Proyecto Fin de Máster
Ingeniería de Telecomunicación

# Análisis de datos de un escenario FIWARE basado en Docker

Autor: Álvaro Martín Rodríguez
Tutor: Antonio Jesús Sierra Collado

**Dpto. Ingeniería Telemática**
**Escuela Técnica Superior de Ingeniería**
**Universidad de Sevilla**

Sevilla, 2018

Proyecto Fin de Máster
Ingeniería de Telecomunicación

# Análisis de datos de un escenario FIWARE basado en Docker

Autor:
Álvaro Martín Rodríguez

Tutor:
Antonio Jesús Sierra Collado

Dpto. Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2018

Proyecto Fin de Carrera: Análisis de datos de un escenario FIWARE basado en Docker

Autor: Álvaro Martín Rodríguez

Tutor: Antonio Jesús Sierra Collado

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2018

El Secretario del Tribunal

*A mi familia*

*A mi pareja*

*A mis amigos*

*A mis profesores*

# Agradecimientos

Hace ya muchos años que comencé mi andadura en la universidad, una andadura que además de llevar en lo alto mucho esfuerzo e ilusión, llegó salpicada por la compañía de muchas personas que aportarían su granito de arena en mi vida, la mayoría de las veces en lo profesional, pero otras muchas también en lo personal. Llegué con una familia que me animaba a conseguir lo que me proponía, amigos con los que compartir tanto los ratos como los retos, pero no quedó ahí la cosa.

Han sido muchos años en esta etapa universitaria, primero con en grado y después en el máster, y si algo han tenido en común, ha sido la gente fantástica con la que te encuentras por el camino. De pronto, te das cuenta de que ese rato del café, esa charla antes de las clases o la conversación de la hora de comer con esas personas con las que compartes risas y sufrimiento (y muchas veces, incluso forma de ver la vida), os ha vuelto amigos inseparables. De esos de los que necesitas saber de vez en cuando, porque si no te pones triste recordando las cosas que ya pasaron…

Y eso que, ¡no todos son compañeros de clase! La universidad es ese sitio donde todo cabe, y tan pronto conoces a alguien que estudia lo mismo que tú, como haces mil amigos nuevos al descubrir un nuevo *hobbie*, de esos que nos distraen y nos hacen pasar tan buenos ratos, como por ejemplo y para qué nos vamos a engañar, bailar. Que no os engañen, una vez se empieza, no se puede parar. Pero eso no es malo, y os daréis cuenta cuando, entre esas personas esenciales que llenan vuestro día a día dándole un color, se encuentren ellas, que se acercan preguntando,

*¿..bailas?*

*Álvaro Martín Rodríguez*

*Sevilla, 2018*

# Resumen

El análisis de los datos se ha convertido hoy en día en una parte fundamental de las tecnologías de la información, y nos permite sacar el máximo partido a los datos. Este trabajo se centra en aplicar un análisis y representación a un escenario consistente en un edificio donde exista una infraestructura de control de entradas, como puede ser un museo. Para ello, se ha desarrollado un escenario completo que comprende las siguientes fases:

Una primera fase basada en la producción de datos, empleando elementos del internet de las cosas como son: sensores, placas Arduino, Raspberry Pi, transceptores radio, etc. En esta fase se utilizan estos elementos para la producción de los datos del escenario, realizando la conexión entre las placas mediante comunicación inalámbrica en la banda 2.4GHz.

Una segunda fase basada en la recolección de los datos de la fase anterior. Para ello se emplea la plataforma FIWARE, concretamente el *context broker Orion*, que hace de intermediario entre el escenario y el sistema de almacenamiento de los datos. Por un lado, los datos son notificados desde una placa Raspberry Pi a Orion context broker, que mantiene los últimos valores obtenidos de los sensores en una base de datos MongoDB. Por otro, Orion a su vez los notifica a la sección de almacenamiento persistente que hemos desarrollado en este prototipo. Estos datos notificados por Orion son almacenados en una base de datos MySQL.

La tercera fase realiza la recolección de datos externos, conectándose a redes sociales para obtener la opinión de los usuarios o visitantes. Concretamente se emplea la red social Facebook, en la que se ha creado una página específica para este proyecto que permite recabar valoraciones de los usuarios.

A continuación, la cuarta y última fase realiza el análisis de los datos, empleando herramientas de procesamiento, análisis y visualización. Estas herramientas comprenden librerías específicas en Python como *Pandas* o *scikit-learn*, que permiten el análisis de los datos, o *matplotlib* que permite la representación y será usada para mostrar los resultados. Estos resultados se almacenan en la base de datos MySQL antes mencionada.

En este trabajo se ha desarrollado un prototipo funcional capaz de monitorizar datos obtenidos de los sensores desplegados en el escenario y realizar un análisis de los datos obtenidos del escenario y las redes sociales.

# Abstract

Nowadays data analysis has become a fundamental part of information technology, because it allows us to and allows us to make the most of the data. This work focuses on the analysis and representation in a scenario consisting of a building that has an infrastructure for ticket control, such as a museum. For this, a complete scenario has been developed that includes the following phases:

A first phase based on internet of things elements such as: sensors, Arduino, Raspberry Pi, radio transceivers, etc. In this phase these elements are used to produce the scenario data, making the connection between the boards through wireless communication in the 2.4GHz band.

A second phase based on the data collection of the previous phase. To do this, the FIWARE platform is used, specifically the Orion context, which acts as an intermediary between the scenario and the data storage system. On one hand, the data from a Raspberry Pi board is notified to an Orion context broker, which keeps the latest data collected from the sensors in a MongoDB database, and notifies them to the persistent storage section that we have developed in this prototype These data are stored in a MySQL database.

The third phase performs the collection of external data, connecting to social networks to obtain the opinion of users or visitors. Specifically, the social network Facebook is used, in which a specific page has been created for this project that allows to obtain users valuations.

Then, the fourth and last phase perform the analysis of the data, employing processing, analysis and visualization tools. These free tools are specific libraries in Python like *pandas* or *scikit-learn*, which allows the analysis of the data, or *matplotlib* that allows the representation and is used to show the results. These results are stored in the aforementioned MySQL database.

In this work we have developed a functional prototype capable of monitoring the data obtained from the sensors deployed on the stage and perform an analysis of the data obtained from the scenario and social networks.

# Índice

**Agradecimientos** **i**

**Resumen** **iii**

**Abstract** **v**

**Índice** **vii**

Índice de Tablas **ix**

Índice de Figuras **xi**

**1 Introducción** **1**

**2 Estado del Arte** **3**

*2.1 Análisis de Datos* *3*

2.1.1 Técnicas 4

2.1.2 Herramientas y frameworks disponibles 4

*2.2 FIWARE* *7*

2.2.1 Cosmos 8

*2.3 Docker* *8*

2.3.1 FIWARE sobre Docker 9

*2.4 Aplicación del Análisis de Datos en la actualidad* *9*

2.4.1 Big Data aplicado al turismo 10

*2.5 Conclusiones* *11*

**3 Diseño del Escenario y Fases** **13**

*3.1 Producción de Datos* *13*

*3.2 Registro de Eventos en FIWARE* *14*

*3.3 Persistencia de los Datos* *15*

*3.4 Análisis de los Datos* *17*

**4 Análisis de los Datos** **19**

*4.1 Objetivo* *19*

*4.2 Consideraciones previas* *19*

*4.3 Técnicas de análisis* *20*

*4.4 Algoritmos e implementación* *21*

4.4.1 Análisis de clusters 21

4.4.2 Análisis de regresión 23

4.4.3 Análisis de opinión o sentimiento 23

4.4.4 Análisis de serie de tiempos 24

*4.5 Persistencia de resultados* *24*

**5 Implementación** **27**
*5.1 Escenario real y escenario lógico* *27*
*5.2 Fase I: Producción de Datos* *30*
5.2.1 Arduino 30
5.2.2 Raspberry Pi 3 34
*5.3 Fase II: Recolección de Datos Internos – FIWARE* *37*
*5.4 Fase III: Recolección de Datos Externos – Facebook* *40*
5.4.1 Creación de una página en Facebook para el caso 40
5.4.2 Uso de la API – Obtención de un *token* 43
5.4.3 Implementación de la aplicación recolectora 46
*5.5 Fase IV: Análisis de los Datos* *47*

**6 Pruebas de Funcionamiento** **49**
*6.1 Fase I: Producción de Datos* *50*
*6.2 Fase II: Recolección de Datos Internos* *51*
6.2.1 Consideraciones previas 51
6.2.2 Recolector de datos de Orion 52
*6.3 Fase III: Recolección de Datos Externos* *54*
6.3.1 Empleo de la página de Facebook 54
6.3.2 Comprobación del *token* obtenido 56
6.3.3 Recolección de datos mediante la aplicación 57
*6.4 Fase IV: Análisis de Datos* *58*
6.4.1 Análisis descartados 64
6.4.2 KPI 65

**7 Conclusiones** **67**
*7.1 Lineas Futuras* *67*

**Referencias** **69**

Glosario **73**

**Anexos** **75**
*1. Compilación en RPi3 del Receptor (RF24, C++)* *75*
*2. Acceso a la Base de Datos mediante Python* *77*
*3. Instalación de librerías y frameworks* *78*

# Índice de Tablas


Tabla 1 – Campos de la tabla Museo 16
Tabla 2 – Campos de la tabla Evento 17
Tabla 3 – Campos de la tabla Facebook 17
Tabla 4 – KPI definidos para el museo 18
Tabla 5 – Datos a procesar 21
Tabla 6 – Clusters o grupos definidos para clasificación 21
Tabla 7 – Consideraciones del análisis de *clusters* 22
Tabla 8 – Definición de la tabla ACluster 25
Tabla 9 – Definición de la tabla ATiempos 25
Tabla 10 – Definición de la tabla AOpinion 25
Tabla 11 – Definición de la tabla ARegresion 25
Tabla 12 – Características hardware de los equipos empleados 29
Tabla 13 – Conexiones NRF24L01 – Arduino UNO 30
Tabla 14 – Conexiones HC-SR04 (1) – Arduino UNO 30
Tabla 15 – Conexiones HC-SR04 (2) – Arduino UNO 31
Tabla 16 – Conexiones DHT11 – Arduino UNO 31
Tabla 17 – Códigos de evento 33
Tabla 18 – Conexiones NRF24L01 – Raspberry Pi 3 34
Tabla 19 – Parámetros mínimos de la prueba del sistema 49
Tabla 20 – Clasificaciones esperadas de los análisis de *clustering* y opinión 61
Tabla 21 – Valores de los KPI definidos en la prueba 65

# Índice de Figuras


Ilustración 1 – Proceso de obtención de inteligencia 3
Ilustración 2 – Interés porcentual en Apache Hadoop [4] 5
Ilustración 3 – Interés porcentual en Apache Spark [6] 5
Ilustración 4 – Tiempo de ejecución de una regresión logística en Hadoop vs Spark [5] 5
Ilustración 5 – Interés porcentual en Python SciPy [8] 6
Ilustración 6 – Interés porcentual en Python NumPy [9] 6
Ilustración 7 – Interés porcentual en Python Pandas [10] 6
Ilustración 8 – Interés porcentual en R (Lenguaje de Programación) [13] 7
Ilustración 9 – Cabecera de infografía comparativa entre R y Python para análisis de datos 7
Ilustración 10 – Interés porcentual en FIWARE [17] 7
Ilustración 11 – Arquitectura FIWARE y Context Broker [18] 8
Ilustración 12 – Estructura de un sistema de contenedores Docker [20] 8
Ilustración 13 – Interés porcentual en Docker [21] 9
Ilustración 14 – Estructura de contenedores de Docker con FIWARE [22] 9
Ilustración 15 – Conclusiones del estudio sobre el Museo Reina Sofía [26] 11
Ilustración 16 – Escenario completo 13
Ilustración 17 – Esquema de captura de datos 14
Ilustración 18 – Detección de entradas o salidas 14
Ilustración 19 – Registro de Eventos en FIWARE 15
Ilustración 20 – Sección de procesado de datos 16
Ilustración 21 – Diagrama de flujo - KMeans 23
Ilustración 22 – Escenario real de la implementación 28
Ilustración 23 – Repartición de tareas por equipos. Pila de software. 29
Ilustración 24 – Conexiones placa Arduino UNO 31
Ilustración 25 – Diagrama de estados implementado en Arduino UNO 32
Ilustración 26 – Vista aérea y lateral del despliegue del sistema Arduino (placas) 33

Ilustración 27 – Sensor de proximidad colocado transversalmente a la zona de paso, verticalmente 34
Ilustración 28 – Conexiones placa Raspberry Pi 3 35
Ilustración 29 - Estructura de directorios de la aplicación de transmisión a Orion 36
Ilustración 30 – Vista aérea del despliegue del sistema Raspberry Pi 37
Ilustración 31 – Estructura de directorios de la aplicación de recolección de Orion 39
Ilustración 32 – Creación de una página en Facebook paso 1 40
Ilustración 33 - Creación de una página en Facebook paso 2 40
Ilustración 34 – Creación de una página en Facebook: selección de categoría 41
Ilustración 35 - Creación de una página en Facebook: nombre de la organización 41
Ilustración 36 - Creación de una página en Facebook paso 3: imagen de la página 41
Ilustración 37 - Creación de una página en Facebook paso 3: descripción de la página 41
Ilustración 38 - Creación de una página en Facebook paso 4: nombre de usuario 42
Ilustración 39 – Identificador de página 42
Ilustración 40 – Graph API GET sobre */me/accounts* 44
Ilustración 41 – Obtención de los datos con el nuevo *token* 44
Ilustración 42 – Información del *token* 45
Ilustración 43 – *Page Token* en la herramienta de gestión 45
Ilustración 44 – *Page token* final obtenido para la aplicación 45
Ilustración 45 – Características del *token* final 46
Ilustración 46 - Estructura de directorios de la aplicación de recolección de Facebook 46
Ilustración 47 – Estructura de directorios de la aplicación de procesado 47
Ilustración 48 – Uso de memoria según Hyper-V 49
Ilustración 49 – Depuración del receptor en C++ 50
Ilustración 50 – Depuración del conjunto receptor 51
Ilustración 51 – Orion en funcionamiento 51
Ilustración 52 – Comprobación del funcionamiento de la base de datos 52
Ilustración 53 – Ejecución del recolector de Orion 54
Ilustración 54 – Error 2006 en la conexión MySQL 54
Ilustración 55 – Acceso a la sección de opiniones en una página de Facebook 55
Ilustración 56 – Opción *escribir una opinión* 55
Ilustración 57 - Apartado de opiniones: comprobación 56
Ilustración 58 – Validación del *Page Token* obtenido mediante petición REST externa 56
Ilustración 59 – Ejecución del recolector de Facebook 57
Ilustración 60 – Primera iteración sobre las opiniones en Facebook 57
Ilustración 61 – Segunda iteración sobre las opiniones en Facebook 58
Ilustración 62 – Cantidad de registros en cada tabla existente en la base de datos 58
Ilustración 63 – Salida del procesado de los datos 59
Ilustración 64 – Porcentaje de ocurrencia de las humedades 59
Ilustración 65 – Porcentaje de ocurrencia de las temperaturas 60

Ilustración 66 – Valoraciones obtenidas de los usuarios en Facebook 60
Ilustración 67 – Resultados del análisis de clusters: KMeans 61
Ilustración 68 – Clasificación KMeans: representación gráfica 61
Ilustración 69 – Resultados del análisis de opinión 62
Ilustración 70 – Resultado cuantitativo del análisis de opinión 62
Ilustración 71 – Ejempos de opiniones: positiva y neutra 63
Ilustración 72 – Ejemplos de opiniones: positiva no pura y negativa 63
Ilustración 73 – Análisis de regresión: registro escrito 63
Ilustración 74 – Análisis de regresión: resultado 64
Ilustración 75 – Fin de la ejecución del análisis 64
Ilustración 76 – Comprobación del almacenamiento en tablas 64
Ilustración 77 – Estructura de directorios del repositorio de la librería RF24 75
Ilustración 78 – Instalación de la librería MySQLdb 77
Ilustración 79 – Prueba de la librería MySQLdb con una base de datos de ejemplo 77

# 1 INTRODUCCIÓN

*Solo hay un principio motriz: el deseo*

Aristóteles

Mientras que una persona es estudiante, busca formarse respecto a los temas que son de su interés o del campo que le apasionan. Ya durante el grado, dispone de diferentes especialidades entre las que elegir, y posteriormente en el máster de nuevo cuenta con asignaturas que le permiten enfocar su aprendizaje. Sin embargo, la inabarcable cantidad de información existente en el campo de las telecomunicaciones, la telemática, y las nuevas tecnologías punteras (inteligencia artificial, análisis de datos, internet de las cosas, etcétera) hace que nos frustremos en nuestro intento por dominar todos estos campos.

Este trabajo ha sido desarrollado con el objetivo de cubrir estos temas, y así ahondar en aquello que no ha sido posible durante las clases. Para ello, podemos definir el **alcance** de nuestro proyecto como sigue:

- Estudio del estado del arte respecto al análisis de datos y el software asociado.
- Diseño de un escenario IoT empleando sensores y software del estado del arte.
- Selección y ajuste de los análisis que se llevarán a cabo sobre los datos
- Implementación de dicho escenario con hardware IoT (Arduino, Raspberry Pi, etcétera).
- Pruebas de funcionamiento o concepto con la implementación realizada.
- Extracción de conclusiones
- Proposición de líneas futuras de trabajo

Para llevar a cabo todas estas tareas, supondremos que el escenario de partida se trata de un museo u otra entidad que busca estudiar o controlar el acceso a sus instalaciones. Con ello, esperamos que el lector aprenda tanto sobre el tema como el alumno al realizar este trabajo.

# 2 ESTADO DEL ARTE

*El progreso consiste en el cambio*

Miguel de Unamuno

Para abordar el proyecto presentado, es fundamental realizar un análisis del estado actual, de manera que comprendamos los componentes implicados, cuales son las opciones disponibles, cuales de ellos emplearemos en nuestra implementación y por qué motivos.

Por ello a continuación realizaremos un estudio de los componentes y facetas de este proyecto, compararemos opciones y herramientas para llevarlo a cabo, y elegiremos los más adecuados para desarrollarlo.

## 2.1 Análisis de Datos

Cuando hablamos de datos, nos referimos a una serie de valores correspondientes a variables o magnitudes. Estos datos de por si pueden no ser relevantes, pero el disponer de ellos en un conjunto suficientemente grande relacionados por otra variable, como por ejemplo el tiempo, puede hacer que extraigamos cierta información de ellos mediante un procesado de estos valores. Esto no tiene por qué ser exclusivamente así, sino que un dato puede aportar información por sí mismo, pero a su vez, si está relacionado con otra variable puede aportar una cantidad mayor aún. Por ejemplo, una temperatura aporta información, pero si está vinculada a un momento concreto del tiempo, puede aportar más aún, ya que se puede relacionar con otras temperaturas en otros momentos diferentes, o si estuviese ligada a un lugar, con otros lugares, y así sucesivamente.

De lo que acabamos de explicar, está claro que hay una información implícita en los datos, pero ¿cómo la podemos obtener? La respuesta claro está, es procesando los datos [1]: correlacionándolos, contándolos, comparándolos, etcétera. Es decir, **relacionando o estructurando los datos de manera significativa**. De esta manera, conseguiremos obtener la información relevante que subyace en este conjunto de datos. Por ejemplo, un montón de temperaturas son simples datos, pero la media de estas temperaturas para un día concreto ya es información para nosotros, pues nos ayuda a decidir posteriormente cosas como la ropa con la que queremos vestir para no pasar frio o calor, etc. Este no es más que el primer paso, pues una vez se dispone de información y se genera ya en las personas y no analíticamente, el **conocimiento como comprensión** de la información. En este punto, influyen la base de conocimiento previo que permite la comprensión, la experiencia, etc. Finalmente, la **inteligencia trata de interpretar la información, reconocer oportunidades** debido a reconocer patrones o estructuras que pueden llevarnos a anticiparnos.

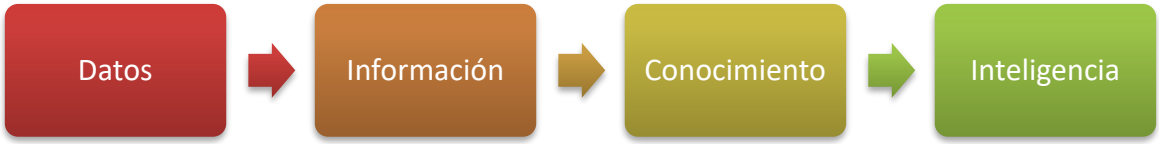


Ilustración 1 – Proceso de obtención de inteligencia

En conjunto, al movernos desde los datos a la inteligencia, nos estamos moviendo desde la **utilidad**

**inmediata,** hacia la **oportunidad**, mediante el **tiempo y el esfuerzo** que requieren las tareas de transformación de una etapa a la siguiente: **procesado, aprendizaje, experiencia**, etc.

### 2.1.1 Técnicas

Para analizar grandes cantidades de datos, hay multitud de técnicas [2] que se diferencian en subgrupos en función del tipo de datos que deseemos analizar. Así, las variables o datos que estudiamos pueden ser, según la clásica diferenciación empleada en estadística:

- **Cuantitativas**: se define su valor mediante un número, y por tanto puede ser **continuas o discretas**. Por ejemplo: el peso de una persona, el número de personas en una sala, una edad, etc.
- **Cualitativas**: se define su valor como un atributo, es decir un valor no numérico, y suelen requerir un instrumento o medio para ser medidas. Por ejemplo: el color, un aroma, un sabor, una impresión, etc.

En ocasiones, tendremos objetos descritos por variables de un tipo u otro, por lo que sería posible aplicar técnicas de que en principio solo serían aplicables a uno de los dos tipos de variables. Estas técnicas son, entre otras:

- **Análisis de clusters**: busca agrupar los objetos por su similitud entre sí, basado en ciertas características. Un posible caso de uso sería separar turistas según lo que más les interesa.
- **Árboles de decisión**: busca obtener una salida o clasificación, mediante una serie de comprobaciones lógicas sucesivas, donde el resultado de la comprobación *i* implica una comprobación *i+1* diferente.
- **Análisis factorial**: pretende relacionar las variables independientes, con las variables dependientes entre sí, aunque sea parcialmente.
- **Análisis de regresión**: busca estimar encontrar la relación total entre variables dependientes y cada una de las variables independientes.
- **Análisis multivariable**: este análisis estudia las variables a la vez, para comparar o correlacionar sus resultados.
- **Análisis de regresión multivariable**: es la combinación de ambos análisis antes explicados.
- **Análisis de segmentación**: trata de separar una serie de variables en segmentos que tienen o parecer tener características en común.
- **Análisis de opinión o sentimiento**: busca categorizar la opinión o actitud de la persona que ha escrito la opinión sobre un tema o problema.
- **Análisis por simulación**: mediante un modelo con las características necesarias para representar un sistema real, una simulación busca emular esta realidad para predecir características o comportamientos.
- **Análisis de serie de tiempos**: se trata de analizar datos que están separados por un mismo intervalo, y ordenados en un eje temporal. Busca poder realizar una predicción, o detectar una desviación en una monitorización si un valor se sale de lo esperado.

### 2.1.2 Herramientas y frameworks disponibles

- **Apache Hadoop** [3]: se trata de un framework de software pensado para ejecutar aplicaciones distribuidas sobre muchos nodos, formando un clúster. Las aplicaciones que pueden ejecutar siguen el modelo MapReduce, y son programadas en Java.

Debido a que Apache Hadoop es un framework pensado para aprovechamiento de muchas máquinas (*cluster*), no tiene mucho sentido emplear esta plataforma para nuestro proyecto. Su limitación al paradigma Map-Reduce ha hecho que el interés en Hadoop se haya visto reducido tras alcanzar un pico en abril de 2015, como

refleja Google Trends[1] [4], y podemos apreciar en el siguiente gráfico, en favor de otras soluciones más flexibles como Apache Spark.


Ilustración 2 – Interés porcentual en Apache Hadoop [4]

- **Apache Spark** [5]: se trata de una plataforma unificada para procesar datos a gran escala. El interés en esta plataforma se ha visto incrementado [6] debido a la flexibilidad que tiene para permitir todo tipo de aplicaciones corriendo sobre su motor, programadas en diferentes lenguajes como Python, Java, o SQL, y a su vez consiguiendo una alta eficiencia en su ejecución.


Ilustración 3 – Interés porcentual en Apache Spark [6]

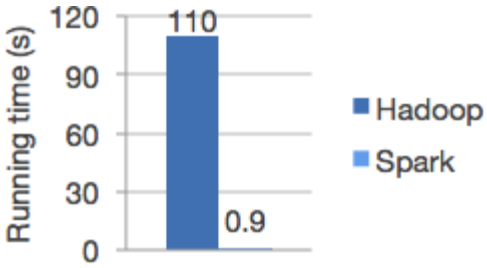


Ilustración 4 – Tiempo de ejecución de una regresión logística en Hadoop vs Spark [5]

Además, Apache Spark no tiene por qué ser ejecutado en solitario ya que el ser un motor (como ellos mismos lo definen), puede funcionar de manera autónoma, sobre Hadoop YARN, en contenedores Docker, etc.

- **SciPy** [7]: se trata de un framework open-source para Python, centrado en las matemáticas, ciencia, ingeniería, etc. En la práctica, es usado para análisis de datos cuando el volumen de estos es no es masivo. Si bien el interés según Google Trends es irregular en SciPy, podemos ver que en promedio es ascendente, además de que este comportamiento está causado por la composición de SciPy, que al estar dividido en diferentes librerías, hacen que la búsqueda sea dirigida directamente a dichas librerías y no a SciPy al completo.

[1] Todos los estudios en Google Trends han sido realizados en el intervalo de tiempo 2004-Actualidad, seleccionando España como región exclusivamente.

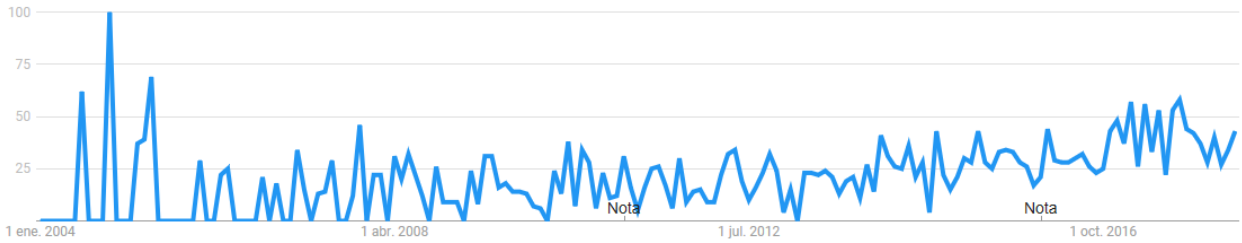


Ilustración 5 – Interés porcentual en Python SciPy [8]

Como hemos comentado, si ahora estudiamos la relevancia de NumPy, una de las librerías más usadas para el análisis de datos de SciPy, comprobamos que la tendencia es indudablemente ascendente en su uso:


Ilustración 6 – Interés porcentual en Python NumPy [9]

También podemos observar este comportamiento en la librería Pandas, parte de SciPy dedicada a las estructuras y el análisis de los datos:


Ilustración 7 – Interés porcentual en Python Pandas [10]

Como pregunta, cabe destacar nuestro interés en saber si el uso de Pandas y el de NumPy es excluyente, complementario, requerido como dependencia, etc. Para resolverlo, podemos fijarnos por ejemplo en un comentario extraido de la red de preguntas y respuestas, Stackoverflow, donde un usuario se planteó la misma pregunta y el autor principal de Pandas le respondió [11]:

---

*De hecho, Pandas provee herramientas de manipulación de datos a alto nivel, construidos empleando NumPy. NumPy, por sí mismo es una herramienta de bajo nivel, comparable al uso de MATLAB. Por otra parte, Pandas provee de funcionalidades para series de tiempo, alineamiento de datos, estadísticos cuando existen datos faltantes (…)*

---

- **R** [12]: se trata de un entorno estadístico y gráfico de libre uso. Su uso se ha visto extendido [13] debido a su facilidad para describir modelos estadísticos, lo que le ha llevado a ser usado principalmente en áreas de ciencias puras o economía, donde no se usaban previamente otros lenguajes de programación. Su crecimiento se ha visto frenado según los analistas, por el uso cada vez más prominente de Python (junto con SciPy) para realizar estos análisis en el entorno ingenieril y el cercano a este. Al ser un entorno con su correspondiente y con ello disponer de su propio lenguaje de programación, requiere de la instalación de su propio software para poder ser ejecutado.

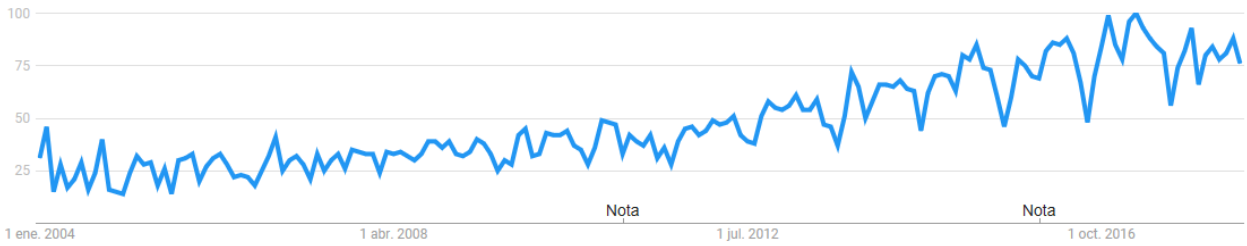


Ilustración 8 – Interés porcentual en R (Lenguaje de Programación) [13]

- Para finalizar, podemos acudir a una infografía disponible en Datacamp [14] que nos permitirá comparar de manera más directa los lenguajes R y Python.


Ilustración 9 – Cabecera de infografía comparativa entre R y Python para análisis de datos

En esta infografía, se comparan aspectos como el objetivo de cada lenguaje, las comunidades que lo usan, las comunidades técnicas donde se pueden resolver dudas al respecto, la facilidad de aprendizaje o la usabilidad, los repositorios existentes e incluso algunos datos numéricos que pueden ser relevantes a la hora de decidirnos por uno u otro.

