



Autores Scardino Laiber, Lorenzo

Zhang Chen, Xingang

CFGS Desarrollo de aplicaciones multiplataforma (DAM)

IES Nicolau Copèrnic, Terrassa

Tutores:

# ÍNDICE

[ÍNDICE 1](#_Toc10675958)

[1. INTRODUCCIÓN 3](#_Toc10675959)

[1.1 PRESENTACIÓN DEL PROYECTO. 3](#_Toc10675960)

[1.2 OBJETIVOS 3](#_Toc10675961)

[1.3 IDEA 4](#_Toc10675962)

[2. PLANIFICACIÓN 5](#_Toc10675963)

[2.1 REQUISITOS 5](#_Toc10675964)

[2.2 NECESIDADES 6](#_Toc10675965)

[2.2.1 NECESIDADES DE HARDWARE 6](#_Toc10675966)

[2.2.2 NECESIDADES DE SOFTWARE 10](#_Toc10675967)

[2.3 DISEÑO 11](#_Toc10675968)

[2.3.1 FUNCIONAMIENTO DEL PROYECTO 11](#_Toc10675969)

[2.3.2 DISEÑO DE LAS CONEXIONES 12](#_Toc10675970)

[2.3.3 DISEÑO DE LA BD 13](#_Toc10675971)

[2.3.3.1 Firebase 13](#_Toc10675972)

[2.3.4 DISEÑO DE LA APP 17](#_Toc10675973)

[2.4 PRUEBAS 24](#_Toc10675974)

[2.4.1 PRUEBAS ARDUINO 24](#_Toc10675975)

[2.4.2 PRUEBAS FIREBASE Y APP 28](#_Toc10675976)

[2.5 PLAN DE TRABAJO 31](#_Toc10675977)

[3 CONCLUSIÓN 32](#_Toc10675978)

[3.1 PROBLEMAS 32](#_Toc10675979)

[3.1.1. Autentificación en la app. 32](#_Toc10675980)

[3.1.2 Utilización de Firebase en java con Maven 33](#_Toc10675981)

[3.1.3 Maven en consola 34](#_Toc10675982)

[3.1.4 Base de datos con minúscula/mayúscula 35](#_Toc10675983)

[3.2 MEJORAS 36](#_Toc10675984)

[3.2.1 Multiplataforma 36](#_Toc10675985)

[3.2.2 Shield Arduino. 36](#_Toc10675986)

[3.2.3 Pronósticos 37](#_Toc10675987)

[4. ANEXO 37](#_Toc10675988)

[4.1 PLAN DE EMPRESA 37](#_Toc10675989)

[4.1.1 Estrategia de marketing 37](#_Toc10675990)

[4.2 MANUAL DE USUARIO 40](#_Toc10675991)

[4.2.1 Configuración de la Raspberry como servidor Local 40](#_Toc10675992)

[4.2.2 Configuración del microcontrolador ESP8266 42](#_Toc10675993)

[4.2.3 Configuración del programa que conecta Firebase con SQL (Raspberry). 43](#_Toc10675994)

[4. 3 ENLACES A GITHUB 45](#_Toc10675995)

[4.6 DATASHEETS 46](#_Toc10675996)

# 1. INTRODUCCIÓN

## 1.1 PRESENTACIÓN DEL PROYECTO.

Nuestro proyecto consiste en diseñar una pequeña estación meteorológica gestionada por un microcontrolador, que detecta e interpreta medidas como temperatura, humedad o la velocidad del viento, y envía esos datos a una base de datos en la nube, los cuales son transmitidos a una app móvil.

Para cumplir la funcionalidad del proyecto, el microcontrolador incorpora diversos sensores para captar las medidas ya mencionadas, todos los sensores se pueden comprar a un precio muy asequible por internet y presentan compatibilidad con Arduino y derivados.

Dentro de la app, los usuarios podrán ver los distintos datos recolectados con la estación meteorológica, junto a una pequeña recomendación sobre qué medio de transporte utilizar para acudir al centro. Además de eso, también disponen de un cuestionario opcional donde podrán indicar si han seguido el consejo ofrecido por la app o en caso contrario, de qué forma han acudido al centro.

Los datos recogidos por la app, se enviarán a un servidor programado en Java que los almacenará en un base de datos SQL para su futura visualización.

## 1.2 OBJETIVOS

A pesar de haber presentado el proyecto como una app de la que pueden disponer y utilizar los alumnos, el objetivo principal es la obtención de datos a través de un pequeño formulario sobre los medios de transporte que utilizan los alumnos para dirigirse al centro. Buscamos pues, con esta aplicación poder ofrecer datos útiles al instituto en caso de necesitarlos para futuros proyectos.

Pongamos por ejemplo que se quiere pedir al ayuntamiento que aporte en accesibilidad al centro construyendo un carril bici, pues con esta aplicación se dispondrán de las herramientas necesarias para obtener datos sobre el número de alumnos que acude diariamente al centro en bicicleta, datos con los cuales se podría respaldar dicha petición.

