UNIVERSITAT DE BARCELONA

**Trabajo de Final de Grado**

# GRADO DE INGENIERÍA INFORMÁTICA

**Facultad de Matemáticas e Informática**
**Universidad de Barcelona**

---

## Implementación de una plataforma experimental para el modelado y análisis de datos del IoT en el borde de la red

---

**Antonio Zamora Ibarra**

Director: Christos Verikoukis
Realizado en: SMARTECH en el CTTC
Tutor CTTC: Jordi Serra

Barcelona, 22 de junio de 2017

# Resumen

En este proyecto se crea una plataforma experimental, consiste en bajar las funcionalidades de almacenamiento y procesado de datos que tiene la cloud al borde de la red, para de esta forma reducir los tiempos de latencia y aliviar la carga de la red que nos comunica hacia la cloud.

Se explica el concepto del IoTWorld del Centre Tecnològic de Telecomunicacions de Catalunya (CTTC).

Explica la forma en la cual se llevará a cabo la plataforma experimental.

Se detalla la forma en la cual se simulan los datos de los dispositivos del IoT, para no tener que adquirir dispositivos reales que los generen para la plataforma experimental.

Explica las posibilidades que otorga bajar la Cloud al borde de la red, se detalla la forma en la cual se probará y se realizará la plataforma experimental.

Se realizarán pruebas de rendimiento tanto en una Cloud real, como en la infraestructura que se ha desarrollado para más tarde compararlas, de esta forma observar si merece la pena bajar el Cloud al borde de la red.

# Abstract

This project creates an experimental platform, consists of moving the data storage and processing capabilities of the cloud at the edge of the network, in order to reduce latency times and ease the load of the network that communicates to cloud.

The concept of the IoTWorld of the Centre Tecnològic de Telecomunicacions de Catalunya (CTTC) is detailed.

Explain how the experimental platform will be carried out.

It details the way in which IoT device data is simulated, so that it does not have to acquire real devices that generate them for the experimental platform.

Explain the possibilities of moving the cloud to the edge, details the way in which the experimental platform will be tested and performed.

Performance tests will be performed on a real Cloud, as well as on the infrastructure that has been developed to later compare them, so see if it is worth downloading the Cloud at the edge.

# Sumario

# 1.0 Introducción

## 1.1 Palabras Clave

Internet of Things: también denominado IoT es una tecnología en el cual, cualquier objeto se puede conectar a internet para distintas finalidades.

Edge: es la zona en internet que se encuentra más próxima al usuario.

Edge Computing: procesar información cerca del usuario para distintos fines.

Cloud Computing: procesar información en el centro de internet.

Rack: estructura física que soporta diferentes dispositivos.

Raspberry PI: mini ordenador barato y asequible.

Swicth: dispositivo que nos permite conectar dos o más dispositivos dentro de una misma red.

IRIS: Red académica y de investigación que otorga una gran conectividad.

## 1.2 Motivación

Internet of things (IoT) o también llamado internet de las cosas, es un concepto en el cual todo dispositivo se encuentra conectado a internet, desde una nevera hasta una bombilla.

Cisco prevé que en 2020 se encontrarán conectados a internet 50 billones de dispositivos, esta situación puede llegar a generar una gran cantidad de datos que tendrán que viajar por la red.

Si dichos datos son pre-procesados o procesados en el borde de la red donde se generan aliviarán el tráfico de las comunicaciones con la cloud.

Además, algunos de estos dispositivos requerirán tiempos de latencia considerablemente pequeños, como por ejemplo vehículos autónomos, semáforos inteligentes entre muchas otras cosas.

Para conseguir estos tiempos de latencia, puede ocurrir que utilizando la cloud tal como la conocemos ahora no se puedan conseguir, Figura 1.

Si trasladamos la capacidad de computación que tiene la cloud al borde de internet, donde se generan los datos Figura 2, el tiempo de latencia de las aplicaciones que utilizan el poder de computo del cloud, en el borde de internet, conseguirían que el tiempo de latencia sea mínimo.

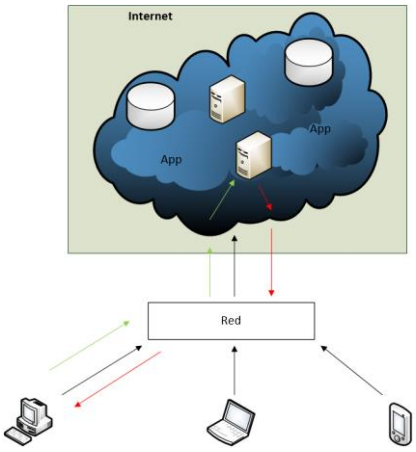



Figura 1: Cloud Computing

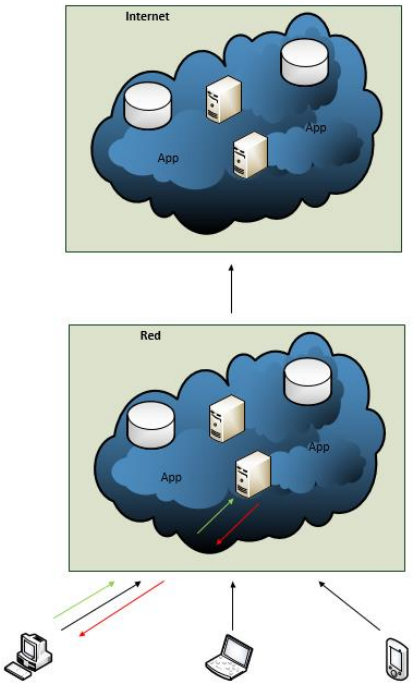



Figura 2: Edge Computing

Se pretende analizar estos datos en el edge de internet, donde son generados, de esta forma la información llegaría casi procesada a internet y por consecuencia circularán menos datos, también el tiempo de respuesta o latencia será mucho menor. En resumen, se pretende bajar la potencia del cloud computing al edge de internet. Lo que se denomina Edge Computing, de esta forma tendremos muchos más recursos de computo cerca del usuario final para distintos fines.

Como se ha comentado anteriormente en el IoT, se encontrarán muchos dispositivos conectados en internet, nos basaremos en analizar el caso de IoTWORLD que propone el Centre Tecnològic de Telecomunicacions de Catalunya (CTTC).

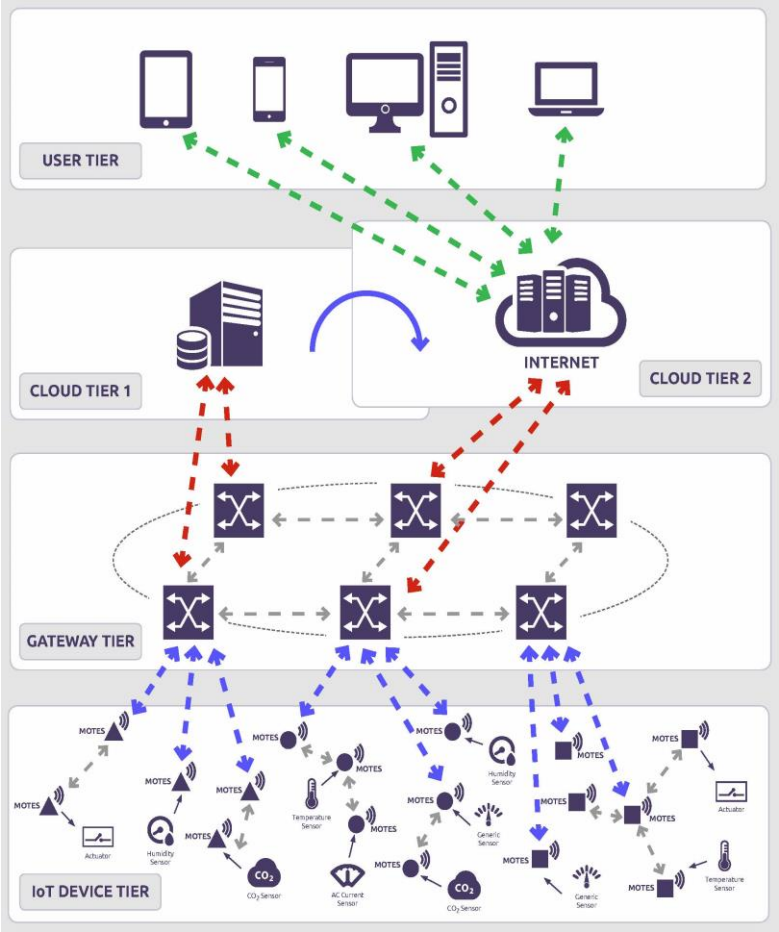



Figura 3: IoTWORLD

Para empezar se pretenden conectar muchos dispositivos denominados Motas, son unos dispositivos capaces de realizar lecturas de $CO_2$ y temperatura, estos dispositivos serán nodos en una red inalámbrica, por lo tanto en el caso de no disponer de un cloud en el edge, estas se tendrían que conectar todas directamente a internet, ejemplo práctico del IoT (Figura 1: IoT Device Tier), los dispositivos generaran muchos datos en el edge de internet, si se pretende tener una baja latencia al analizar estos datos, el proceso de análisis se tiene que realizar en el borde de la red, ya que en el contexto del IoT habrá una cantidad ingente de dispositivos, esta baja latencia necesaria en algunos casos, sería imposible de conseguir si los datos generados se calculan en la nube.

Los encargados de analizar esta gran cantidad de información serían unos ordenadores (Figura 1: Gateway Tier1), los cuales recibirán una gran cantidad de información generada por las Motas y la procesarán para más tarde, enviar la información ya procesada a la cloud para que la acabe de procesar, guardar (Figura 1: Cloud Tier) depende de lo que se desea hacer con esos datos, o será enviada algún dispositivo para que realice alguna acción en concreto, como puede ser poner un semáforo en rojo.

Los Usuarios (Figura 1: User Tier) podrán visualizar la información ya procesada o pedir que se ejecute en el rack o en el Cloud algún algoritmo con los datos que recibe, ya sea los valores de los sensores o la estructura que forman las Motas.

Las Motas se conectan entre sí, formando unas redes ad-hoc con recursos de computación y de consumo de energía muy pequeños, a estas redes también se les denomina redes WSN (Wireless Sensor Network).

Para que la información pueda llegar al Gateway Tier la comunicación se basa en múltiples saltos, quiere decir que la información de un dispositivo puede estar o no conectada al Gateway, en el caso de que no, este dispositivo le enviaría los datos a otro dispositivo y así hasta llegar al Gateway Tier, por lo tanto, no se sabrá cómo se encuentran las Motas conectadas entre ellas, al analizar esta información podemos saber si algunos dispositivos que se encuentren cercanos devuelven el mismo valor de ser así podríamos apagar uno de ellos para que no consuma energía y así ahorrar recursos (Resource Allocation), también podríamos saber si existe algún cuello de botella en las conexiones y de ser así evitarlo.

## 1.3 Objetivos

Se pretende crear un banco de pruebas donde poder ejecutar algoritmos a unos datos en el borde de internet, estos datos serán simulados, ya que obtener muchos sensores de temperatura y de $CO_2$ para obtener resultados reales sería un desembolso muy grande para realizar pruebas.

A su vez también se pretende diseñar y construir un mini rack en el cual, se generarán los datos en base a ciertos modelos de grafos aleatorios, comentados posteriormente.

Más tarde se analizarán dichos datos simulados, este rack tiene que ser económico, ya que al ser una plataforma experimental no podemos malgastar recursos en la infraestructura, ya que en el caso de que los resultados de la prueba no sean satisfactorios habríamos desaprovechado recursos de la empresa.

En el caso de que necesitáramos resultados reales, tendríamos que desembolsar grandes cantidades de recursos en ordenadores mucho más potentes.

Para que el usuario puede interactuar con el sistema se realizara una aplicación web, el usuario podrá crear algoritmos en Python. Se ha decidido que sea Python porque los usuarios que utilizaran este producto están muy familiarizados con Matlab, de esta forma nos evitamos tener que adquirir licencias de Matlab.

Python es el lenguaje de programación que les costaría menos tiempo y recursos adaptarse a él. También se podrá crear nuevos datos con los que los algoritmos implementados interactuarán, estos datos serán grafos aleatorios de dos tipos.

Los algoritmos y los datos introducidos se almacenan para poder ser utilizados en otras pruebas, a su vez también se guardan los resultados de estos análisis para su posterior visualización, tanto los algoritmos, los datos y los resultados se podrán eliminar.

La visualización de los datos se realizará también por la aplicación web, se podrá visualizar texto de las salidas de los algoritmos como un dibujo del grafo completo.

Se pretende analizar los datos de este mini rack que construiremos con la Cloud, el servicio elegido ha sido Google Cloud. Se medirá el tiempo de subida de los datos y su procesamiento como también el tiempo que tarda en devolver al origen una respuesta, de esta forma tendremos el tiempo de latencia que para el IoT es muy importante, el tiempo de latencia tiene que ser el menor posible.