## 2.2 FIWARE

La plataforma FIWARE surgió en el año 2014 aproximadamente a partir de las propuestas del Horizonte 2020 de la Unión Europea. Se trata de un intento de **estandarizar una nube que permita conectar el IoT** [15], estando ya finalizado, aunque su desarrollo continúa siendo dirigido por la propia comunidad de FIWARE. Si estudiamos el interés en esta plataforma, encontramos que su inicio fue explosivo, debido a la gran aglomeración de empresas y noticias que generaron contenido y búsquedas en su planteamiento, desarrollo y lanzamiento, que después se vio reducido. Sin embargo, podemos empezar a observar un leve repunte en el interés por esta plataforma, ya que algunas empresas como Telefónica buscan convertirlo en el estándar de facto mediante su uso, que comenzó a impulsarse en 2015 [16].


Ilustración 10 – Interés porcentual en FIWARE [17]

Para lograr esta conectividad de elementos y a su vez permitir la adición de otras funcionalidades, **el elemento principal** de la nube de FIWARE es el llamadado **ORION Context Broker** [18], un intermediario que permite la conectividad a todo tipo de elementos, bien mediante su API NGSI que emplea REST, o bien mediante agentes. Estos agentes no son más que módulos intermediarios que pueden traducir otros formatos o protocolos de comunicación a NGSI, de manera que se posibilita de esta forma la conectividad con otros dispositivos que no puedan establecer comunicaciones mediante REST. Por otro lado, aplicaciones externas de desarrolladores pueden comunicarse con Orion a través de la API REST mencionada.

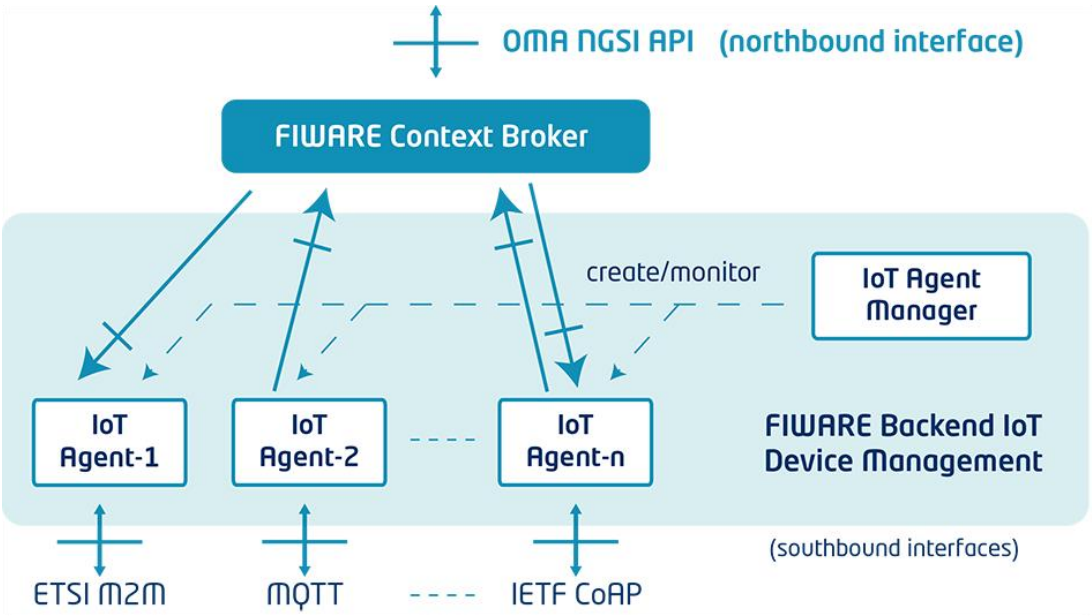


Ilustración 11 – Arquitectura FIWARE y Context Broker [18]

Todos los componentes de FIWARE son, según su propia nomenclatura, denominados Generic Enablers (GEs), dado que proveen nuevas características al sistema que otros componentes no pueden ofrecer: persistencia de los datos, comunicación, autenticación segura, etc.

### 2.2.1 Cosmos

Uno de los GE disponibles en FIWARE es Cosmos [19], destinado a permitir el análisis Big Data una vez que los datos han sido almacenados de manera persistente. Cosmos permite el análisis de los datos mediante aplicaciones del tipo *map&reduce*, o mediante Apache Hive.

## 2.3 Docker

Docker es un **sistema de contenedores de software**, que empaqueta el software en estos contenedores, de manera que contienen todo lo necesario para funcionar. Esto facilita el despliegue y mantenimiento de las aplicaciones, ya que para que la propia aplicación funcione evitamos requerir unas dependencias previas a la máquina *host*, provee de una capa a la vez de seguridad y de abstracción al aislar los contenedores del resto de dicha máquina, y evita la sobrecarga de necesitar ejecutar un sistema operativo completo sobre el host.

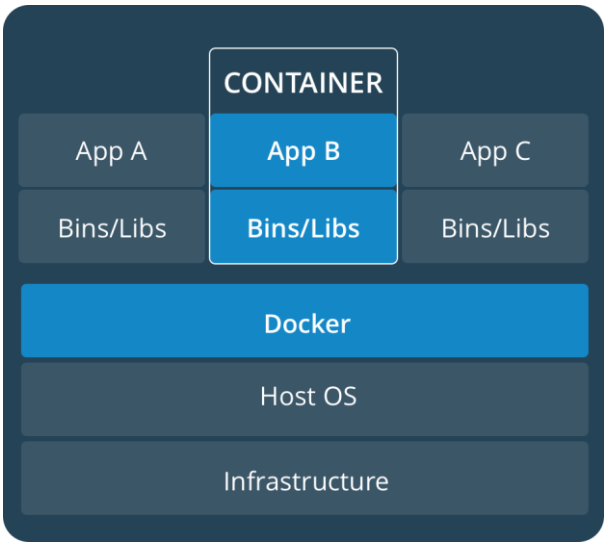


Ilustración 12 – Estructura de un sistema de contenedores Docker [20]

El interés en Docker ha sido creciente desde su lanzamiento, y nuestro uso de este sistema viene condicionado frente a otros sistemas de contenedores debido a que, además de ser el sistema más extendido y usado de este tipo, **dispone de contenedores FIWARE** que son los que deseamos emplear en nuestro proyecto.


Ilustración 13 – Interés porcentual en Docker [21]

### 2.3.1 FIWARE sobre Docker

De los múltiples componentes que existen en la arquitectura de FIWARE, la siguiente imagen [22] refleja cuales son aquellos que existen como contenedores en Docker, con un color oscuro los propios de FIWARE, y en un color más claro aquellos relacionados y que también existen y por tanto es viable emplearlos en un despliegue. Como podemos comprobar, **Orion está disponible**, y a su vez sería necesario emplear un contenedor de MongoDB para poder desplegar Orion y tenerlo en funcionamiento.

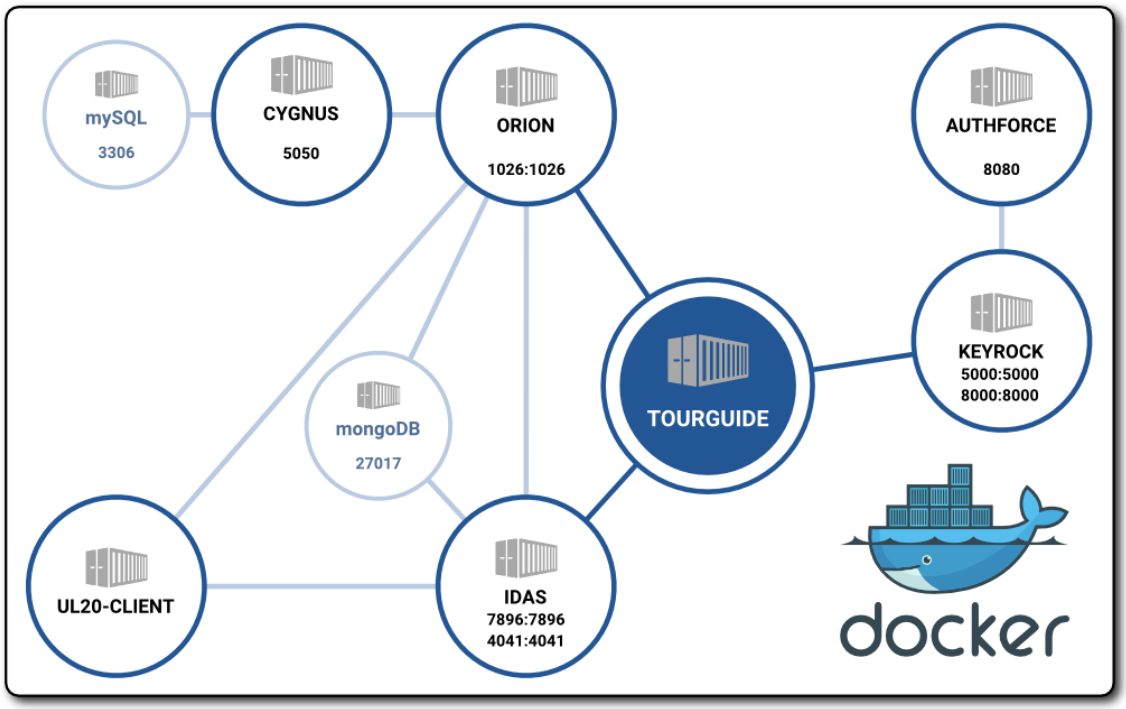


Ilustración 14 – Estructura de contenedores de Docker con FIWARE [22]

## 2.4 Aplicación del Análisis de Datos en la actualidad

La aplicación de las técnicas de análisis de datos a gran escala o *Big Data* llevan un tiempo entre nosotros, y las compañías y empresas privadas lo aplican para mejorar sus procesos o encontrar nuevas oportunidades de negocio. Sin embargo, empieza a ser frecuente incluso en entornos públicos, dando lugar a las denominadas *Smart Cities*, ciudades inteligentes donde todo es susceptible de ser conectado como un dispositivo IoT, desde sensores de temperatura hasta semáforos, o indicadores de nivel en contenedores de basura. Esto se esta llevando a cabo en Santander [23], donde el análisis de datos permite calcular rutas óptimas para recoger la basura según como de llenos están sus contenedores, etc:

*Toda la información recogida por los diferentes dispositivos es, además, puesta a disposición de los propios ciudadanos, que, bien a través de aplicaciones móviles o de los diferentes paneles informativos desplegados por la ciudad, pueden conocer en tiempo real, entre otras cosas, el estado del tráfico en cada momento para así poder diseñar la ruta menos congestionada. (…)*

Cabe destacar que esto ha sido llevado a cabo por Telefónica, mediante su plataforma *Thinking City*:

*La plataforma Thinking City de Telefónica es la base de ese gran cerebro de la ciudad. La plataforma ofrece API abiertas que permiten integrar datos tanto de los servicios urbanos (agua, residuos, energía, transporte, parques y jardines, etc.) como de los sistemas tradicionales existentes en el Ayuntamiento (Policía, Servicios Sociales…). Estos datos se almacenan en un repositorio de datos común, que es la base para el desarrollo y la implementación de toda la inteligencia de la ciudad.*

Y si accedemos al enlace que proporcionan para esta plataforma [24], encontramos que se encuentra basada en FIWARE como eje central.

### 2.4.1 Big Data aplicado al turismo

El uso o aplicación de las técnicas de Big Data en el turismo no es una idea original, sino que lleva tiempo siendo gestada tanto por organismos públicos [25] como por entidades privadas que buscan aplicar el Big Data para extraer inteligencia de los datos de los que disponen, permitiendo con ello la toma de decisiones tácticas y estratégicas que permitan mejorar el servicio, potenciar los negocios, reducir costes o aumentar las ganancias.

Concretamente, encontramos como ejemplo más reciente en España el caso del museo Reina Sofía [26]. Para llevar el análisis a cabo, el museo realizó acopio de los datos durante cinco meses, para después procesarlos y mejorar la experiencia de sus usuarios. Tomaron datos tanto internos como externos, siendo estos segundos los más importantes, debido a los pocos datos internos de los que cuentan en este tipo de instalaciones sobre los usuarios. Como detalle a mencionar, se encuentra que el museo ha anonimizado los datos, de manera que están desvinculados de las personas que los han producido, no poniendo en peligro por tanto su privacidad. Estos datos externos, incluían:

- Meteorología
- Escucha de Redes Sociales (Twitter …)
- Calendario de Festividades
- Datos de Movilidad

En base a estos datos, el estudio elaborado por el equipo de Movistar extrae una serie de conclusiones, centradas en valores como la asistencia al museo, la propia climatología, la nacionalidad de los visitantes, etc. Podemos ver un ejemplo de las conclusiones extraidas del estudio en la siguiente ilustración:


Ilustración 15 – Conclusiones del estudio sobre el Museo Reina Sofía [26]

## 2.5 Conclusiones

Como hemos podido observar, hay muchas técnicas y herramientas para llevar a cabo análisis de datos. De las distintas herramientas, emplearemos el framework de Python: **SciPy**, concretamente la librería **Pandas**, ya que nos permite:

- Procesar de forma flexible tanto conjuntos de datos grandes como pequeños. En caso de ser conjuntos muy grandes, podría combinarse por ejemplo con Apache Spark para usar un cluster.
- Evitar la instalación de un IDE para ejecutarlo, como ocurre con R.
- Disponer de una gran documentación web sobre su uso, debido a su extendido uso.
- Tener la capacidad, mediante librerías especificamentes diseñadas para ello como Pandas, de analizar los datos de los que disponemos.

En cuanto a técnicas, el conjunto disponible es elevado, y cada una de ellas podrá implementarse con diferentes algoritmos. Dado que hasta ahora no hemos estudiado nuestros datos disponibles, no podemos tomar una decisión sobre estas técnicas, por lo que esto será estudiado más en profundidad en el punto 4, Análisis de los Datos.

Por otro lado, emplearemos **FIWARE** como intermediario para conectar nuestro escenario IoT. La razón de uso de esta plataforma se encuentra en dos motivos principales:

- Permitir conectar un escenario IoT completo, sean cuales sean los componentes que lo conforman. Si bien nuestro escenario puede ser suficientemente pequeño como para no requerir el uso de esta plataforma, es útil de cara a la conceptualización del trabajo, su uso en un escenario de mayor tamaño y para el propio aprendizaje.
- La no existencia de plataformas similares que no sean privadas, por tanto, la única alternativa abierta a su uso para todo el mundo.

En tercer lugar, emplearemos **Docker** como sistema de contenedores para desplegar los componentes de FIWARE que necesitemos emplear. De esta manera, conseguiremos:

- Ahorrar tiempo y dificultades en el despliegue y configuración de los GE de FIWARE.
- Facilitar la actualización o reemplazo de componentes desplegados en contenedores.

Finalmente, hemos podido observar como el análisis de datos está a la orden del día, aplicándose estas técnicas

a ámbitos cada vez más diversos, como puede ser el turismo en el caso de uso que hemos expuesto. Emplearemos esta referencia como base para realizar un estudio de un pequeño escenario planteado por el alumno.

# 3 DISEÑO DEL ESCENARIO Y FASES

*La nueva información hace posible las nuevas ideas*

Zig Ziglar

El escenario planteado para llevar a cabo nuestro proyecto, incluirá distintos componentes, necesarios para las distintas fases o actividades implicadas para conseguir nuestro objetivo: la obtención de inteligencia a partir de los datos de entrada. Separando en función de estas fases, tendremos una topología general como la que sigue en la figura 16, que iremos desgranando en mayor detalle.

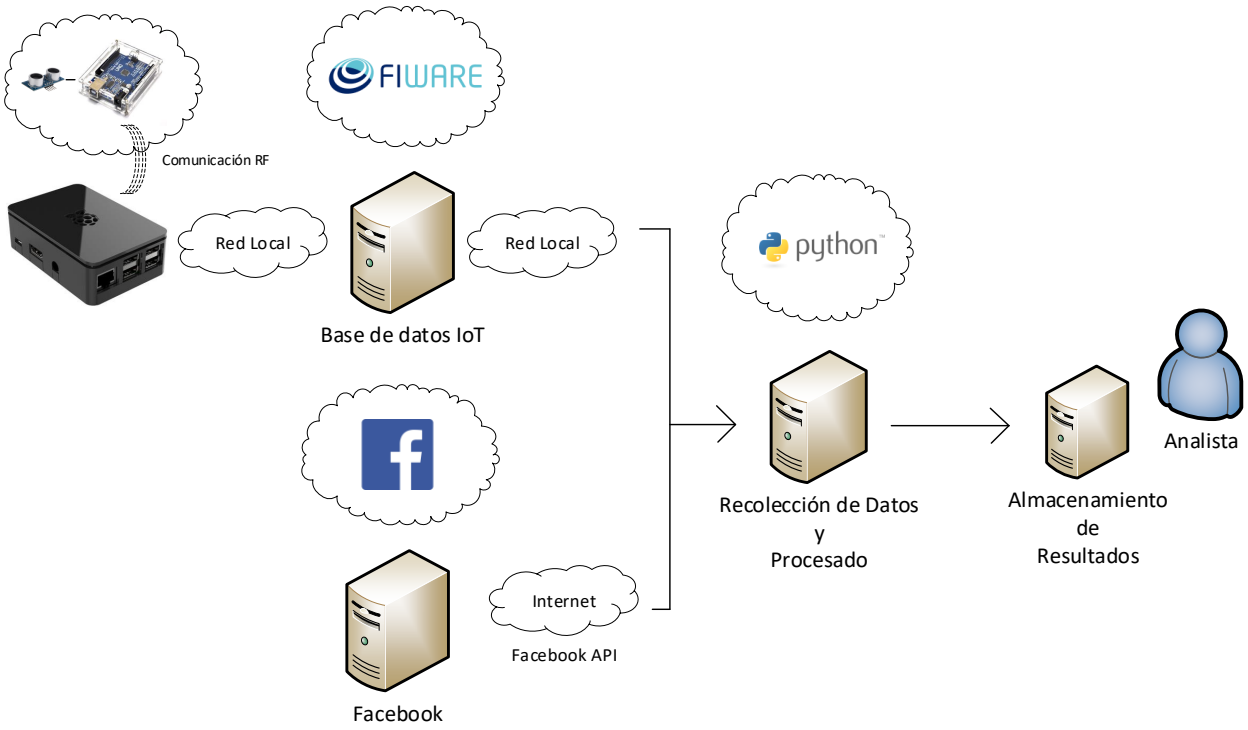


Ilustración 16 – Escenario completo

## 3.1 Producción de Datos

La captura de los datos en un escenario real implicaría, en el caso del turismo, la captura de eventos en los controles de entrada y salida de las instalaciones visitables, así como de la recolección de datos vía API de los servidores de la red social escogida para estudiar las reacciones de los visitantes, en nuestro caso, Facebook.

Dado que no contamos con una instalación de este tipo, la captura de estos datos se hará mediante una instalación que se realizará mediante una placa Arduino a la que se conectarán dos sensores de proximidad por ultrasonido. Con esto se capturarán datos similares a la entrada y salida de los usuarios de una instalación, donde los tornos o arcos de entrada/salida toman constancia de estos eventos. La placa se comunicará a su vez por radiofrecuencia (RF) a 2.4GHz con una Raspberry Pi 3, que hará la función de concentrador o hub,

enviando estos datos a la plataforma FIWARE. Esto puede verse con detalle en la figura siguiente:

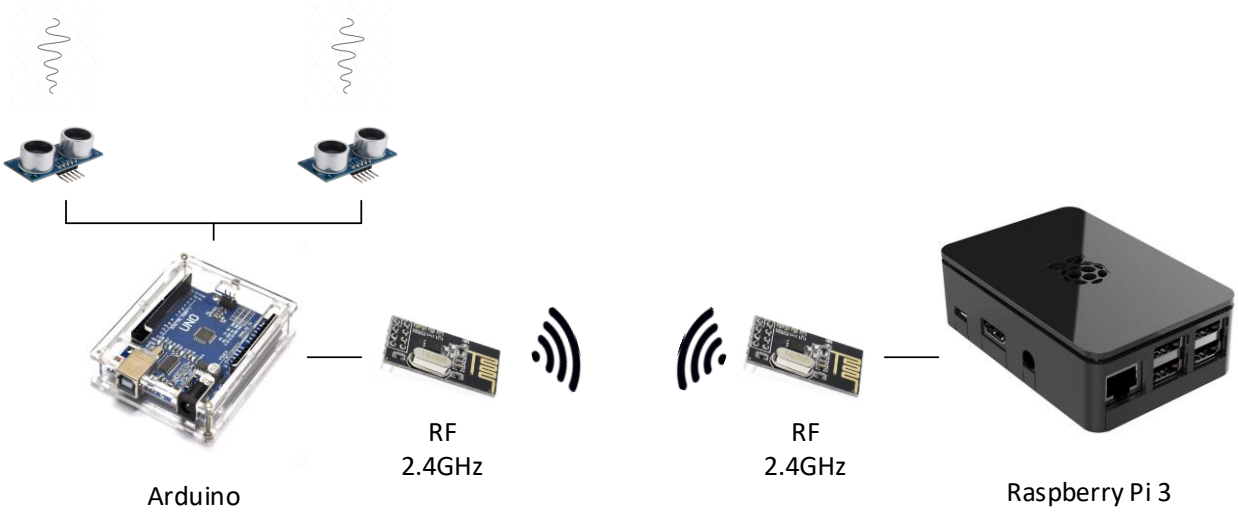


Ilustración 17 – Esquema de captura de datos

El método de funcionamiento es sencillo: los dos sensores de proximidad se encuentran situados perpendicularmente a la entrada a una estancia, y paralelos entre sí, separados medio metro aproximadamente. Cuando una persona accede a dicha sala o instalación, el sensor colocado más cerca de la entrada se activa, desactiva, y esto se repite con el segundo sensor. Detectamos así por tanto una entrada a la instalación, mientras que, si el orden de activación de los sensores es el contrario, lo que habremos detectado será una salida. Un ejemplo de entrada puede observarse en la figura 18.

Otras combinaciones posibles, como la activación y desactivación de un sensor, sin que a continuación se produzca la misma cadena en el otro sensor, serán descartadas al ser consideradas incorrectas, pudiendo significar que la persona se ha dado la vuelta, por ejemplo, y finalmente no ha entrado o salido de la sala.

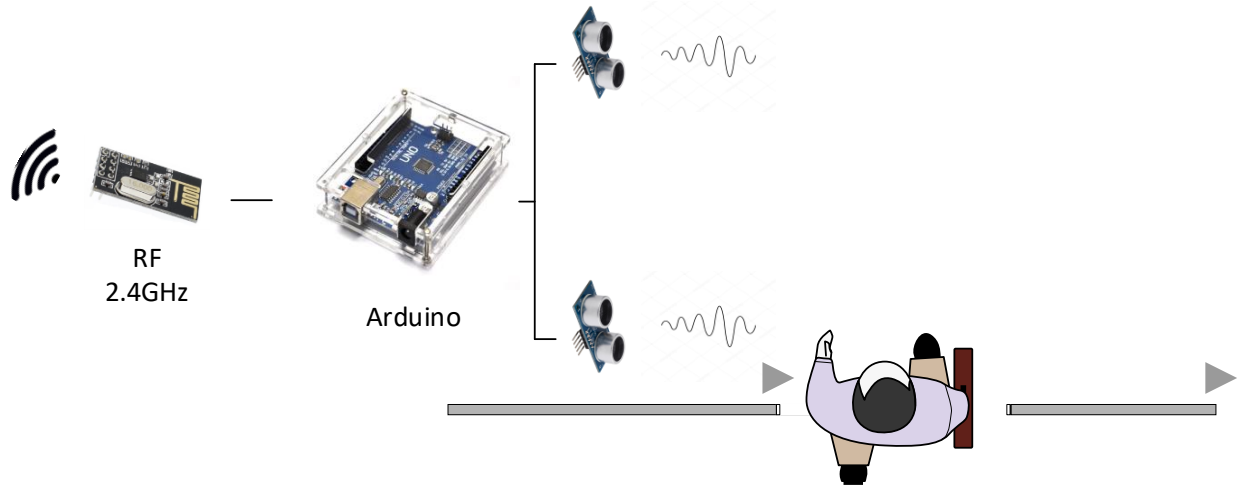


Ilustración 18 – Detección de entradas o salidas

Una vez reconocida una entrada o salida, generaremos un evento en la placa Arduino, que será transmitido al concentrador (Raspberry Pi), donde se dotará al evento de información extra: la fecha y hora en la que se ha registrado en el sistema. Una vez listo, el evento será insertado en el sistema FIWARE.

## 3.2 Registro de Eventos en FIWARE

Una vez hemos capturado y enriquecido los eventos con la información deseada, estos han de ser insertados en

la plataforma FIWARE. Concretamente, emplearemos ORION Context Broker [27], el intermediario de esta plataforma que es el que se comunica con los agentes IoT, que le envían los eventos o datos.

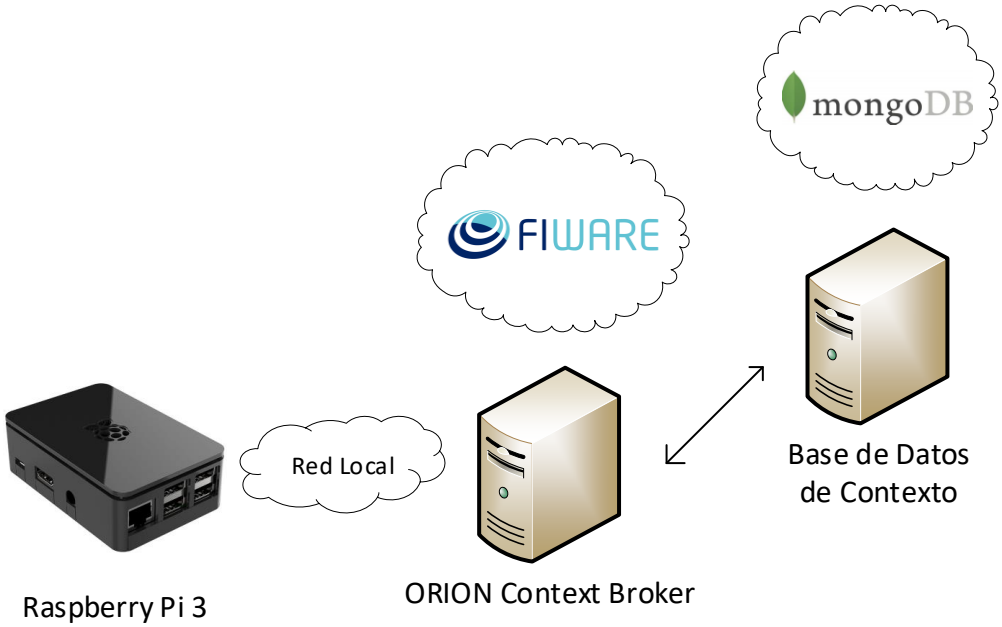


Ilustración 19 – Registro de Eventos en FIWARE

Concretamente, emplearemos los elementos que necesitemos sobre Docker, de manera que aprovechemos el rápido despliegue que permiten los sistemas de contenedores, y la fácil configuración que podremos hacer con este sistema.

El intermediario ORION es el elemento al que deberán conectarse los elementos externos a esta plataforma, por lo que es fundamental desplegarlo. Tras él, será necesario desplegar MongoDB, para que pueda almacenar los valores que reciba de los agentes. Mediante este enlace, será almacenado en mongoBD el contexto, es decir no se almacena el histórico de valores, sino el valor del contexto, el valor asociado a la entidad actualmente. Cada vez que un valor sea actualizado, este sustituirá al valor que hubiese anteriormente.

Al desplegar ORION mediante Docker, encontramos que los datos no son persistentes [28], por lo que, si detenemos el contenedor de MongoDB por algún motivo, perderemos los datos almacenados en él. Siguiendo la documentación propia de MongoDB para Docker que puede verse en [29], podríamos crear volúmenes en Docker que nos permitiesen almacenar de manera persistente la información en este punto. Sin embargo, en nuestro escenario supondremos que todos los elementos del sistema funcionan permanentemente, de manera que los datos no son perdidos en este punto, ya que son recolectados por el bloque de procesado, y además el sistema ORION más MongoDB nunca es desactivado (**sus contenedores Docker siempre se encuentran activos**).

## 3.3 Persistencia de los Datos

De manera que el acceso a los datos esté estandarizado y sea igual para todos los datos por parte del programa encargado de realizar el análisis, los datos serán almacenados todos previamente en una base de datos. De esta manera, tendremos dos tablas, una para los datos obtenidos mediante los sensores, y otra para los datos extraidos de Facebook.

Para disponer de los datos almacenados de esta forma, el apartado de la recolección de datos previa al procesado se separará en dos programas. Así, pasamos de una estructura general como se ve en la ilustración 16, a la estructura real implementada que se puede ver en la ilustración 20.