Ya hablando de temas más técnicos, el objetivo es utilizar los conocimientos obtenidos a lo largo del grado para:

* Crear un prototipo de estación meteorológica con arduino, hacer un control de los datos y enviar la información ya procesada a una base de datos.
* Mostrar en una app móvil desarrollada para android los datos obtenidos con los sensores de una forma clara y concisa, evitando lo máximo posible el *bloatware****.***
* Crear una base de datos en línea para que los usuarios puedan acceder a la información donde quiera que estén.
* Crear una base de datos local donde guardar la información más relevante y crear formas de visualizar dichos datos.

## 1.3 IDEA

Inicialmente nos encabezamos al final del curso sin tener muy claro que queríamos hacer como proyecto, sol sabíamos que tenía que ser algo relacionado con la electrónica, por lo que acudimos a nuestra tutora Esther en busca de alguna inspiración, y tras conocer nuestro interés por la electrónica, nos ofreció la idea de hacer este proyecto, el cual más tarde, ayudó Jordi Cárdenas a desarrollar y dar un enfoque distinto, así que por ello les damos las gracias.

# 2. PLANIFICACIÓN

Uno de los puntos más importantes a la hora de realizar cualquier proyecto, es la planificación, se debe tener claro desde el primer momento que se quiere hacer y cómo se quiere hacer. Para ayudarnos a conseguir este objetivo, hemos utilizado Trello, un gesto de proyectos online, gracias a él, hemos podido organizarnos mejor el tiempo y también nos ha ayudado a seguir una pauta de trabajo ordenada y continua. (Link a Trello en el anexo).

Lo primero que debemos tener claro para desarrollar nuestro proyecto son los requisitos y las necesidades, qué elementos necesitamos y cómo vamos a usarlos.

## 2.1 REQUISITOS

Para poder calificar este proyecto como exitoso, los requisitos mínimos que debemos cumplir son

* + Tener una estación meteorológica funcional que sea capaz de captar la temperatura, la humedad relativa, calcular la velocidad del viento en Km/h, el nivel de precipitaciones por hora (ml/h), la presión atmosférica, la sensación térmica y la cantidad de polvo que hay en el aire.
  + Disponer de un servidor local funcional escrito en Java que sea capaz de interpretar los datos percibidos por la estación meteorológica y que mismamente, pueda enviarlos a una base de datos remota Firebase.
  + Crear una base de datos en la nube utilizando Firebase que esté disponible en todo momento por los usuarios de nuestra APP
  + Diseñar e implementar una app que según los datos obtenido en firebase recomiende un medio de transporte a los alumnos.
  + Pasar datos de Firebase a SQL exitosamente.
  + Crear una aplicación en Java que sea capaz de interpretar los datos de la base de datos en SQL y mostraros en un formato cómodo para la visualización.

## 2.2 NECESIDADES

Debido a la naturaleza de nuestro proyecto, hemos decidido hacer una distinción entre nuestras necesidades; necesidades de Hardware y necesidades de Software.

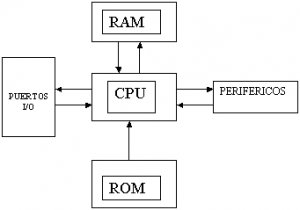
### 2.2.1 NECESIDADES DE HARDWARE

Para poder realizar mediciones sobre el medio y transformarlas en señales eléctricas que el ordenador pueda interpretar, necesitamos dos cosas básicas, sensores y un microcontrolador.

#### 2.2.1.1 MICROCONTROLADOR (MCU)

Un microcontrolador (o MCU) es un pequeño circuito integrado que se compone de tres elementos principales, una central de procesamiento (CPU), memorias RAM y ROM y líneas de entrada y salida (perifericos), y lo más importante de todo, es programable.

Vendría a ser algo así como un pequeño “ordenador”, con muchas comillas.



1Diseño componentes de un microcontrolador

El propósito principal de un microprocesador es la rápida ejecución de un programa (el cual diseñamos en un ordenador y grabamos en la memoria del mcu permanentemente) que realice una tarea específica.

Y es aquí donde viene la diferencia frente a ordenadores o Microprocesadores, los MCU solo son capaces de ejecutar un único programa (o sketch) simultáneamente. Por ello suelen ser empleados para realizar tareas que no requieran de una alta capacidad de cálculo, ni almacenar una gran cantidad de datos, como por ejemplo, el manejo de sensores, relojes, detectores de luz.. etc.

También cuentan con el atractivo de ser altamente modulables, ya que con la ayuda de la gran cantidad de periféricos disponibles, un mismo mcu puede realizar infinidad de tareas con tan solo un simple chip.

**Cual elegir?**

Recientemente el mercado de los microprocesadores ha aumentado exponencialmente su oferta gracias a nuevas alternativas como Arduino que ha apostado por crear no tan solo un hardware , sino también un software fácil de usar y accesible para todo el mundo, y es debido a ello que contamos con una gran diversidad de MCU de los cuales elegir, así que, es necesario crear una lista de requisitos básicos para poder acotar la lista de dispositivos que pueden servirnos en nuestro proyecto.

Los requisitos mínimos son los siguientes:

* Facilidad de uso.
* Especificaciones.
* Conexión Wifi.
* Y por encima de todo asequibilidad.