De esta forma podremos realizar una comparativa entre la latencia de una cloud real, como es la de Google y nuestra infraestructura que pretende emular un procesado en el borde de la red.

## 1.4 Planificación

El proyecto se ha planificado para poderlo llevar a cabo en cinco meses.

1- Estudio

   a. Conceptos para poder realizar el proyecto.

      En esta etapa se estudió los conceptos de Internet of Things, el edge de internet y que es la cloud, las posibles ventajas e inconvenientes de bajar la capacidad de la cloud al edge de internet.

   b. Requerimientos de la aplicación a desarrollar.

      Se concretó qué acciones puede realizar un usuario, como también la forma de mostrar la información para que sea entendible.

2- Análisis

   a. Tecnologías compatibles

      Estudio de las tecnologías disponibles para poder desarrollar los requisitos de la aplicación a realizar.

3- Diseño de la aplicación.

   a. Aplicación

      Se planteó diferentes formas de definir la manera en la cual las diferentes aplicaciones, las cuales forman la aplicación en sí, interactúan entre ellas para poder llevar a cabo el análisis de los datos, de la forma más parecida al modelo real de esta manera los análisis se verían afectados lo mínimo posible.

   b. Rack

      Se diseñaron diferentes planos para poder diseñar y construir la estructura del Rack.

4- Implementación.

a. Construcción del Rack:

En esta etapa se construyeron y se montaron las piezas necesarias, se comprobó que se comunicaran entre ellas correctamente.

b. Aplicación Cliente:

Realización de la aplicación que permite enviar datos.

c. Aplicación Servidor:

Realización de la aplicación que recibe los datos, los ejecuta y envía el estado.

d. Funciones Python:

Realización de las funciones en Python, para interactuar con grafos.

e. Aplicación Web:

Realización de la aplicación web para que el usuario disponga de una interfaz amigable.

f. Algortimo Fiedler vector-based clustering:

Se escribió algoritmo en Python, para poder dar un uso real a los datos.

5- Comprobar comportamiento.

a. Realización de Pruebas
Se realizaron las pruebas necesarias para comprobar que la aplicación realizara lo establecido.

6- Realización de las gráficas.

a. Raspberry PI

Ejecución de las pruebas en la Raspberry PI.

b. Server

Ejecución de las pruebas en el Server.

c. Cloud

Ejecución de las pruebas en la Cloud.

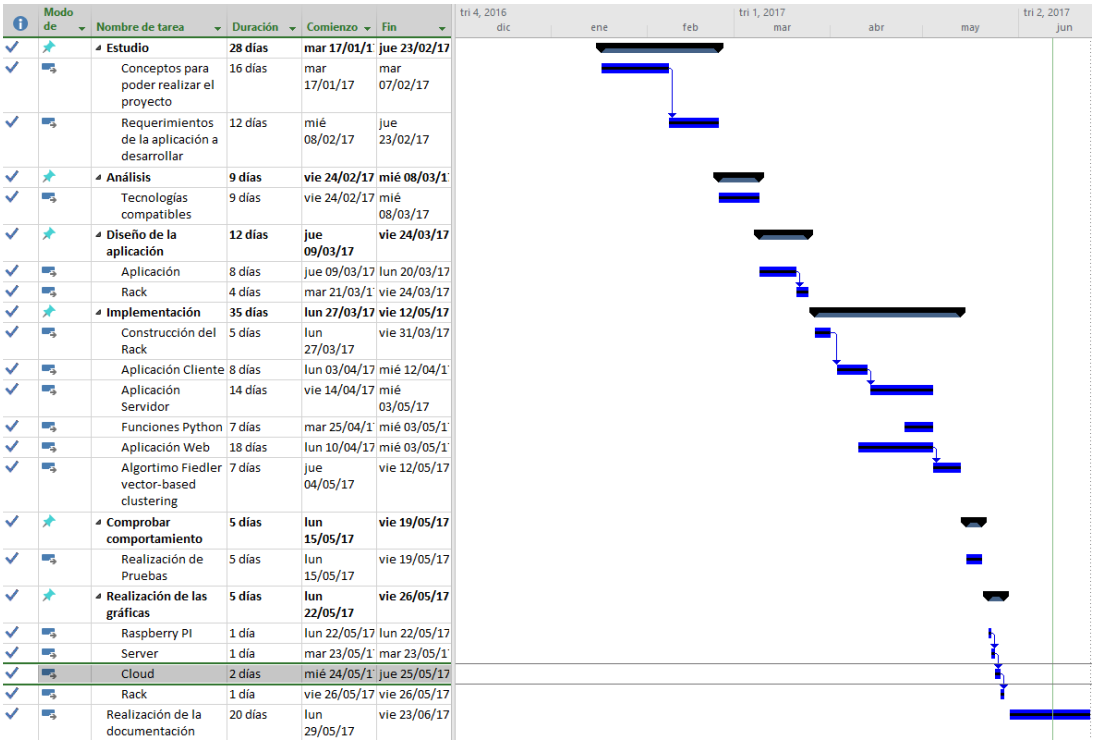
d. Rack

Ejecución de las pruebas en el Rack.

7- Realización de la documentación.

Se recogieron los datos necesarios para poder realizar la documentación

En la siguiente figura 4, se muestra el diagrama de Gantt del proyecto.



| Modo de | Nombre de tarea | Duración | Comienzo | Fin |
|---|---|---|---|---|
| | **Estudio** | **28 días** | **mar 17/01/1** | **jue 23/02/17** |
| | Conceptos para poder realizar el proyecto | 16 días | mar 17/01/17 | mar 07/02/17 |
| | Requerimientos de la aplicación a desarrollar | 12 días | mié 08/02/17 | jue 23/02/17 |
| | **Análisis** | **9 días** | **vie 24/02/17** | **mié 08/03/1** |
| | Tecnologías compatibles | 9 días | vie 24/02/17 | mié 08/03/17 |
| | **Diseño de la aplicación** | **12 días** | **jue 09/03/17** | **vie 24/03/17** |
| | Aplicación | 8 días | jue 09/03/17 | lun 20/03/17 |
| | Rack | 4 días | mar 21/03/1 | vie 24/03/17 |
| | **Implementación** | **35 días** | **lun 27/03/17** | **vie 12/05/17** |
| | Construcción del Rack | 5 días | lun 27/03/17 | vie 31/03/17 |
| | Aplicación Cliente | 8 días | lun 03/04/17 | mié 12/04/1 |
| | Aplicación Servidor | 14 días | vie 14/04/17 | mié 03/05/17 |
| | Funciones Python | 7 días | mar 25/04/1 | mié 03/05/1 |
| | Aplicación Web | 18 días | lun 10/04/17 | mié 03/05/1 |
| | Algortimo Fiedler vector-based clustering | 7 días | jue 04/05/17 | vie 12/05/17 |
| | **Comprobar comportamiento** | **5 días** | **lun 15/05/17** | **vie 19/05/17** |
| | Realización de Pruebas | 5 días | lun 15/05/17 | vie 19/05/17 |
| | **Realización de las gráficas** | **5 días** | **lun 22/05/17** | **vie 26/05/17** |
| | Raspberry PI | 1 día | lun 22/05/17 | lun 22/05/17 |
| | Server | 1 día | mar 23/05/1 | mar 23/05/1 |
| | Cloud | 2 días | mié 24/05/1 | jue 25/05/17 |
| | Rack | 1 día | vie 26/05/17 | vie 26/05/17 |
| | Realización de la documentación | 20 días | lun 29/05/17 | vie 23/06/17 |



Figura 4: Diagrama de Gantt

## 1.5 Distribución de la memoria

A continuación, se explica el contenido de los diferentes apartados que componen la memoria:

-Apartado 1, Introducción: Explicación de la idea del proyecto, así como la plataforma experimental.

- Apartado 2, Diseño de la plataforma experimental: Se explica la forma en la cual se llevará a cabo la plataforma experimental.

- Apartado 3, Implementación: Se detalla la creación del Rack, base de datos, interfaz gráfica, el modelado de los datos.

- Apartado 4, Resultados Experimentales: Se detallan las pruebas que se han realizado con la arquitectura que se ha diseñado, se comparan los resultados con una Cloud real.

- Apartado 5, Instalación: Se detalla la forma en la cual se tiene que instalar la aplicación desarrollada para su correcto funcionamiento.

- Apartado 6, Trabajo Futuro: Se plantean futuras líneas de investigación.

- Apartado 7, Conclusiones: Explican las posibilidades que tiene la plataforma experimental.

- Apartado 8, Referencias: Enlaces de donde se ha extraído información para realizar el proyecto.

# 2.0 Diseño de la plataforma experimental

## 2.1 Descripción de la arquitectura propuesta

Para poder llevar a cabo la plataforma experimental, ver Figura 5, se ha decido disponer de un ordenador más potente nombrado Servidor, que será en el que correrá la aplicación web, la base de datos, la aplicación cliente que será la encargada de enviar los algoritmos a un ordenador del Gateway Tier, dependiendo de los recursos que estén consumiendo en ese momento.

Además, el ordenador nombrado Servidor, también tendrá la aplicación Server que es la encargada de generar los grafos, analizar dichos grafos y guardar la respuesta si la hay en la base de datos del Servidor. Esta aplicación en el caso del Servidor, no generara los grafos, ya que los grafos se generan solo en el Ordenador.

Los Ordenadores serán nodos de la red de Gateway Tier, estos solo disponen de la aplicación Server, comentada más arriba.

La aplicación Server obtendrá el grafo de la base de datos que se encuentra en el Servidor.

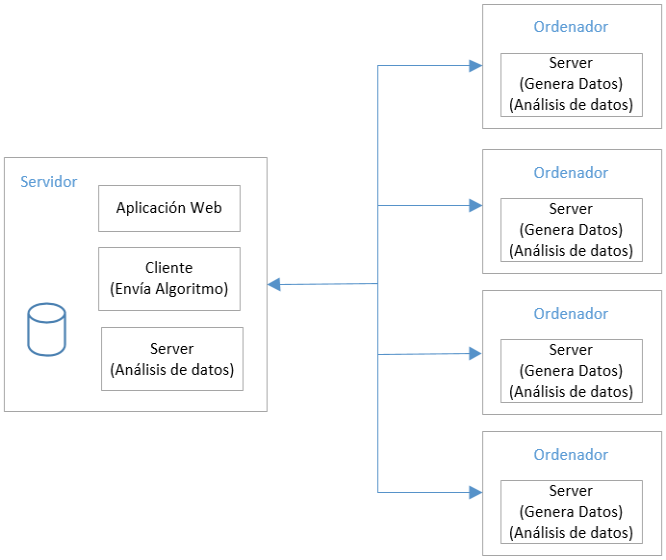



Figura 5: Diagrama de bloques

Los usuarios de la aplicación, podrán acceder a ella a través de cualquier navegador, conectándose a través de internet al Servidor, el cual tiene la Aplicación Web, ver Figura 6.

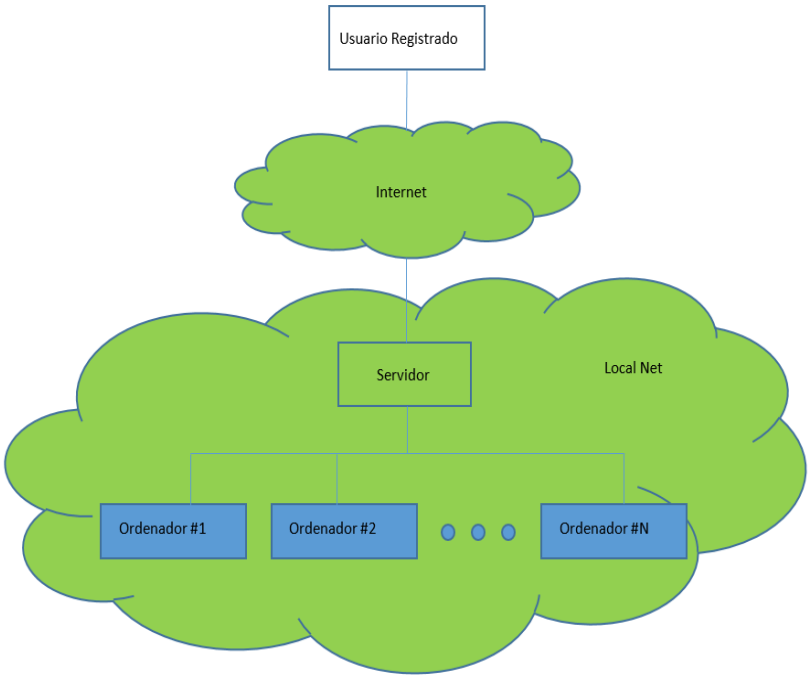



Figura 6: Diagrama de bloques

## 2.2 Requerimientos

Disponer de una infraestructura que realice la función de Gateway Tier, simular en el Gateway Tier una red de nodos que serían las Motas en el modelo real.

Tener a nuestra disposición un ordenador, para realizar la interacción con el usuario, también realizar el procesado de los datos.

