalCode": "01240"
12-     },
13-     "metadata": {}
14-   },
15-   "aggregateRating": {
16-     "type": "AggregateRating",
17-     "value": {
18-       "reviewCount": 1,
19-       "ratingValue": 3
20-     },
21-     "metadata": {}
22-   },
23-   "capacity": {
24-     "type": "PropertyValue",
25-     "value": 120,
26-     "metadata": {}
27-   }
28- }

```

Figure 13 .- NGSI v2 restaurant entity provided by the Tour Guide Application

Next step would be to start gathering IoT data provided by the sensors placed at the restaurants.
For doing so it would be needed to

1/ Register IoT devices and bind them to restaurant entities

```

1  {
2  "devices": [
3  {
4    "device_id": "restaurant-sensor-0115206c51f60b48b77e4c937835795c33bb953f",
5    "protocol": "UL20",
6    "entity_name": "0115206c51f60b48b77e4c937835795c33bb953f",
7    "entity_type": "Restaurant",
8    "attributes": [
9    {
10     "object_id": "t",
11     "name": "temperature",
12     "type": "number"
13   }
14   ],
15   "static_attributes": [
16   ]
17 }
18 ]
19 }

```

Figure 14 .- Device registration for gathering IoT data

2/ POST sensor readings to the corresponding IoT Agent

```

curl http://idas:7896/iot/d?k=tourguide-devices&i=restaurant-sensor-
0115206c51f60b48b77e4c937835795c33bb953f

t|25

```

3/ Check that restaurant data is updated with readings coming from IoT devices