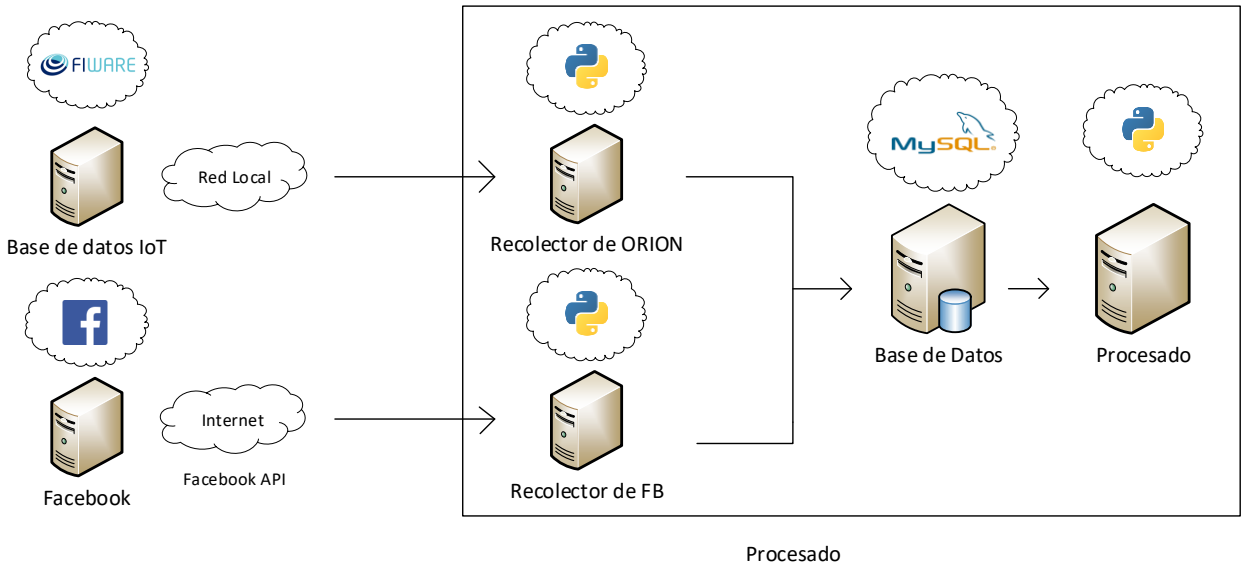


Ilustración 20 – Sección de procesado de datos

El primer programa, será el encargado de recibir los eventos procedentes de Orion mediante un sistema de publicador/suscriptor, esto es, el programa se suscribirá a Orion para ser notificado de los eventos que este registre sobre nuestro escenario. Cuando el programa recibe uno de estos datos, lo registra en la base de datos asociada, de manera que el programa de análisis pueda acceder a dicho histórico que ha sido almacenado de manera persistente. Dado que la información puede ser bien eventos de entrada o salida del lugar, o bien de temperatura y humedad, se emplearán dos tablas en la base de datos, una para cada tipo de información.

En segundo lugar, tendremos un programa que se encontrará funcionando de manera permanente como el primero, pero en lugar de recibir eventos será él mismo quien se encargue de pedir a los servidores de la red social mediante su API la información deseada. Cada vez que la información recibida incluya nuevos datos que no han sido recibidos en peticiones anteriores, serán almacenados en la base de datos, pero en una tabla diferente a la empleada por el programa anterior.

Para el almacenamiento crearemos tres tablas diferentes, teniendo en cuenta que los campos de texto pueden ser de tamaño fijo o variable según el tipo de campo [30], y que el tipo *varchar* hace que el **tamaño se ajuste al contenido**, será este el tipo de campo que usaremos para el texto. De cara a facilitar la implementación, permitiremos el uso de campos de texto para almacenar las fechas, ya que de esta manera evitaremos necesitar convertir las fechas del formato empleado en Python al formato empleado en MySQL, y viceversa al leerlo, en caso de que fueran distintos.

- *Museo*: será la tabla que almacene los datos generales del museo, es decir, la fecha y hora en la que se registran los datos, así como la temperatura y la humedad registradas en ese momento.

| Campo | Tipo |
|---|---|
| Fecha | VARCHAR(20) |
| Temperatura | FLOAT |
| Humedad | INT |

Tabla 1 – Campos de la tabla Museo

- *Evento*: será la tabla en la que se almacene la fecha y hora en la que se registra un evento, así como el evento asociado, pudiendo ser este o entrada o salida.

| Campo | Tipo |
|---|---|
| Fecha | VARCHAR(20) |
| Evento | VARCHAR(10) |

Tabla 2 – Campos de la tabla Evento

- *Facebook*: es la tabla que almacenará las opiniones de los usuarios emitidas sobre el museo, junto con su valoración numérica. Emplearemos además un campo que denominaremos MD5 [31], debido a que emplearemos el algoritmo del mismo nombre para obtener una huella de las opiniones. De esta manera, al cuando el programa recolector de opiniones busque si una opinión ya existe en la base de datos, no tendrá necesidad de comparar las opiniones una por una en la base de datos con todos los caracteres que estas implican, sino que podrá comparar las huellas que son de un tamaño mucho menor.

| Campo | Tipo |
|---|---|
| MD5 | VARCHAR(100) |
| Opinion | VARCHAR(1200) |
| Nota | INT |

Tabla 3 – Campos de la tabla Facebook

## 3.4 Análisis de los Datos

Finalmente, debemos procesar los datos para obtener la información relevante en nuestro caso de uso. Además, buscaremos realizar agregaciones de mayor nivel de abstracción, para tener indicadores de mayor relevancia para nuestro escenario del museo, de manera que podamos facilitar la toma de decisiones final en base a estos valores.

Esto se conseguirá mediante:

1. La definición de indicadores clave de rendimiento, comúnmente denominados por sus siglas en ingles: ***Key Performance Indicator*, KPI** [32]. Estos indicadores son valores que podemos calcular, y que nos van a indicar el rendimiento de un proceso, de manera que, si el indicador está bien diseñado, mediante el valor de este indicador podremos detectar si el proceso está funcionando incorrectamente, y por tanto intentar corregirlo para mejorar. Esto a su vez, corregiría el valor de este KPI, que pasaría a reflejar esta mejoría del funcionamiento del proceso con un valor más cercano al considerado óptimo para ese proceso.

A la hora de definir estos KPI, se siguen unos principios que se recogen en el acrónimo **SMART**, por sus siglas en inglés:

- E**s**pecíficos (Specific): medir o recoger algo concreto, no dejando lugar a las ambigüedades.
- **M**edibles (Measurable): mediante algún método o instrumento, representable numéricamente.
- **A**lcanzables (Achievable): realistas, no debe fijarse un KPI de algo que es imposible de conseguir por diferentes motivos.
- **R**elevantes (Relevant): para el proceso, afectando a su rendimiento o representándolo.
- **T**emporales (Timely): ser posible hacer un seguimiento de su evolución en el tiempo.

Según estas premisas, elaboraremos indicadores KPI adecuados que reflejen el comportamiento y la gestión del museo. Las distitnas ténicas, algoritmos, métodos o valores que puedan calcularse, podrán ser directamente KPI, o bien valores intermedios para calcular estos KPI. Supondremos algunos valores necesarios sobre el tamaño del museo o el número de visitas para definir estos KPI.

- Visitas por día: deben alcanzar un mínimo para que el museo cumpla su propósito de divulgación, así como para ingresar lo suficiente para poder mantener su colección. El umbral debe tener en cuenta la

temporada baja de visitas.

- Satisfacción: entendida mediante la fórmula que aparece a continuación, evalúa cuál es el porcentaje de visitantes satisfechos con su visita. El total de visitantes tenidos en cuenta es de aquellos que expresan su opinión mediante algún medio como las redes sociales, y no sobre el total real de visitas. Dado que siempre habrá algo que mejorar, no es realista esperar un 100%, pero la cifra de un 90% parece razonable como objetivo a cumplir.

**Ecuación 3–1.** Cálculo del KPI de satisfacción en base a las opiniones de los visitantes.

$$Satisfacción = \frac{Favorables}{Favorables + Desfavorables} \tag{3–1}$$

- Concurrencia: se trata de la cantidad de visitantes que tiene el museo en un momento cualquiera. Este parámetro busca repartir las visitas a lo largo de todo el horario de apertura del museo, para mejorar la experiencia de los visitantes y mejorar aspectos de seguridad.
- DVP: duración de la visita promedio. Debe ser un tiempo promedio entre el tiempo de visita de 1 sola exposición, y el tiempo de visita de la exposición completa. Se aceptan variaciones altas, dado que hay muchos factores que pueden afectar a este valor, como la visita acompañada por niños pequeños. Un valor excesivamente bajo, indicaría la falta de interés del público en lo que se expone.
- Valoración: del museo, en promedio entre 1 y 5 con todas las opiniones expresadas por los visitantes a través de las redes sociales. Como objetivo, similar a la satisfacción, esperaremos una nota por encima del 4, mientras que por debajo del 3 consideraremos necesarias acciones para solucionarlo.

En la siguiente tabla, recogeremos los KPI antes mencionados, indicando su valor óptimo u objetivo, así como el umbral que indicaría un valor que necesitaría acciones correctoras. Elaboraremos primeramente una tabla con valores reales para un museo, si bien debemos recordar, que en el apartado 6, Pruebas de Funcionamiento, verificaremos con una tabla similar y diferentes valores, ya que el entorno controlado de pruebas no es un museo real.

| KPI | Objetivo | Umbral |
|---|---|---|
| Visitas por día | 5.000 | 1.000 |
| Satisfacción | 90% | 50% |
| Concurrencia | 100 | 25 |
| DVP | 1h 30m | 45m |
| Valoración | 4 | 3 |

Tabla 4 – KPI definidos para el museo

Posteriormente en el apartado 4, Análisis de los Datos, se indicará con qué tipo de análisis se calcularán estos KPI.

2. La **generación de datos auxiliares**, como pueden ser la identificación del sentimiento de los visitantes en satisfechos o insatisfechos de manera individual con la visita, la relación de las visitas con las horas del día para permitir la toma de decisiones posterior al análisis en base a toda la información disponible, etc.

# 4 ANÁLISIS DE LOS DATOS

*Los datos no son información, la información no es conocimiento.*

Clifford Stoll

El análisis de los datos es la parte esencial de este proyecto, y por tanto profundizaremos en ella algo más que en las demás fases que hemos visto anteriormente. Debemos recordar al lector, que en el apartado 2.1 correspondiente al estado del arte, se ha proporcionado una visión general sobre algunas técnicas existentes para el análisis de datos, así como herramientas y marcos de trabajo disponibles para realizar dicho procesamiento.

Así pues, una vez obtenidos los datos tal y como se ha especificado en el apartado 3 de este trabajo, estos se deben procesar para obtener la información relevante subyacente, mediante la relación y estructuración de estos.

Una vez que se han proceado los datos, estos se almacenan para poder ser empleados en la toma de decisiones. Sin embargo, el proceso no acaba aquí, ya que el procesado tiene una temporalidad, lo que hace que estos puedan ser a su vez agregados por períodos (o no) y estudiados en base al momento temporal al que pertenecen, de manera que podamos obtener una mayor inteligencia de ellos mediante la correlación de los datos y al momento al que pertenecen. De la misma forma, podría hablarse de una localidad, u otro parámetro que permita relacionarlos conforme a un valor común.

## 4.1 Objetivo

Nuestro objetivo final a la hora de estudiar los datos es obtener una serie de indicadores de rendimiento (KPI) con los que caracterizar el funcionamiento del museo, así como **información relevante que pueda explicar el por qué** de los valores obtenidos.

No debe confundirse esta última idea, ya que los KPI como comentamos anteriormente deben ser medibles, con justificar numéricamente estos valores, sino con una explicación respecto a los visitantes o las condiciones concretas del museo o del entorno, es decir, una **explicación semántica**.

## 4.2 Consideraciones previas

Antes de comenzar el análisis, deben ser tenidas en cuenta unas consideraciones previas relativas a los datos que se emplean en el estudio [33]. Se puede considerar esta fase como una fase de estudio de los datos, en el sentido de que busca caracterizarlos antes de la fase real de procesamiento, en la que se busca mediante los datos una respuesta a una pregunta o situación planteada.

- Calidad de los datos.
- Calidad de las medidas.
- Transformaciones iniciales de los datos.
- Características de los datos.

- Limpieza de los datos

La realización de estas comprobaciones se hace mediante métodos estadísticos, como son por ejemplo el cálculo de estadísticos como la media, la varianza, etcétera para caracterizar la calidad de los datos; o métodos experimentales como es la medición de la desviación en medidas sucesivas para la calidad de las medidas.

Nosotros:

- No estudiaremos la calidad de los datos, ya que **su caracterización estadística** es la propia información sobre la que versa parte del **estudio que queremos realizar** según nuestros KPI.
- Caracterizaremos las medidas por la **fiabilidad de los dispositivos empleados** en la implementación.
- Realizaremos pequeñas transformaciones iniciales de los datos, ya que los datos originales proveídos por los sensores que se utilizarán en la implementación no son siempre relevantes para el sistema. Por ejemplo, no es relevante la medida de los sensores de proximidad, sino dato inferido de una entrada o una salida mediante la relación de medidas obtenidas por ambos sensores y la temporalidad entre dichas medidas.
- El mismo razonamiento aplicado a la calidad de los datos, puede ser aplicado a su caracterización.
- La limpieza de los datos busca errores, duplicaciones e incompletitudes en los datos. Debido al tratamiento previo que se les en nuestro sistema IoT, no necesitaremos realizar este apartado, ya que:
    - Los datos son **atómicos**, por lo que **no pueden estar incompletos**: entradas/salidas, temperatura o humedad.
    - Los datos **no pueden estar duplicados**, ya que cada uno se corresponde con un momento temporal concreto en el que se procesa, por lo que no pueden existir dos procesados en el mismo momento.
    - Los datos pueden contener errores:
        - Por error del hardware: rebotes adicionales de las ondas ultrasónicas de los sensores de proximidad, que provoquen activaciones adicionales no previstas en el algoritmo de reconocimiento de entradas y salidas.
        - Por error del algoritmo: (de reconocimiento implementado en el hardware), y por tanto no predecibles al estar fuera del alcance de la depuración previa que se lleve a cabo tras su diseño e implementación.
        - Por error en la transmisión radio: que se ve solucionado mediante el uso de una conexión TCP en lugar de UDP.

      Sin embargo, los dos errores que pueden alcanzar el estadio final, es decir la fase de procesado, no son ni siquiera reconocibles salvo que se hiciese un reconocimiento dato a dato en tiempo de ejecución, y bajo supervisión humana. Por tanto, podremos realizar correcciones limitadas sobre ellos que se explicarán en el apartado de implementación. Concretamente, la única corrección que podremos aplicar será:

        - Corregir eventos imposibles: eliminar registros de salidas que no sean posibles, debido a que el contador de personas dentro de la estancia sea cero cuando dicho evento se registre.

## 4.3 Técnicas de análisis

Una vez tenidas en cuenta las consideraciones previas, vamos a determinar las técnicas y tipos de análisis que vamos a realizar. Para ello, comencemos recordando mediante la siguiente tabla, los datos de los que podemos disponer para el análisis:

| Dato | Intervalo de valores | Unidad |
|---|---|---|
| Temperatura | -50 a 50 | ºC |
| Humedad | 0 a 100 | % |
| Evento | Entrada/Salida | - |
| Opinión | 1200 | Caracteres |
| Valoración | 1 a 5 | - |

Tabla 5 – Datos a procesar

Con estos datos, debemos obtener información relevante para nuestro caso de uso y, además, ser capaces de calcular los KPI antes definidos. Debemos recordar que cada uno de estos datos, **tiene asociado el tiempo en el que ha entrado al sistema** (salvo las opiniones y valoraciones, que tienen asociados el momento en el que fueron emitidos). Para ello aplicaremos, de las técnicas expuestas en el apartado 2.1.1 de este proyecto, las siguientes:

- **Análisis de clusters:** clasificaremos a los clientes/visitantes en:

| Grupo Original | Clasificación |
|---|---|
| Visitantes | Satisfechos / Insatisfechos |

Tabla 6 – Clusters o grupos definidos para clasificación

- **Análisis de regresión**: intentaremos relacionar los visitantes, con la hora del dia para adivinar sus horas preferidas para este tipo de actividad. Cabe destacar que la diferencia en este caso concreto con el análisis de serie de tiempos, es que buscamos establecer la relación entre las dos variables, tiempo y entradas, mientras que el otro pretende solamente realizar unos cálculos estadísticos en base al eje temporal.
- **Análisis de opinión o sentimiento**: sobre las opiniones de los visitantes en las redes sociales.
- **Análisis de serie de tiempos**: se empleará para calcular los KPI antes acordados:
    - Visitas por día
    - Concurrencia
    - DVP

    Además, sin ser soportados directamente por un eje temporal, pero también relacionados con los mismos métodos estadísticos, calcularemos:

    - Satisfacción
    - Valoración
    - Porcentaje de presencia u ocurrencia de:
        - Temperaturas
        - Humedades

## 4.4 Algoritmos e implementación

Hasta este punto, hemos definido los análisis o técnicas que vamos a realizar, con vistas a la información que queremos obtener o los datos que queremos relacionar. En este punto vamos a ver como se implementarán estos análisis, es decir, con qué algoritmos o formas de trabajo se van a conseguir realizar.

### 4.4.1 Análisis de clusters

Para clasificar en grupos o clusters distintos, debemos determiner tres cosas previamente:

1. Los datos que caracterizan a cada individuo a clasificar.
2. El algoritmo por el que se va a realizar la clasificación.
3. El número de grupos o clusters en los que deseamos clasificar a los individuos.

En nuestro caso, vamos a emplear el único dato por el que podemos clasificar a los visitantes: la valoración que han dejado sobre su visita al museo, que es una nota numérica entre 1 y 5 sin decimales. Dado el escaso margen de información y de diferenciación en este único dato que se emplea, el número de clusters será dos, siendo uno de estos grupos visitantes satisfechos con su experiencia, y el otro aquellos visitantes insatisfechos. Finalmente, queda por decidir el algoritmo por el que vamos a realizar la clasificación, para el que elegiremos el conocido **KMeans**. Reunamos por tanto estos datos en una tabla antes de continuar.

| | Valor |
|---|---|
| Datos | [1, 5] |
| Clusters o grupos | 2 |
| Algoritmo | KMeans |

Tabla 7 – Consideraciones del análisis de *clusters*

#### 4.4.1.1 KMeans

El algoritmo KMeans es un método de agrupamiento que busca clasificar los datos en K *clusters* o grupos, mediante una metodología sencilla basada en asignar a cada individuo la pertenencia al grupo que se encuentra más cerca de él. Esta cercanía se calcula como la distancia del inviduo, al valor medio de cada grupo, y de esta manera se elige la distancia más corta para asignar el grupo adecuado. Sin embargo, la distancia no tiene por qué ser euclídea, sino que puede definirse empleando otra definición, como puede ser una distancia de Hamming, etc. Veamos como sería el funcionamiento de un algoritmo de KMeans:

1. Inicializariamos un vector de longitud K, con las medias de cada grupo, eligiendo aleatoriamente dichos elementos sobre el grupo.
2. Se calcula la distancia de cada elemento a los grupos, y se le asigna la pertenencia al más cercano.
3. Se recalculan las medias de los grupos, ahora que cuentan con nuevos miembros.
4. Repetimos los pasos 2 y 3.
5. Si se llegase al paso 3, y no no hubiese cambios en los miembros de ningún grupo, se ha alcanzado una solución estable.

Como detalle, cabe destacar que esta solución puede ser un mínimo local y no absoluto: esto puede solucionarse repitiendo el proceso con un nuevo punto de partida, de manera que sea posible alcanzar un resultado distinto. Plasmemos este funcionamiento en un diagrama de flujo.

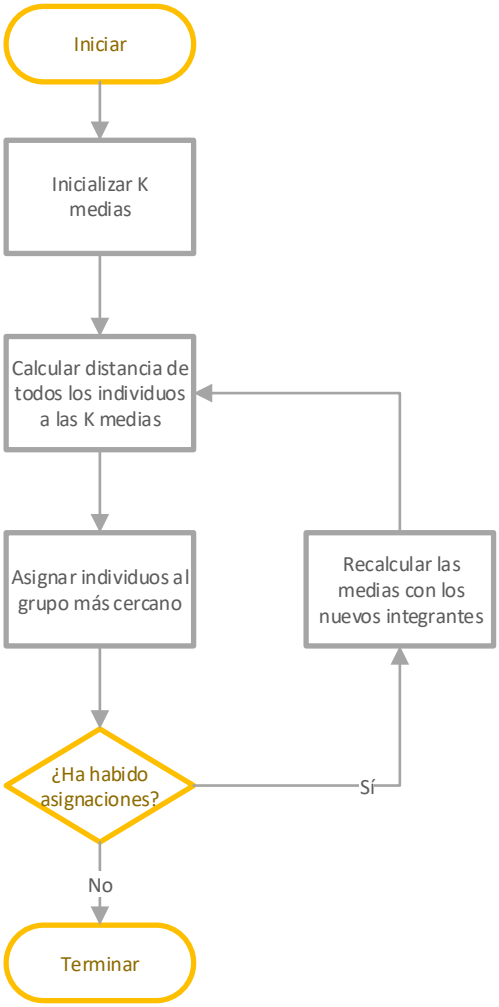


Ilustración 21 – Diagrama de flujo - KMeans

### 4.4.2 Análisis de regresión

En el análisis de regresión, buscaremos relacionar la variable temporal, medida con una precisión de horas, con el número de visitas registradas. De esta manera, esperamos encontrar una relación entre el tiempo y las entradas al recinto, permitiéndonos detectar así las horas más favorables y desfavorables en cuanto a visitas, y dejando a un analista posterior la labor de determinar el motivo de la relación que pudiese establecerse.

Para realizar este análisis, no necesitaremos emplear ningún algoritmo ni metodología especial, por lo que nos limitaremos a realizar conteos en las diferentes horas durante el tiempo que se ejecute la prueba, para posteriormente representar el resultado obtenido gráficamente, basándonos en las librerías *Numpy*, *Pandas* y *Matplotlib*.

### 4.4.3 Análisis de opinión o sentimiento

Clasificar la opinión de los visitantes, nos basamos exclusivamente en aquella que estos han expresado a través de las redes sociales, y la analizaremos con una variación del conocido TF o term frequency (frecuencia de términos). Esta técnica se basa en la frecuencia de aparición de un término en un texto, para saber cómo de importante es dentro del mismo [34].

Debido a que las opiniones de los usuarios son textos de una longitud extremadamente corta comparado con los conjuntos de datos a los que habitualmente se aplica la técnica de los TF, elaboraremos una implementación personalizada y simplificada, que además no se centrará en la importancia de cada término para el texto, sino que **agrupará los términos por signficado**, y **buscará el grupo de términos más**

**importante**. De esta forma, tendremos un grupo de términos que denoten positividad en la opinión del visitante y viceversa. Estos términos se buscarán en las opiniones recopiladas, y aquel grupo de términos más presente en la opinión, hará que la opinión sea clasificada como perteneciente a él, pudiendo ser: **positiva**, **negativa**, o **neutra** en caso de que el número de términos presentes en ambos grupos sea el mismo.

La elección de estos términos debe ser cuidadosa: el uso de términos débilmente positivos o negativos es poco indicativo para clasificar una opinión de una forma u otra, y estos términos permiten ser acompañados de una negación previa que puede dar la vuelta al significado del término. De esta forma, puede ser posible encontrar dos variantes en significado como son:

*La exposición es **buena** porque (…)*

*La exposición **no** es **buena** porque (…)*

Para solucionar este problema:

- Buscaremos centrarnos en términos más extremos o superlativos de manera que este uso sea menos probable, debido a que la lengua española cuente con un antónimo cuyo uso sea más generalizado que la negación

*La exposición es **excelente** debido a (…)*

*La exposición es **horrible** debido a (…)*

- De igual forma, el uso de adverbios que cuenten con un antónimo:

*La exposición está **excelentemente** organizada ya que (…)*

*La exposición está **horriblemente** organizada ya que (…)*

- Emplearemos adjetivos que implícitamente impliquen negatividad o positividad, de manera que su uso para expresar lo contrario sea poco habitual y de esta manera, asumible el error de emplearlos para clasificar:

*(…) el precio de la entrada es **disparatado** ya que (…)*

*(…) muy **divertido** para los niños (…)*

### 4.4.4 Análisis de serie de tiempos

Finalmente, este análisis es el más amplio en cuanto a cálculos de diversa índole ya que, al basarse en el empleo de datos ordenados sobre un eje temporal, estos entran en este tipo de análisis, para dar como resultado, entre otros, los diferentes KPI que antes definimos.

Como en el caso de el análisis de regresión, estos cálculos se realizan empleando las diferentes librerías antes comentadas, sin requerir algoritmos especiales para ser obtenidos.

## 4.5 Persistencia de resultados

Una vez realizados los análisis indicados, será necesario almacenarlos para asegurar su persistencia, y permitir así su uso posterior por un analista en caso de no ser posible o necesario su uso inmediato una vez calculados. Para ello, emplearemos la misma base de datos empleada para la persistencia de datos procedentes de Orion, pero crearemos tablas especiales para almacenar los resultados de los diferentes análisis. Estas tablas estarán definidas como se puede ver en las siguientes tablas, donde el nombre de las tablas corresponde al nombre de cada análisis, con una “A” inicial para especificar que la tabla corresponde a los resultados de un análisis.

| Campo | Tipo |
|---|---|
| MD5 | VARCHAR(100) |
| Clasificacion | INT |

Tabla 8 – Definición de la tabla ACluster

| Campo | Tipo |
|---|---|
| Titulo | VARCHAR(20) |
| Valor | FLOAT |

Tabla 9 – Definición de la tabla ATiempos

| Campo | Tipo |
|---|---|
| MD5 | VARCHAR(100) |
| Clasificacion | VARCHAR(10) |

Tabla 10 – Definición de la tabla AOpinion

| Campo | Tipo |
|---|---|
| Hora | VARCHAR(10) |
| Entradas | INT |

Tabla 11 – Definición de la tabla ARegresion

El empleo de campos del tipo VARCHAR, concretamente en la primera columna de todas las tablas incluyendo a la tabla ARegresion, pese a que aparentemente sería más adecuado un tipo numérico o de hora/fecha, se debe a la intención de facilitar la implementación de una función común y estándar para su uso al trabajar con todas estas tablas.

# 5 IMPLEMENTACIÓN

*El auténtico genio consiste en la capacidad para evaluar información incierta, aleatoria y contradictoria.*

Winston Churchill

Una vez visto el diseño del sistema y como se analizarán los datos, pasaremos a discutir a continuación los detalles más técnicos y concretos de la implementación. Para hacer esto, separaremos la implementación en fases, correspondientes con las diferentes secciones de trabajo con los datos. Así, tendremos:

1. Fase I de producción de datos: comprende desde la **generación de datos** por parte de los sensores empleados en nuestra solución hardware, hasta **su notificación al sistema FIWARE** a través del intermediario Orion.
2. Fase II de recolección de datos internos: comprende la tarea de recolección de todos los datos que son registrados en Orion, y su **almacenamiento en la base de datos** elegida (MySQL) para su acceso estandarizado por parte del programa de análisis de los datos.
3. Fase III de recolección de datos externos: comprende la tarea **de recolección de los datos producidos en Facebook**, es decir todas las opiniones de los usuarios junto con su valoración numérica, y su almacenamiento en la base de datos para su acceso estandarizado por parte del programa de análisis de los datos.
4. Fase IV de análisis de los datos: comprende la tarea de análisis de los datos disponibles en la base de datos, mediante las técnicas elegidas en el punto 4 anterior, mediante el uso del marco de trabajo SciPy y en especial la librería Pandas.

Sin embargo, antes de proceder con estas fases, debemos explicar las diferencias entre la implementación lógica y la implementación física o real llevada a cabo.

## 5.1 Escenario real y escenario lógico

Dado que vamos a realizar una implementación del sistema diseñado, comentaremos a continuación cuales han sido sus componentes, los diferentes programas implementados y sobre qué equipos y/o tecnologías han sido desplegados.

Partiendo de la base de la ilustración 16, en la que veíamos el escenario completo diseñado, mostraremos ahora la implementación completa que se ha realizado, atendiendo a los equipos utilizados, y después la representaremos mediante un diagrama similar a la de dicha ilustración, y un diagrama de capas:

- Sensores: se conectan mediante el uso de *pines* a la placa Arduino.
- Transceptores RF 2.4GHz: se conectan mediante *pines* a las placas Arduino UNO y Raspberry Pi 3.
- Arduino UNO: trabaja con los sensores para recabar los datos de temperatura, humedad, entradas o salidas de la estancia en la que controlan el acceso y salida, y transmite los datos y eventos percibidos mediante el transceptor que tiene conectado.

- Raspberry Pi 3: implementa dos programas para la recepción de datos y su introducción en la plataforma FIWARE:
    - Receptor RF: programado en C++, debido a la disponibilidad única de la librería empleada para el uso del transceptor en este lenguaje: RF24 [35].
    - Concentrador: programado en Python, debido a la facilidad para realizar peticiones REST para comunicarse con Orion, se encarga de transmitir los eventos y datos recabados por el receptor RF a la plataforma FIWARE.
- Servidores de Facebook: a los que nos conectaremos para recabar los datos de la red social. Están fuera de nuestro control.
- Máquina host 1: máquina que ejecuta el sistema operativo Windows 10 como sistema principal, sobre el que ejecutaremos, con Hyper-V como hipervisor:
    - Máquina virtual de 64 bits con Linux Mint, sobre la que instalaremos Docker. Sobre ella se ejecutarán:
        - Contenedores: ejecutando
            - Por un lado, la plataforma FIWARE, compuesta por Orion y MongoDB
            - Por otro lado, la base de datos MySQL donde se recolectan los datos para su análisis.
        - Las herramientas de recolección de datos:
            - Recolector de datos de Orion
            - Recolector de datos de Facebook
- Máquina host 2: máquina que ejecuta el sistema operativo Windows 10 como sistema principal, sobre la que ejecutaremos, con VMWare Workstation Player como hipervisor:
    - Máquina virtual de 32 bits con Linux Mint, sobre el que ejecutaremos:
        - Programa analizador de datos

Todos estos elementos, se encuentran conectados como refleja la siguiente ilustración:

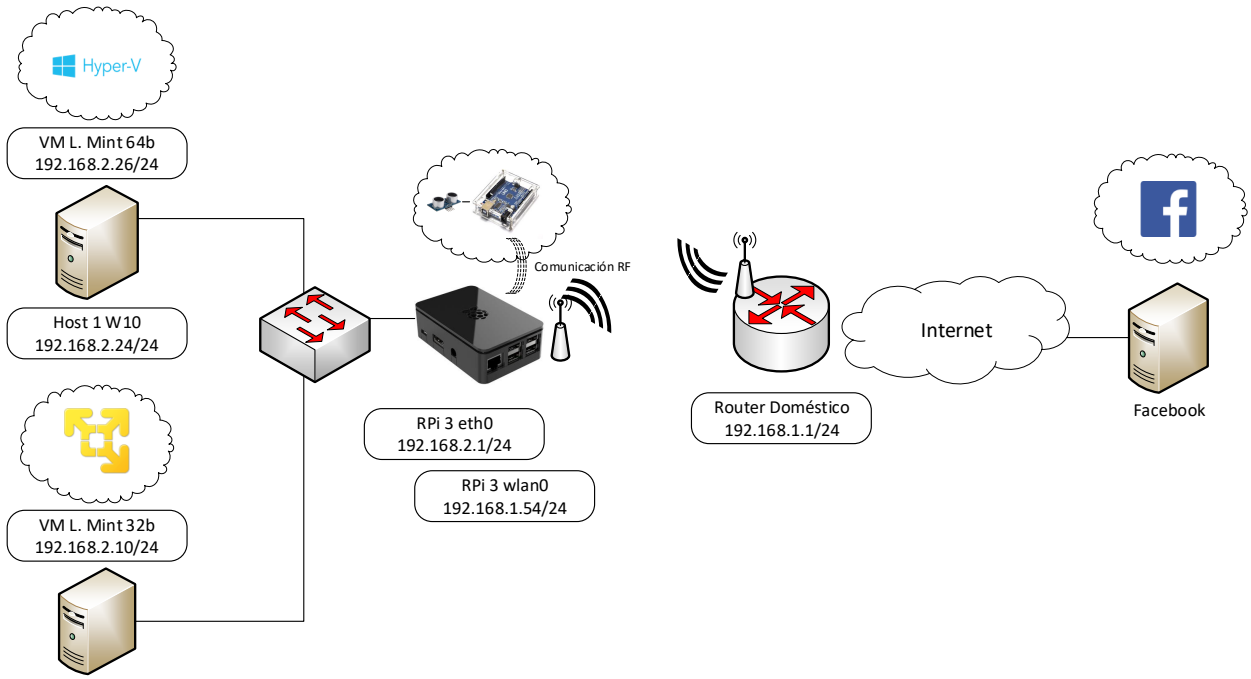


Ilustración 22 – Escenario real de la implementación

Como se puede apreciar, la Raspberry Pi 3, además de **ejecutar sus programas** como parte del sistema, también **hace de encaminador** para los dos equipos empleados en el proyecto.