Un Usuario registrado podrá crear la red de nodos que uno desee y visualizar la matriz de adyacencia, esto se realizará con grafos, subir y crear algoritmos de análisis de datos que más tarde interactuarán con la red de nodos. Esta interactuación del algoritmo con los nodos, generará un resultado que más adelante un usuario podrá visualizar a través de la página web.

Los Usuarios también serán capaces de poder eliminar una red de nodos, algoritmos y resultados.

Un Usuario solo podrá ver sus propios algoritmos, sus propias redes de nodos y sus propios resultados.

El usuario podrá decidir si ejecutar el algoritmo a una red de nodos, en el ordenador que simula la Cloud o en los ordenadores del Gateway Tier.

## 2.3 Casos de Uso

En la aplicación solo existe un solo caso de uso, es el del Usuario Registrado, Figura 7.

Un Usuario Registrado puede:

Añadir un Algoritmo:

Se pueden generar nuevos algoritmos escritos en lenguaje Python, para que analicen con los Datos generados.

Eliminar un Algoritmo:

Permite eliminar un algoritmo que se encuentre en la aplicación, ya sea porque no se usa o por que se ha subido uno más actualizado.

Generar Nuevos Datos:

Permite generar un nuevo banco de pruebas (Grafo), este puede ser del tipo ER o SBM, dependiendo del tipo de grafo que se desea crear, se tendrá que introducir unos valores u otros.

Visualizar Matriz de Adyacencia:

Nos permite ver las conexiones entre los nodos, para poder generar un algoritmo para unos datos específicos, por si hemos localizado algo interesante en los datos y lo deseamos analizar con más profundidad.

Eliminar Datos:

Nos permite eliminar unos datos ya generados.

Ejecutar Algoritmo con unos Datos en el Cloud:

Aplica a los datos un algoritmo para obtener información de los datos, ya sea la cantidad de nodos o el número de comunidades que existen, este algoritmo se procesa en el Cloud.

Ejecutar Algoritmo con unos Datos en las Raspberry PI:

Igual que el comentado anteriormente, pero esta vez se procesa en las Raspberry PI.

Visualizar Resultados:

Nos permite ver el output del algoritmo, ya sea texto o una imagen.

Eliminar Resultados:

Nos permite eliminar un resultado.

Visualizar API:

Nos permite ver la documentación de la API.

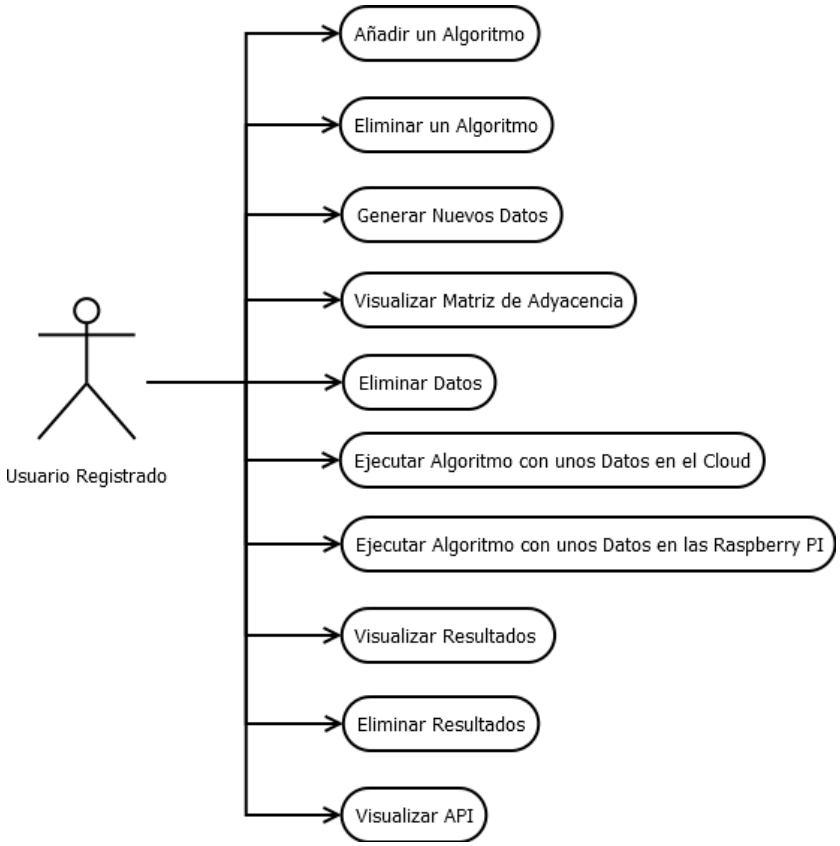



Figura 7: Caso de Uso de un usuario registrado

# 3.0 Implementación

## 3.1 Descripción del Rack de Raspberry PI

Para poder realizar la plataforma experimental los ordenadores del Gateway Tier de la Figura 3 comentada anteriormente, serán sustituidos por un rack de Raspberry PI, de esta forma será asequible y escalable.

Una Raspberry PI es un ordenador de bajo coste con el que se puede realizar muchas funciones, la versión 3 que es la que estamos utilizando, tiene un procesador 1,2 GHz Quad-core ARM Cortex-A53, dispone de 1GB de RAM, 4 USB y un puerto Ethernet.

El Rack será el encargado de procesar los datos de los dispositivos del IoT que se encuentren conectados a él, de esta forma al realizar un primer procesamiento de los datos aliviara la carga de las redes que nos comunican con el cloud de internet, al realizar un primer procesamiento también aliviara la carga del cloud que se localiza en internet.

Además, el rack que sería un nodo en el Gateway Tier podrá comunicarse entre ellos para realizar el procesamiento de los datos recibidos, o simplemente enviarlo al cloud de internet para que este realice todo el procesado de los datos.

El rack está formado por 2 torres de 4 raspberry Pi versión 3 cada una. Dos switch y 2 fuentes de alimentación para alimentar a las Raspberry Pi.

Para poder realizar un prototipo funcional se plantearon diferentes distribuciones de como montar el rack, ya que se creía que la distancia entre las torres podría afectar a la comunicación entre estas mediante wifi, nuestra prueba la comunicación se realiza por el puerto Ethernet de la placa.

Se realizó un análisis para comparar si existían interferencias entre ellas y descubrimos que no, por lo tanto, la distancia entre las torres fue la mínima para poder introducir todos los elementos que la componen.

Para la base del rack se ha decido que el material sea metacrilato, las sujeciones de las torres a la base se ha realizado con tornillos, las sujeciones de las fuentes de alimentación y de los switch se ha realizado con velcro, de esta forma se puede volver a restructurar de una forma rápida y sencilla, ya que si solo queremos disponer de 2 torres con 2 raspberry pi cada una solo necesitaríamos un swicth.

El Rack de Raspberry PI se diseñó en dos niveles, en el primer nivel se encuentran las fuentes de alimentación y los swicth, en el segundo se encuentran las torres. A continuación, se muestran los diseños de la placa de metacrilato que separan los 2 niveles Figura 8, como también la base de esta Figura 9, estos dos niveles se crearon en QCAD para poder cortarlos.

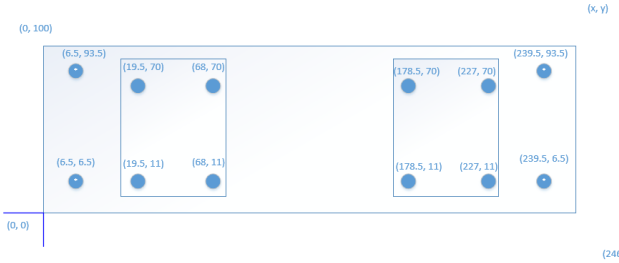



Figura 8: Nivel 1

Figura 9: Base

Se planteó la idea de poder mover las torres del segundo nivel como uno deseara, como también añadir las que uno necesitara para realizar sus pruebas, por ejemplo poner las torres en forma de anillo, pero más tarde esta idea fue desechada por el problema de adherir las torres al metacrilato sin que las torres se inclinaran y con ello la posibilidad de que están torres se cayeran, se pensó en la idea de poner un riel y desplazarlas por el como si fuera una cortina pero no solucionaba el problema de poder colocarlas libremente, ya que solo se podrían mover libremente en un eje.

El esquema del modelo final es el siguiente.


Figura 10: Rack de Raspberry Pi

De este modo si se necesita escalar se pueden añadir todos los racks que se deseen.

La idea original consistía en crear todas las torres de Raspberry PI que se desearan y después estas introducirlas en una estructura de metal, como si fueran componentes de un ordenador de sobremesa, de esta forma seria fácil ordenarlas y refrigerarlas.

En el caso de que la plataforma experimental fuera exitosa y dispusiéramos de suficientes recursos económicos podríamos sustituir las Raspberry PI por ordenadores más potentes.

## 3.2 Interfaz Grafica

La forma en la cual un usuario se comunica con la aplicación es una interfaz gráfica, la interfaz se encuentra basada en web.

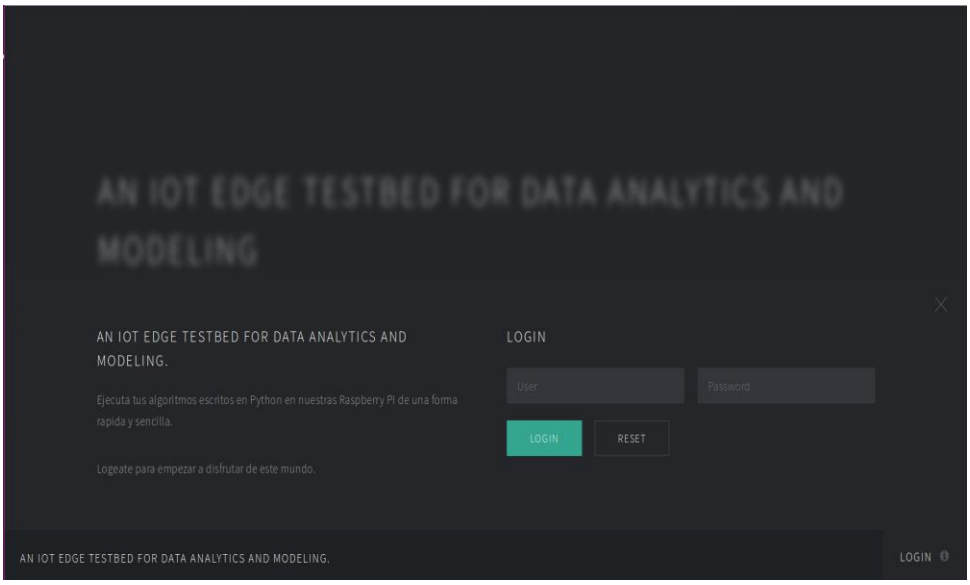



Figura 11: Página principal

En la página principal o index, de la aplicación, un usuario se tiene que autentificar para poder empezar a utilizar la aplicación Figura 11, para ello se encuentran los campos de usuario y contraseña.

La aplicación no dispone de un formulario de registro ya que, si deseas utilizar la aplicación y con ello las Raspberry PI debes ponerte en contacto con el propietario de estas, si te da permiso este, te enviara un usuario y contraseña para poder utilizarlas.

Una vez introducidos correctamente, en este caso el nombre de usuario es test y la contraseña test, seremos redirigidos a la página donde se encuentran los algoritmos Figura 12.

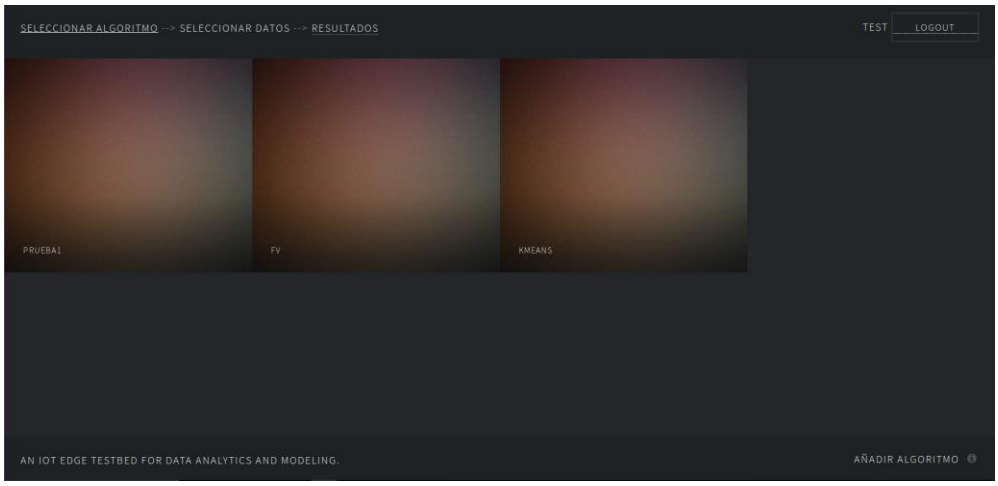



Figura 12: Pagina con todos los algoritmos de un usuario

En esta página podemos subir un nuevo algoritmo en Python para que este analice de alguna forma con el grafo, ya sea obteniendo la cantidad de nodos, las aristas que tiene, las comunidades, etc.