```

11  {
12  "id": "0115206c51f60b48b77e4c937835795c33bb953f",
13  "type": "Restaurant",
14  "address": {
15    "type": "PostalAddress",
16    "value": {
17      "streetAddress": "Cuesta de las Cabras Aldapa 2",
18      "addressRegion": "Araba",
19      "addressLocality": "Alegria-Dulantzi",
20      "postalCode": "01240"
21    },
22    "metadata": {}
23  },
24  "name": {
25    "type": "none",
26    "value": "Elizalde",
27    "metadata": {}
28  },
29  "temperature": {
30    "type": "number",
31    "value": "25",
32    "metadata": {}
33  }
34 }
35 ]

```

Figure 15 .- Restaurant entity enriched with IoT data

Similarly, other services could be tweaked and configured by users, learning step by step all the FIWARE internals. The final aim of the Tour Guide Application is not the functionality offered itself but illustrating and helping to understand FIWARE, its Open APIs and accompanying architectural patterns.

The screen captures shown below depict the look and feel of the client application. Being able to deploy and launch such application will be the final step to be performed by successful learners.

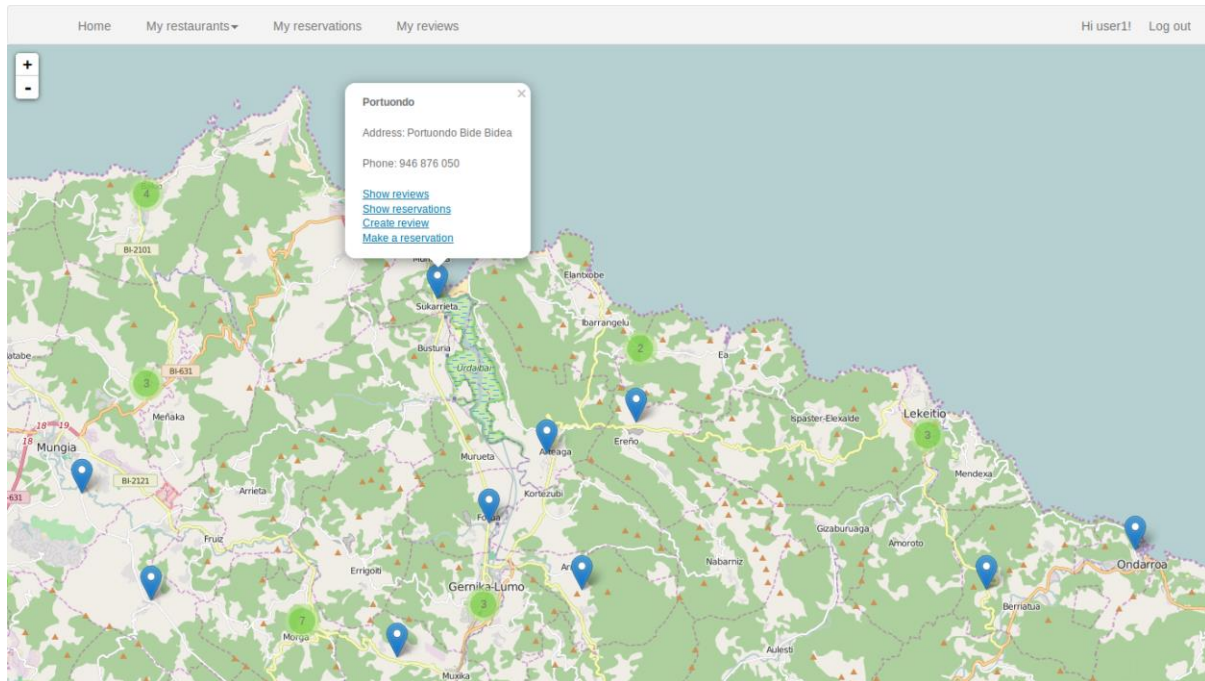


Figure 16 .- Restaurant entities pinned on a map

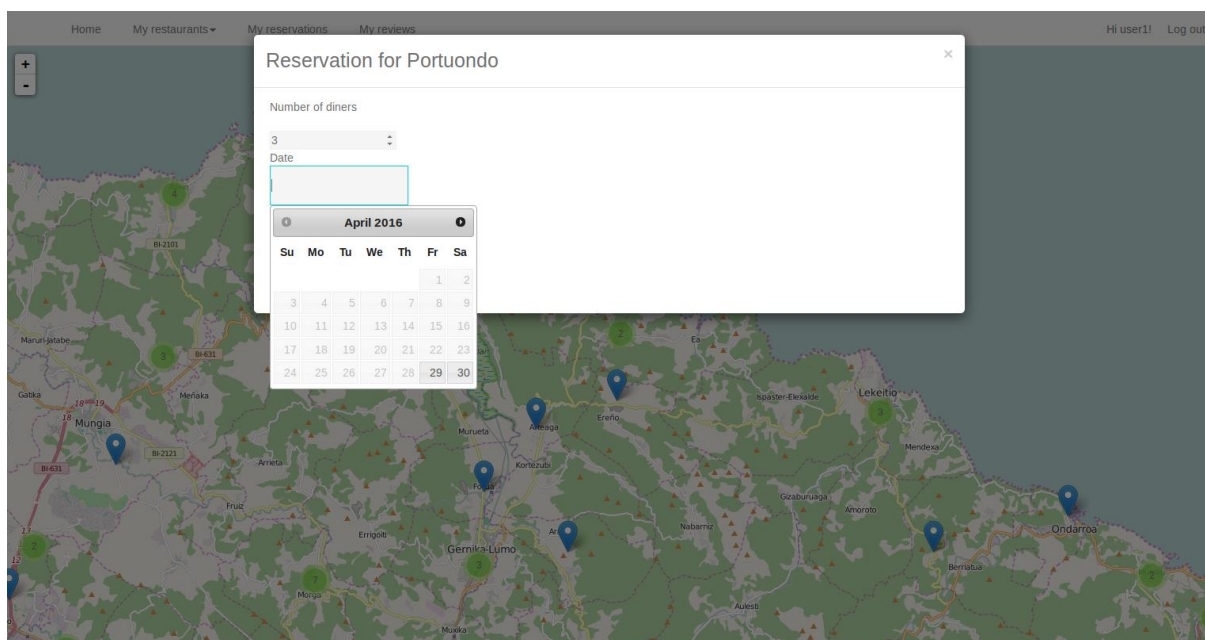


Figure 17 .- Adding a new reservation with a restaurant

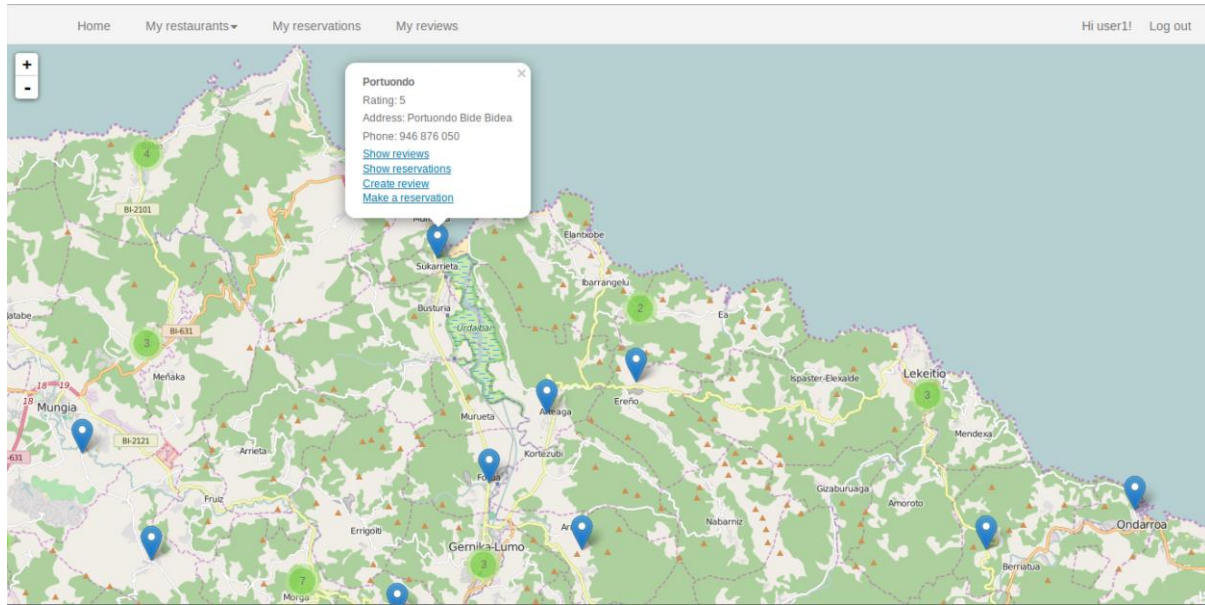


Figure 18 .- Ratings of a restaurant

5 Conclusions and future work

The FIWARE Tour Guide and its accompanying reference tutorial application are a considerable step forward aimed at engaging even more developers in FIWARE. The project has worked intensively during this period to enable a sustainable collaboration environment to develop both artefacts. Although current results are promising and are starting to be fruitful, more work has to be conducted during the following months:

- Further alignment of the Tour Guide with the work produced by the technical chapters, ensuring there is bidirectional feedback. Particularly it is important that the Tour Guide Application uses exactly the same Docker descriptors as the official ones released by the GERis.
- Improve the structure of the Tour Guide Application so that it is more friendly for learning purposes. This will imply a refactoring to avoid automated procedures and substitute the former by manual procedures that will help learners to grasp what is happening behind the scenes
- A step by step guide to ease the learning process, following a try and tweak schema. This will be complementary to the previous point.
- Total alignment between the examples of the Tour Guide and the Tour Guide Application data, so that learners can experiment easily and conveniently.
- Use the Tour Guide application data as part of the API Cookbook of GERis. Particularly it is planned to do it with the [FIWARE NGSiv2 cookbook](#).
- In cooperation with the FIWARE Cloud Chapter, deploy an instance of the Tour Guide Application in the FIWARE Lab using the new Docker container service developed under such chapter.

All the future work is being tracked at <https://github.com/Fiware/tutorials.TourGuide-App/issues> together with Jira, so that the work is properly planned as part of FIWARE's Agile process.

Annex 1

Original Tour Guide Application Requirements

Introduction