Si atendemos a la repartición de tareas que se hace sobre cada equipo, podemos representar el siguiente diagrama:

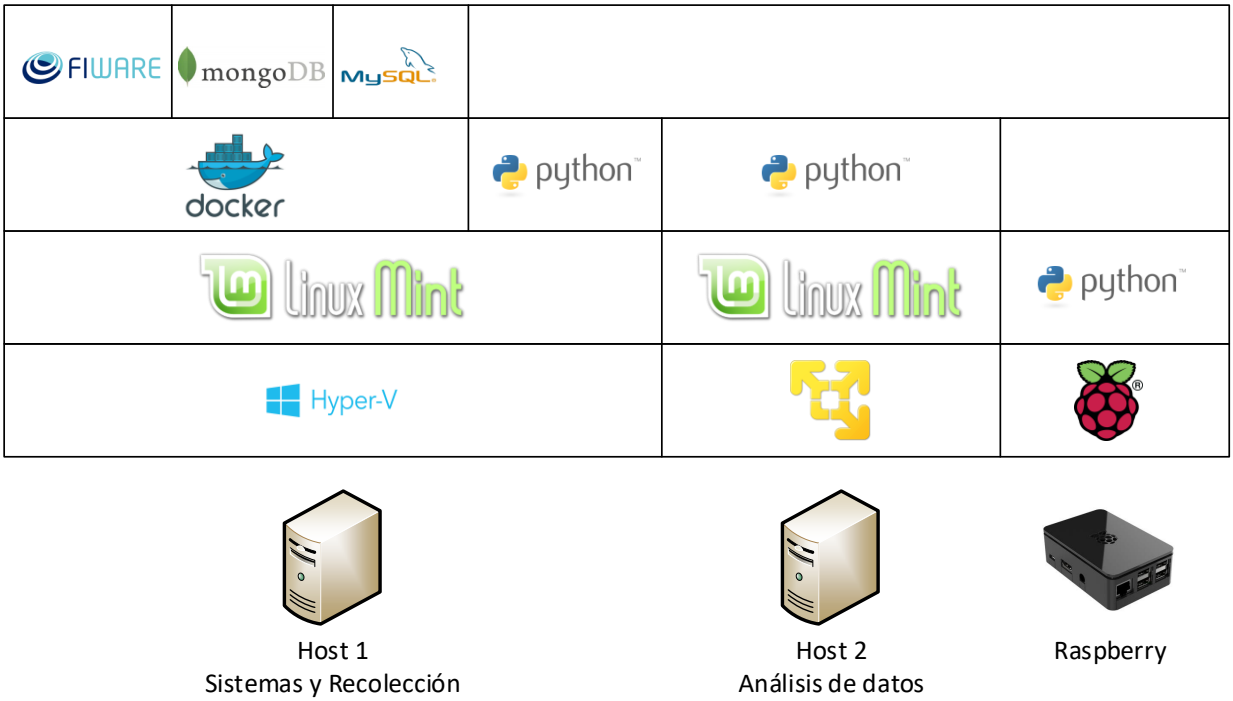


Ilustración 23 – Repartición de tareas por equipos. Pila de software.

Para finalizar, especificaremos las características de estos equipos y máquinas virtuales, con vistas a esclarecer al lector la capacidad necesaria requerida para replicar el sistema en caso de que fuese necesario. Para ello, estas características se encuentran recogidas en la siguiente tabla:

| Equipo | Sistema Operativo | Características |
|---|---|---|
| Host 1 | Windows 10 | CPU: 4 núcleos, 3.8GHz<br>RAM: 8GB |
| VM Linux Mint 64b | Linux Mint 64b | CPU: 1 núcleo, 3.8GHz<br>RAM: 4GB |
| Host 2 | Windows 10 | CPU: 4 núcleos, 3.4GHz<br>RAM: 8GB |
| VM Linux Mint 32b | Linux Mint 32b | CPU: 2 núcleos, 3.4GHz<br>RAM: 2GB |
| Raspberry Pi 3 | Raspbian | CPU: 4 núcleos, 1.2GHz<br>RAM: 1GB |

Tabla 12 – Características hardware de los equipos empleados

Debe tenerse en cuenta, de que estos valores son representativos de los sistemas utilizados en la implementación realizada, asegurando un correcto funcionamiento del sistema gracias a ellos. Valores superiores siempre seguirían siendo válidos, mientras que valores inferiores también podrían ser posibles en ciertos casos, por los que se recomienda realizar pruebas de funcionamiento con menos recursos en caso de duda para verificar su viabilidad.

A continuación, detallaremos la implementación realizada atendiendo a la división de fases antes realizada.

## 5.2 Fase I: Producción de Datos

Comenzaremos la fase de producción de datos desde los más básicos que se pueden generar: los correspondientes a los sensores. Buscamos emplear dos sensores de proximidad mediante ultrasonido, y un sensor de temperatura y humedad para complementarlos, teniendo así otros datos del escenario. Por tanto, emplearemos:

- Sensor HC-SR04 de proximidad mediante ultrasonido (x2): este módulo cuenta con una precisión de hasta 0.3cm (aunque en nuestra aplicación no será necesaria tal precisión), y es capaz de diferenciar distancias desde 2cm hasta 450cm. Es un dispositivo digital que consta de cuatro pines: alimentación, tierra, un pin de disparo para realizar la lectura de la proximidad, y un pin de eco por donde se transmite la respuesta.
- Sensor DHT11 de temperatura y humedad: se trata de un dispositivo capaz de reconocer temperaturas en un rango entre los 0º y los 60º centígrados, con una precisión de ±1º, a la vez que es capaz de situar la humedad en el ambiente en un valor entre el 10% y el 90%. Es un sensor digital, y consta de tres pines: alimentación, tierra y un pin para la comunicación de los datos.
- Transceptor NRF24L01: se tratra de un transceptor que trabaja en la banda de los 2.4GHz, concretamente en el rango 2.400GHz~2.525GHz.

### 5.2.1 Arduino

Comenzaremos por la placa Arduino, viendo como conectar los diferentes componentes, para lo que elaboraremos en primer lugar unas tablas indicativas para conectar los distintos elementos con Arduino. A continuación, presentaremos el esquema, donde cabe destacar que el sensor DHT11 del que disponemos en nuestra implementación es de 3 pines frente a los 4 que aparecen en el diagrama. Esta diferencia es trivial, pues en el modelo de 4 pines, dos de ellos van a tierra, por lo que nosotros obviaremos la conexión de uno de estos terminales en el esquema.

| NRF24L01 | Arduino |
|---|---|
| 1 – GND | GND |
| 2 – VCC | 3.3V |
| 3 – CE | 9 |
| 4 – CSN | 10 |
| 5 – SCK | 13 |
| 6 – MOSI | 11 |
| 7 - MISO | 12 |
| 8 - IRQ | - |

Tabla 13 – Conexiones NRF24L01 – Arduino UNO

Como puede apreciarse, el pin 8 no será necesario conectarlo ya que no vamos a emplear las interrupciones que soporta el dispostivo.

| HC-SR04 | Arduino |
|---|---|
| GND | GND |
| VCC | 5V |
| TRIG | 3 |
| ECHO | 2 |

Tabla 14 – Conexiones HC-SR04 (1) – Arduino UNO

| HC-SR04 | Arduino |
|---|---|
| GND | GND |
| VCC | 5V |
| TRIG | 6 |
| ECHO | 5 |

Tabla 15 – Conexiones HC-SR04 (2) – Arduino UNO

| DHT11 | Arduino |
|---|---|
| GND | GND |
| VCC | 5V |
| DATA | 4 |

Tabla 16 – Conexiones DHT11 – Arduino UNO

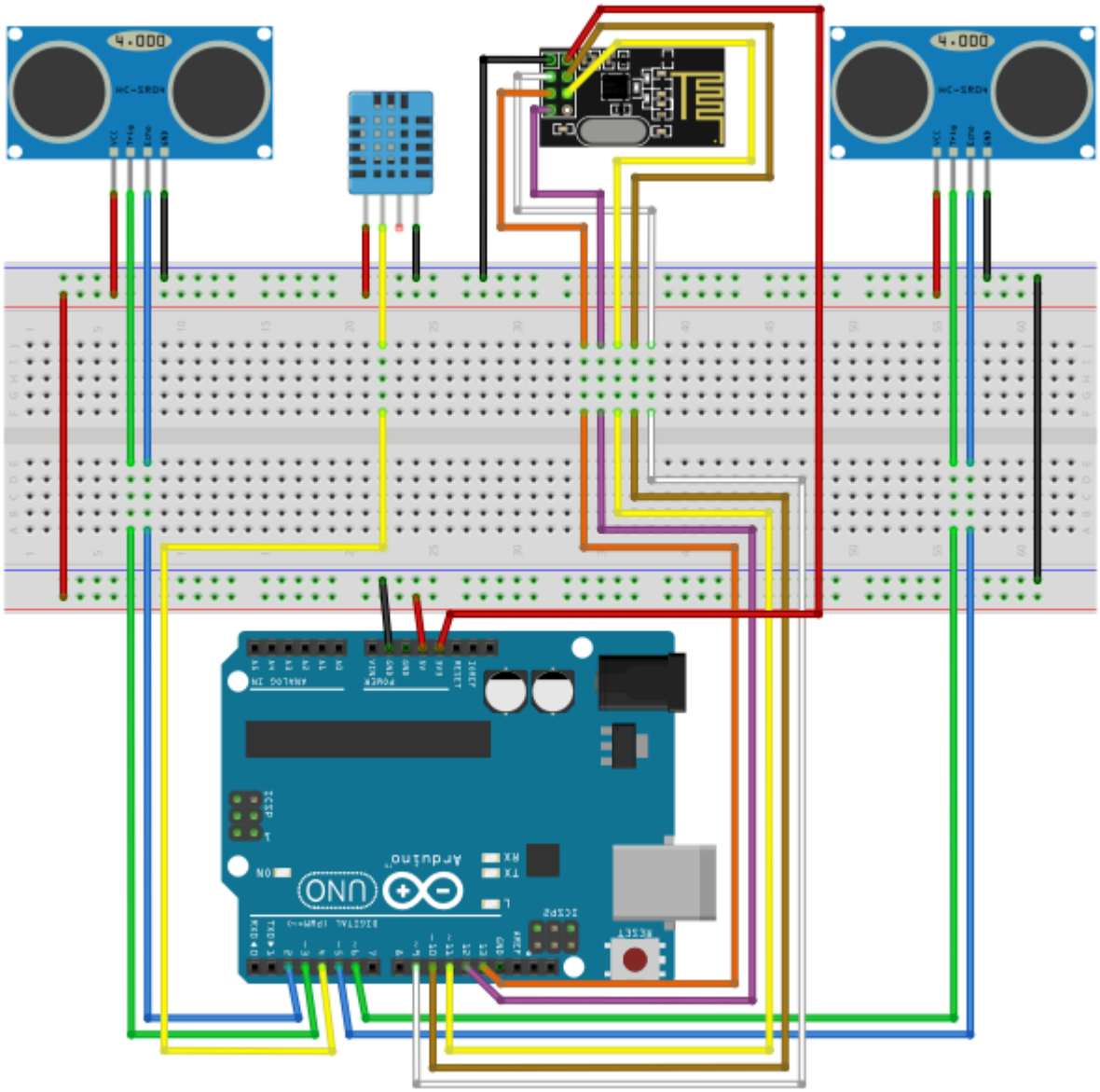

Ilustración 24 – Conexiones placa Arduino UNO

En base a este esquema, implementaremos a continuación el programa que regirá el comportamiento de nuestra placa. Debemos tener en cuenta que:

- Debemos leer **temperatura y humedad** con relativa frecuencia, suficiente para que sea informativo y percibamos cambios si algo va mal, y a la vez un tiempo prudencial para que el dato no sea absurdo. Estableceremos un valor conservador de 10 segundos entre lecturas de estos valores.
- Debemos detectar los **pasos por los sensores de proximidad** de manera instantánea, para acto seguido determinar si se ha pasado por ambos, y ver qué sensor ha sido el primero y cual el segundo en ser superados, de manera que podamos determinar si el cruce de la persona ha sido una **entrada o una salida**. Para ello, diseñaremos un diagrama de estados que nos permita identificar estas situaciones:

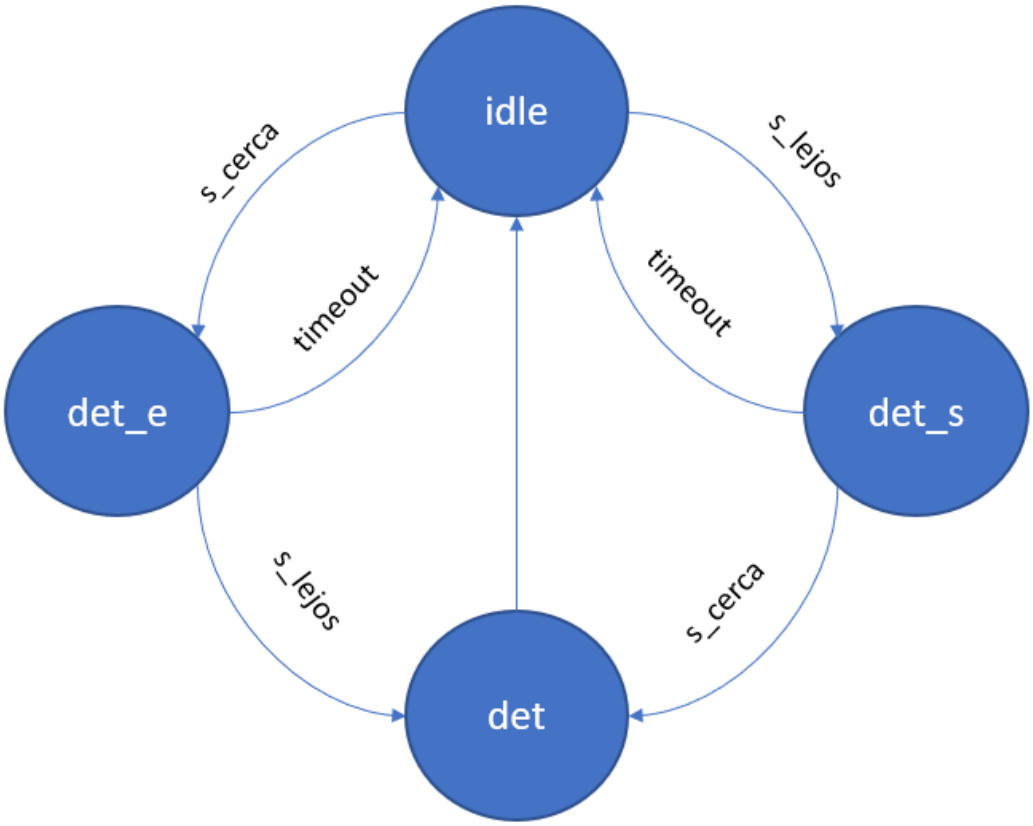


Ilustración 25 – Diagrama de estados implementado en Arduino UNO

Como podemos comprobar, en función de que sensor se activa primero, estaremos detectando una entrada o una salida. Una vez se activa un sensor, tendremos un tiempo para que se active el sensor contrario, de manera que se complete la detección. Si no se activase el otro sensor en ese tiempo previsto, se produciría un *timeout*, de manera que volveríamos al estado de reposo. Este *timeout* ha sido **calculado de manera experimental**, ha sido denominado en el programa como *CROSS_TIME*, y se ha establecido en un segundo, ya que es un tiempo suficiente para que la persona pueda cruzar por delante de los sensores.

Además, independientemente de esta máquina de estados, llevaremos una cuenta temporal para que cada 10 segundos, se produzca una lectura del sensor DHT11 para obtener la temperatura y la humedad. Esto podría ser implementado con una máquina de estados de dos estados, pero al poder controlarse mediante una única variable y debido a su simplicidad, lo mantendremos de esta forma en lugar de implementar dichos estados.

Finalmente, debemos hablar de la transmisión radio que se realiza. Debemos tener en cuenta:

- **El canal de transmisión**: debe coincidir con el valor usado en el otro extremo, para que la frecuencia de trabajo coincida en ambos transceptores y puedan comunicarse entre sí.
- **La tasa de transmisión**: debe coincidir igualmente para permitir la comunicación.

- Las tuberías, es decir, los **identificadores de las entidades en la comunicación** deben coincidir el identificador del nodo destino en Arduino (transmisor), con el identificador de nodo receptor en este programa (Raspberry).

Mientras que el envío de los datos se realizará siguiendo el formato:

**Formato de envío**

```
valor_humedad valor_temperatura
0 valor_evento
```

De esta manera, si el primer valor es un 0, automáticamente entenderemos en el receptor que el segundo valor es un evento, mientras que si el valor es distinto de cero significará que la pareja de valores son humedad y temperatura. Cabe recordar, que el sensor DHT11 solo es capaz de medir la humedad entre un 10% y un 90%, por lo que nunca producirá un valor de 0% que pudiera ser confundido con el 0 que identifica al evento. Así, los eventos irán identificados por los valores 1 y 2 tal que:

| Evento | Código numérico |
|---|---|
| Entrada | 1 |
| Salida | 2 |

Tabla 17 – Códigos de evento

Por último, una foto real del escenario montado puede verse en las siguientes ilustraciones:

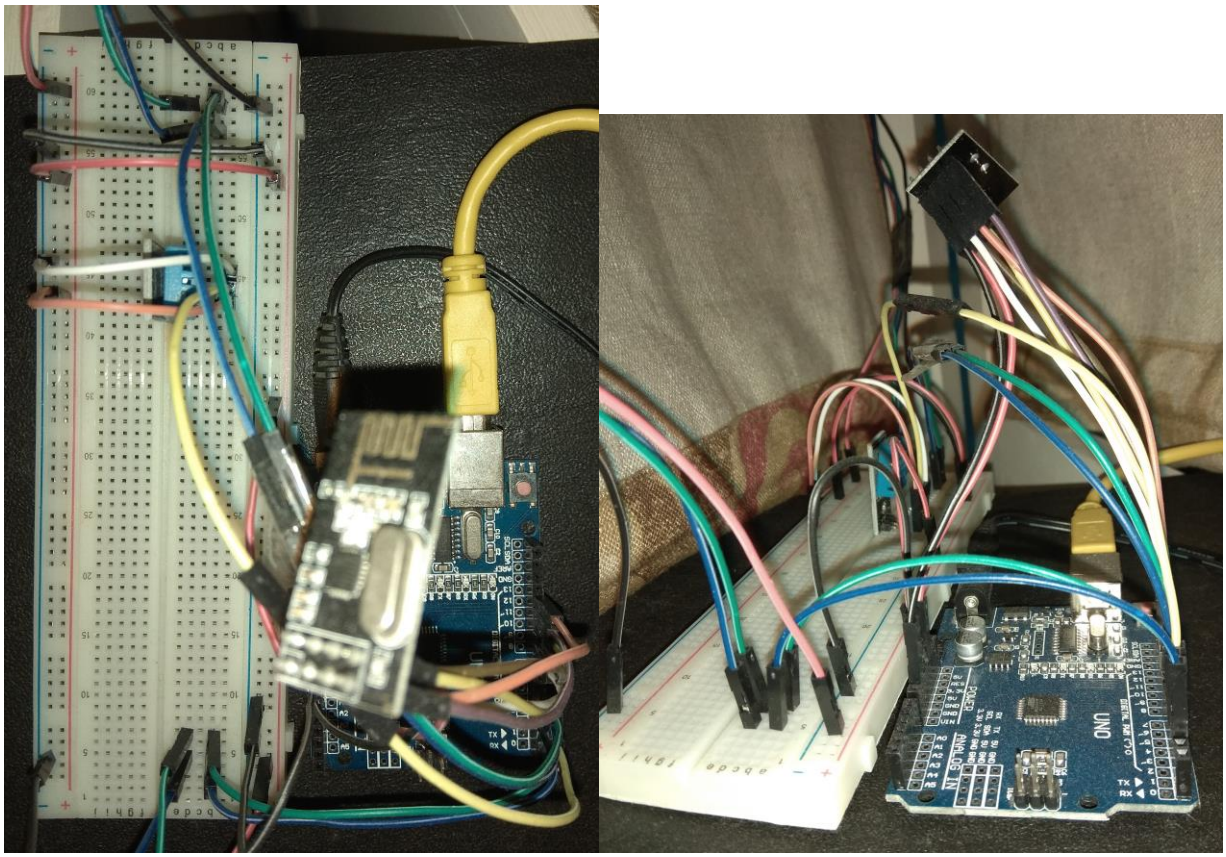

Ilustración 26 – Vista aérea y lateral del despliegue del sistema Arduino (placas)


Ilustración 27 – Sensor de proximidad colocado transversalmente a la zona de paso, verticalmente

### 5.2.2 Raspberry Pi 3

Una vez finalizada la implementación de la placa Arduino, veamos la implementación realizada en la Raspberry Pi 3. El transceptor NRF24L01, se conectará a la placa como ilustra el siguiente esquema, en función de la tabla que se muestra a continuación:

| NRF24L01 | RPi 3 GPIO | RPi 3 SPI |
|---|---|---|
| 1 – GND | 9GND | 9 |
| 2 – VCC | 3.3V | 1 |
| 3 – CE | GPIO 22 | 15 |
| 4 – CSN | GPIO 8 | SPI_CEO 24 |
| 5 – SCK | GPIO 11 | SPI_SCLK 23 |
| 6 – MOSI | GPIO 10 | SPI_MOSI 19 |
| 7 - MISO | GPIO 9 | SPI_MISO 21 |
| 8 - IRQ | - | - |

Tabla 18 – Conexiones NRF24L01 – Raspberry Pi 3

Como puede apreciarse, el pin 8 no será necesario conectarlo ya que no vamos a emplear las interrupciones que soporta el dispostivo.

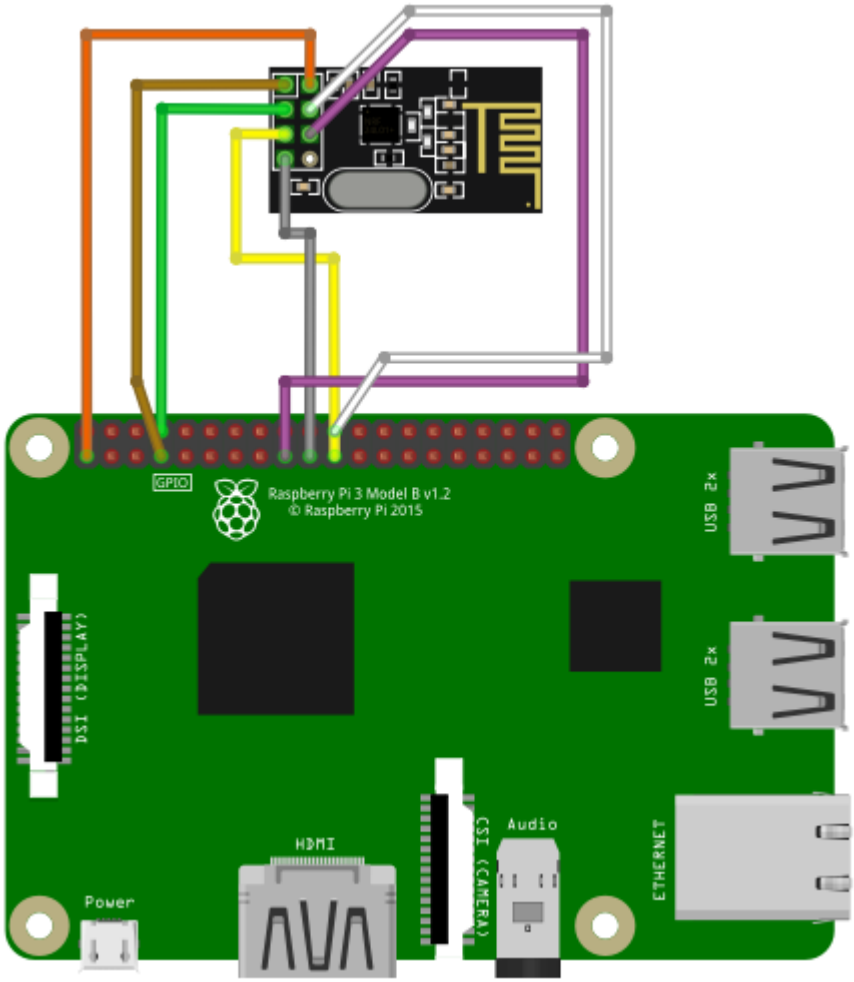


Ilustración 28 – Conexiones placa Raspberry Pi 3

Procederemos a implementar los programas necesarios para llevar los datos desde los sensores hasta Orion. Del punto anterior, partimos disponiendo de una transmisión empleando el transceptor RF24 en 2.4GHz, donde se transmiten en cada ocasión dos números, que según sus valores estarán asociados a la pareja temperatura-humedad, o serán indicativos de un evento bien de entrada o de salida.

Para recibir estos valores, y debido al uso de la librería *RF24*, el programa de recepción de estos valores deberá emplear la misma librería para ser capaz de decodificar los mensajes al seguir el mismo formato de transmisión. Por ello, este programa estará realizado en el **lenguaje C++**, único lenguaje para el que está disponible la librería.

Como parámetros a configurar para la recepción de estas comunicaciones, deberemos establecer:

- **El canal de transmisión**: debe coincidir con el valor usado en el otro extremo, para que la frecuencia de trabajo coincida en ambos transceptores y puedan comunicarse entre sí.
- **La tasa de transmisión**: debe coincidir igualmente para permitir la comunicación.
- Las tuberías, es decir, los **identificadores de las entidades en la comunicación** deben coincidir el identificador del nodo destino en Arduino (transmisor), con el identificador de nodo receptor en este programa (Raspberry).

Este programa imprimirá, por la **salida estándar**, los datos en un formato similar al que los recibe en la comunicación con Arduino, de forma que deberemos redirigir esta salida como **entrada estándar** del programa concentrador o *hub*, que reconocerá los datos de igual manera. Para ello, ejecutaremos los programas en *bash* o terminal redirigiendo la salida tal que:

**Código de ejecución de los programas con uso de tubería**

```
./programa_1 | ./programa_2
```

Deberemos realizar el programa que se encargará de realizar las peticiones REST para insertar los valores en la plataforma FIWARE. Este programa estará realizado en Python, debido a la facilidad de codificación de estas tareas en este lenguaje frente a C++, y tomará los datos mediante su entrada estándar, que se comunicará directamente con la salida estándar del receptor en C++ para tomar sus valores. En cuanto a la estructura, la aplicación principal se encuentra en el fichero *Hub.py*, que importa a su vez el fichero *CManager.py* con la clase del mismo nombre en su interior. Esta clase implementa los métodos para conectarse con Orion y realizar las peticiones REST que necesita el programa principal para realizar su tarea.

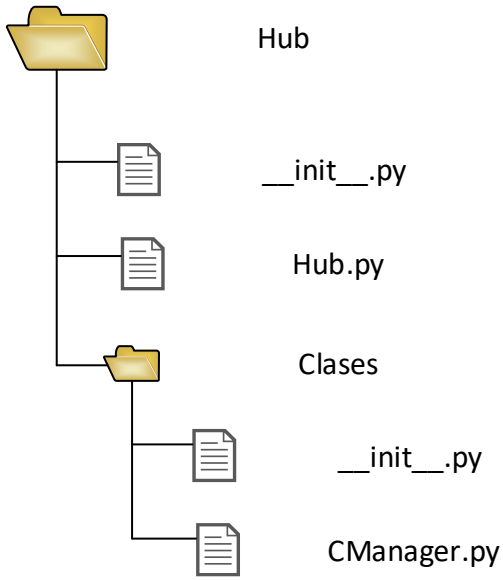


Ilustración 29 - Estructura de directorios de la aplicación de transmisión a Orion

Este programa empleará las librerías *json* y *httplib*, de manera que seamos capaces de realizar estas tareas. Tras iniciarse, el programa buscará crear las entidades correspondientes en Orion, si bien es posible que ya existan y por tanto no puedan ser creadas, por ejemplo, si ya se ha ejecutado anteriormente el programa. A partir de este momento se realizarán peticiones PUT que actualicen el valor de dichas entidades con los valores recibidos por la entrada estándar. En caso de que haya algún problema con la comunicación, haremos que nuestro programa intente crear una nueva conexión con Orion y en caso de no ser posible, detenga su bucle de transmisión de datos.