Una vez seleccionado el algoritmo que deseamos aplicar a los datos, nos aparecerá la página en la cual podremos seleccionar los datos Figura 13.

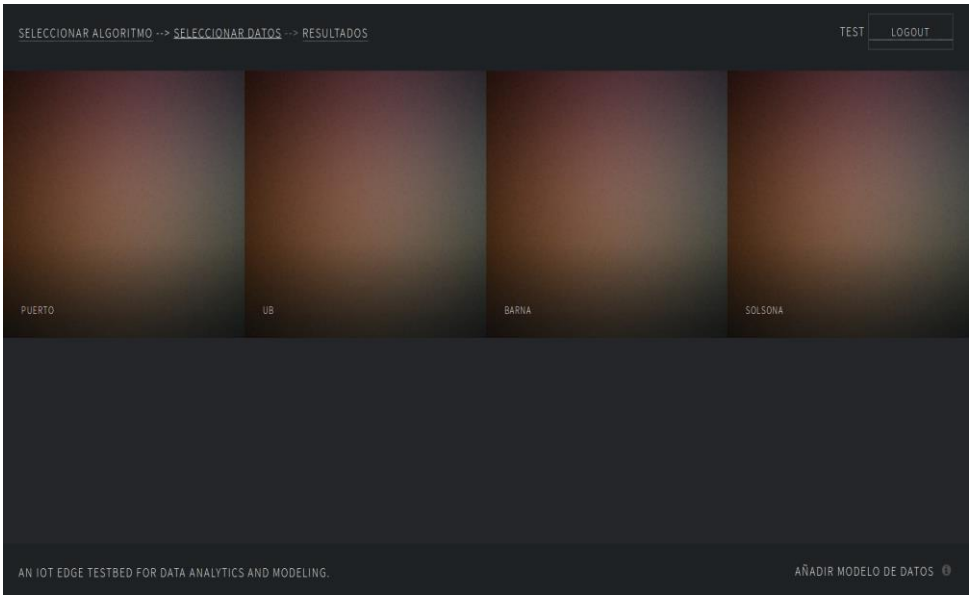



Figura 13: Pagina con todos los datos de un usuario

Podemos seleccionar unos datos ya existentes, como por ejemplo el Puerto de Barcelona, o crear unos datos (Grafo) nuevo Figura 14.

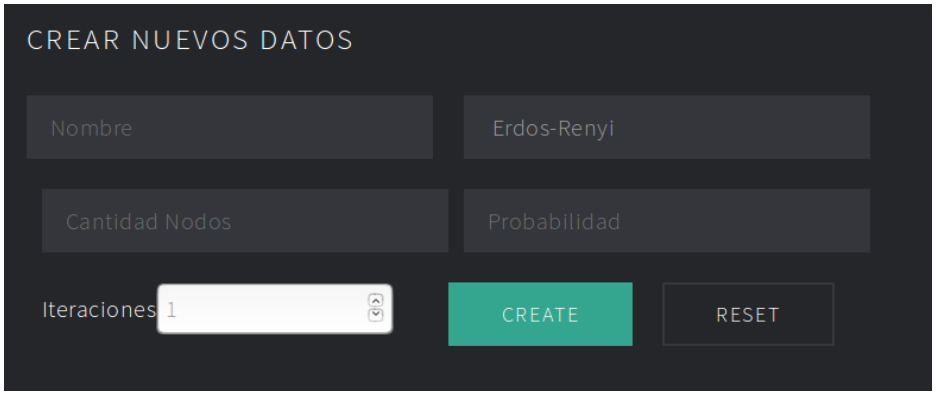



Figura 14: Formulario a rellenar para crear unos nuevos datos

En el caso de crear unos datos nuevos, se nos mostrara esta ventana en la cual podremos darle un nombre al grafo, seleccionar el tipo de grafo que estamos generando (ER o SBM), e introducir los datos generadores dependiendo del tipo de grafo a generar, también podremos crear el grafo con una cantidad de iteraciones. En cada iteración se crea un grafo nuevo con los datos generadores que se han introducido, de esta forma podemos comprobar si la probabilidad media concuerda con la probabilidad que hemos introducido, como es un proceso aleatorio puede no coincidir en algunos casos.

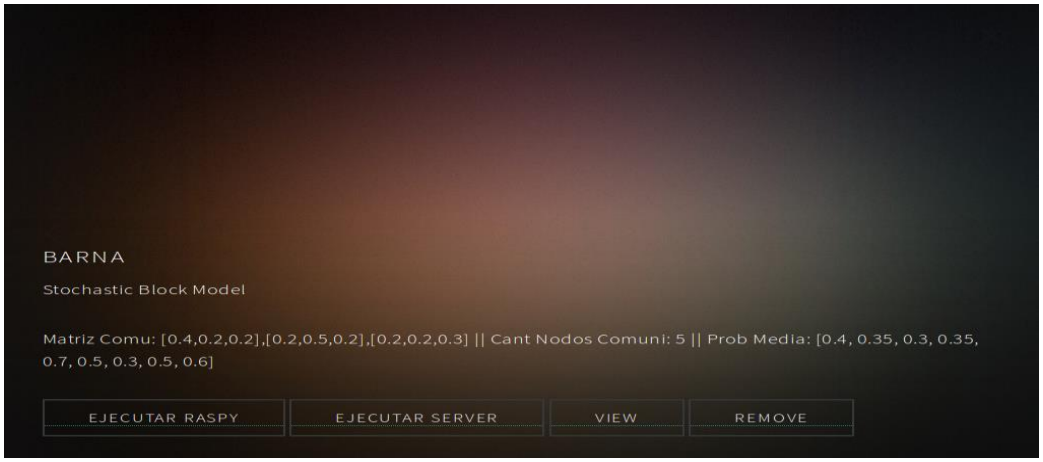



Figura 15: Detalles de un dato

Si escogemos unos datos ya generados Figura 15, nos aparecerá más información del dato escogido, como los datos generadores que han creado el grafo y la probabilidad media, nos aparecerán 4 opciones:

- Ejecutar el algoritmo con los datos en una de las Raspberry PI.

- Ejecutar el algoritmo con los datos en el Server.

- Mostrar la matriz de adyacencia del grafo, esto nos permite generar algoritmos específicos para este dato si deseamos comprobar ciertas cosas, o se ha generado un grafo interesante que deseamos analizarlo más profundamente.

- Eliminar el dato.

Una vez escogido donde lo deseamos analizar, tendremos que ir a la ventana de resultados Figura 16 para poder visualizar el resultado del algoritmo con dicho dato.

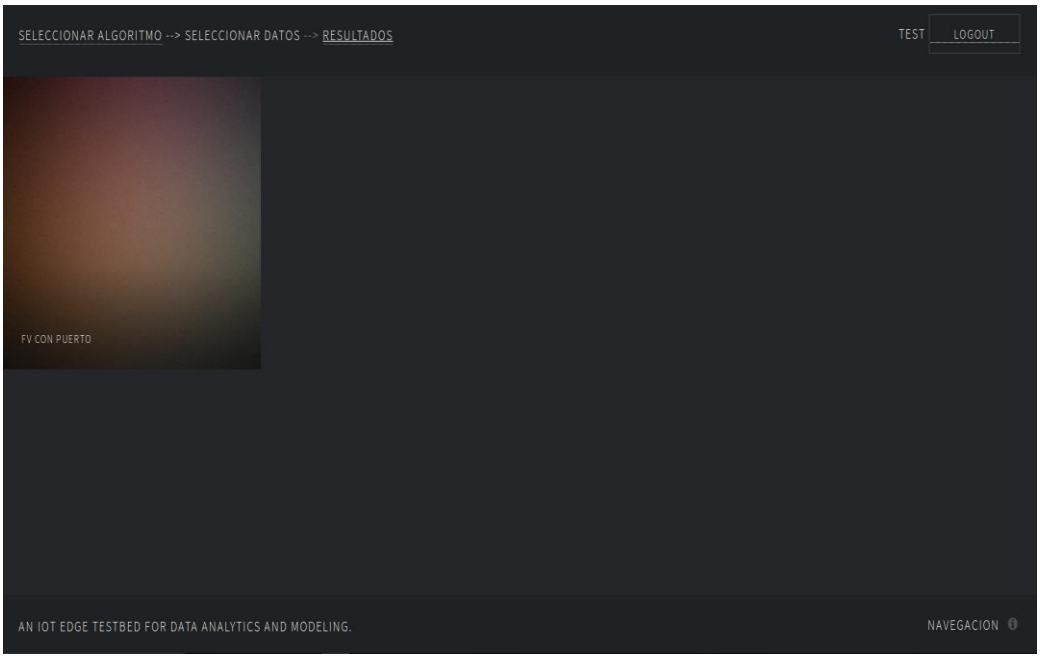



Figura 16: Pagina en la que se encuentran todos los resultados ejecutados

En esta página nos aparecerán todos los resultados que han sido lanzados, para poder visualizar un resultado tendremos que seleccionar el resultado y nos aparecerán unas opciones, una es ver el resultado y la otra es eliminar el resultado.

Si el algoritmo que hemos seleccionado anteriormente nos dibuja un grafo, por ejemplo, el algoritmo nos dibuja las comunidades, nos mostrará algo parecido a la figura 17 y si es solo texto nos mostrara algo parecido a la figura 18.

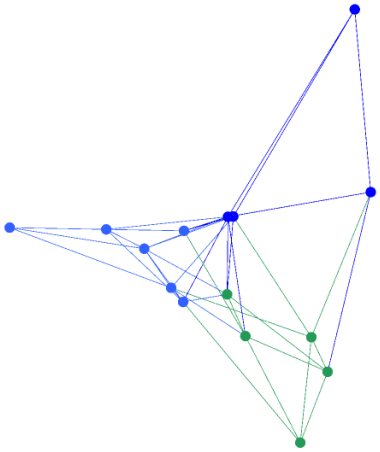


Figura 17: Dibujo de un Grafo

[(0, 32), (0, 2), (0, 59), (0, 36), (0, 8), (0, 74), (0, 11), (0, 12), (0, 45), (0, 46), (0, 80), (0, 18), (0, 22), (0, 91), (0, 29
(1, 78), (1, 84), (1, 86), (1, 87), (1, 90), (1, 91), (1, 94), (2, 64), (2, 69), (2, 6), (2, 39), (2, 8), (2, 11), (2, 44), (2, 45)
(3, 71), (3, 8), (3, 11), (3, 12), (3, 13), (3, 77), (3, 17), (3, 83), (3, 94), (3, 86), (3, 23), (3, 88), (3, 15), (3, 29), (3, 6
(5, 71), (5, 64), (5, 9), (5, 74), (5, 43), (5, 44), (5, 78), (5, 16), (5, 82), (5, 20), (5, 53), (5, 54), (5, 55), (5, 24), (5, 8
(6, 60), (6, 66), (6, 69), (6, 71), (6, 77), (6, 82), (6, 83), (6, 84), (6, 91), (7, 15), (7, 17), (7, 21), (7, 26), (7, 35), (7,
91), (7, 94), (8, 13), (8, 24), (8, 26), (8, 31), (8, 34), (8, 40), (8, 41), (8, 42), (8, 53), (8, 62), (8, 63), (8, 67), (8, 68),
(9, 95), (9, 12), (9, 45), (9, 47), (9, 52), (9, 54), (9, 89), (9, 25), (9, 59), (9, 69), (10, 34), (10, 99), (10, 37), (10, 70)
(11, 28), (11, 29), (11, 38), (11, 48), (11, 49), (11, 50), (11, 54), (11, 55), (11, 56), (11, 60), (11, 67), (11, 77), (11,
47), (12, 50), (12, 58), (12, 75), (12, 77), (12, 85), (12, 90), (12, 99), (13, 21), (13, 24), (13, 29), (13, 52), (13, 54),
(14, 44), (14, 76), (14, 84), (14, 41), (14, 73), (14, 63), (15, 96), (15, 33), (15, 34), (15, 76), (15, 44), (15, 27), (15,
78), (16, 79), (16, 48), (16, 18), (16, 83), (16, 85), (16, 29), (16, 62), (16, 95), (17, 35), (17, 66), (17, 36), (17, 42),
(18, 68), (18, 69), (18, 95), (18, 79), (18, 24), (18, 87), (18, 88), (18, 36), (18, 47), (18, 93), (18, 62), (18, 31), (19,
26), (19, 92), (19, 95), (20, 33), (20, 67), (20, 81), (20, 49), (20, 71), (20, 79), (20, 22), (20, 21), (20, 86), (20, 90),
(21, 59), (21, 60), (21, 61), (21, 63), (21, 65), (21, 82), (21, 85), (21, 95), (21, 96), (22, 23), (22, 27), (22, 28), (22,
92), (22, 99), (23, 66), (23, 37), (23, 81), (23, 34), (23, 79), (23, 51), (23, 56), (23, 88), (23, 98), (23, 63), (24, 96),
(25, 67), (25, 37), (25, 42), (25, 71), (25, 44), (25, 96), (25, 92), (25, 80), (25, 43), (25, 51), (25, 99), (25, 41), (25,
68), (26, 69), (26, 70), (26, 80), (26, 81), (26, 85), (26, 92), (26, 98), (27, 64), (27, 33), (27, 71), (27, 40), (27, 42),
(28, 51), (28, 52), (28, 87), (28, 56), (28, 90), (28, 79), (28, 30), (28, 31), (29, 33), (29, 97), (29, 72), (29, 73), (29,
82), (30, 90), (30, 59), (31, 35), (31, 68), (31, 33), (31, 43), (31, 76), (31, 50), (31, 93), (31, 48), (31, 83), (31, 57),
(32, 65), (32, 69), (32, 74), (32, 80), (32, 84), (32, 94), (32, 95), (32, 96), (32, 98), (33, 37), (33, 40), (33, 42), (33,
77), (34, 80), (34, 81), (34, 51), (34, 49), (34, 62), (35, 97), (35, 67), (35, 37), (35, 81), (35, 44), (35, 77), (35, 91),
(36, 43), (36, 48), (36, 51), (36, 86), (36, 55), (36, 88), (36, 54), (36, 42), (36, 37), (37, 38), (37, 40), (37, 51), (37,
46), (38, 67), (38, 85), (38, 55), (38, 60), (38, 94), (39, 96), (39, 97), (39, 67), (39, 66), (39, 72), (39, 73), (39, 79),
(40, 78), (40, 59), (40, 50), (41, 45), (41, 48), (41, 50), (41, 54), (41, 57), (41, 58), (41, 66), (41, 70), (41, 78), (41