This document is intended to enact requirements for a FIWARE App to be linked to the FIWARE Developers Tour Guide. The work will be divided into two phases corresponding to version 1 and version 2 of the application. Separate sections for each version are provided in order to facilitate work planning.

Overall description of the application

The proposed tutorial application is a smart, context-aware application which allows to manage large Restaurant chains which are operating worldwide. You can think off Starbucks or McDonalds as examples. These businesses usually operate as a franchise. The application is meant for both franchise managers and for customers. To this aim, the main functionalities that will be provided by the application are:

- Admit Customer reservations in accordance with current occupation and reservations made
- Register customer reviews according to different criteria (service, food, etc.).
- Real-time control of different parameters at each restaurant location (occupation, temperature (at cuisine and at the dining room), light conditions, noise levels, stocks, energy consumption, state of the refrigeration system, ...)
- Short time historic data of the different parameters monitored
- Publication of open data concerning the most relevant information about the different restaurant locations, grouped by different properties, namely location.
- Data analytics intended to determine the most efficient and effective restaurant locations and to suggest improvement actions
- Data analytics intended to determine / predict optimal food stocks in each restaurant location per date or time of year

Detailed description of functionalities (v1)

A detailed description of functionalities to be implemented for v1 is provided below.

User profiles

- Three different user profiles are identified:
 - **End users** who are restaurant customers. They can book seats and publish reviews about the service received at specific restaurant locations belonging to the franchise.
 - **Franchise managers**. They are managers / owners of one or more specific restaurants adhered to the franchise
 - **Global Managers**. They are managers at the origin organisation and can get access to all the information about reviews and restaurant locations. They use the information provided by the application in order to monitor the performance of the different restaurants

Users will enter into the application by introducing login and password. Depending on their profile they will have different functionalities available.

Customer Reservations and Reviews

A customer will be able to apply for a reservation (day, time and number of people). The system will decide if the reservation can be granted or not.

Once the service has been provided the customer can register a new review about the restaurant covering the following aspects:

- Service quality
- Food quality
- Place conditions

A review will include a general comment and a punctuation between 1 and 10 (maximum), for each aspect. Once a review is posted the system will recalculate the average score of the restaurant.

At any point in time a customer can get access to a map with restaurants and ratings and consult the different reviews made by other customers.

Real time Management of Restaurants

At any point in time a manager will be able to get access to the restaurants under his duty. He will have two different kind of views:

- The map view which will allow him to see where his restaurants are located and nearby points of interest, etc.
- A control dashboard which will allow him to visualize and manage the different facilities

From the Map View a manager will be able to obtain a summary of relevant information about each restaurant:

- Average score
- Current State (open, closed, on vacation)
- Current level of occupancy

Once a specific restaurant is chosen, from the Map view the manager can get access to the Control Dashboard. In order to speed up the process of finding a restaurant a search facility will be provided.

The Control Dashboard will provide the following applications:

- **Dining Room Control.** Allows to monitor different ambient parameters at the dining room. For v1 only temperature and ambient humidity will be controlled. With regards to temperature, a desired temperature can be set. As a result, the system will automatically actuate over the air-conditioning or the heating system. Manual control over these systems will be available as well. Short time historic graphs will be provided.
- **Cuisine Control.** It allows to control relevant parameters at the cuisine. At this stage temperature and ambient humidity will only be monitored. Short time historic graphs will be provided.
- **Occupation monitoring.** Allows to know the current state of the different tables. A 3D model will model the dining room, tables, seats, etc. so it will be easy to see the real time status of the place. Graphs about occupation in short time history will be provided
- **Customer Satisfaction.** It will allow to query reviews, know about average rates, etc.

Publication of data as Open Data

The following information will be published as open data (per restaurant location, per hour and date):

- Occupancy levels.
- Dining room temperature and ambient humidity

Big Data Analysis

This will be part of v2.

Real time analysis of streaming sources (cameras)

This will be part of v2.

Non-Functional Requirements

For v1 the front-end will be a (web) desktop application, capable of being consumed from tablet and mobile devices as well. A specific mobile experience is not required for v1, although it might be required for v2.

Implementation technologies outside FIWARE must be open source.

In order to facilitate comprehension by developers who might use different technologies and libraries, the software must be implemented using as few external libraries as possible. Particularly, the Web Application must be implemented using vanilla HTML5, Javascript and CSS. Javascript frameworks (jQuery, Angular, etc.) are not allowed. The application must work on modern Web Browsers, namely Google Chrome and Mozilla Firefox (including iOS and Android devices).

Simulators to create a large data sets are going to be required and will be specified separately.

Wireframes will be provided with suggestions about the specific UX to be implemented. Final visual designs for the tutorial application might be provided by Ogilvy (TBC).

The application must be deployable as local Docker containers or in the FIWARE Lab.

The expected FIWARE GEs to be used are those present at the FIWARE Developers Tour Guide, namely:

- Orion Context Broker
- IDAS
- PEP Proxy
- Key Rock / Access Control
- Complex Event Processing
- Short Time Historic (STH)

- Wirecloud
- 3D GEs