**Código de ejecución del conjunto**

```
./RF24/examples_linux/receiver | python2 ./Hub/Hub.py
```

Finalmente, aquí una ilustración del despliegue del Sistema llevada a cabo:

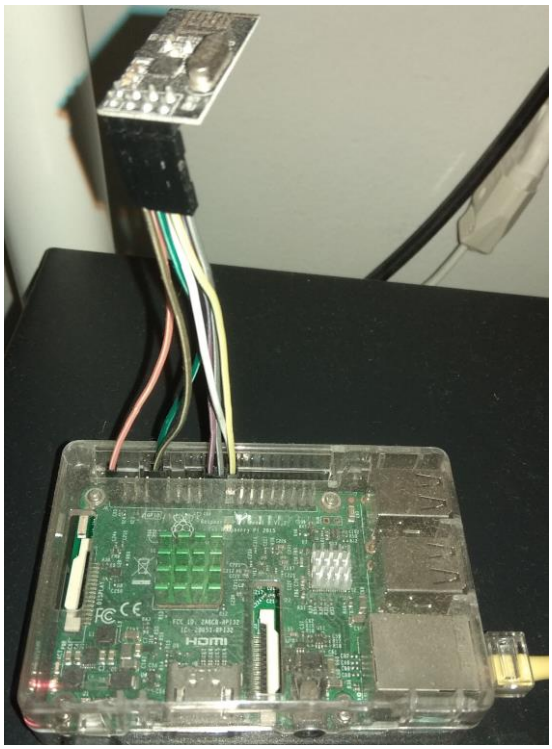

Ilustración 30 – Vista aérea del despliegue del sistema Raspberry Pi

## 5.3 Fase II: Recolección de Datos Internos – FIWARE

La implementación de los componentes FIWARE para la recolección de datos se llevará a cabo empleando el sistema de contenedores Docker. De esta manera, se minimizan los esfuerzos dedicados al despliegue y configuración, a la par que conseguimos disponer de las dependencias necesarias sin necesidad de prepararlas expresamente con anterioridad.

Para emplear esta plataforma, hay que tener en cuenta que Docker funciona sobre sistemas de 64 bits, y aunque dispone de una implementación para Windows, es un sistema más maduro en Linux. Por ello, emplearemos una máquina virtual de 64 bits, con sistema operativo Linux Mint de 64 bits para instalar en ella Docker. Una vez hecho esto, emplearemos la herramienta *docker-compose* para desplegar un servicio multicontenedor, con:

Un primer contenedor, de *MongoDB*, que sirve para almacenar el contexto de las entidades que maneja Orion.

Un contendor de *ORION Context Broker*, el elemento mediador en el sistema FIWARE. Este contenedor se conecta con el primero indicándolo mediante un parámetro en la sección *links*, a la vez que se añade un comando para realizar la conexión. A su vez, será necesario vincular el puerto empleado por Orion dentro del contenedor con el puerto de la máquina host en el que se desea que esté accesible la conexión con Orion (si fuese el caso). Como nosotros deseamos que Orion sea accesible para entidades y programas que no están en su mismo contenedor/entorno, vincularemos el mismo puerto dentro y fuera de Docker.

En conjunto, el código para generar nuestro contenedor será escrito en un fichero llamado *docker-compose.yml*, y se ejecutará el comando d*ocker-compose* en el directorio donde se encuentre dicho fichero para poner en funcionamiento este contenedor. La ejecución y comprobación de funcionamiento de este código puede verse en el correspondiente apartado del punto 6, dedicado a pruebas de funcionamiento.

**docker-compose.yml**

```
mongo:
    image: mongo:3.2
    command: --nojournal
```

```
Orion:
    image: FIWARE/Orion
    links:
        - mongo
    ports:
        - "1026:1026"
    command: -dbhost mongo
```

Una vez se han transmitido los datos a Orion, debemos extraerlos y almacenarlos en una base de datos, de manera que el acceso a estos por parte del programa encargado de analizarlos sea un acceso estandarizado. Para ello, desplegaremos una base de datos MySQL en un contenedor Docker [36], empleando de nuevo la herramienta *docker-compose*:

**docker-compose.yml**

```
version: '3.3'

services:
   db:
        image: mysql:5.7
        volumes:
            - mysqldb:/home/btc/mysql
        restart: always
        ports:
            - "3306:3306"
        environment:
            MYSQL_ROOT_PASSWORD: TFMUSAIT
            MYSQL_DATABASE: tfm
            MYSQL_USER: alvmarrod
            MYSQL_PASSWORD: alvmarrodtfm

volumes:
    mysqldb:
```

Elaboraremos ahora el programa recolector de datos de Orion, el eje central de este apartado. Se trata de nuevo de un programa elaborado en Python, debido a que, en esta ocasión, debemos suscribirnos y recibir peticiones REST de tipo PUT, que nos informarán de la actualización de las entidades a las que nos suscribamos. Crearemos una estructura para el programa como la que se ve en la siguiente ilustración:

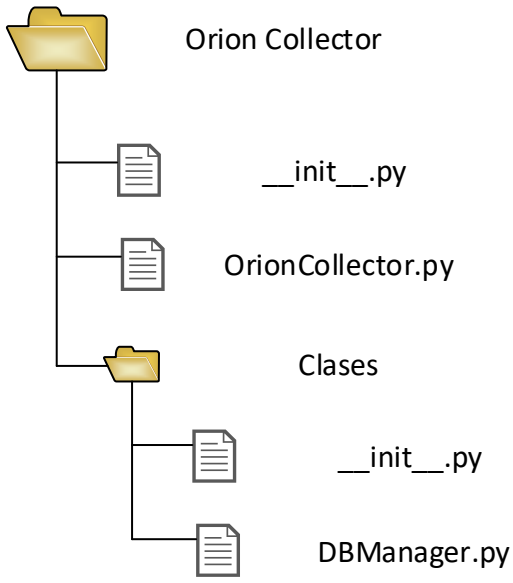


Ilustración 31 – Estructura de directorios de la aplicación de recolección de Orion

La aplicación principal se encuentra en el fichero *OrionCollector.py*, que importa a su vez el fichero *DBManager.py* con la clase del mismo nombre en su interior. Esta clase implementa los métodos para trabajar con la base de datos MySQL que necesita el programa principal para almacenar los datos. Este programa principal se basará en el acumulador proporcionado en el tutorial de la API de Orion disponible en [37], que es un ejemplo ya preparado para procesar las peticiones que recibe, y escribir los datos en un fichero de texto. Aunque esta funcionalidad es parecida a la que nosotros queremos implementar, nuestras modificaciones eliminarán este comportamiento y añadiremos la recogida de los datos en la base de datos.

Debido a la implementación de este programa, su ejecución no será plana, sino que habrá que **ejecutarlo con una serie de parámetros**. Concretamente, el programa podrá recibir los parámetros:

- port: para indicar el puerto que empleará el servidor para escuchar las notificaciones a las que nos hemos suscrito. Es decir, se trata del **puerto al que buscará comunicarse Orion** para enviar dichas notificaciones.
- url: se trata de la url a la que deberán acceder las notificaciones de la suscripción. En caso de que las notificaciones no viniesen a esta url, se procesarían de diferente forma a lo esperado, originando mensajes de error en función de la url a la que se accede o provocando diferentes acciones.
- host: para indicar el host de la base de datos.
- pretty-print: modo de impresión por pantalla ordenado y más legible.
- v: modo detallado de información por pantalla.

Con estos parámetros, el programa original acumulador se ejecutaría tal como indica el siguiente ejemplo, que veremos personalizado posteriormente en el punto 6.

**Ejemplo de ejecución del acumulador del tutorial**

```
./accumulator.py --port 1028 --url /accumulate --host ::1 --pretty-print -v
```

## 5.4 Fase III: Recolección de Datos Externos – Facebook

### 5.4.1 Creación de una página en Facebook para el caso

Necesitaremos disponer de una página en Facebook con la que representar a nuestro museo para su estudio, donde se publiquen las opiniones de los visitantes, y donde deberemos acudir para recabarlas. Para ello, necesitaremos primero una cuenta normal de usuario en la red social, y procederemos de la siguiente manera:

- Buscaremos la opción *crear página*, donde seleccionaremos la tipología *compañía, organización o institución.*

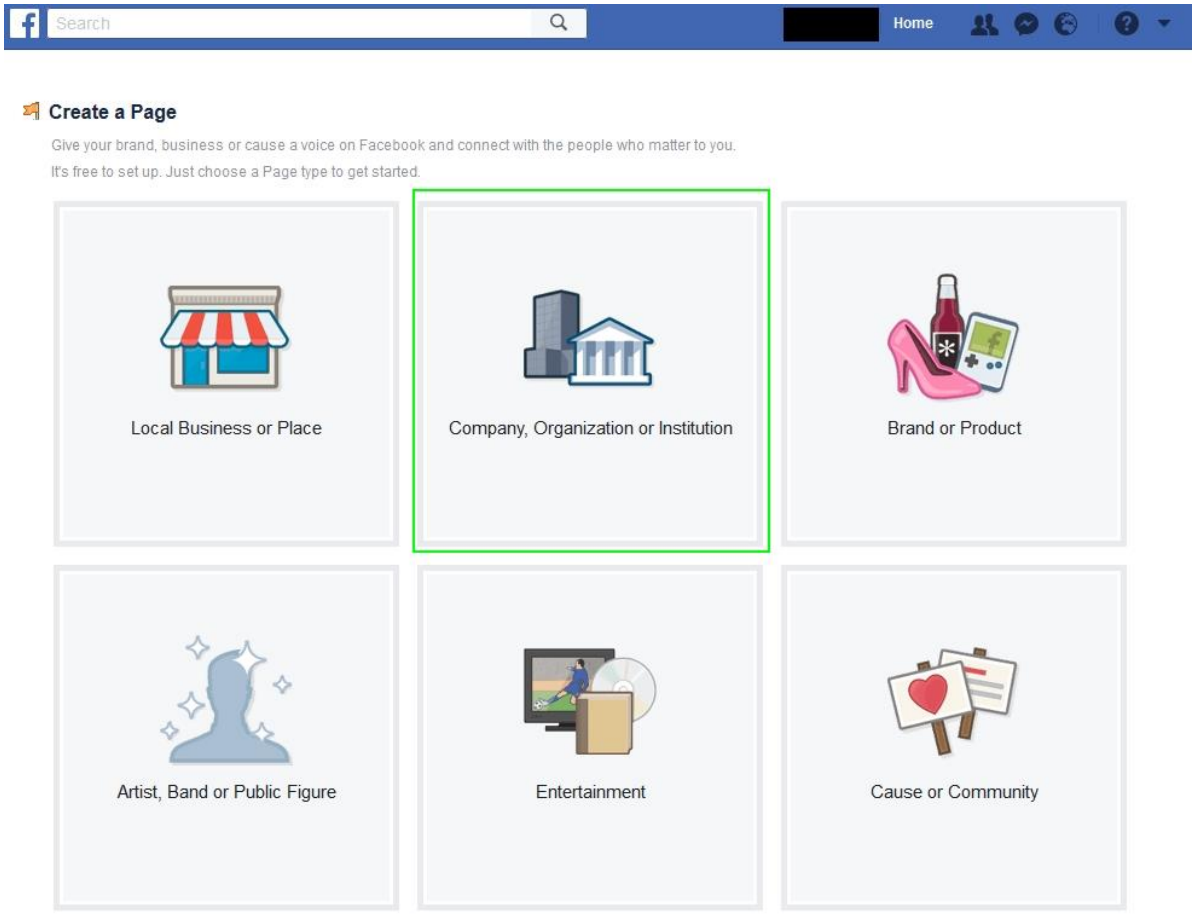


Ilustración 32 – Creación de una página en Facebook paso 1

- Ahora debemos elegir una categoría para la organización, para lo que elegiremos *Universidad*.

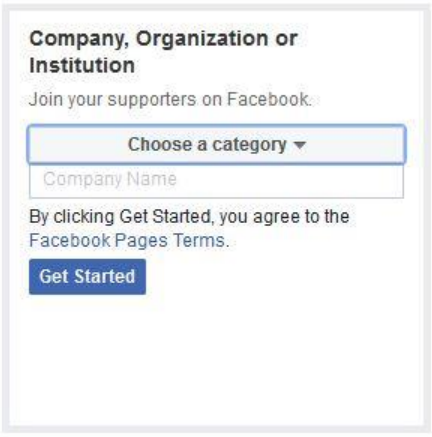


Ilustración 33 - Creación de una página en Facebook paso 2

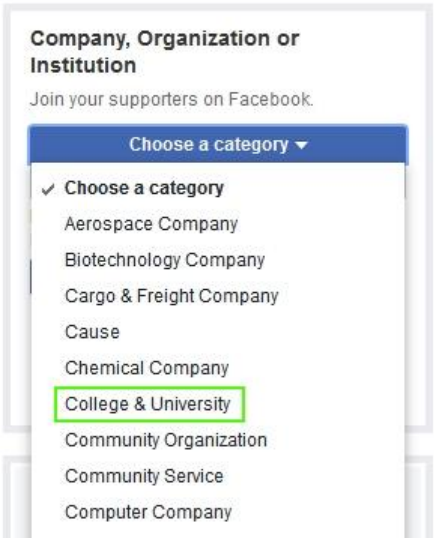


Ilustración 34 – Creación de una página en Facebook: selección de categoría

- Debemos poner un nombre a la página, en nuestro caso emplearemos las siglas del alumno, junto con las siglas del trabajo fin de máster: *TFM AMR*.

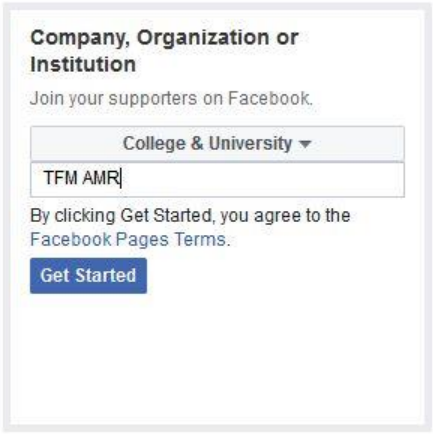


Ilustración 35 - Creación de una página en Facebook: nombre de la organización

- Acompañaremos en el siguiente paso, la creación de la página con una descripción y una imagen que la represente. Para ello, hemos creado una pequeña imagen que incorpore los logos de las 4 tecnologías fundamentales que se emplean en este trabajo: Docker, FIWARE, Raspberry y Python.

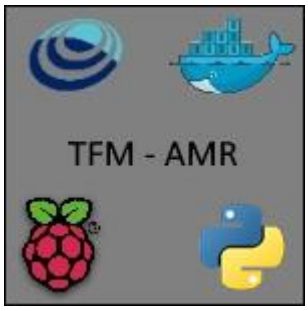


Ilustración 36 - Creación de una página en Facebook paso 3: imagen de la página


Ilustración 37 - Creación de una página en Facebook paso 3: descripción de la página

- Finalmente, deberemos especificar un equivalente a un de nombre de usuario para la página, para lo que volveremos a emplear las siglas antes establecidas. Este nombre de usuario no nos será de ninguna utilidad posteriormente, por lo que el lector puede olvidar este valor sin mayor importancia.

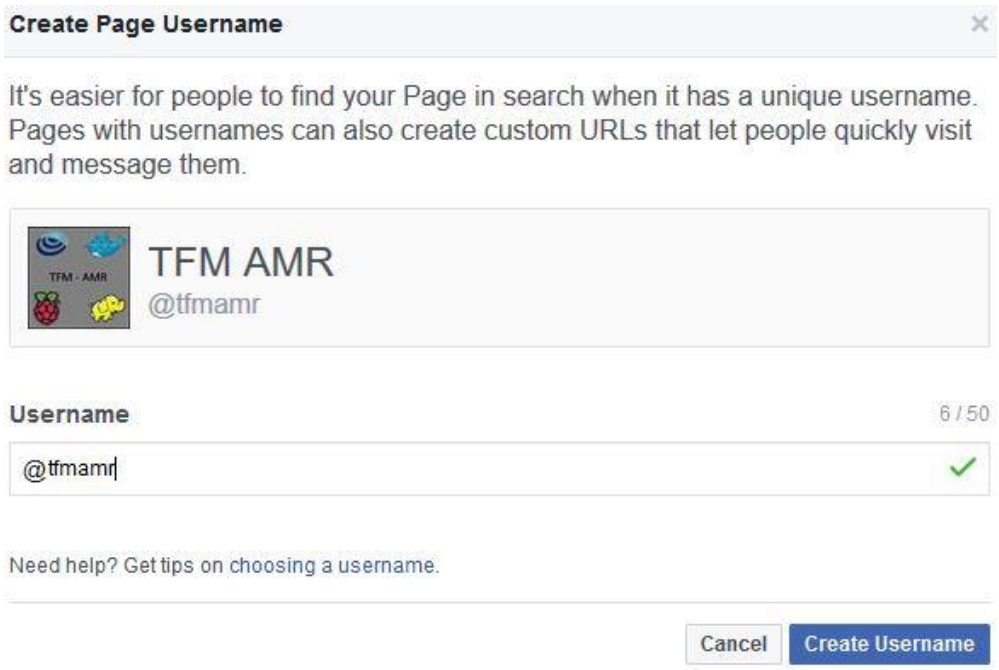


Ilustración 38 - Creación de una página en Facebook paso 4: nombre de usuario

Con esto tenemos nuestra página en Facebook operativa, y veremos en el punto 6.3.1 cómo utilizarla. Sin embargo, si que será importante el identificador numérico de la página, que podremos observar en en apartado *más información*:


Ilustración 39 – Identificador de página

### 5.4.2 Uso de la API – Obtención de un *token*

A la hora de extraer los datos de Facebook, será necesario emplear su API para realizar la lectura de los apartados deseados. Concretamente, nos interesan las opiniones o *reviews* de la página de nuestro "museo". Según la documentación de la API [38], tendremos que hacer una petición web con la siguiente estructura:

**Petición HTTP**

```
GET /v3.0/{page-id}/ratings HTTP/1.1
Al servidor: graph.facebook.com
```

Teniendo en cuenta que nuestra ID es 769750519900195 como vimos anteriormente, la petición quedaría de la siguiente forma:

**Petición HTTP – Caso de uso del proyecto**

```
GET https://graph.facebook.com/v3.0/769750519900195/ratings
```

La respuesta a esta petición vendrá compuesta por dos campos, *data* y *paging*, de la forma:

**Respuesta de la petición HTTP**

```
{
    "data": [],
    "paging": {}
}
```

Como podemos ver, *data* es una lista, de forma que tendremos que analizarla para extraer de ella los datos que queremos obtener.

Sin embargo, estas peticiones no pueden ser realizadas sin un *token* que permita el acceso a la API [39]. De entre los distintos tipos de *tokens* existentes, para acceder a la información de una página necesitaríamos un *Page Access Token*, pero este token no permite ser incrustado en una aplicación de escritorio debido a que este estado no permitiría que el secreto o el *token* fuesen seguros. De esta forma, es necesario usar un *Client Token*, que se obtendrá realizando el procedimiento reflejado en la documentación para desarrolladores de la siguiente referencia: [40].

Brevemente, podemos decir que el proceso pasa por crear y conectarnos a una cuenta de Facebook, a partir de esta cuenta crear una cuenta de desarrollador, y a partir de este punto, crear un *client token* para nuestra aplicación. Sin embargo, al final nos encontramos con que un *client token* no es válido para obtener los datos de la página existente, por lo que necesitamos un *Page Access Token*. A continuación, desarrollaremos el método empleado para conseguirlo, basado principalmente en la respuesta de un usuario de la red StackOverflow a una pregunta sobre como acceder a páginas desde una aplicación nativa [41].

Comenzamos creando una aplicación en el panel de desarrollador, a la que queda asignada un identificador ID que la representa en el entorno de Facebook. Ahora, accediendo al Graph API Explorer [42], elegimos nuestra nueva aplicación para trabajar con su *token*. Pulsamos sobre la opción para obtener el *token* de usuario, y añadimos los permisos de control de página y de eventos: *manage_pages* y *user_events*. Ahora, realizamos una petición HTTP GET sobre la API, con la ruta */me/accounts*. De esta petición, obtenemos una respuesta como se puede apreciar en la siguiente ilustración, que incluye un *Access Token* de nuestra página. A su vez, podemos ver el ID (aunque ya lo teníamos anteriormente) de la página en cuestión.

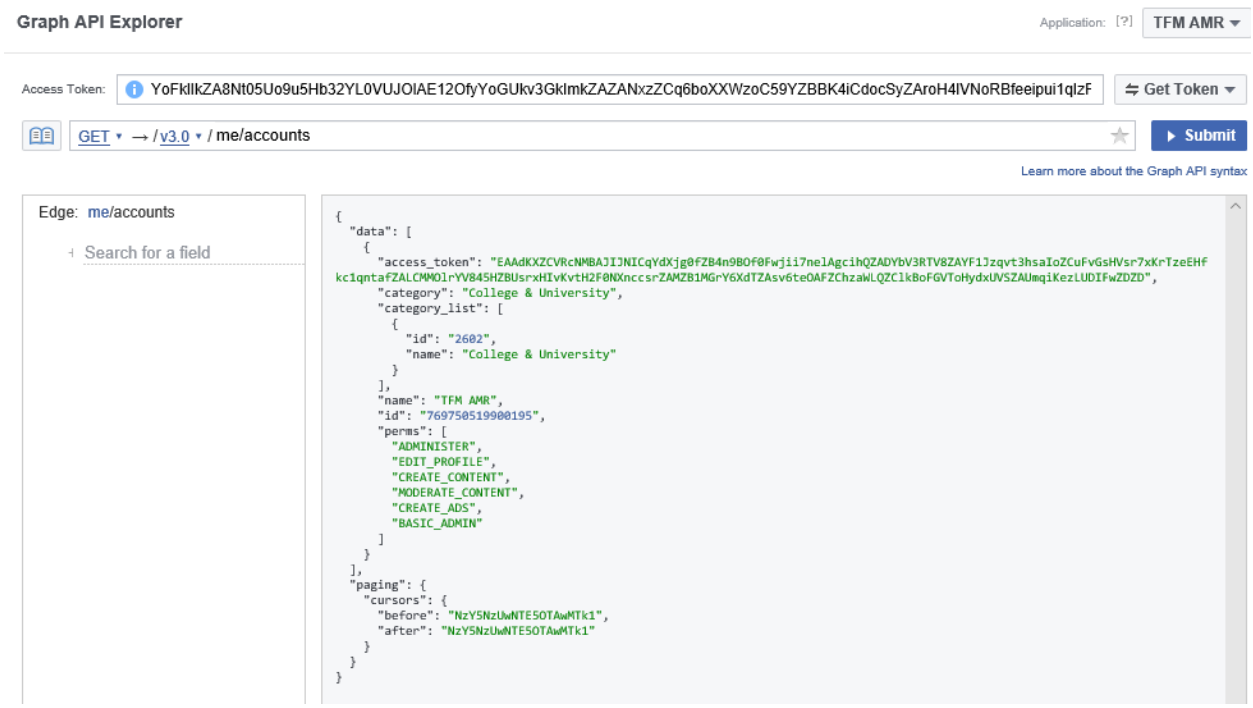


Ilustración 40 – Graph API GET sobre */me/accounts*

De este punto, obtenemos el Access token mostrado, y cambiamos nuestra petición a la original deseada, uniendo la ID de la página con los datos que queremos obtener: */ID/ratings* y le añadimos el *Access_token* que hemos visto, de manera que podamos acceder a los datos:

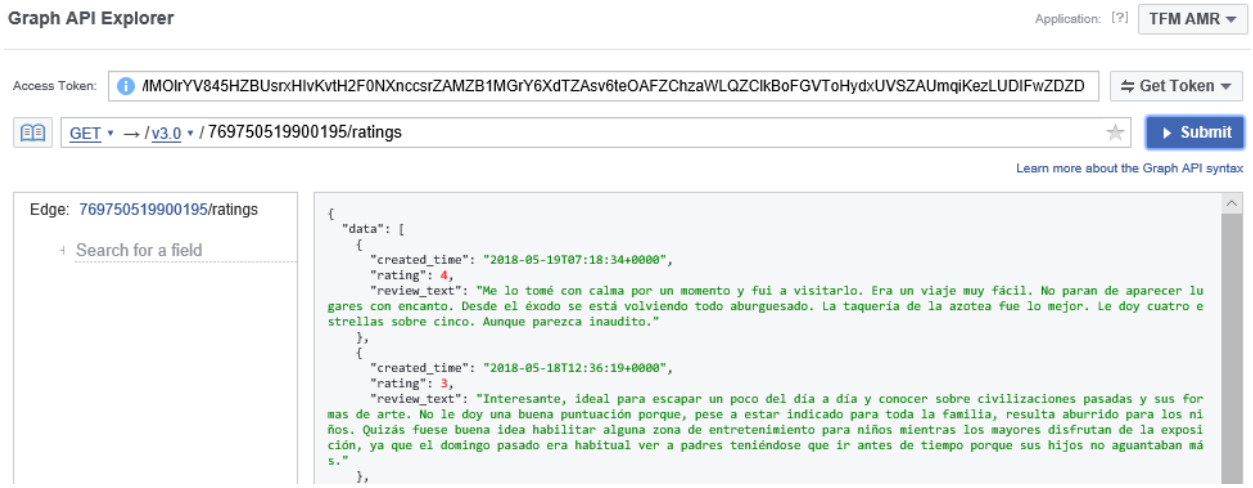


Ilustración 41 – Obtención de los datos con el nuevo *token*

Ahora vamos a hacer que no expire, para poder usarlo en nuestra aplicación. Para ello, vamos al símbolo de información superior que podemos ver en la imagen, y abrimos el *token* en la herramienta disponible para ello.

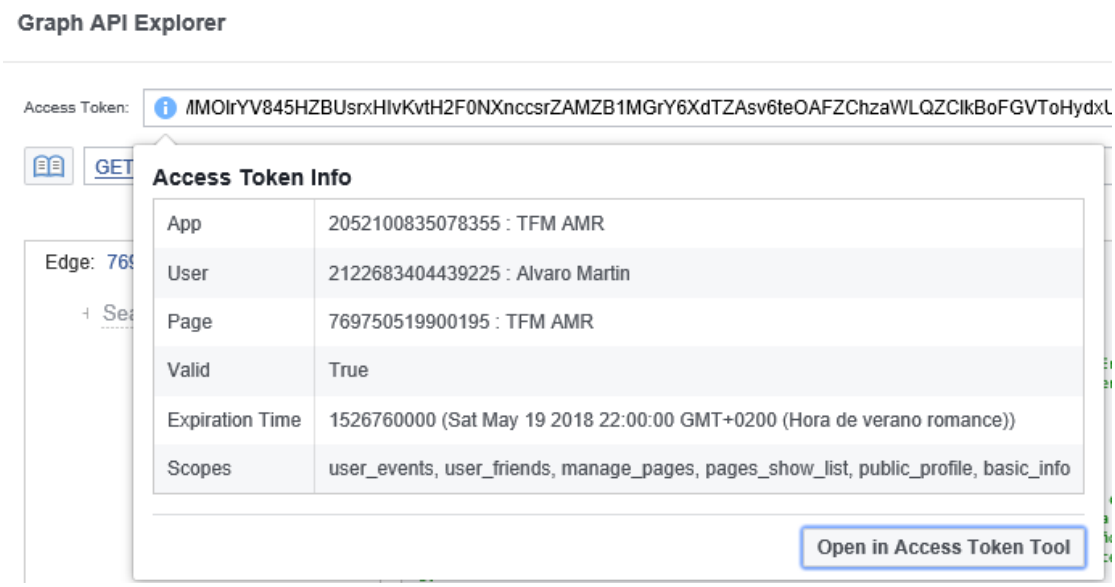


Ilustración 42 – Información del *token*

Podemos ver que caduca tras dos horas de uso aproximadamente desde que ha sido generado. Al abrirlo con la herramienta, sin embargo, podemos extender su duración.