Figura 18: Salida en caso de que sea texto

## 3.3 Modelado y generación de Datos

Se pretende analizar la estructura en la que se encuentran conectadas las Motas, como adquirir el dispositivo completo es costoso, la placa que contiene el emisor Wifi y el procesador son 75€, a este dispositivo hay que sumarle el precio de un sensor de temperatura, este sensor cuesta unos 15€ y el sensor de $CO_2$, este dispositivo es el más caro, ya que ronda los 85€, por lo tanto, un nodo completo de la red ronda los 175€.

Como se ha comentado conectar muchas motas es un proceso caro y complicado, para solucionar este problema, se ha decidido simularlas, ya que es un método muy barato.

A su vez simular las Motas también aporta mucha más flexibilidad, porque se pueden considerar diferentes valores de los datos, además también se pueden optar por diferentes topologías de conexión entre ellas, de una forma rápida y precisa, lo cual es muy útil antes de utilizarlos en un entorno real. Ya que se pueden observar las limitaciones y las bondades de los algoritmos generados.

Para simular la estructura de conexión de las motas se generan 2 tipos de grafos aleatorios, el Erdös-Rényi (ER) y el Stochastic Block Model (SBM).

Los grafos nos permiten representar las redes de sensores de una manera sencilla, ya que cada Mota representa un nodo del grafo y las conexiones que realiza la Mota con los demás dispositivos se representa mediante aristas.

Por lo tanto, un nodo es la representación de un objeto y una arista es la comunicación que existe entre dos nodos.

Una vez generado el grafo se puede obtener su matriz de adyacencia, la cual nos permitirá obtener información muy útil del grafo, como por ejemplo saber las conexiones que tiene un nodo con todos los demás.

Existe una probabilidad de que un nodo se conecte a otro, usaremos la probabilidad de Bernoulli, esta probabilidad nos dirá la cantidad de nodos que estarán conectados con otros, a mayor probabilidad habrá más conexiones entre ellos.

### 3.3.1 Erdös-Rényi

El grafo aleatorio ER nos permite conectar nodos entre sí con una probabilidad, de esta forma cada vez que generemos el grafo las conexiones entre ellos serán distintas por lo tanto tendremos un grafo diferente cada vez, podríamos decir que cada vez que se ejecute el algoritmo tendríamos una comunidad de nodos. El ER no es muy útil, pero nos permitirá analizar una comunidad de una forma sencilla ya que si queremos analizar un conjunto de comunidades tenemos el SBM.

Para generar un grafo ER se necesita el número total de nodos que se desean y la probabilidad de conexión entre ellos.

$$N_{ij} = B(P)$$

Para saber si existe conexión entre un nodo "i" y otro nodo "j", se realiza la probabilidad de Bernoulli de la probabilidad introducida al generar el grafo, todos los nodos tienen la misma probabilidad de estar conectados entre ellos.

### 3.3.2 Stochastic Block Model

El grafo aleatorio SBM es un mucho más real que el citado anteriormente, esta forma de generar grafos aleatorios nos permite obtener diferentes comunidades de nodos, los nodos de una misma comunidad se conectan entre ellos con una misma probabilidad y esta comunidad se conecta con los nodos de las otras comunidades con otra probabilidad, cada comunidad tiene su probabilidad, de esta manera podemos configurar el grafo para que si dos comunidades que se encuentren cercanas entre sí, la probabilidad de conexión entre estas dos sea mayor que si estas comunidades estuvieran alejadas.

Para generar el SBM se necesita el número de nodos por comunidad y una matriz cuadrada, en la cual se introduce la probabilidad de conexión entre los nodos de una misma comunidad y la probabilidad de conexión con los nodos de las demás comunidades.

Una vez tenemos la matriz cuadrada de probabilidades (P), se realiza el producto de Kronecker con el número de nodos que deseamos que tenga cada comunidad ($J_K$).

$$A = P \otimes J_K$$

Esta operación nos devuelve una matriz cuadrada, con las probabilidades de conexión entre cada uno de los nodos, ya sea de una misma comunidad o no.

### 3.3.3 Análisis De Datos

Para saber las comunidades que hay en un grafo empleamos el algoritmo Fiedler vector-based clustering, ver Apartado 8: Referencias, en el cual se tiene en cuenta el vector de Fiedler para segmentar el grafo en comunidades.

El vector de Fiedler se obtiene de la matriz Laplaciana de un grafo, para ello obtenemos el segundo eigenvalue no nulo de la matriz, el autovector asociado es el vector de Fiedler.

Con dicho vector se puede segmentar cualquier grafo en dos comunidades utilizando los valores que nos da el vector ya que los nodos de un vector que tengan valores positivos nos indican que son parte de una comunidad, los valores de los nodos que sean negativos son otra comunidad, para finalizar los valores que sean cercanos al 0, son los nodos que conectan una comunidad con la otra. De esta forma si repetimos en proceso, se puede segmentar un grafo en N comunidades.

El Algoritmo Fiedler vector-based clustering sería el siguiente:

1- Se calcula el vector de Fiedler.
2- Los valores positivos de los nodos del vector se interpretan como una comunidad.
3- Después se eliminan esas aristas.
4- Se comprueba si ya tenemos N - 1 comunidades, sino es así se vuelve al paso 1.
5- Los valores negativos de los nodos del vector se interpretan como una comunidad.

## 3.4 Desarrollo

A la hora de desarrollar la aplicación se ha decidido emplear JavaServer Page (JSP) para mostrar información al usuario, podría haber desarrollado también con una aplicación de escritorio convencional, para el servidor web se ha decidido utilizar Apache Tomcat.

A la hora de dar a la página web un aspecto visual más cuidado se ha empleado JQuery que es una biblioteca de Java Script, para poder desarrollar el código de una manera más sencilla.

La aplicación está compuesta de dos partes, una que se emplea para la interacción del usuario con la aplicación, la página web, la otra es la que se encarga de aplicar el algoritmo a los datos.

Cuando el usuario mediante la interfaz web decida ejecutar un algoritmo sobre unos datos, el servidor introduce unos parámetros a una aplicación escrita en C++ y la lanza.

Dependiendo de los parámetros recibidos, la aplicación que se ha decidido llamarla Cliente, ejecutara él envió del algoritmo a la aplicación Server que se encuentra en las Raspberry PI.

En el caso de que el usuario decida ejecutar el algoritmo en el Cloud será la aplicación Server del Cloud la que recibirá los datos.

Si los parámetros que ha recibido el Server, resulten ser para generar nuevos datos este generará el nuevo grafo, incluida matriz de adyacencia de este. Cuando concluya la generación de los datos, este los guarda en la base de datos que se encuentra en la Cloud.

En el caso de que los parámetros que se ha recibido, resultan ser para aplicar un algoritmo a unos datos.

La aplicación server descargara el algoritmo en Python, como también el grafo de la base de datos, se ha decidido realizarlo de esta manera para que la Raspberry PI no cargue con una base de datos en local, ya que los recursos son limitados y los datos son simulados.

En el campo real no tendría que descargar estos datos ya que estarían conectados a ella, cuando la descarga de la información necesaria se finalice, se ejecutará el algoritmo escrito en Python sobre el grafo.

Esta acción devuelve en caso de que el algoritmo lo desee, un archivo de texto con el output del algoritmo y un json del grafo, más tarde esta información se guarda en la base de datos para que el usuario la pueda consultar desde la página web.

Para la correcta visualización del grafo se ha decidido utilizar Sigma.js, que es una librería de JavaScript para poder dibujar grafos de una manera sencilla y elegante.

Como disponemos de un Rack de Raspberry PI, tenemos que decidir a que Raspberry de las 4 disponibles le enviamos el grafo a analizar, para ello se ha creado un servicio REST para que cada 5 segundos, la aplicación server que se encuentra en cada Raspberry PI, envié su rendimiento actual al server de esta forma obtiene la que dispone de más recursos disponibles y envía el grafo a analizar a esta.

Para calcular el rendimiento de la Raspberry PI se utiliza el tanto por cien de procesador que está utilizando y la memoria RAM que está en uso.

## 3.5 Base De Datos

A la hora de almacenar los datos, se ha decidido utilizar una base de datos, de todas las que existen actualmente, para la plataforma experimental nos hemos inclinado por MySQL.

Se contempló la posibilidad de utilizar MongoDB, esta base de datos es una base de datos NoSQL, por la gran velocidad de la lectura de los datos y por la alta escalabilidad que tiene.

Pero al final se descartó por no cumplir las propiedades ACID, una de las propiedades es muy importante para el proyecto ya que no nos asegura en ningún momento, que las escrituras en la base de datos se realicen, ya que en algunos casos pueden fallar, entonces podría darse la casualidad que algún dato importante no se escribiera, por consecuencia este dato lo perderíamos.

A continuación, se muestra el diagrama de Entidad Relación (ER) de la base de datos Figura 19.

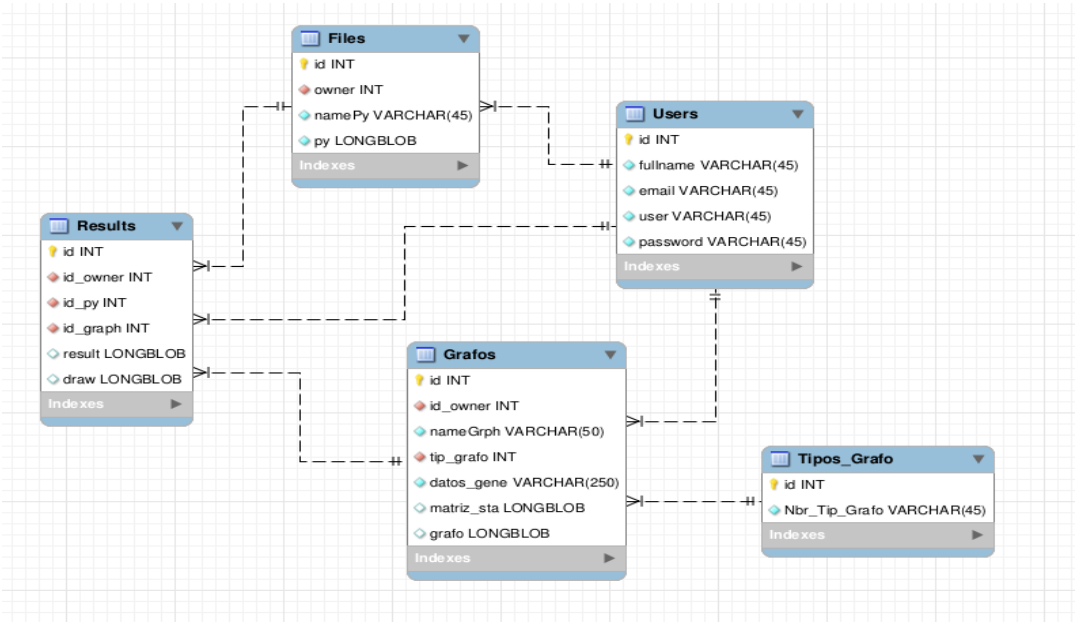



Figura 19: Diagrama de Entidad Relación

Users: En dicha tabla se guardan la información personal del usuario, como es el nombre de usuario y contraseña con las que podremos acceder a la aplicación.