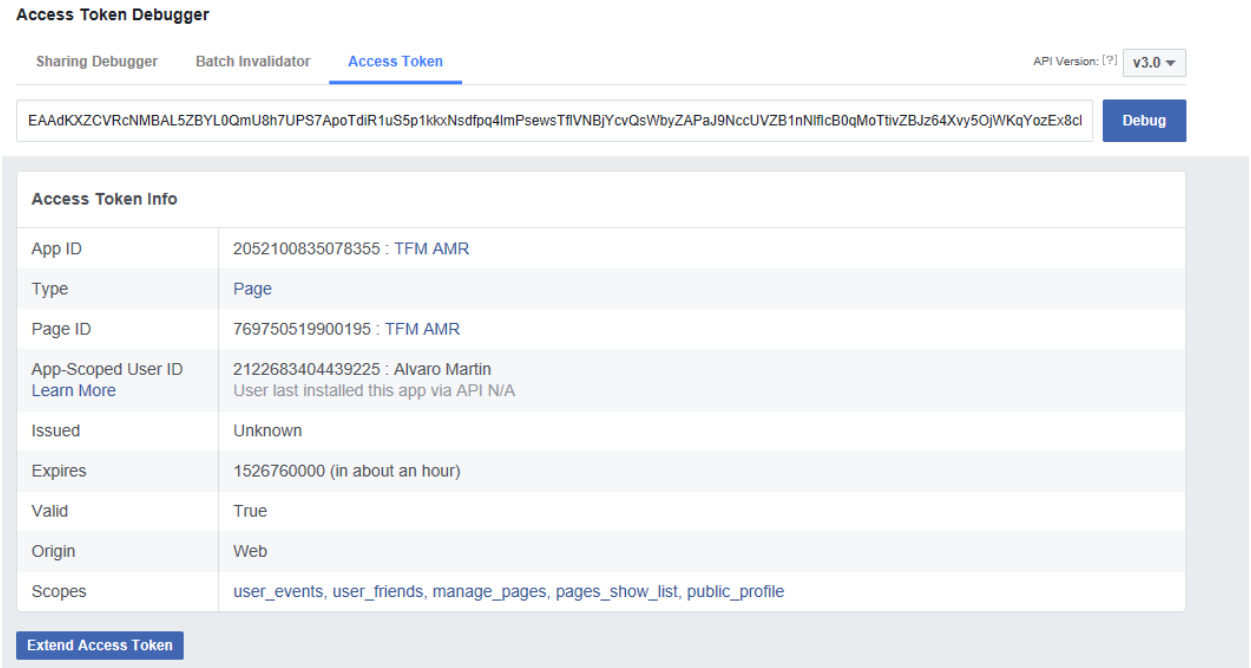


Ilustración 43 – *Page Token* en la herramienta de gestión

Ahora, pulsamos sobre el botón de la parte inferior derecha, consiguiendo así que no caduque en un periodo de dos meses. Concretamente, obtenemos un nuevo *token* con una **duración infinita** según el mensaje obtenido en la interfaz, para ser exactos:

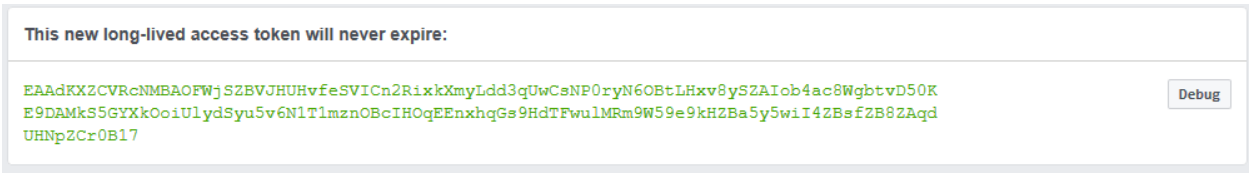


Ilustración 44 – *Page token* final obtenido para la aplicación

Si le damos a *debug*, podemos comprobar sus características:

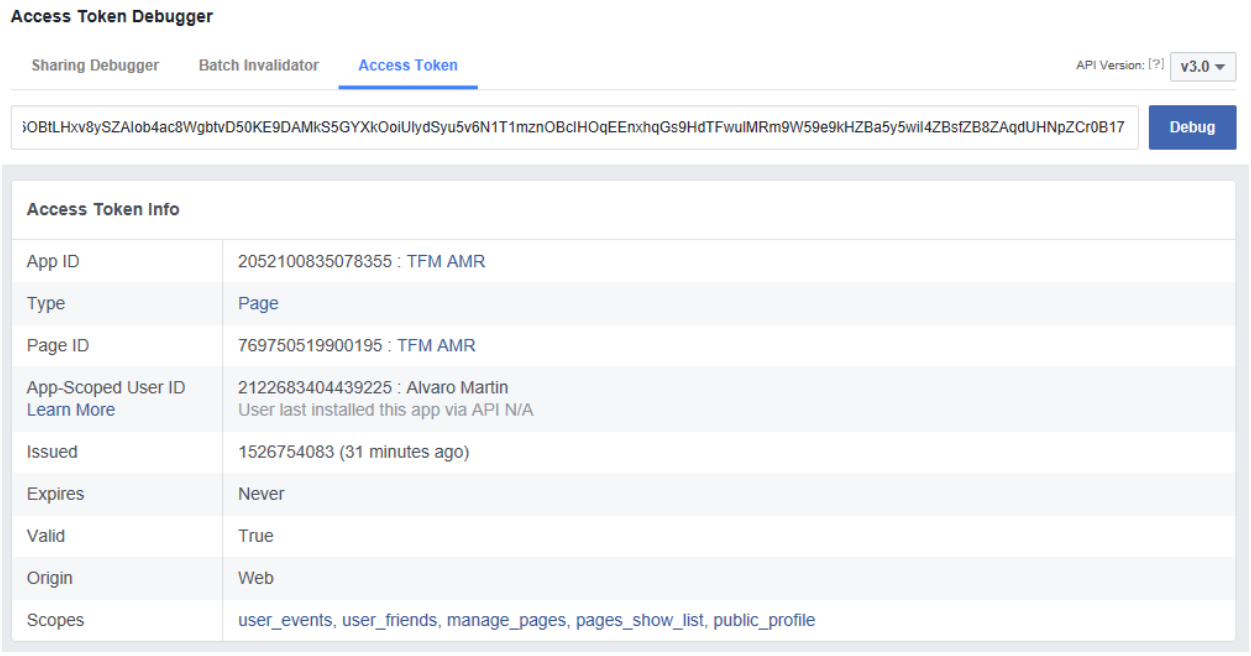


Ilustración 45 – Características del *token* final

### 5.4.3 Implementación de la aplicación recolectora

Ahora que ya tenemos el *token* necesario para la aplicación, podemos pasar a su desarrollo: la aplicación principal se encuentra en el fichero *FBCollector.py*, que importa a su vez el fichero *ConManager.py* con la clase del mismo nombre en su interior. Esta clase implementa los métodos para trabajar con la base de datos MySQL que necesita el programa principal para almacenar los datos con los que aquí trabajamos en la base de datos, así como las funciones necesarias para conectarse con Facebook y recabar las opiniones de los usuarios.

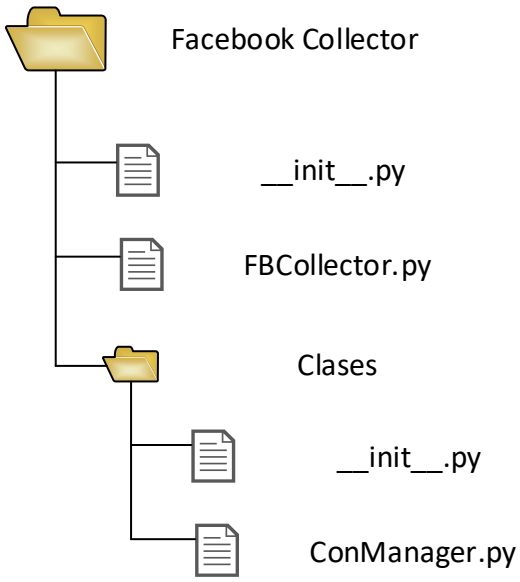


Ilustración 46 - Estructura de directorios de la aplicación de recolección de Facebook

La aplicación funcionará ejecutando de manera periódica una petición a los servidores de Facebook para recibir todas las opiniones de nuestra página, que procederá a analizar. Para ello, se iterará sobre todas las opiniones, realizando:

1. Calcular el MD5 a la opinión recibida.
2. Comprobar si hay coincidencia del MD5 con algún MD5 ya almacenado en la base de datos.
3. Si no lo hubiera, almacenar dicha opinión en la base de datos, junto con su valoración y su MD5.

Cuando se acaba de procesar la última opinión, el programa entra en modo de espera durante el tiempo configurado, que estableceremos en 5 minutos o 300 segundos, que es un tiempo aceptable para buscar nuevas opiniones en una red social (aunque podríamos emplear un tiempo incluso mayor). Durante ese tiempo, el usuario puede introducir una entrada para deter el programa, o en caso contrario, no introducir nada de manera que el programa vuelve a ejecutar la tarea iterativa antes descrita.

Los pasos 1 y 2 cobran ahora sentido, debido a que cuando se interroga a Facebook por las opiniones asociadas a la página, su respuesta incluye todas las opiniones disponibles, por lo que también se recibirán opiniones que ya se han almaceando en la base de datos, de ahí la necesidad de comparar primero si no existe previamente antes de almacenarla para evitar repeticiones.

## 5.5 Fase IV: Análisis de los Datos

Finalmente, debemos proceder con el análisis según lo estipulado anteriormente en el punto 4. Para ello, crearemos un programa capaz de recabar los datos de la base de datos MySQL que creamos y rellenamos anteriormente. Posteriormente, el programa los analizará y tras mostrarlos por pantalla al usuario que ha ejecutado el análisis, almacenará los resultados en la base de datos, en las tablas especificadas en el apartado 4.5, Persistencia de los resultados.

El formato seguido para la aplicación es similar al ya empeado con las anteriores aplicaciones que hemos realizado en Python, con una estructura de directorios y ficheros como la que sigue:

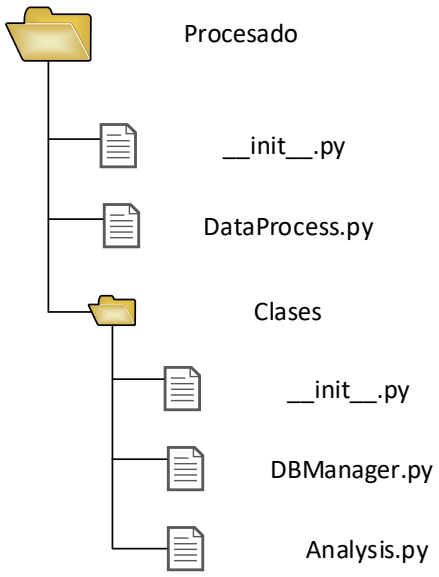


Ilustración 47 – Estructura de directorios de la aplicación de procesado

En esta ocasión, como podemos comprobar tenemos dos ficheros auxiliares en lugar de uno: el primero, *DBManager.py* se encarga como es habitual de las conexiones y funciones relacionadas con la base de datos, mientras que el fichero *Analysis.py* se encarga de llevar a cabo las tareas de análisis.

En el caso del fichero *DBManager.py*, la estructura de las funciones es similar a la empleada en los programas anteriores, pero con pequeñas diferencias:

- La función preparada para crear tablas nuevas no conocerá las tablas que se usaban para almacenar los datos del escenario, sino que creará nuevas tablas para almacenar los resultados del análisis.
- La función preparada para leer datos de las tablas no empleará directamente la función de lectura de la librería MySQLdb antes empleada, sino que pasará la conexión abierta a las funciones que integra Pandas para cargar los datos de las tablas directamente en *dataframes*, que es la clase por defecto con la que se trabaja en Pandas con los datos.

En cuanto al fichero *Analysis.py*, este fichero cuenta con todos los tipos de análisis en su interior, por lo que:

- Necesitamos las librerías:
  - *datetime*, para trabajar con las marcas temporales de fecha y hora.
  - *matplotlib*, para representar los resultados.
  - *numpy*, para emplear algunas funciones incluidas en esta librería en el análisis.
  - *pandas*, para realizar todo el grueso del análisis en base al tipo *dataframe* que esta librería define, entre otras utilidades.
  - *scikit-learn*, para poder implementar y aplicar el algoritmo KMeans para hacer los grupos o clusters [43].
- Internamente, se organiza en funciones, atendiendo a:
  - **F. Auxiliares**: de tipo matemático como truncar, calcular porcentajes con un número deseado de decimales, etc.
  - **F. de Serie de Tiempos**: encargadas de realizar las tareas del análisis de serie de tiempos, incluyendo cálculos, conteo de eventos o representación de los resultados.
  - **F. de Clusters**: encargadas de aplicar el algoritmo KMeans mediante la librería *scikit-learn* sobre los datos a clasificar.
  - **F. de Opinion**: encargadas de analizar las opiniones de los usuarios, así como de representar los resultados.
  - **F. de Regresion**: encargadas de contar los eventos de entrada por horas para hallar la relación, y representarlo en un diagrama de barras.

# 6 PRUEBAS DE FUNCIONAMIENTO

*Poseer información es una cosa. Otra muy diferente es saber lo que significa y cómo utilizarla.*

Jeff Lindsay

En continuación, ejecutaremos una prueba de concepto con vistas a comprobar el correcto funcionamiento del escenario, del software diseñado, de las implementaciones hardware y software realizadas, así como la utilidad de los análisis realizados, que esperamos consigan clasificar y extraer la información deseada.

Ejecutaremos a continuación una prueba de nuestro sistema, que se caracterizará por contener, como mínimo, con los datos recogidos en la siguiente tabla:

| Parámetro | Valor mínimo |
|---|---|
| Tiempo de captura de datos con los sensores | 5 días |
| Nº de Opiniones | 20 |

Tabla 19 – Parámetros mínimos de la prueba del sistema

Mostraremos aquí la salida obtenida para la ejecución de los diferentes programas, si se ha encontrado algún error significativo y su solución, y finalmente el análisis de datos en la fase IV para los valores recogidos en la tabla que acabamos de ver. Antes de comenzar con estas fases, remarcaremos un detalle sobre la implementación realizada que se comentaba en el punto 5.1 de este proyecto.

En ella, se especificó el hardware con el que funcionaba la máquina virtual de Linux Mint de 64 bits sobre el hipervisor Hyper-V. Este hipervisor permite comprobar el uso que se le están dando a los recursos asignados, por parte de las máquinas virtuales en funcionamiento. Concretamente, nos interesan los valores de la memoria RAM asignada a la máquina virtual, que podemos ver en la siguiente captura:

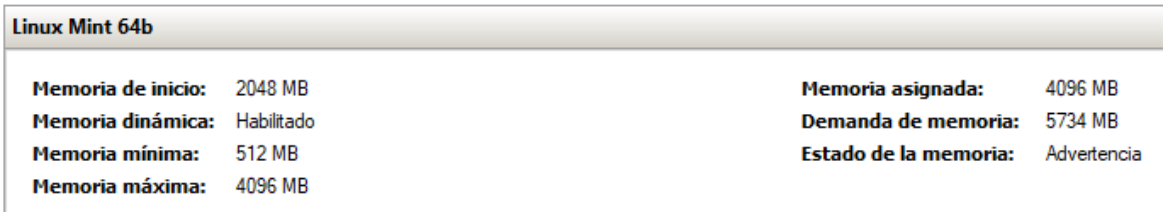


Ilustración 48 – Uso de memoria según Hyper-V

Como podemos comprobar, pese a que hemos permitido emplear a la máquina virtual hasta 4GB de memoria RAM, está demandando casi 6GB. A pesar de ello, funciona correctamente ya que emplea memoria de intercambio o *memoria swap*. Sin embargo, nos gustaría hacer notar al lector que en caso de disponer de una mayor cantidad de memoria RAM, sería conveniente su adjudicación a esta máquina virtual si piensa ejecutar los mismos elementos que nosotros sobre ella.

Para finalizar, y antes de proceder a la primera fase, dado que en ocasiones se emplean mensajes en formato JSON tanto por parte de nuestras aplicaciones como de respuesta por algunas entidades del sistema,

emplearemos un **servicio de validación de formato** en JSON para comprobar la corrección de estos mensajes durante la depuración, como el que puede encontrarse en la referencia [44].

## 6.1 Fase I: Producción de Datos

Para determinar el correcto funcionamiento de este apartado del sistema, procedemos en el siguiente orden:

1. Ponemos en funcionamiento la placa Arduino UNO ya programada.
2. Ponemos en funcionamiento el receptor C++. En la siguiente ilustración, podemos ver una ejecución del programa con pequeñas modificaciones para mostrar por pantalla (salida estándar) los valores recibidos con un formato amigable para el usuario.

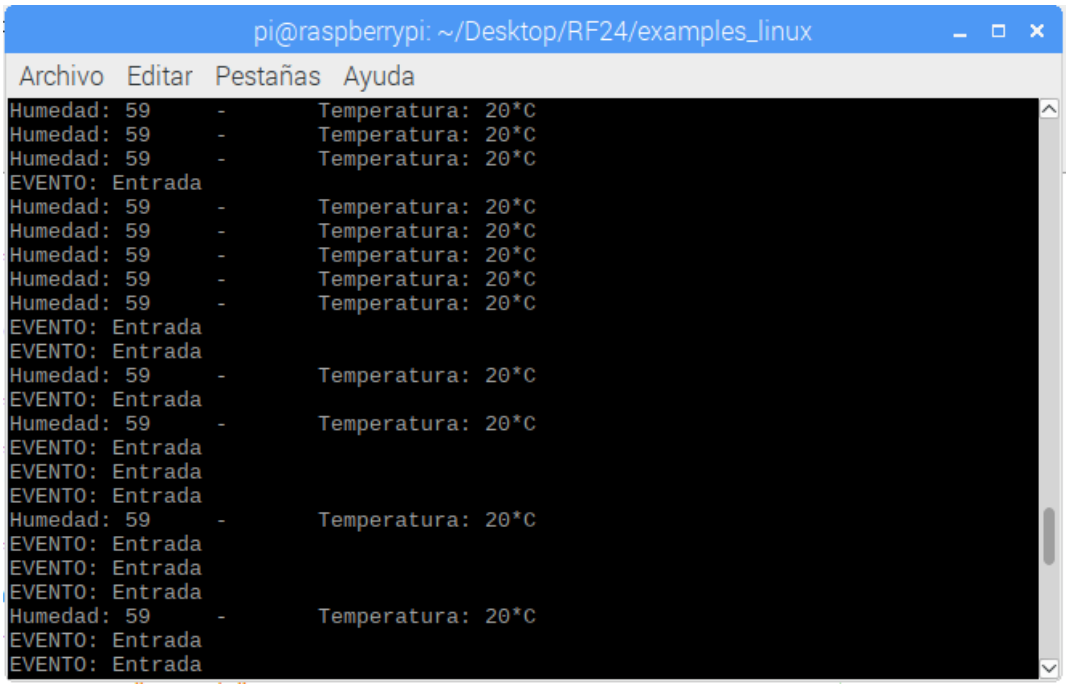


Ilustración 49 – Depuración del receptor en C++

3. Ponemos en funcionamiento el concentrador realizado en Python, que por un lado recibe los valores del programa receptor, y los introduce en Orion. La salida estándar del programa receptor anterior, es dirigida como entrada estándar a este programa, lo cual se realiza ejecutando ambos programas a la vez mediante el script *RxHub.sh*, como se aprecia en la siguiente ilustración:

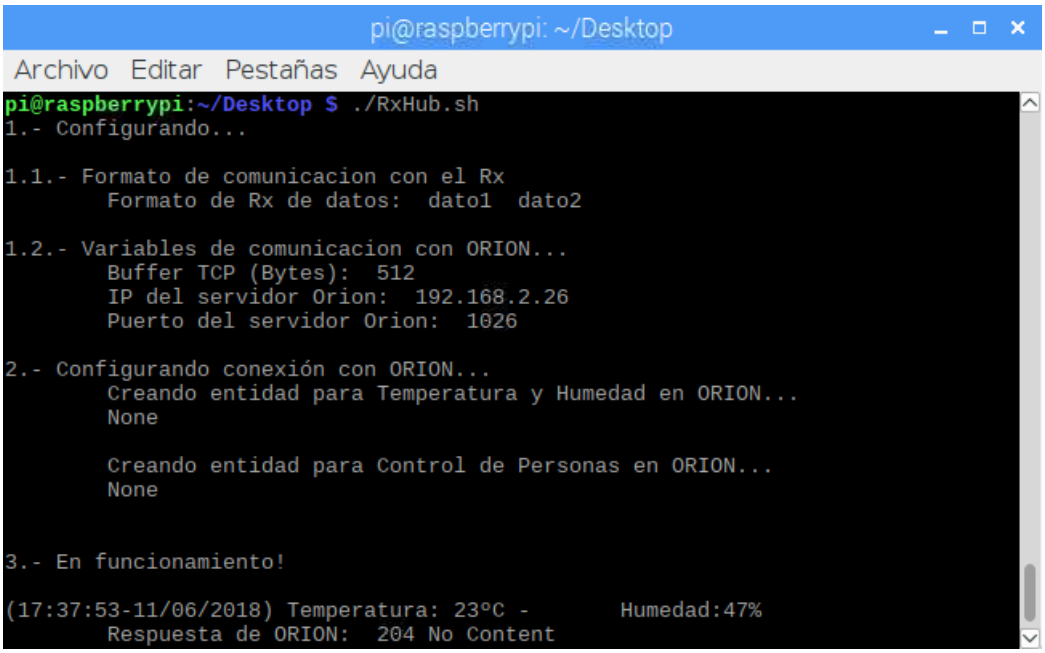


Ilustración 50 – Depuración del conjunto receptor

Podemos observar como respuesta *None* en la creación de las entidades, debido a que **estas entidades ya existían** en Orion cuando se ejecutó el programa para realizar esta captura. Asimismo, comprobamos que la respuesta de Orion es la correspondiente al código 204, que es la respuesta predeterminada de Orion para mensajes de tipo actualización como el que se ha producido sobre la entidad de temperatura y humedad del museo. Con esta comprobación realizada, tenemos que Orion se encuentra desplegado y en funcionamiento, lo cual también podemos comprobar interrogando mediante un navegador web a la dirección url:

```
IP:1026/version
```

A lo que obtenemos como resultado:

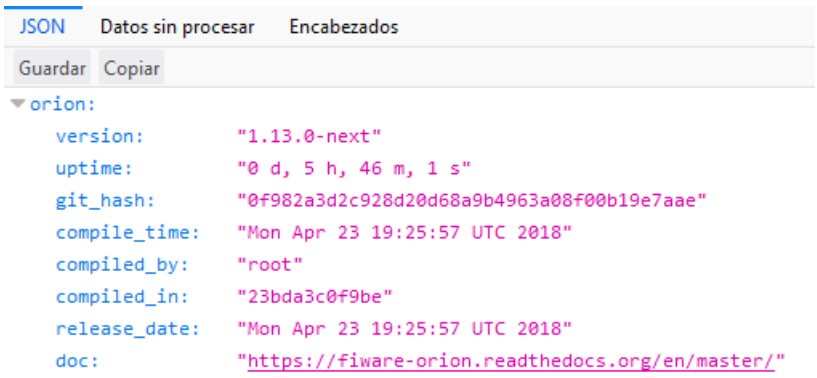


Ilustración 51 – Orion en funcionamiento

Podemos observar que Orion nos responde con información genérica, entre la que se incluye su versión o el tiempo que lleva activo en el momento de la consulta.

## 6.2 Fase II: Recolección de Datos Internos

### 6.2.1 Consideraciones previas

Debemos tener en cuenta que, para recabar los datos internos, es decir aquellos que provienen de la plataforma FIWARE, necesitaremos:

- Que Orion esté en funcionamiento (comprobado en el punto 6.1)
- Que la base de datos MySQL esté disponible

Para comprobar que la base de datos está disponible, realizaremos una pequeña prueba de concepto directamente sobre el intérprete de Python, que se realiza como complemento a la explicación del uso de la librería empleada en el anexo *Acceso a la Base de Datos mediante Python*. Sin embargo, añadiremos aquí un pequeño test, en el que comprobaremos el acceso a las tablas empleadas por el sistema dentro de la base de datos, así como la longitud de los datos en estas en el momento de su ejecución. Dado que la foto ha sido tomada con el sistema ya en marcha, las tablas ya cuentan con datos en su interior, por lo que no están vacias:

```
btc@btc-vm ~/dataCollect $ python dbTest.py
La tabla Facebook tiene 15 registros
La tabla Museo tiene 54517 registros
La tabla Evento tiene 1194 registros
```

Ilustración 52 – Comprobación del funcionamiento de la base de datos

### 6.2.2 Recolector de datos de Orion

Si ya tenemos activa la parte del sistema de producción de datos, estos ya estarán siendo insertados en Orion, por lo que ahora debemos activar el programa suscriptor que recupera el valor de las entidades definidas. Para ello, el primer paso será suscribirnos a tales entidades en Orion, eligiendo como parámetro disparador de actualización la fecha y hora asociada a la entidad, de manera que en el suscriptor obtengamos todas las actualizaciones para la entidad, ya que el valor de fecha y hora siempre variará, por lo que Orion nos mantendrá informados de estos cambios, aunque otros valores como la temperatura no cambien. Estas suscripciones se harán sobre la url:

```
IP:1026/v2/subscriptions
```

Siendo estas:

**Petición de suscripción para la temperatura y humedad**

```
curl -v 192.168.2.26:1026/v2/subscriptions -s -S -H 'Content-Type: application/json' -d @- <<EOF
{
  "description": "Suscripcion para obtener T y H del MUSEO",
  "subject": {
    "entities": [
      {
        "id": "Museo",
        "type": "Edificio"
      }
    ],
    "condition": {
      "attrs": [
        "Temperatura",
        "Humedad",
        "Fecha"
      ]
    }
  },
  "notification": {
    "http": {
      "url": "http://192.168.2.26:1028/pushData"
    },
    "attrs": [
```

```
        "Temperatura",
        "Humedad",
        "Fecha"
      ]
    },
    "expires": "2040-01-01T14:00:00.00Z",
    "throttling": 5
  }
  EOF
```

**Petición de suscripción para los eventos de entrada o salida**

```
  curl -v 192.168.2.26:1026/v2/subscriptions -s -S -H 'Content-Type:
application/json' -d @- <<EOF
  {
    "description": "Suscripcion para obtener eventos del MUSEO",
    "subject": {
      "entities": [
        {
          "id": "Evento",
          "type": "ControlAcceso"
        }
      ],
      "condition": {
        "attrs": [
          "Evento",
          "Fecha"
        ]
      }
    },
    "notification": {
      "http": {
        "url": "http://192.168.2.26:1028/pushData"
      },
      "attrs": [
        "Evento",
        "Fecha"
      ]
    },
    "expires": "2040-01-01T14:00:00.00Z",
    "throttling": 5
  }
  EOF
```

Aquí se ha mostrado la suscripción para realizarla sobre un terminal empleando la herramienta *curl*, pero **nosotros la hemos incrustado en el programa realizado**. El formato seguido ha sido una suscripción para cada entidad, pero sería posible suscribirse a varias entidades con una única suscripción más compleja.

Ahora, si ejecutamos nuestro programa recolector de datos, obtenemos la salida que se ve en la siguiente ilustración, durante la ejecución inicial del programa. La llamada al programa debe efectuarse con una serie de parámetros, que recordamos fueron comentados en el apartado 5.3, concretamente:

**Ejecución del programa recolector de datos de Orion**

```
 ./OrionCollector.py –host 192.168.2.26 --port 1028 --url /pushData --
pretty-print -v
```

```
btc@btc-vm ~/dataCollect/Orion $ python OrionCollector.py --host 192.168.2.26 --port 1028 --url /pushData --pretty-print -v
verbose mode is on
port: 1028
host: 192.168.2.26
server_url: /pushData
pretty: True
https: False
No se ha podido crear la tabla 'Museo'
No se ha podido crear la tabla 'Evento'
 * Serving Flask app "OrionCollector" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://192.168.2.26:1028/ (Press CTRL+C to quit)
=====================================

19:40:54-11/06/2018     H: 50   T: 24

172.17.0.3 - - [11/Jun/2018 19:40:55] "POST /pushData HTTP/1.1" 200 -
=====================================

19:40:54-11/06/2018     H: 50   T: 24
```

Ilustración 53 – Ejecución del recolector de Orion

Podemos comprobar que las dos tablas necesarias no son creadas, ya que han sido creadas en ejecuciones anteriores del mismo programa. Por otro lado, el programa se pone a la escucha en el puerto 1028, y comienza a recibir de manera correcta las actualizaciones de temperatura y humedad de Orion.

Cabe destacar que, durante la ejecución de esta prueba, **se detectó un fallo** en el manejo de la conexión con la base de datos MySQL, que venía producido porque la conexión con la base de datos caducaba produciendo el error con código 2006 como se ve en la siguiente ilustración:

```
172.17.0.3 - - [08/Jun/2018 16:46:09] "POST /pushData HTTP/1.1" 500 -
=====================================

16:46:20-08/06/2018     H: 66   T: 23

[2018-06-08 16:46:19,482] ERROR in app: Exception on /pushData [POST]
Traceback (most recent call last):
  File "/usr/local/lib/python2.7/dist-packages/flask/app.py", line 2292, in wsgi_app
    response = self.full_dispatch_request()
  File "/usr/local/lib/python2.7/dist-packages/flask/app.py", line 1815, in full_dispatch_request
    rv = self.handle_user_exception(e)
  File "/usr/local/lib/python2.7/dist-packages/flask/app.py", line 1718, in handle_user_exception
    reraise(exc_type, exc_value, tb)
  File "/usr/local/lib/python2.7/dist-packages/flask/app.py", line 1813, in full_dispatch_request
    rv = self.dispatch_request()
  File "/usr/local/lib/python2.7/dist-packages/flask/app.py", line 1799, in dispatch_request
    return self.view_functions[rule.endpoint](**req.view_args)
  File "./dataCollector.py", line 377, in record
    write_data(datos)
  File "./dataCollector.py", line 119, in write_data
    # raise error
OperationalError: (2006, 'MySQL server has gone away')
172.17.0.3 - - [08/Jun/2018 16:46:19] "POST /pushData HTTP/1.1" 500 -
```

Ilustración 54 – Error 2006 en la conexión MySQL

Esto lo solucionaremos **controlando las excepciones**, de manera que cuando el error venga de la función write_data como se aprecia en la ilustración, se cerrará la conexión existente (si la hubiere) y **se creará una nueva conexión** con la base de datos, de forma que se solucione el error en caso de ser el aquí mencionado.

## 6.3 Fase III: Recolección de Datos Externos

### 6.3.1 Empleo de la página de Facebook

Una vez se dispone de una página de Facebook operativa, buscaremos emplear la opción de *opinión* o *review* (en inglés), para valorar el sitio o servicio que representa la página, en nuestro caso un museo. Para ello, no hay más que acceder a la página, y buscar el apartado con dicho nombre, disponible tanto en el menú lateral

como en el menú superior de la página:

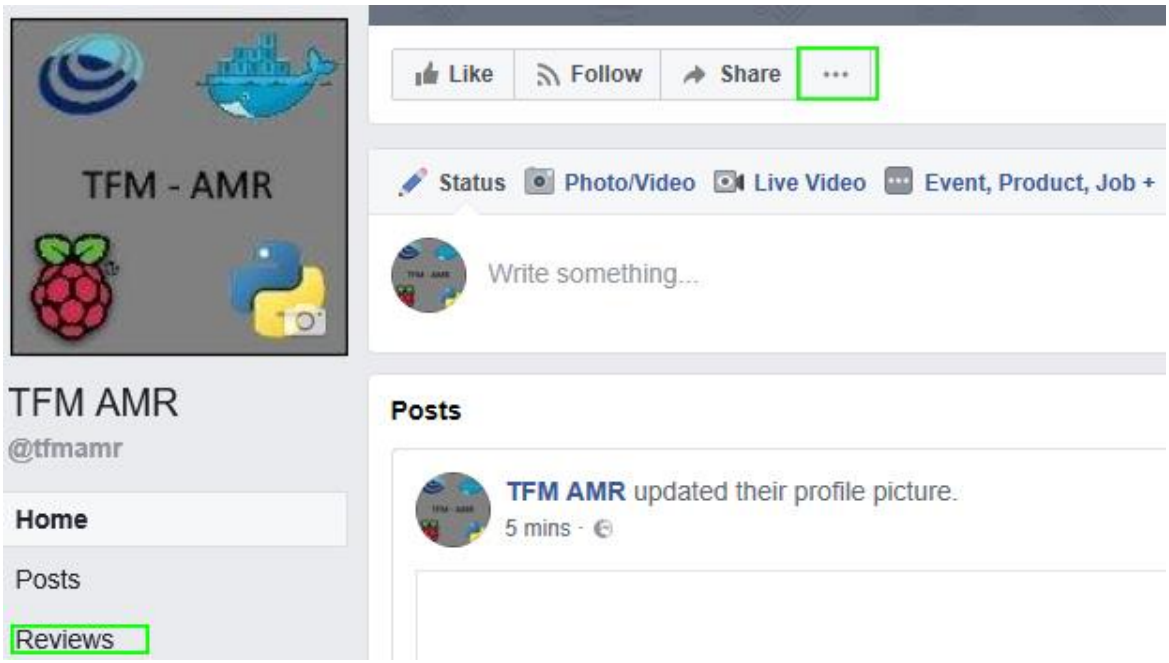


Ilustración 55 – Acceso a la sección de opiniones en una página de Facebook

Ahora, solo debemos pulsar el botón *escribir una opinión* o *write a review*, también disponible en el menú superior:

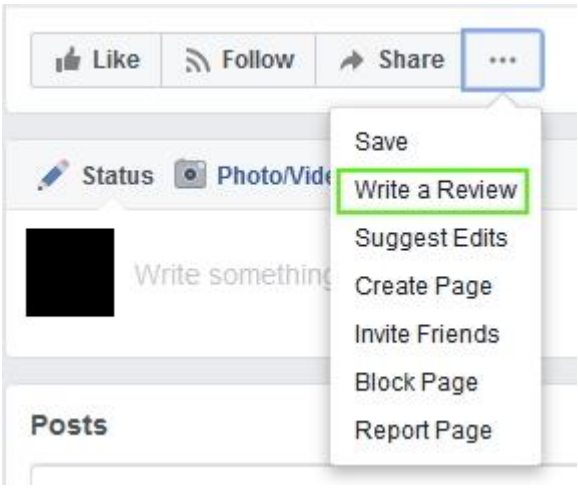


Ilustración 56 – Opción *escribir una opinión*

Las opiniones constan de un apartado escrito y una valoración numérica, y son visibles para todos los usuarios que acceden a dicha página. Con la colaboración de otras personas ajenas a este trabajo final de master, podemos comprobar que todo funciona de manera correcta al verificar la existencia de opiniones en este apartado.


Ilustración 57 - Apartado de opiniones: comprobación

### 6.3.2 Comprobación del *token* obtenido

Ahora debemos comprobar que el *Page Token* finalmente obtenido es válido para nuestra aplicación. Para ello, probamos a realizar una petición desde una extensión para peticiones REST desde el navegador web, incluyendo este *token* como credencial, y verificamos que el resultado es válido.

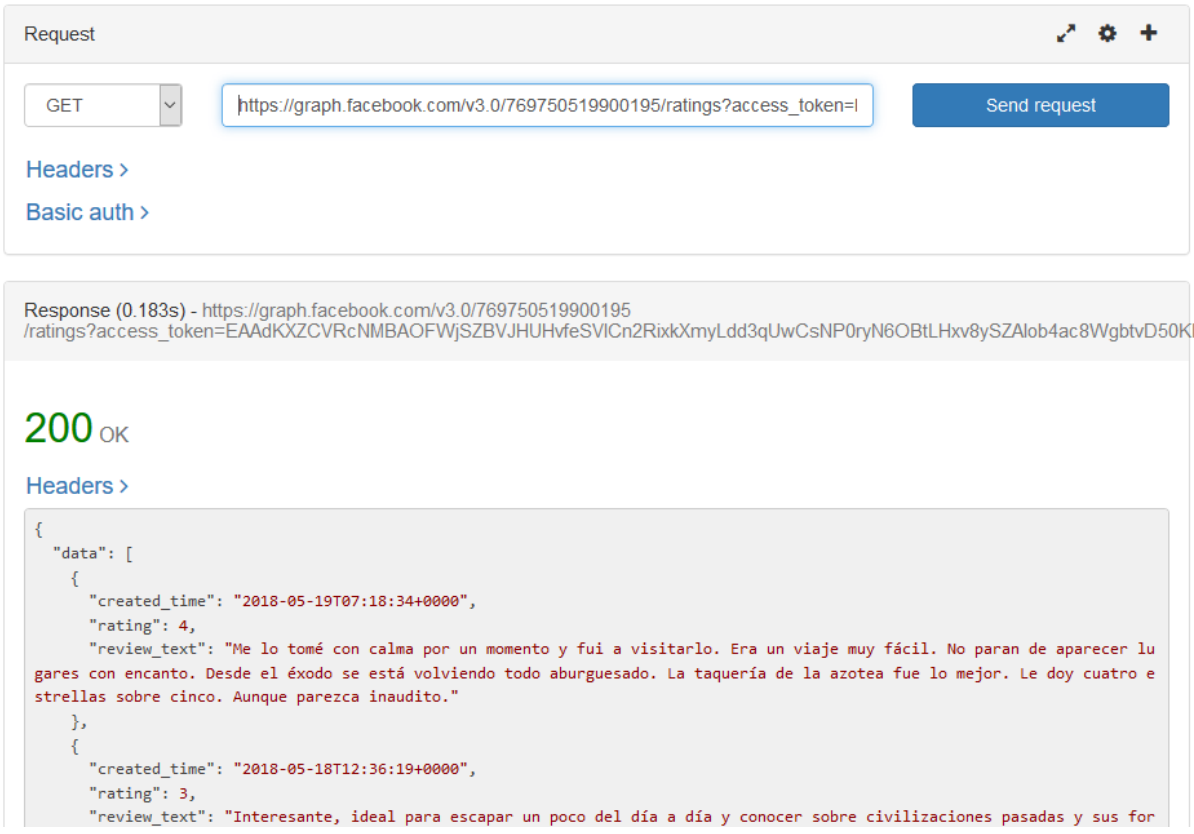


Ilustración 58 – Validación del *Page Token* obtenido mediante petición REST externa

### 6.3.3 Recolección de datos mediante la aplicación

Ahora, ya podemos verificar que todo funcione en nuestra propia aplicación Python. Para ello, ejecutaremos nuestra aplicación, de manera similar a la fase II, ejecutando la aplicación disponiendo de la base de datos para almacenar los datos.

```
btc@btc-vm ~/dataCollect/Facebook $ python FBCollector.py
1.- Configurando...

        IP MySQL:  192.168.2.26

        Facebook Page ID:  769750519900195

2.- Preparando conexiones y pasos previos...

        Conectando con la BBDD del sistema...
        Creando la tabla 'Facebook' en la BBDD...
                Error al crear la tabla, es posible que ya exista
                No se ha podido crear la tabla 'Facebook'

3.- Buscando datos...
        Control: 2018-06-11 11:59:58.809153
        Creando conexion con Facebook...
        Conectando con la BBDD...

        Obteniendo opiniones del Museo...

        Obteniendo opiniones previamente almacenadas en la BBDD...
        Procesando opinion: 1
                Buscando coincidencias...
```

Ilustración 59 – Ejecución del recolector de Facebook

Siguiendo con las pruebas realizadas anteriormente, en este caso la tabla no es creada en la base de datos debido a que ya existe, y se procede a **analizar las opiniones existentes** en la página de Facebook correspondiente a esta prueba simulando un museo. Podemos comprobar que se están buscando coincidencias previas en la base de datos, para ver si esa opinión ya se encuentra almacenada. Como no lo están, son todas almacenadas, como se ve en la siguiente ilustración:

```
                Buscando coincidencias...
                No se han encontrado coincidencias
        Transmitiendo a la BBDD...
        Procesando opinion: 8
                Buscando coincidencias...
                No se han encontrado coincidencias
        Transmitiendo a la BBDD...
        Procesando opinion: 9
                Buscando coincidencias...
                No se han encontrado coincidencias
        Transmitiendo a la BBDD...
        Procesando opinion: 10
                Buscando coincidencias...
                No se han encontrado coincidencias
        Transmitiendo a la BBDD...
        Procesando opinion: 11
                Buscando coincidencias...
                No se han encontrado coincidencias
        Transmitiendo a la BBDD...
        Cerrando conexion con Facebook...
        Cerrando conexion con la BBDD...

Si desea no seguir ejecutando el programa, pulse N (60s):
```

Ilustración 60 – Primera iteración sobre las opiniones en Facebook

Posteriormente, el programa repite la operación pasado un tiempo, en busca de nuevas opiniones. En este caso, aquellas opiniones que ya existen en la base de datos son ignoradas, como se puede comprobar en la siguiente ilustración:

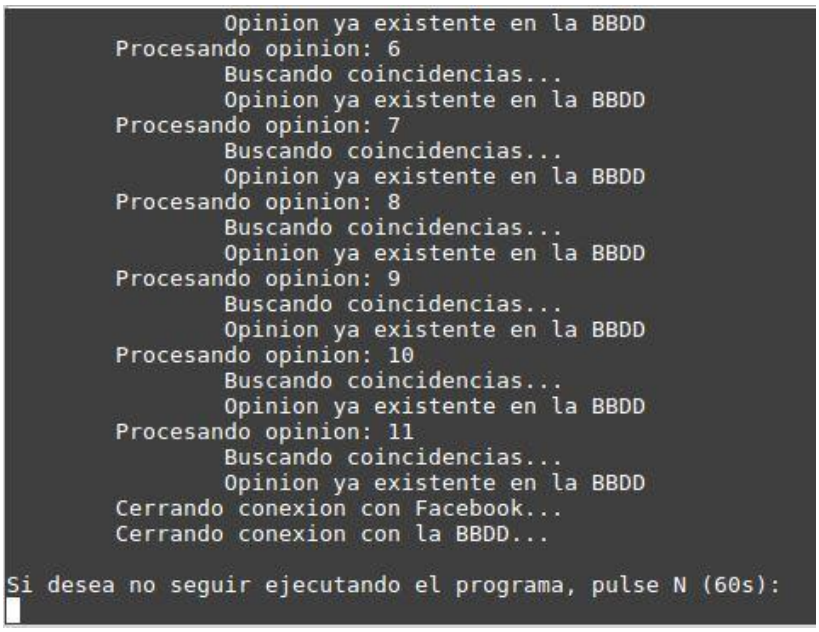


Ilustración 61 – Segunda iteración sobre las opiniones en Facebook

En caso de que hubiese nuevas opiniones en la página de Facebook en una iteración, que no hubiesen sido encontradas previamente, estas se añadirían a la base de datos para que estuviesen disponibles para el apartado de análisis.

Con este apartado, queda comprobado que nuestros programas funcionan correctamente para recolectar los datos y almacenarlos en la base de datos que hemos dispuesto. Queda pendiente como colofón, comprobar que la base de datos está almacenando estos registros. Para ello, creamos un pequeño programa que comprueba esta cantidad, y lo mostramos a continuación:

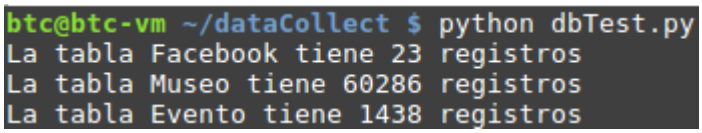


Ilustración 62 – Cantidad de registros en cada tabla existente en la base de datos

## 6.4 Fase IV: Análisis de Datos

Finalmente llega la parte central, el propio análisis o procesado de los datos. Comenzamos ejecutando nuestro programa, que procede a conectarse a la base de datos MySQL para obtener todo lo necesario, realiza pequeñas conversiones de tipos previas que son necesarias, como son fechas que estaban almacenadas como cadenas de texto y que son pasadas a tipos más adecuados, o la limpieza de los datos para eliminar aquellos que se detecten como inválidos, para comenzar a emitir resultados por pantalla como se puede apreciar en la siguiente ilustración:

```
btc@btc-virtual-machine ~/PROCESADO $ python DataProcess.py
1.- Configurando variables...
2.- Conectando con la base de datos...
3.- Cargando los datos...

4.- Preparando analisis...

        4.1.- Convirtiendo strings en fechas...
        4.2.- Eliminando datos no validos...
                Datos eliminados:  68

5.- Analisis de SERIE de TIEMPOS

        5.1.- Visitas por dia 98.0

        5.2.- Concurrencia minima:  0
        5.3.- Concurrencia maxima:  46

        5.4.- Satisfaccion:  82.6 %

        5.5.- Valoracion umbral:  3
        5.6.- Valoracion:  3.52

        5.7.- Porcentaje de presencia de cada temperatura
        Temperatura:  23.0  - Presencia:  43.4222096368 %
        Temperatura:  22.0  - Presencia:  9.98256255868 %
        Temperatura:  24.0  - Presencia:  30.2055233468 %
        Temperatura:  25.0  - Presencia:  13.7219249743 %
        Temperatura:  26.0  - Presencia:  2.66777948343 %

        5.8.- Porcentaje de presencia de cada humedad
        Humedad:  50  - Presencia:  3.67080495402 %
        Humedad:  51  - Presencia:  2.11037751315 %
        Humedad:  52  - Presencia:  3.72892975841 %
        Humedad:  53  - Presencia:  4.42046589266 %
```

Ilustración 63 – Salida del procesado de los datos

Además de los resultados numéricos, se lleva a cabo una representación de los porcentajes de presencia de las temperaturas y humedades, en gráficas separadas que permiten visualizar la distribución de estos valores:

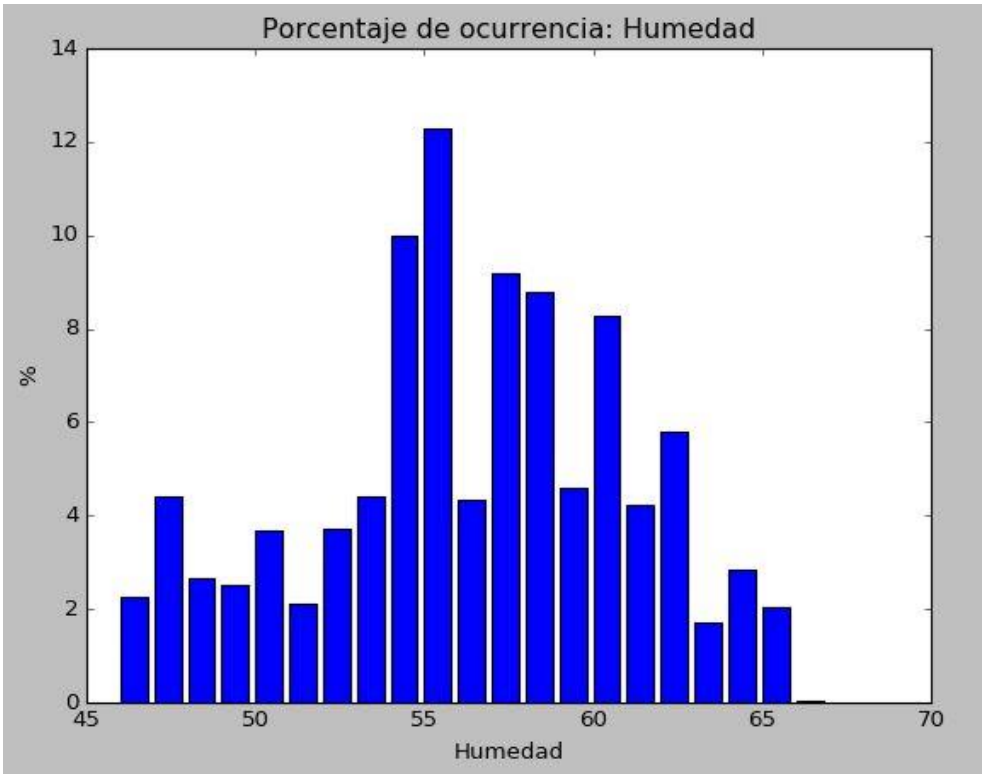


Ilustración 64 – Porcentaje de ocurrencia de las humedades

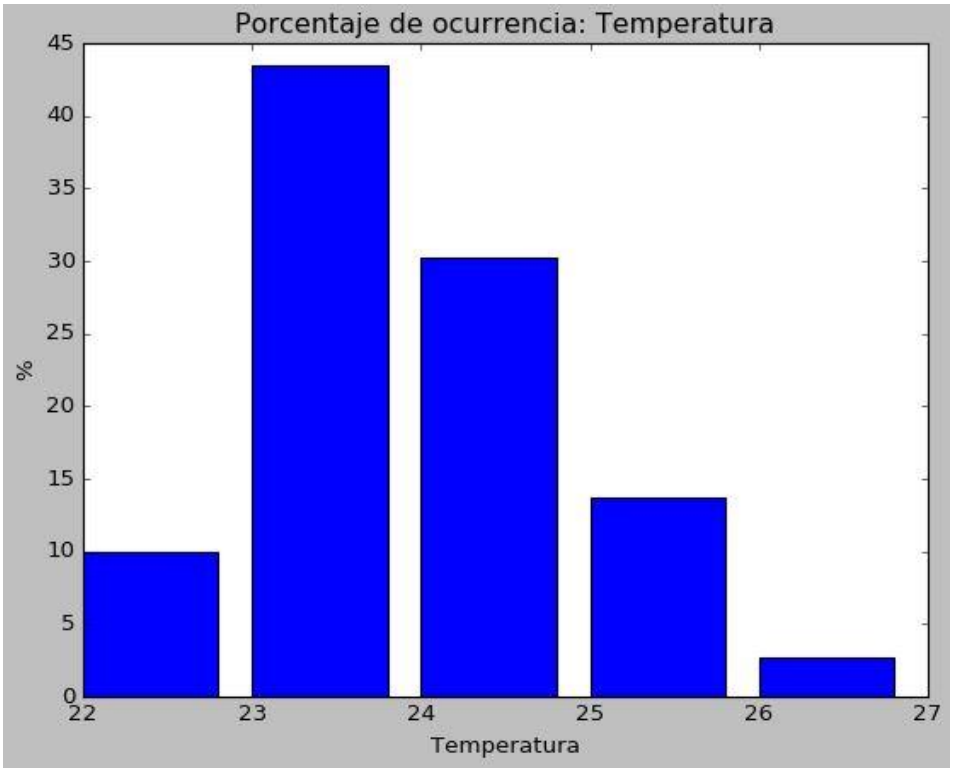


Ilustración 65 – Porcentaje de ocurrencia de las temperaturas

A continuación, se realiza el análisis de clusters para el que se aplica el algoritmo KMeans. Para ello, antes de comenzar veremos las valoraciones que tenemos en Facebook, y las clasificaremos manualmente para probar la eficiencia de este método, así como el análisis de opinión posterior.


Ilustración 66 – Valoraciones obtenidas de los usuarios en Facebook

Estas opiniones serán consideradas como se aprecia en la siguiente tabla, dejando la opción de 3 estrellas a libre interpretación en la clasificación de usuarios. Esto se debe a que, si la nota general de los usuarios fuese positiva, podríamos interpretar que es un usuario satisfecho pero con ciertos detalles a mejorar, sin llegar a desaconsejar la experiencia, mientras que si la nota general de los usuarios fuese negativa, podríamos entender esta valoración como que el usuario ha visto ciertas cosas de interés, pero en general no se llega a recomendar la experiencia. Dejaremos pues, que el algoritmo de clustering decida qué hacer con estos usuarios, y aparecerán sumados a uno de los dos grupos, satisfechos o insatisfechos:

| Valoración | Clasificación de Usuarios | Total | Clasificación de Opinión | Total |
|---|---|---|---|---|
| 5 Estrellas | Satisfecho | 14 | Positiva | 14 |
| 4 Estrellas | Satisfecho | | Positiva | |
| 3 Estrellas | - | 5 | Neutra | 5 |

| | | | | |
|---|---|---|---|---|
| 2 Estrellas | Insatisfecho | 4 | Negativa | 4 |
| 1 Estrella | Insatisfecho | | Negativa | |

Tabla 20 – Clasificaciones esperadas de los análisis de *clustering* y opinión

Ahora podemos proceder con los resultados del análisis de clusters:

```
6.- Analisis de Clusters

        6.1.- Calculando clusters...
                Visitantes satisfechos:  19
                Visitantes no satisfechos:  4
        6.2.- Dibujando resultado...
```

Ilustración 67 – Resultados del análisis de clusters: KMeans

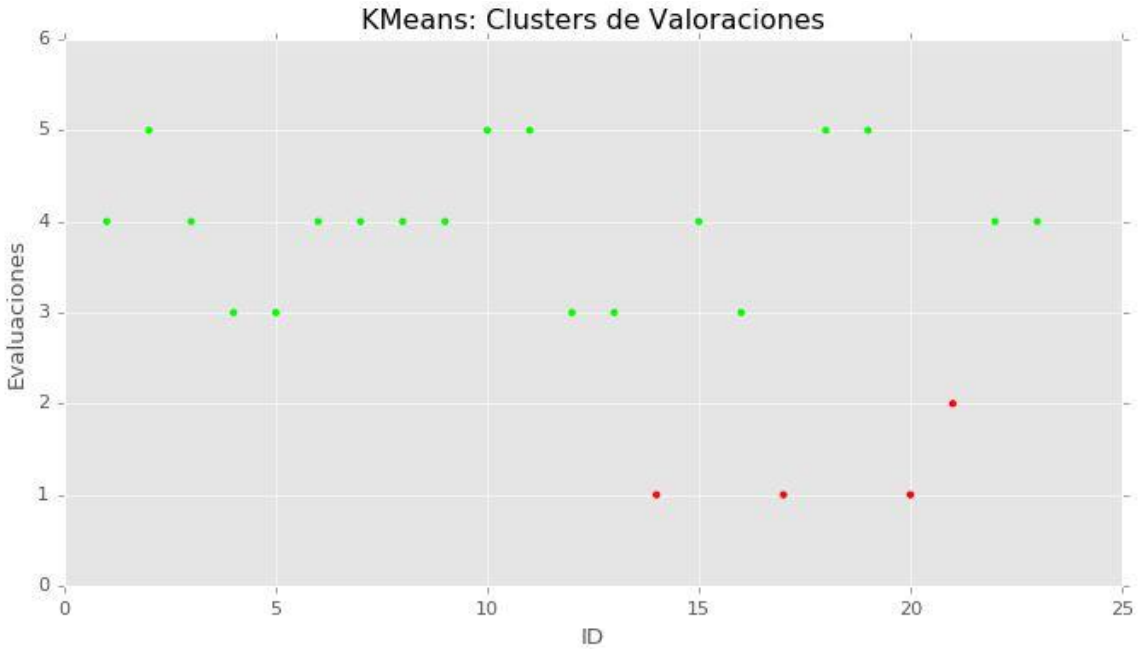


Ilustración 68 – Clasificación KMeans: representación gráfica

El resultado casa con lo esperado, ya que por el funcionamiento de KMeans, en el que recordamos que se asignan los individuos al grupo más cercano, el grupo de usuarios satisfechos debe generar una media más próxima al 4 que al 5, ya que hay más valoraciones con 4 estrellas que con 5. En el grupo de usuarios insatisfechos, el resultado es el contrario, teniendo una media más cercana al 1 que al 2. Por todo ello, las valoraciones de 3 puntos se encuentran **más cerca de la media del grupo de usuarios satisfechos**, y son clasificados como tal.

Si pasamos al análisis de opinión, donde cada una se encuentra representada por su *hash* MD5, obtenemos:

```
7.- Analisis de Opinion

        Opinion MD5( 1d2c89eef4a7ab186df6827764b91c2b ):  POSITIVA
        Opinion MD5( 334e0b45a0e932642f10920f70d701de ):  NEUTRA
        Opinion MD5( 3b2e3682c9d7babadf1b21279173c57f ):  POSITIVA
        Opinion MD5( 41078b5a300762334f9fca9966cfc104 ):  NEUTRA
        Opinion MD5( 62884ec294030f0edebb4577603c0cd0 ):  POSITIVA
        Opinion MD5( 756f1f3247cc2fcdc5308027eb6753fd ):  POSITIVA
        Opinion MD5( 8d6b01e79595f9fb5a4db9037d6ee82c ):  NEGATIVA
        Opinion MD5( a0fb50fe6cae66a62c29436978e1fff4 ):  POSITIVA
        Opinion MD5( b3c2db4a6a83d9561e3b0d1a5fd14b35 ):  NEUTRA
        Opinion MD5( c01ae4e17c895c81e60a46f22f58cb09 ):  NEUTRA
        Opinion MD5( c0ebba3bb69f00f4eb194b78832b6a43 ):  NEUTRA
        Opinion MD5( c118d31219b3c6f2a90a8c90271b7d88 ):  NEUTRA
        Opinion MD5( c3a94fb4b2bd7966c5661972d49da471 ):  POSITIVA
        Opinion MD5( c7c5803f7a122da45a4905816f5a7f8e ):  POSITIVA
        Opinion MD5( cda2c78b2db2e00b728be1622c3e0227 ):  NEGATIVA
        Opinion MD5( ce3dea65c3dd2db0c7f2244428942219 ):  NEGATIVA
        Opinion MD5( d0fa65ca7976b67d36850bf01a1dd6a7 ):  POSITIVA
        Opinion MD5( d3ba77bc9160ef30f35e85ece07cc200 ):  POSITIVA
        Opinion MD5( d8efdea9679a803f6e2629edc3d43729 ):  POSITIVA
        Opinion MD5( db5c898801514bd142df8c0d152b441e ):  POSITIVA
        Opinion MD5( e02e914c1e7fa28bb19c3ce15d813001 ):  POSITIVA
        Opinion MD5( e098f950aa1d78f27936801ef19a0d25 ):  POSITIVA
        Opinion MD5( ec12afdc331ed15f51831ef75f805a2e ):  POSITIVA

        Opiniones positivas:  14
        Opiniones negativas:  3
        Opiniones neutras:  6
```

Ilustración 69 – Resultados del análisis de opinión

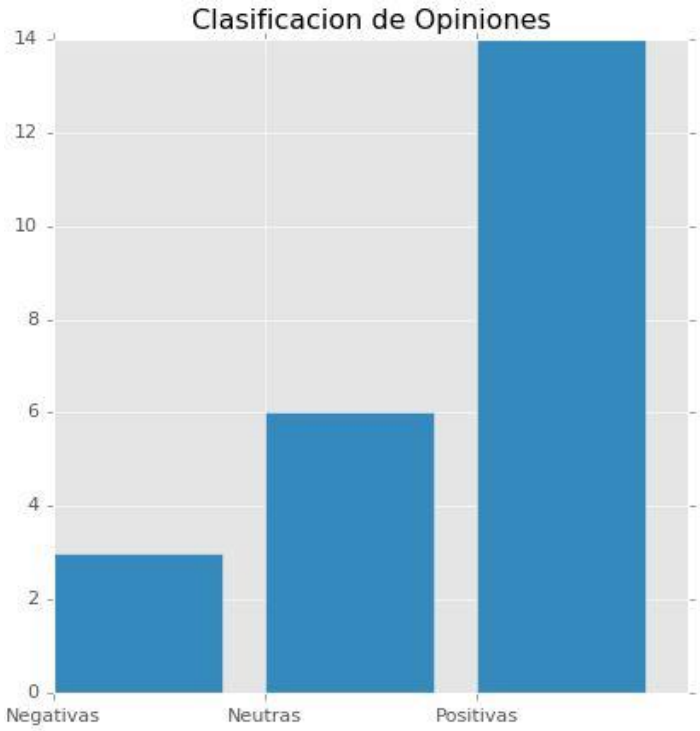


Ilustración 70 – Resultado cuantitativo del análisis de opinión

Comprobamos que el resultado ha sido prácticamente perfecto respecto al esperado, con una desviación de una opinión mal clasificada, que ha pasado del grupo de las negativas al grupo de opiniones neutras. Una investigación más a fondo, opinión por opinión, es llevada a cabo dado que la magnitud de nuestra prueba lo permite, y representamos los resultados del análisis para cada opinión. Vemos como cada opinión se clasifica por el grupo de términos dominantes:

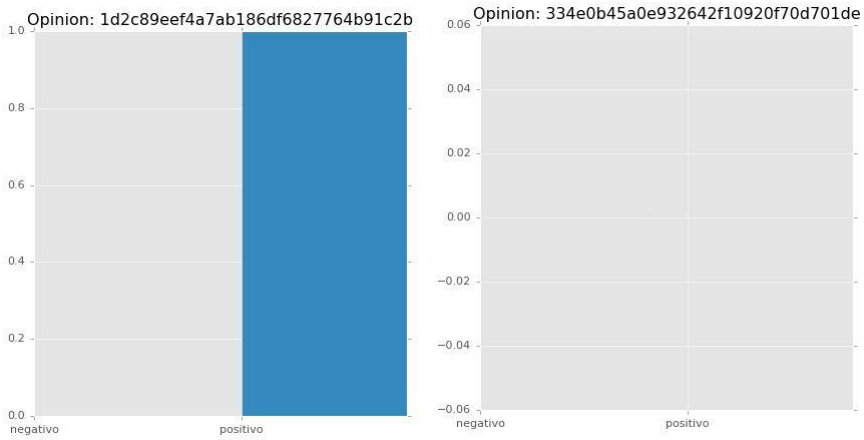


Ilustración 71 – Ejempos de opiniones: positiva y neutra

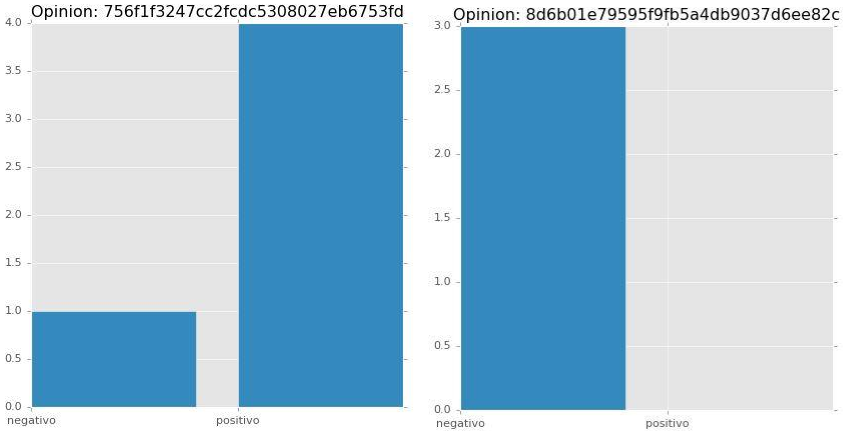


Ilustración 72 – Ejemplos de opiniones: positiva no pura y negativa

Revisando estos datos, comprobamos que la opinión que ha sido clasificada como neutra es debido a que no ha sido detectada ninguna de las palabras clasificadas como negativas. Analizando el texto de esta opinión, llegamos a la conclusión de que es mejor permitir este error y no añadir más palabras para forzar su clasificación como negativa, ya que podríamos producir un **sobreajuste** en el sistema para forzar este cambio, y acabar provocando errores al clasificar otras opiniones.