Files: En esta tabla se guardan los algoritmos, el nombre del mismo y el archivo Python que contiene las instrucciones para realizar el algoritmo. Nos guardamos el “id” del usuario que ha subido el algoritmo, para más tarde poder mostrar solo todos los algoritmos de un usuario.

Grafos: Contiene los datos de la simulación (Grafos), los parámetros empleados para realizar el grafo como también su matriz de adyacencia que nos puede ayudar a la hora de diseñar un algoritmo. Guardamos el tipo de grafo y el usuario que lo ha generado, para poder filtrar los grafos más tarde.

Tipos_Grafo: Contiene el tipo de grafo, en el estado que se encuentra actualmente la aplicación solo disponemos de dos, a posteriori se podrían implementar más.

Results: En esta tabla se guarda la salida de ejecutar un algoritmo con unos datos, se guarda el output del algoritmo y el json resultante si los hay. Nos guardamos el “id” del algoritmo empleado en el análisis, también el “id” del grafo y el del usuario. Para saber que usuario ha generado este resultado, como el grafo que ha utilizado y el algoritmo.

# 4.0 Resultados Experimentales

## 4.1 Introducción

Actualmente existen tres grandes empresas que ofrecen servicios de Cloud computing, estos tres son Amazon, Microsoft y Google. Los servicios en el mismo orden son Amazon Web Services, Azure y Google Cloud.

Podríamos decir que la nube es el siguiente paso lógico del hosting dedicado, ya que, si creamos una página web y la queremos publicar, tendríamos que alquilar un servidor dedicado en internet o comprar uno nosotros y pedir a alguna empresa que nos coloque nuestro servidor con los suyos, a esto último se le denomina housing. El problema del hosting dedicado es que estamos pagando por recursos que muy probablemente no usaremos nunca ya que si nuestra página web es poco visitada esos recursos no se utilizan, si un día nuestra página web tiene muchas visitas el hosting dedicado se quedaría corto de recursos y tendríamos que alquilar uno más grande sino nuestro servidor caería, por otro lado sabemos que nuestra página la mayoría de las visitas serian durante el día no durante la noche, por lo tanto estamos pagando el alquiler de un hosting que durante la noche no se usa.

Al usar cloud solo se nos cobraría por las horas que se está utilizando o cuando realizamos alguna de las operaciones CRUD a una base de datos, las tarifas dependen de la compañía que escojamos.

Con esto quiero decir que si tenemos muchas visitas no nos tendremos que preocupar en comprar un hosting más grande ya que la cloud nos otorgará más potencia de cálculo cuando la necesitemos, en el caso de aplicar un algoritmo a un grafo, donde el grafo es mucha información, nos otorgara una cantidad de recursos de computación increíble solo cuando la necesitemos. Al realizar el análisis en la cloud pasamos de analizar un grafo que tardaría minutos en ser procesado a segundos.

El problema viene con las conexiones a internet ya que si tenemos una conexión lenta no merece la pena ya que el tiempo de subida del grafo es mayor al tiempo de procesado en el edge.

Como las Motas estarán dentro de un edificio o una zona al aire libre, la conexión que tendrán a internet para poder realizar el análisis en la cloud puede no sea muy buena, por ello merece la pena bajar la cloud al edge. Ya que al realizar este movimiento

evitamos el problema de la velocidad de subida y por lo tanto ganamos mucho en la reducción de los tiempos de latencia.

En el caso de que un semáforo con un micrófono detecte el sonido característico de una ambulancia, si el análisis de estos datos de audio se realiza en el cloud, el semáforo no recibirá la orden de ponerse en verde para que la ambulancia pase, cuando recibiría la señal de ponerse en verde puede que estuviera en el siguiente semáforo, esta situación se debe a que el tiempo de latencia es muy alto.

Al bajar el cloud al borde de la red, se conseguiría bajar el tiempo de latencia y en la situación comentada anteriormente, el semáforo se pondría en verde antes de que la ambulancia pasara por él, esta es una de las muchas situaciones en las cuales este concepto podría salvar vidas, ya que la ambulancia llegaría lo antes posible al hospital y el paciente podría ser tratado rápidamente, salvando así su vida.

Para la comparativa se usará un ordenador que realizara la función de una Raspberry PI, pero más potente, este ordenador tiene un Intel Core 2 Quad a 2.4GHz e 4GB de RAM, se le ha denominado Server. La Cloud real de Google Cloud. Una sola Raspberry PI y el Rack de Raspberry PI.

Las velocidades de subida de las comparativas son reales, para la obtención de la velocidad de subida de las conexiones 3G y 4G, se ha empleado un Samsung Galaxy S6 en una población de la provincia de Castellón.

## 4.2 Rendimiento de una Raspberry Pi

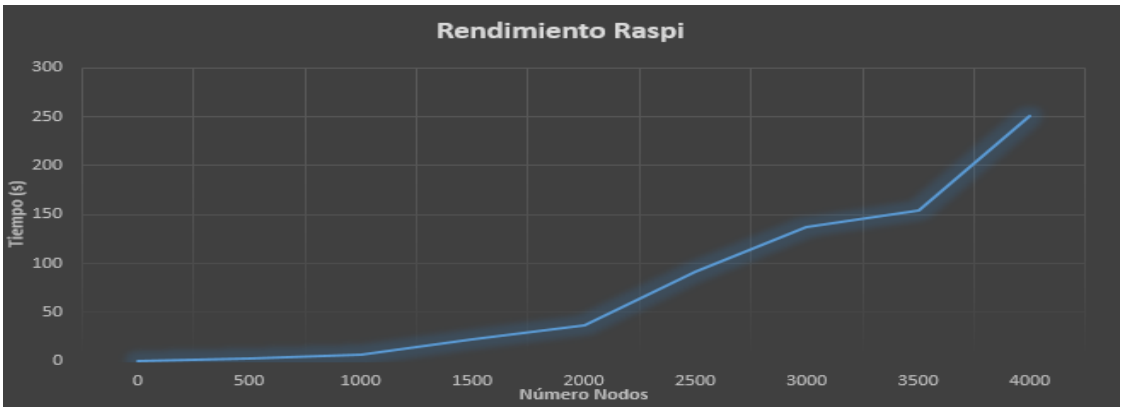



Figura 20: Rendimiento de una Raspberry PI

En la Figura 20, podemos observar el tiempo que tarda una sola Raspberry PI en realizar el algoritmo Fiedler vector-based clustering sobre un grafo. El tiempo que tarda en analizar 2000 nodos es un tiempo aceptable, pero a los 2500 nodos el tiempo que tarda al analizar 500 nodos más es más del doble.

## 4.3 Rendimiento del Server

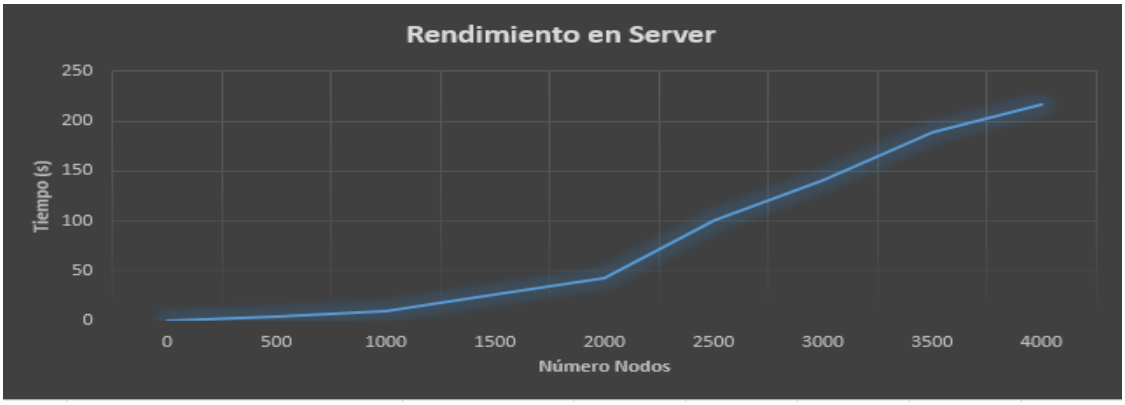



Figura 21: Rendimiento del ordenador Server

Esta gráfica es la mismo que la anterior, pero en vez de analizar el grafo en una Raspberry PI lo estamos ejecutando en el servidor, podemos observar que la gráfica es muy parecida a la anterior, al pasar de los 2000 nodos el tiempo se dispara, esto es debido a que el Server aparte de analizar gestiona el servicio web y la base de datos.

## 4.4 Rendimiento Google Cloud con ADSL

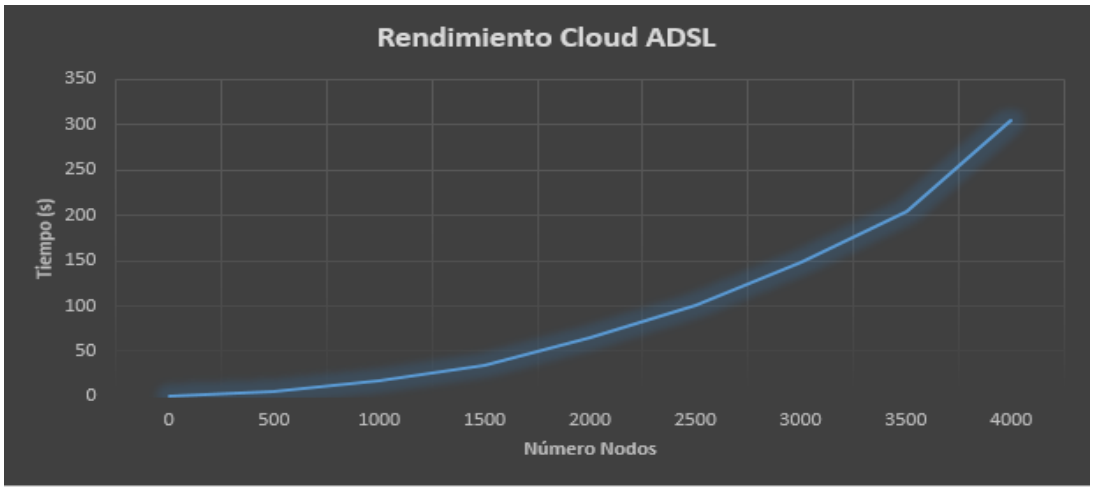



Figura 22: Rendimiento de la Cloud con una conexión ADSL

Podemos observar Figura 22 que el rendimiento de la Cloud con una conexión ADSL es muy parecida a la del Server esto es debido por la velocidad de subida, ya que la velocidad de subida es de 0.066 MB, por lo tanto, un grafo muy grande, un grafo de 3000 nodos pesa unos 10MB, tardara mucho tiempo en subirlo a la Cloud para poder empezar su análisis, una vez allí el análisis se realiza en segundos, el análisis con el grafo más grande es de 10s, si lo comparamos con la Raspberry PI el cloud es 25 veces más rápido.

## 4.5 Comparativa Google Cloud ADSL vs Raspberry PI

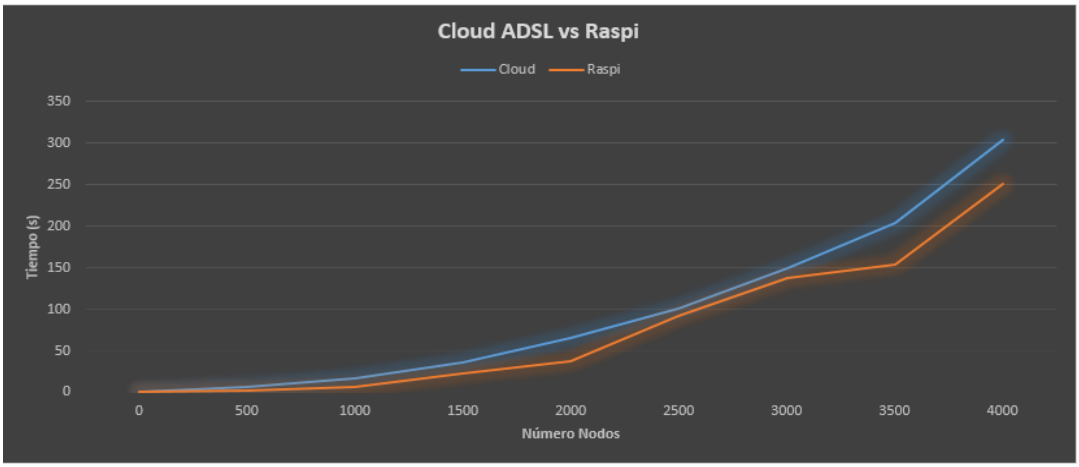



Figura 23: Rendimiento de la Cloud con ADSL contra una Raspberry PI

Tenemos la comparativa de una Raspberry PI con el Cloud con una conexión ADSL, se puede observar que la Raspberry PI es más rápida que la Cloud, pero no se obtiene una ventaja significativa, si en el caso de la conexión de las Motas seria de una conexión ADSL, merecería la pena utilizar el Cloud ya que nos ahorramos los problemas de mantenimiento que pueden ocasionar las Raspberry PI.