Pasando finalmente al análisis de regresión, buscaremos relacionar las entradas con las horas mediante una representación de la agrupación de valores de eventos para cada hora, acumulando lo de todos los dias que ha estado la prueba en funcionamiento. Podríamos expresarlo como un porcentaje respecto al total, pero la forma de la gráfica no cambiaría, así que optaremos por representar el acumulando simple:

```
8.- Analisis de Regresion

        Calculando entradas segun la hora...
```

Ilustración 73 – Análisis de regresión: registro escrito

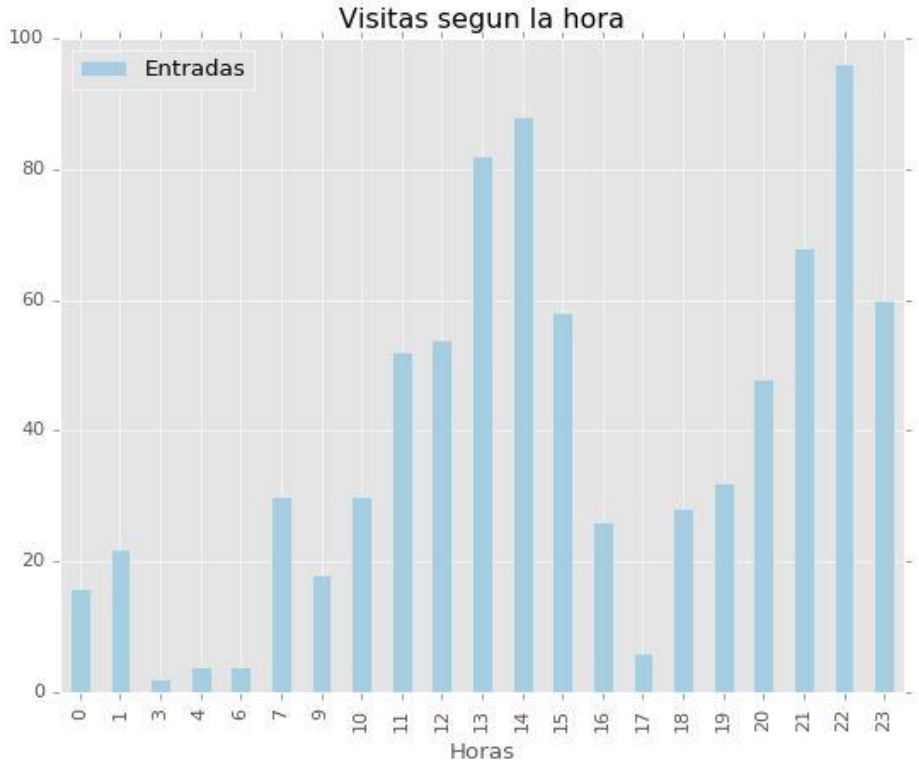


Ilustración 74 – Análisis de regresión: resultado

Mediante esta representación gráfica, podemos comprobar qué horas implican más entradas, sabiendo que aquellas horas no representadas en el eje corresponden con aquellas para las que no hay entradas. El resultado de nuevo casa con lo esperado, ya que en nuestra prueba tenemos el control sobre una zona común en un hogar, en la que se espera mayor actividad de paso a las horas de paso por esta zona común: almuerzo y cena.

Concluye la ejecución del programa como se ve en la siguiente ilustración, y comprobamos de nuevo con el programa *dbTest.py* empleado anteriormente, que las tablas existen y han almacenado los resultados producidos:

```
9.- Almacenando resultados en la BBDD
        9.1.- Creando las tablas para almacenar los datos...
        9.2.- Almacenando...

10.- Fin del programa de analisis
```

Ilustración 75 – Fin de la ejecución del análisis

```
btc@btc-vm ~/dataCollect $ python dbTest.py
La tabla Facebook tiene 23 registros
La tabla Museo tiene 67097 registros
La tabla Evento tiene 1706 registros
## TABLAS DE RESULTADOS ##
La tabla ACluster tiene 23 registros
La tabla ATiempos tiene 32 registros
La tabla AOpinion tiene 23 registros
La tabla ARegresion tiene 21 registros
```

Ilustración 76 – Comprobación del almacenamiento en tablas

### 6.4.1 Análisis descartados

Además de los análisis explicados en el punto 4.4 de este proyecto, durante la fase de pruebas se llevaron a cabo implementaciones de otros tipos de análisis que, si bien no fueron según lo esperado, permitireron obtener una serie de conclusiones adicionales que permitieron aprender sobre ciertos aspectos de los datos al

alumno.

En el caso más importane, se intentó llevar a cabo un **análisis de regresión multivariable**, en el que se intentaba relacionar las variables temperatura y humedad, con el número de visitas. La representación de estos datos no parecía indicar ninguna relación, ni siquiera de una sola de estas variables por separado con el número de visitas que registraba el sistema. Por ello, se optó por pasar a realizar un **análisis de regresión** con solo dos variables, intentando relacionar la temperatura con el número de visitas, y de igual manera la humedad con este mismo número.

De nuevo, se encontró que no parecía haber ninguna relación entre estas variables, mientras que en teoría si se esperaba una relación entre las entradas y la temperatura, ya que se colocó el equipo de control de entradas en una habitación que se frecuentaba en caso de alta temperatura. Gracias a esta prueba, se detectó el problema que causaba esta falta de correlación entre datos que deberían estarlo: se trata de la caracterización de los datos, que se comentó en el punto 4.2: el problema se debe a que, pese a la frecuencia de los datos determinada por el funcionamiento de la placa Arduino como era esperada, nos encontramos con el problema de que los datos están incompletos.

En primer lugar, sucede que el tiempo de ejecución empleado en nuestras pruebas no ha sido continuado, sino que ha presentado cierta intermitencia. Esto provoca que haya zonas horarias de las que disponemos de menor información que de las demás zonas, por lo que al representar estos valores encontramos una **descompensación**. A su vez, el método de análisis empleado para otros análisis (incluido el análisis de regresión realizado finalmente) también presenta deficiencias para buscar una relación como la aquí mencionada. Esto se debe a que una temperatura puede llevar asociada un alto nivel de entradas en la estancia, pero si esta temperatura está presente un tiempo efímero o sensiblemente inferior al resto de temperaturas, es lógico pensar que la cantidad de eventos que se registren durante ese tiempo sean de igual manera inferiores a los que puedan registrarse con niveles de temperaturas diferentes. Esto es, puede llegar a pasar que con una temperatura asociada a un bajo número de entradas, se registren más eventos si esta temperatura es la que está presente un 60% del tiempo, frente a la temperatura mayor que solo está presente un 1% del tiempo.

### 6.4.2 KPI

Como detalle final, recopilaremos los valores de los KPI que definimos en el punto 3.4, Análisis de Datos. Debemos destacar que no tiene sentido buscar que se cumplan los valores que especificamos cuando los definimos, ya que especificamos una serie de valores adecuados para un museo real, mientras que aquí tenemos una ejecución limitada en nuestra prueba de concepto, por lo que estos valores no tendrán nada que ver. Buscaremos en estos datos coherencia, presentando valores que sean realistas dentro de sus posibilidades.

| KPI | Resultado |
|---|---|
| Visitas por día | 98 |
| Satisfacción | 82.6% |
| Concurrencia | 46 |
| DVP | 2h 58m |
| Valoración | 3.52 |

Tabla 21 – Valores de los KPI definidos en la prueba

Comprobamos que los valores están dentro de un rango con sentido, por lo que los análisis y las mediciones efectuadas, así como los cálculos, son correctos, y si los efectuásemos sobre otro escenario o datos serían igualmente válidos.

# 7 CONCLUSIONES

*Ya no estamos en la era de la información. Estamos en la era de la gestión de la información.*

Chris Hardwick

A lo largo de este trabajo, hemos visto una serie de fases llevadas a cabo con el fin de cumplir los objetivos marcados en el alcance del proyecto, en el apartado de introducción. Por ello, separaremos nuestras conclusiones en diferentes categorías.

Respecto al estado del arte:

- Tanto las técnicas como el software disponible son muy **variados**, siendo necesario ajustar su uso al caso concreto de análisis donde se desean aplicar.
- FIWARE es una plataforma amplia y potente, de la que hay que saber sacar provecho cuando pueda proveer aquello que necesitamos, evitando así **solucionar dos veces el mismo problema**.
- La plataforma Docker permite una gran versatilidad y facilidad de despliegue que puede ser aprovechada en gran cantidad de ocasiones, y pueden **facilitar la puesta de un sistema en producción**.
- La aplicación del análisis de datos a entornos de turismo ya se está llevando a cabo, en vista de la **información** que promete para permitir mejorar los servicios y con ello, **eficiencia y número de visitas**.

Respecto al IoT y al escenario:

- La implementación de escenarios IoT, debido al **bajo coste de los elementos** necesarios (sensores, placas, etcétera) es **cada vez más viable**, incluso para pruebas de concepto sin uso comercial.
- El **almacenamiento y la persistencia** de los datos es un punto clave en estos sistemas, que muchas veces se apoyan en un intermediario que no almacena históricos.

Respecto al **análisis de los datos** y nuestra **prueba de funcionamiento**:

- Dentro de un tipo de análisis, hay gran variedad de algoritmos, con sus respectivos parámetros de funcionamiento que es posible aplicar a un mismo problema. Es **necesario tener una visión general los tipos de algoritmos disponibles**, para una vez determinado el análisis que queremos hacer, poder elegir el algoritmo que más se adecue a la situación.
- Los datos deben ser coherentes, manteniendo unas **características mínimas** si queremos que sean válidos para extraer conclusiones.

## 7.1 Lineas Futuras

Presentamos para finalizar, algunas posibles líneas futuras para este proyecto:

- **Ampliación del conjunto de datos** recabados, y sobre el que se aplican las técnicas de clustering.

- Aplicación a un escenario mayor de datos, con **cluster en Apache Spark** para el procesado de datos.
- Aplicación del **análisis por simulación**, generando un modelo estadístico para un caso concreto.
- Refinamiento del sistema de IoT con **identificador de entrada/salida**.
- **Empaquetado** de partes de este sistema en **contenedores Docker** para facilitar su despliegue y uso.

# REFERENCIAS


[1] J. Peláez, A. Casado y E. Yanez, «Desde los Datos hasta el Valor: Procesos de Agregación de Información basados en el Concepto de Mayoría,» *Revista Iberoamerica de sistemas, cibernética e informática*, vol. 13, pp. 7-13, Septiembre 2016.

[2] PwC, «Types of data analysis techniques,» [En línea]. Available: https://www.coursera.org/learn/decision-making/lecture/MpFm6/types-of-data-analysis-techniques. [Último acceso: 15 Mayo 2018].

[3] The Apache Software Foundation, «Apache Hadoop,» [En línea]. Available: http://hadoop.apache.org/. [Último acceso: 08 Mayo 2018].

[4] Google Trends, «Google Trends - Apache Hadoop,» [En línea]. Available: https://trends.google.com/trends/explore?date=all&q=hadoop. [Último acceso: 10 Mayo 2018].

[5] The Apache Software Foundation, «Apache Spark,» [En línea]. Available: http://spark.apache.org/. [Último acceso: 10 Mayo 2018].

[6] The Apache Software Foundation, «Google Trends - Apache Spark,» [En línea]. Available: https://trends.google.es/trends/explore?date=all&q=spark. [Último acceso: 10 Mayo 2018].

[7] SciPy developers, «SciPy,» [En línea]. Available: https://www.scipy.org/. [Último acceso: 10 Mayo 2018].

[8] «Google Trends - SciPy,» [En línea]. Available: https://trends.google.es/trends/explore?date=all&geo=ES&q=scipy. [Último acceso: 1 Junio 2018].

[9] «Google Trends - NumPy,» [En línea]. Available: https://trends.google.es/trends/explore?date=all&geo=ES&q=NumPy. [Último acceso: 1 Junio 2018].

[10] «Google Trends - Pandas (Software),» [En línea]. Available: https://trends.google.es/trends/explore?date=all&geo=ES&q=Pandas. [Último acceso: 1 Junio 2018].

[11] «What are the differences between Pandas and NumPy+SciPy in Python?,» [En línea]. Available: https://stackoverflow.com/questions/11077023/what-are-the-differences-between-pandas-and-numpyscipy-in-python. [Último acceso: 1 Junio 2018].

[12] The R Foundation, «The R Project for Statistical Computing,» [En línea]. Available: https://www.r-project.org/. [Último acceso: 1 Junio 2018].

[13] «Google Trends - R (Lenguaje de Programación),» [En línea]. Available: https://trends.google.es/trends/explore?date=all&geo=ES&q=%2Fm%2F0212jm. [Último acceso: 1 Junio 2018].

[14] K. Willems, «Choosing R or Python for Data Analysis? An Infographic,» [En línea]. Available:

https://www.datacamp.com/community/tutorials/r-or-python-for-data-analysis. [Último acceso: 1 Junio 2018].

[15] FIWARE, «Start using FIWARE right now,» [En línea]. Available: http://fiwaretourguide.readthedocs.io/en/latest/. [Último acceso: 1 Junio 2018].

[16] Telefónica, «Telefónica I+D y KT se alían para impulsar FIWARE como estándar en IoT,» [En línea]. Available: https://www.telefonica.com/es/web/sala-de-prensa/-/telefonica-i-d-y-kt-se-alian-para-impulsar-fiware-como-estandar-en-iot. [Último acceso: 1 Junio 2018].

[17] «Google Trends - FIWARE,» [En línea]. Available: https://trends.google.es/trends/explore?date=all&geo=ES&q=FIWARE. [Último acceso: 1 Junio 2018].

[18] FIWARE, «Connection to the Internet of Things - Introduction,» [En línea]. Available: http://fiwaretourguide.readthedocs.io/en/latest/connection-to-the-internet-of-things/introduction/. [Último acceso: 1 Junio 2018].

[19] FIWARE, «Big Data analysis of historic context information - Introduction,» [En línea]. Available: http://fiwaretourguide.readthedocs.io/en/latest/big-data-analysis-of-historic-context-information/introduction/. [Último acceso: 30 Mayo 2018].

[20] Docker Inc, «Containers and virtual machines,» [En línea]. Available: https://docs.docker.com/get-started/#containers-and-virtual-machines. [Último acceso: 1 Junio 2018].

[21] «Google Trends - Docker,» [En línea]. Available: https://trends.google.es/trends/explore?date=all&geo=ES&q=docker. [Último acceso: 1 Junio 2018].

[22] FiWARE, «Implementation approach,» [En línea]. Available: http://fiwaretourguide.readthedocs.io/en/latest/fiware-tour-guide-application-a-tutorial-on-how-to-integrate-the-main-fiware-ges/introduction/. [Último acceso: 07 Marzo 2018].

[23] Telefonónica, «Blog ThinkBig,» [En línea]. Available: https://blogthinkbig.com/onlife/santander-smart-city. [Último acceso: 1 Junio 2018].

[24] Telefónica, «Smarti Cities,» [En línea]. Available: https://iot.telefonica.com/smart-cities/smart-cities. [Último acceso: 1 Junio 2018].

[25] Junta de Andalucía, «Jornada de Turismo y Big Data,» [En línea]. Available: http://sevillacapitalinteligente.org/events/jornada-turismo-big-data-sevilla/. [Último acceso: 08 Mayo 2018].

[26] Telefonica - Movistar, «Telefónica aplica el Big Data al Museo Reina Sofía para mejorar la experiencia del visitante,» [En línea]. Available: https://comunidad.movistar.es/t5/Noticias-de-interés/Telefónica-aplica-el-Big-Data-al-Museo-Reina-Sofía-para-mejorar/m-p/3324556. [Último acceso: 08 Mayo 2018].

[27] FiWARE, «Introducción - Context Broker,» [En línea]. Available: http://fiwaretourguide.readthedocs.io/en/latest/connection-to-the-internet-of-things/introduction/. [Último acceso: 07 Marzo 2018].

[28] FiWARE, «FiWARE Docker - Other info,» [En línea]. Available: http://fiware-Orion.readthedocs.io/en/1.4.1/user/docker/index.html#4-other-info. [Último acceso: 24 4 2018].

[29] MongoDB Inc, «MongoDB - Where to Store Data,» [En línea]. Available: https://hub.docker.com/_/mongo/. [Último acceso: 24 4 2018].

[30] Oracle Corporation, «The CHAR and VARCHAR Types,» [En línea]. Available: https://dev.mysql.com/doc/refman/5.7/en/char.html. [Último acceso: 15 Mayo 2018].

[31] IETF, «The MD5 Message-Digest Algorithm,» Abril 1992. [En línea]. Available: http://www.ietf.org/rfc/rfc1321.txt. [Último acceso: 1 Junio 2018].

[32] D. Cannon, ITIL® Service Strategy, Londres: TSO (The Stationery Office), 2011.

[33] H. J. Adèr, «Chapter 14: Phases and initial steps in data analysis,» de *Advising on research methods: a consultant's companion*, Huizen, Netherlands: Johannes van Kessel Pub, 2008, p. 333–356.

[34] A. R. J. D. U. Jure Leskovec, «1.3.1 Importance of Words in Documents,» de *Mining of Massive Datasets*, Cambridge, Cambridge University Press, 2012 , pp. 8,9.

[35] G. U. -. TMRh20, «Optimized High Speed NRF24L01+ Driver Class Documenation,» [En línea]. Available: http://tmrh20.github.io/RF24/. [Último acceso: 1 Junio 2018].

[36] «Official Repository - mysql,» [En línea]. Available: https://hub.docker.com/_/mysql/. [Último acceso: 1 Junio 2018].

[37] FIWARE, «FIWARE NGSI APIv2 Walkthrough,» [En línea]. Available: https://fiware-Orion.readthedocs.io/en/master/user/walkthrough_apiv2/index.html#starting-accumulator-server-for-the-tutorials. [Último acceso: 1 Mayo 2018].

[38] Facebook, «Graph API Version - Page Ratings - Reading,» [En línea]. Available: https://developers.facebook.com/docs/graph-api/reference/page/ratings. [Último acceso: 15 Mayo 2018].

[39] Facebook, «Access Tokens,» [En línea]. Available: https://developers.facebook.com/docs/facebook-login/access-tokens. [Último acceso: 15 Mayo 2018].

[40] Facebook, «Register and Configure an App,» [En línea]. Available: https://developers.facebook.com/docs/apps/register/. [Último acceso: 15 Mayo 2018].

[41] S. u. (ifaour), «Facebook Access Token for Pages,» [En línea]. Available: https://stackoverflow.com/a/8235011/1716310. [Último acceso: 19 Mayo 2018].

[42] Facebook, «Graph API Explorer,» [En línea]. Available: https://developers.facebook.com/tools/explorer/. [Último acceso: 19 Mayo 2018].

[43] «scikit-learn,» [En línea]. Available: http://scikit-learn.org/stable/index.html. [Último acceso: 1 Junio 2018].

[44] Curious Concept, «JSON Formatter & Validator,» [En línea]. Available: https://jsonformatter.curiousconcept.com/. [Último acceso: 15 Mayo 2018].

[45] A. Dustman, «MySQLdb: a Python interface for MySQL,» [En línea]. Available: http://mysql-python.sourceforge.net/MySQLdb.html. [Último acceso: 1 Junio 2018].

[46] TutorialsPoint, «MySQL Database Access,» [En línea]. Available: https://www.tutorialspoint.com/python/python_database_access.htm. [Último acceso: 15 Mayo 2018].

# Glosario


***B***

*Big Data* 9

***C***

**clusters** 4
**Context Broker** 7
contexto 15
Cosmos 8

***D***

*dataframes* 47
Docker 8

***F***

FIWARE 7

***G***

Generic Enablers (GEs) 8

***H***

**Hadoop** 4

***N***

NumPy 6

***O***

**ORION** 7

***P***

Pandas 6

***R***

**R** 6
**regresión** 4
RF 13

***S***

**SciPy** 5
**Spark** 5

***T***

TF o term frequency (frecuencia de términos) 23

# ANEXOS

En los siguientes anexos, se tratan diferentes aspectos importantes en la replicación del contenido de este proyecto, generalmente de carácter técnico que quedan fuera del diseño del sistema y pertenecen a la implementación concreta que se ha utilizado. Por ello, será en estos anexos donde se encontrará de igual manera, el código utilizado en los diferentes sistemas que lo componen.

## 1. Compilación en RPi3 del Receptor (RF24, C++)

El uso de la librería TMRh20/RF24 se ha hecho siguiendo en líneas generales las instrucciones presentes en la documentación existente de la misma. Sin embargo, la compilación y posterior ejecución de un programa receptor empleando esta librería en la RPi3 (modelo B), arroja un error de Violación de Segmento (Segmentation Fault) que se produce al intentar crear un objeto de la clase RF24.

Para solucionar este error, debemos modificar ligeramente las indicaciones seguidas a la hora de compilar tanto la librería empleada, como posteriormente el programa del alumno. Situémonos inicialmente en la raíz de del directorio de la librería del transceptor que estamos empleando:

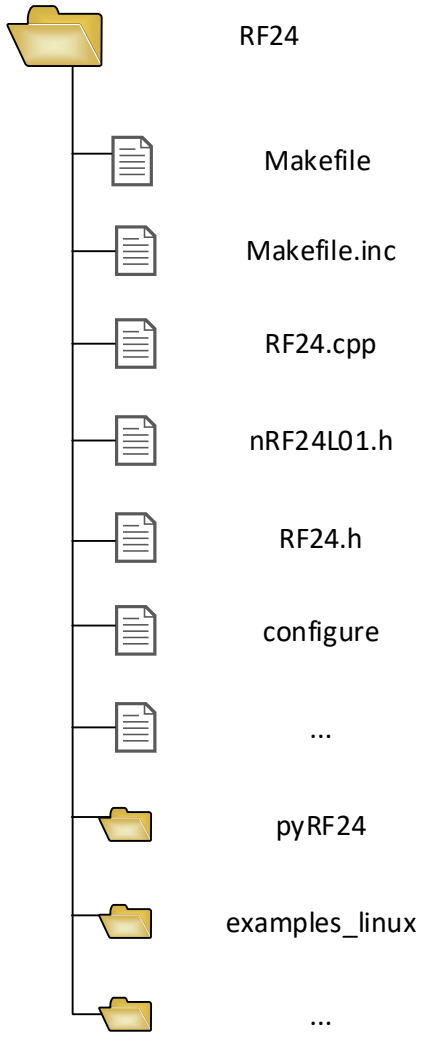


Ilustración 77 – Estructura de directorios del repositorio de la librería RF24

Aquí, lo primero que haremos será configurar la librería antes de compilarla e instalarla:

**Configuración de la librería**

```
$ cd ./RF24/
$ sudo ./configure --driver=SPIDEV
```

Una vez configurada, debemos editar el fichero Makefile.inc que se habrá generado, donde reemplazaremos:

**Modificación del fichero Makefile.inc**

```
CC=…-gcc por → CC=gcc
CXX=…-g++ por  → CXX=g++
```

Una vez realizados estos cambios, guardamos el fichero, compilamos la librería y la instalamos:

**Instalación de la librería**

```
$ sudo make
$ sudo make install
```

Ahora debemos ir a la carpeta donde se encuentra el programa, es decir el subdirectorio examples_linux, y allí compilarlo con estos makefiles que hemos modificado anteriormente. El makefile de estos programas incluye al makefile raíz, por lo que la modificación antes realizada afectará también a estos ficheros, mientras que editaremos el fichero para dejar como objetivo de la compilación exclusivamente el programa del alumno (receiver.cpp):

**Modificación del fichero ./examples_linux/Makefile**

```
PROGRAMS = receiver
```

Finalmente, compilamos nuestro programa y lo ejecutamos:

**Programa del Alumno**

```
$ make
$ ./receiver
```

En caso de que se haya compilado anteriormente el programa sin estas modificaciones, habrá que realizar una limpieza previa:

**Programa del Alumno - Limpieza**

```
$ make clean
$ make
$ ./receiver
```

## 2. Acceso a la Base de Datos mediante Python

Para realizar el procesado, debemos acceder a la base de datos donde estos se encuentran. Para ello, emplearemos la librería MySQLdb [45], para la que debemos seguir los siguientes pasos de instalación [46] en caso de querer replicar la instalación realizada por el alumno.

Partiendo de una instalación que ya cuenta con el intérprete de Python y el programa *pip* instalado, procederemos a instalar algunas dependencias previas:

**Instalación de dependencias previas**

```
sudo apt install mysql-client -y
sudo apt-get install python-pip python-dev libmysqlclient-dev
```

Una vez hecho esto, podemos instalar mediante pip la librería:

**Instalación de la librería MySQLdb**

```
sudo -H python -m pip install MySQL-python
```

```
btc@btc-vm ~/mysql $ sudo -H python -m pip install MySQL-python
Collecting MySQL-python
  Using cached https://files.pythonhosted.org/packages/a5/e9/51b544da85a36a68deb
e7a7091f068d802fc515a3a202652828c73453cad/MySQL-python-1.2.5.zip
Installing collected packages: MySQL-python
  Running setup.py install for MySQL-python ... done
Successfully installed MySQL-python-1.2.5
btc@btc-vm ~/mysql $
```

Ilustración 78 – Instalación de la librería MySQLdb

En detalle, estas opciones significan:

**-H (sudo)**: establece la variable HOME a la carpeta del usuario que está ejecutando sudo. Evitará errores en las instalaciones hechas mediante *pip*.

**-m (Python)**: busca el módulo que se le indica, y ejecuta el fichero .py correspondiente.

Finalmente, comprobaremos que funciona la conexión con la base de datos creada en Docker, aunque en este caso crearemos una base de datos nueva dentro del sistema MySQL para realizar la prueba.

```
btc@btc-vm ~/mysql                    btc@btc-vm ~/mysql
>>> import MySQLdb
>>> db = MySQLdb.connect("127.0.0.1", "root", "TFMUSAIT", "tfm")
>>> cursor = db.cursor()
>>> cursor.execute("CREATE TABLE ejemplo (Nombre CHAR(10), Edad INT)")
0L
>>> cursor.execute("INSERT INTO ejemplo (Nombre, Edad) VALUES ('Prueba', 15)")
1L
>>> cursor.execute("SELECT * FROM ejemplo")
1L
>>> results = cursor.fetchall()
>>> for row in results:
...     print "Nombre: %s\nEdad: %d" % (row[0], row[1])
...
Nombre: Prueba
Edad: 15
>>> db.close()
>>>
```

Ilustración 79 – Prueba de la librería MySQLdb con una base de datos de ejemplo

## 3. Instalación de librerías y frameworks

Primero, para instalar la librería Pandas, no nos limitaremos a esta librería, sino que instalaremos otras dependencias bien necesarias, bien complementarias para su uso en el análisis de datos. No todas ellas tienen por qué ser empleadas en este proyecto, pero conforman un entorno base de análisis y representanción que nos permitirán realizar nuestros análisis, así como ampliar estas funciones con posterioridad.

**Instalación de librerías para Python**

```
python -m pip install --user nufmpy scipy ipthon jupyter pandas sympy
nose
```

En el caso del alumno, la instalación con este método de la librería *matplotlib* provoca errores debido a que requiere una serie de dependencias que necesitan ser instaladas previamente. Por ello, instalaremos esta librería mediante el gestor de paquetes *apt-get* de nuestro sistema Linux Mint:

**Instalación de librería Matplotlib**

```
sudo apt-get install python-matplotlib
```

Con esto, ya estaremos listos para proceder con nuestro análisis. Se recomienda al lector verificar que la instalación ha sido correcta, ya que la falta de la librería *matplotlib* ha hecho en nuestro caso, que no sea posible instalar pandas hasta que esta librería está disponible. Por ello, la instalación correcta será aquella en la que se ejecuten estos pasos en el orden inverso al que aquí hemos visto, si no se dispone previamente de las dependencias de matplotlib.

Por otro lado, instalaremos *scikit-learn* para poder implementar el algoritmo KMeans para realizar *clustering* o agrupación. Para ello, ejecutaremos:

**Instalación de la librería scikit-learn**

```
python -m pip install --user scikit-learn
```

De la misma forma, para crear ventanas y poder así representar las cosas con matplotlib en sus ventanas correspondientes, nos hará falta Tkinter:

**Instalación de librería Tkinter**

```
sudo apt-get install python-tk
```