## 4.6 Rendimiento Google Cloud con IRIS

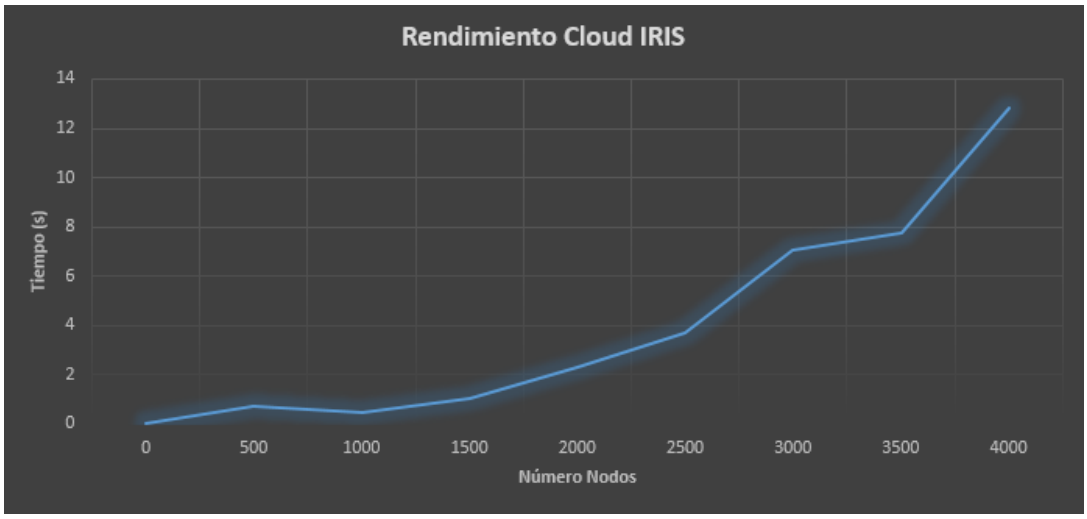



Figura 24: Rendimiento de la Cloud con una conexión IRIS

Se puede observar que el rendimiento de la Cloud con una velocidad de subida de 9MB es espectacular, tenemos que el grafo más grande lo sube a la Cloud y lo analiza en 12.5s, una Raspberry PI analizaría en 12.5s un grafo de 1250 nodos más o menos. En el caso de disponer de una conexión a internet en todo el mundo con una velocidad de subida de 9MB, el cloud sería la mejor opción siempre y cuando no dispongamos de un Rack de Raspberry PI, esta situación la comentaremos más adelante.

## 4.7 Rendimiento Google Cloud con 4G

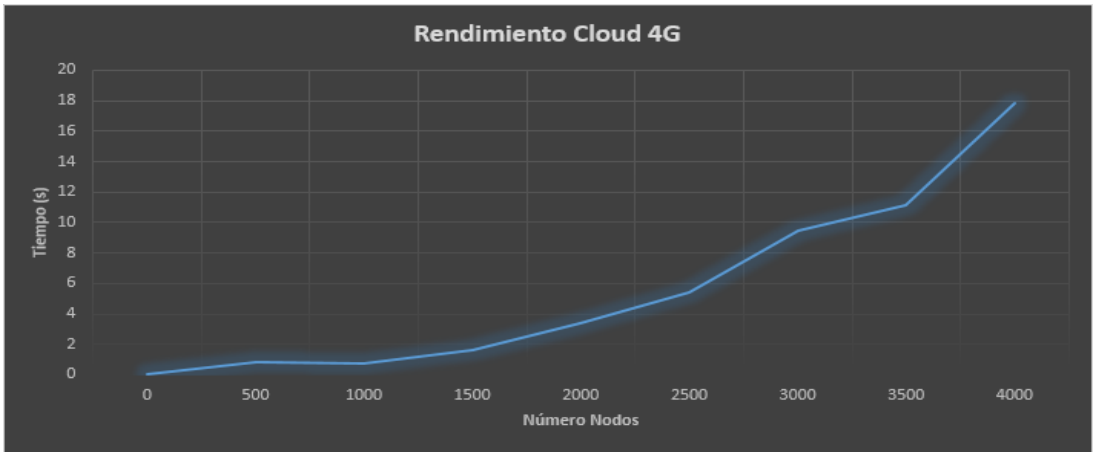



Figura 25: Rendimiento de la Cloud con una conexión 4G

Rendimiento de la Cloud con una conexión 4G Figura 25, la velocidad de subida con una conexión 4G es de 2.7MB, como se puede observar da unos tiempos muy aceptables, ya que la conexión 4G está bastante extendida se puede considerar una solución viable sí solo disponemos de una Raspberry PI.

## 4.8 Rendimiento Google Cloud con 3G

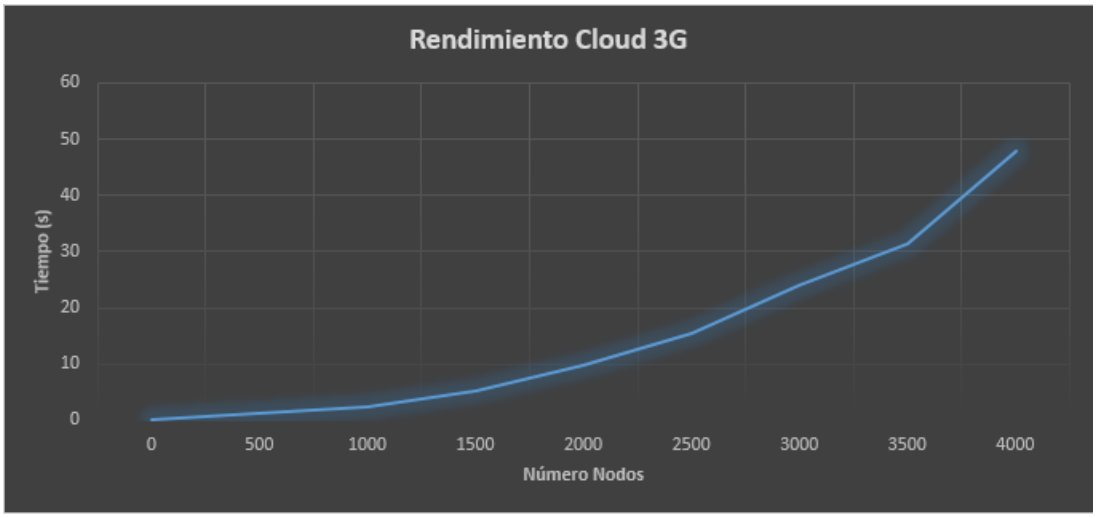



Figura 26: Rendimiento de la Cloud con una conexión 3G

Rendimiento de la Cloud con una conexión 3G Figura 26, la velocidad de subida es de 0.127MB, los tiempos si los comparamos con la Raspberry PI o Server son muy buenos, pero demasiado altos si los comparamos con la conexión de 9MB.

## 4.9 Rendimiento del Rack de Raspberry PI

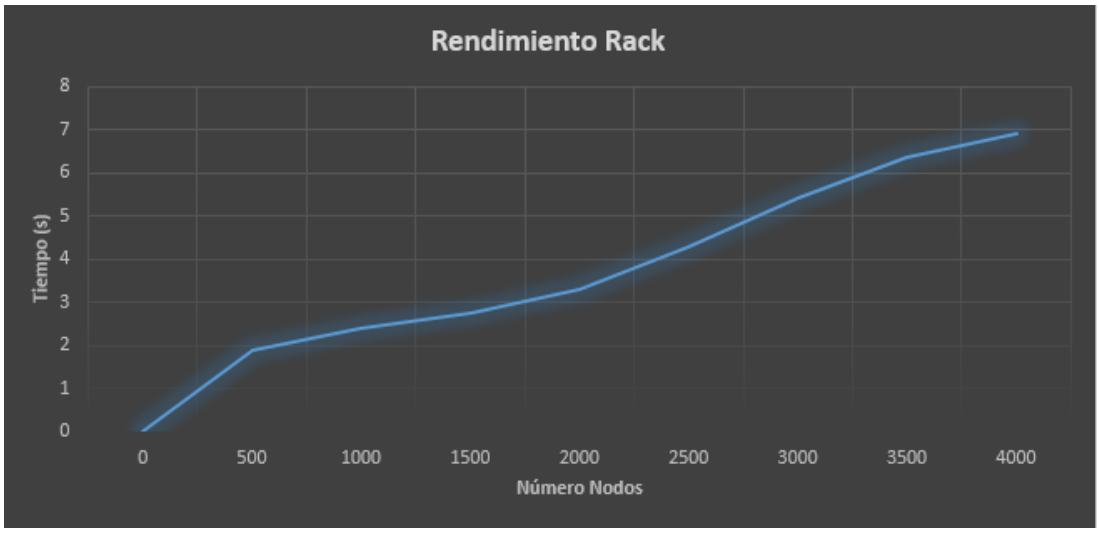



Figura 27: Rendimiento del Rack de Raspberry PI

Rendimiento del Rack de Raspberry PI, son 4 Raspberry PI trabajando a la vez, esta grafica muestra el tiempo que tardaría en analizar los grafos, en la gráfica no se ha tenido en cuenta el tiempo que tardarían en comunicarse entre ellas para poder tener una visión global del grafo, por tanto, está trabajando con comunidades independientes, esta situación es favorable si tenemos diferentes Motas en una misma zona, sin comunicación entre ellas.

## 4.10 Comparativa Google Cloud vs Raspberry PI

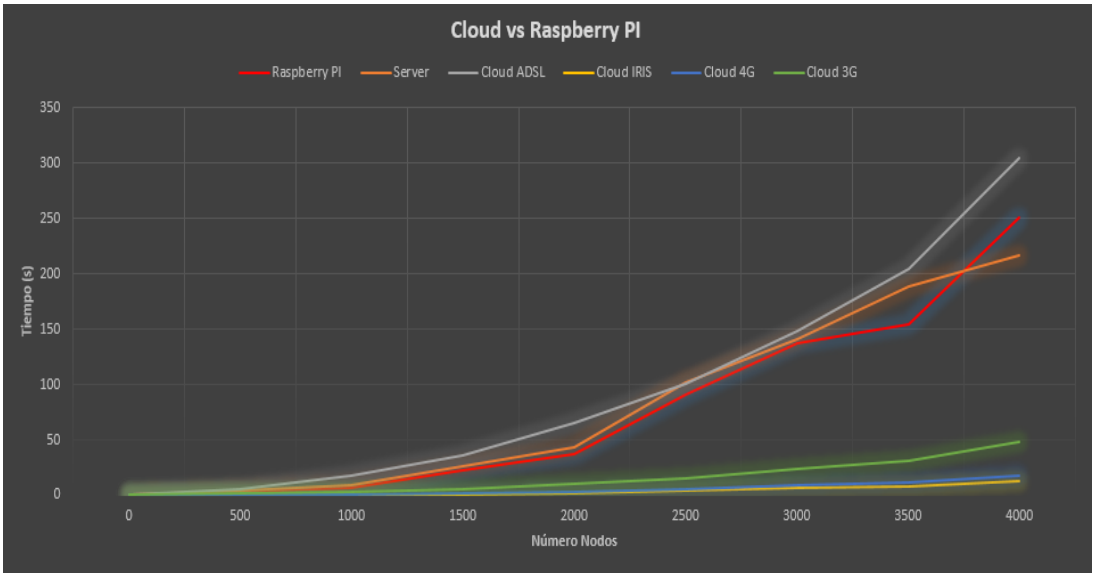



Figura 28: Rendimiento de la Cloud contra una Raspberry PI

Comparamos todas las anteriores soluciones con una sola Raspberry PI Figura 28, podemos observar que si solo disponemos de una Raspberry PI en el edge, donde se generan los datos, no merece la pena trabajar en el edge es mejor utilizar la cloud, ya que con la conexión de subida que nos proporciona una conexión 4G obtendremos muy buenos resultados, en el caso de que la zona donde se encuentra la Raspberry PI no disponemos de una conexión 4G, la red 3G también otorga buenos resultados.

Si deseamos analizar los datos en una zona rural donde no dispongamos de conexión 3G y por lo tanto tampoco 4G sería mejor la Raspberry PI, esta solución es buena donde la conexión a internet no sea óptima.

## 4.11 Comparativa Google Cloud vs Rack Raspberry PI

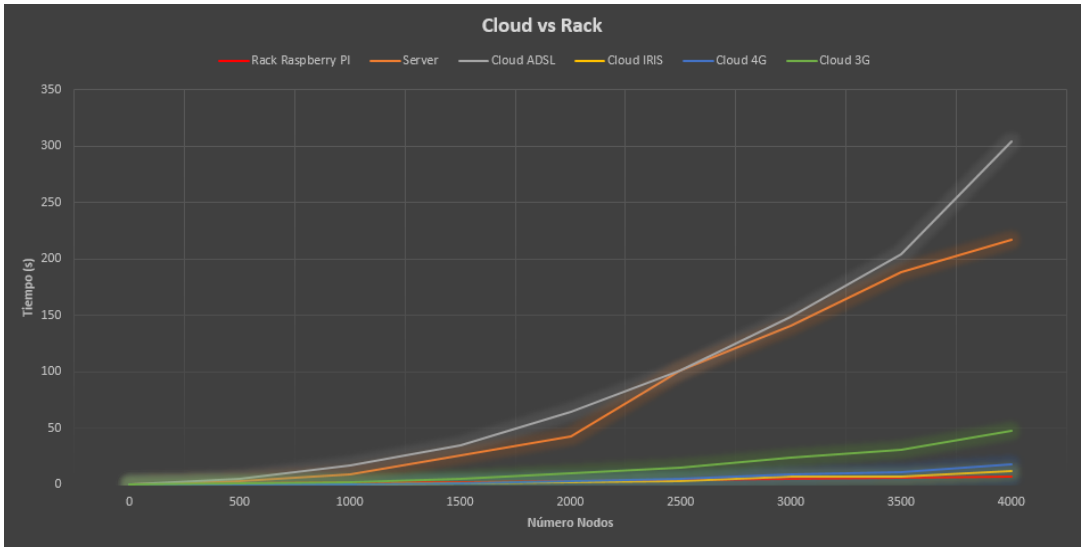



Figura 29: Rendimiento de la Cloud con ADSL contra el Rack de Raspberry PI

Comparamos todas las anteriores soluciones con el Rack de Raspberry PI Figura 29, como se puede observar la solucion de disponer de un Rack en el edge de internet es una solucion muy optima, ya que nos evitamos el problema de la conexión a internet, que es donde se encuentra nuestro cuello de botella.

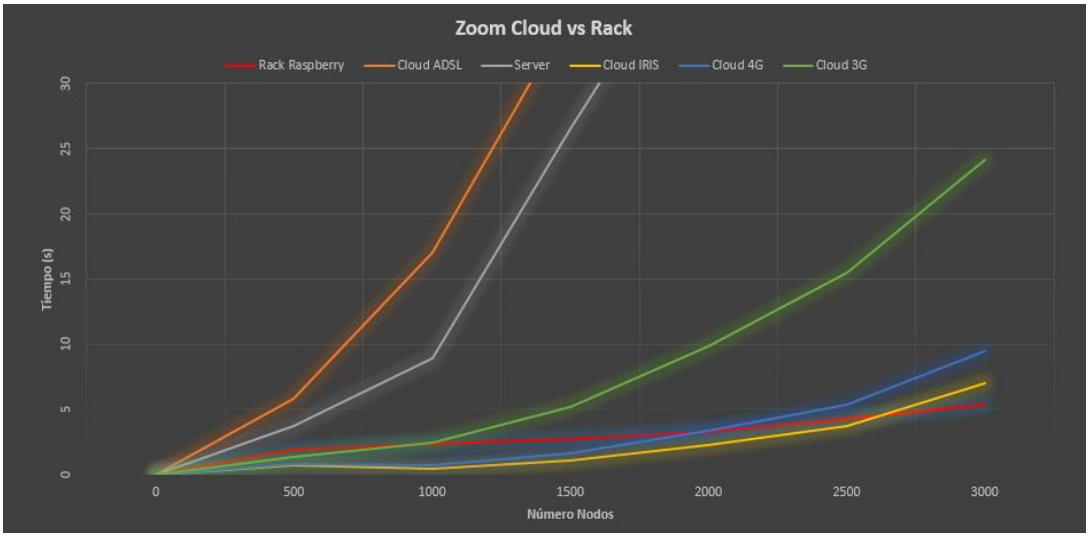



Figura 30: Zoom de la comparativa de la Figura 29

Ampliacion de una seccion de la grafica anterior Figura 30, ya que no se puede visualizar los valores correctamente, al disponer del Rack la solucion de 3G ya no resulta favorable, ya que es demasiado lenta, si tenemos en cuenta una pequeña cantidad de nodos, la conexión de 4G y la subida de 9MB son un poco superiores. En caso cantrario, una gran cantidad de nodos, se puede observar que el Rack obtiene mejores resultados que hasta con una velocidad de subida de 9MB, por lo tanto, disponer de un Rack en el edge obtendremos los mejores resultados y de esta forma evitamos el cuello de botella que tenemos con la conexión a internet al trabajar con la cloud, por no mencionar la latencia que sera muy baja, para obtener una respuesta muy rapida.

# 5.0 Instalación

## 5.1 Requisitos

Para poder instalar la aplicación es necesario disponer de un pc que realizara la función de server o “Edge Cloud” donde se encuentra la página web, el usuario se comunica con la aplicación mediante la página, se podría haber hecho una aplicación de escritorio, pero se decidió realizarla en un entorno web, ya que se encontraría alojada con la pagina que permite visualizar los datos de las Motas.

## 5.2 Instalamos MySQL

Para poder instalar MySQL en el server ejecutamos en una terminal “sudo apt-get install mysql-server” cuando nos pregunte por la contraseña introducimos cttc2017, instalamos las librerías para poder interactuar con MySQL “sudo apt-get install libmysqlclient-dev”.

Nos movemos a la ubicación etc/mysql/mysql.conf.d/ y modificamos el archivo mysqld.cnf, en dicho archivo vamos a la línea donde pone bind-address = 127.0.0.1 y la modificamos por bind-address = 0.0.0.0, de esta forma permitimos conexiones exteriores.

En el mismo archivo también modificamos el valor de max_allowed_packet a 128MB, al introducir esta modificación le damos permiso a MySQL para que guarde archivos de menos de 128MB, si no se modifica no nos permitiría guardar grafos muy grandes y por ello pesados.

Reiniciamos el servicio de MySQL con “sudo systemctl restart mysql”.

En el caso de que nos conectamos a mysql con el usurio root tendremos que ejecutar estas líneas en mysql, para poder acceder al terminal de mysql escribimos en terminal “mysql –u root –p”:

```
GRANT ALL PRIVILEGES ON *.* TO root@'%' IDENTIFIED BY 'password';

FLUSH PRIVILEGES;
```

## 5.3 Instalamos java 8

Para instalar java 8 la versión de Oracle, primero tendremos que añadir un el repositorio donde se encuentra, introducimos en la terminal “sudo add-apt-repository ppa:webupd8team/java”, a continuación realizamos un update con “sudo apt-get update”, instalamos java 8 con “sudo apt-get install oracle-java8-installer”.

## 5.4 Instalamos Tomcat 8

Nos lo descargamos Tomcat 8 de la página oficial, lo descomprimimos y lo movemos a /etc/tomcat/ (si la carpeta no se encuentra la creamos). Creamos el Daemon de tomcat 8 y

copiamos estas 2 librerías en la carpeta lib del tomcat (jstl-1.2, mysql-connector-java-5.1.40-bin)

Copiamos la carpeta de la página web (se encuentra dentro de build) en la carpeta webapp de tomcat.

## 5.5 Creamos la Base de Datos

Nos logeamos a la base de datos "mysql -h localhost -u root -p"

Creamos la base de datos "create database cttc" e importamos los datos (Tablas, Relaciones, etc) con "mysql -h localhost -u root -p cttc < bdd.sql"

Comprobamos si se ha realizado correctamente, para ello podemos realizar un select de todas las tablas, para seleccionar una base de datos se utiliza el comando "use cttc" en este caso.

## 5.6 Preparación de la aplicación Cliente (Todo en el Server)

Creamos la carpeta uploadFiles en /etc/, movemos la carpeta Client dentro de la carpeta /etc/uploadFiles.

En el archivo ./Client/main.cpp modificamos la configuración de la BDD, para poner la que queremos utilizar. A continuación, compilamos Client con el comando make (contiene un archivo makefile).

## 5.7 Preparación Raspberry PI

Creamos la carpeta uploadFiles en /etc, movemos a este directorio las carpetas Server y Cre_Grafos.

Modificamos el archivo config.cfg, en él se configura la ip del servidor donde se encuentra la base de datos y el número de la raspberry pi siempre empezando desde 0.

Compilamos la aplicación Server con make (contiene un archivo makefile).

Ejecutamos "sudo pip install networkx" para instalar la librería que utilizamos para generar los grafos en Python, más tarde ejecutamos "sudo apt-get install python-matplotlib" con este comando instalamos la librería matplotlib para Python la cual nos permite visualizar grafos entre otras funciones interesantes que podremos utilizar, a continuación instalamos unas librerías que nos permitirán trabajar con matrices de una forma rápida y sencilla "sudo apt-get install python-numpy python-scipy", ejecutamos "sudo apt-get install libmysqlclient-dev" como hemos comentado anteriormente nos permite conectarnos con la base de datos.

Creamos el daemon, para ello movemos el archivo raspserver a /etc/init.d

Ejecutamos "sudo systemctl daemon-reload" para recargar los daemon, ya que sino el que acabamos de mover no funcionaría, lo inicializamos con "sudo systemctl start raspserver" en este punto la aplicación ya enviara el estado de la Raspberry PI al ordenador donde tenemos instalado la página web, de esta forma decidirá enviar el grafo a analizar en la Raspberry PI que tenga más recursos disponibles.

Repetir el procedimiento en las cuatro Raspberry PI.

## 5.8 Preparación Server Cloud (nuestro server)

Movemos a este directorio las carpetas Server y Cre_Grafos a /etc/uploadFiles

Modificamos el archivo config.cfg, en él se configura la ip del servidor donde se encuentra la base de datos en este caso seria 127.0.0.1, ya que la base de datos se encuentra en el mismo ordenador.

En /etc/uploadFiles/Server comentamos la línea que empieza por thread t1 en main.cpp, de esta forma no enviara el estado del Server a la aplicación web, ya que no la tenemos en cuenta.

Compilamos Server con make (contiene un archivo makefile).

A continuación, seguimos los pasos descritos en la sección anterior (Preparación Raspberry PI)

AVISO: Si no se utilizan estas contraseñas y direcciones, la página web fallara, para arreglarlo tendríamos que modificar valores de las clases y volver a compilar la página web, si se desea comprobar si ha funcionado bien se guarda la salida en unos archivos en /var/log.

# 6.0 Trabajo Futuro

Sustituir el protocolo de comunicación http por https, de esta forma la aplicación obtendría mayor seguridad.

Instalar OpenNebula en el Rack de Raspberry PI, te permite crear una cloud privada, más tarde se generarían las gráficas y se compararían los resultados.

Cambiar la base de datos, como el diagrama de Entidad Relación, para dar mayor escalabilidad al sistema.

Comprimir los grafos utilizando algún algoritmo de compresión de texto, de esta forma viajara menos información por la red.

Crear más algoritmos para el análisis de grafos.

Notificación al acabar de generar un grafo, también se podría aplicar a los resultados.

# 7.0 Conclusiones

Al finalizar el tiempo programado para este proyecto, se han podido cumplir los objetivos esperados. Al realizar el apartado de los experimentos, nos hemos dado cuenta del potencial de la Cloud y de la importancia que tiene en nuestras vidas.

Bajar la Cloud al borde de la red, es una muy buena opción para bajar los tiempos de latencia, y con ello conseguir que el tiempo de respuesta sea menor. En el caso de que las Motas realizaran las lecturas de CO2 y los semáforos de la ciudad estuvieran conectadas a la Cloud en el borde de la red.

Se podría llegar a conseguir que el nivel de contaminación de las grandes ciudades, disminuyera considerablemente, ya que podrías redirigir el tráfico en tiempo real y de esta forma reducir la contaminación de una zona.

Si en un futuro la plataforma experimental que se ha realizado en el proyecto, se lograra implementar en el Internet actual, las posibilidades serian infinitas, no solo en el ámbito del IoTWORLD, sino en muchos otros ámbitos.

A la vez también podría ocasionar serios problemas a la infraestructura actual de internet, este es un punto a tener muy en cuenta.

# 8.0 Referencias

[1] A. Bertrand and M. Moonen, "Seeing the Bigger Picture: How Nodes Can Learn Their Place Within a Complex Ad Hoc Network Topology," in *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 71-82, May 2013.

http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6494673&isnumber=6494646

[2] K. Avrachenkov, L. Cottatellucci and A. Kadavankandy, "Spectral properties of random matrices for stochastic block model," *2015 13th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, Mumbai, 2015.

http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7151116&isnumber=7151020

[3] Network Analysis and Modeling, CSCI 5352, Lecture 16, Prof. Aaron Clauset, 5 November 2013

http://tuvalu.santafe.edu/~aaronc/courses/5352/fall2013/csci5352_2013_L16.pdf

[4] IoTWORLD

http://iotworld.cttc.es/