Teniendo en cuenta estos aspectos y por encima de todo, la recomendación de nuestro profesor Iván Cantón, nos decantamos por un ESP8266, en concreto, el incluido en la placa de desarrollo NodeMCU 3.

NodeMcu es una popular placa de desarrollo basada en el mcu de 32-bits Tensilica L106. Que se emplea sobretodo en proyectos con componentes IoT debido a su módulo wifi integrado, dispone también de unas compatibilidades CASI-total con el software y componentes para arduino, que siendo el más popular en el mercado, es un gran plus. El principal inconveniente de esta placa, eso sí, es que solamente dispone de un pin de entrada analógico por lo que se limita el uso a sensores digitales y como mucho uno analógico.

Más detalles sobre NodeMCU se pueden encontrar en su datasheet en el Anexo.

#### 2.2.1.2 SENSORES

El siguiente paso es ver que mediciones debemos hacer para poder proporcionar los datos adecuados a nuestra estación meteorológica. Típicamente, sería necesario obtener:

* Temperatura.
* Humedad.
* Sensación térmica (fórmula).
* Nivel de precipitaciones.
* Velocidad del viento.
* Presión Atmosférica.

Además de estos datos, a nosotros también nos ha interesado medir la cantidad de luz ambiente y el nivel de partículas en el aire, para así también poder tener en cuenta a la gente que presente alergias al polen o al polvo, porque aunque haga un día fantástico, si eres alérgico al polen, quizás venir en bici no sea la mejor idea.

Sabiendo ya nuestras necesidades y buscando una vez más, los productos con la mejor relación precisión/asequibilidad, hemos optado por utilizar los siguientes componentes:

* DHT22, sensor que permite una medición simultánea de temperatura y humedad.
* BME280, sensor que mide la temperatura, la humedad y la presión.
* Sensores de efecto hall, sensores que producen voltaje cuando detectan un campo magnético.
* gp2y1010 sensor de calidad del aire/ partículas de polvo.
* bh1750 sensor digital de nivel de luz (lux).

De nuevo, la información técnica de estos sensores se encontrará en los datasheets del anexo

#### 2.2.1.3 RASPBERRY PI

Para realizar la segunda parte del proyecto, se va a necesitar un elemento que sea capaz de hacer un control de posibles errores de medición de nuestro microcontrolador (debido a que la capacidad de la memoria del mismo es limitada) y que nos sirva a la vez de servidor local, que en conjunto con un servidor en la red, almacene los datos de la aplicación.

**Que es una raspberry?**

Raspberry Pi es una placa que funciona como un ordenador de bajo coste y al cual complementa ofreciendo pines GPIO, abriendo así la oportunidad de conectar sensores o tro tipo de periféricos .Es comúnmente utilizada en proyectos de electrónica que requieran manejar elementos más complejos y que por ende, no sea posible manejarlo con microcontroladores como arduino. Nosotros necesitamos ejecutar una base de datos SQL y un servidor Java que se conecte con la misma, operaciones que dentro de lo que cabe, son simples, por lo que el uso de un “servidor” en el sentido tradicional es un malgasto de recursos, y a la vez son demasiado complejos para que los ejecute otro arduino, por lo que una raspberry Pi vendría siendo una solución intermedia entre ambos.

### 2.2.2 NECESIDADES DE SOFTWARE

Para la realización del proyecto serán necesarios conocimientos en los siguientes ámbitos:

* Java.
* Android.
* Bases de datos SQL.
* Firebase.
* Conocimientos mínimos de los funcionamientos de las conexiones entre los elementos de una misma red.
* Conocimientos básicos sobre electrónica.
* Conocimientos sobre el manejo básico de sistemas operativos GNU/Linux.

El software necesario para desarrollar el proyecto ha sido:

* Arduino IDE
  + Utilizado para el desarrollo del sketch que hace funcionar la estación meteorológica.
* Netbeans IDE
  + Utilizado para crear el servidor local en Java.
* Android Studios IDE
  + Utilizado para la creación de la aplicación móvil.
* Raspbian
  + Utilizado como SO en la Raspberry pi
* Libre office & Word
  + Como editores de texto para la memoria.
* ThinkerCad
  + Utilizado para planificar los esquemas eléctricos.
* Trello
  + Utilizado como gestor de proyecto.
* Putty
  + Utilizado para conectarse mediante protocolo SSH al servidor local en raspberry.
* Etcher
  + Utilizado para quemar la imagen del ISO de la raspberry en una tarjeta SD.

## 2.3 DISEÑO

### 2.3.1 FUNCIONAMIENTO DEL PROYECTO

La estación meteorológica *per se*, consta de un microprocesador conectado a los sensores que a su vez, sirviéndose de un módulo wifi que ya lleva integrado, se comunica con el servidor local, en el que, a través de un programa Java , se reciben y procesan las lecturas de los sensores.

Al recibir los datos, el servidor realiza una serie de pruebas para controlar los errores y elimina los datos que no se encuentren dentro de un valor lógico. Tras esto, envía los datos a una base de datos cloud en Firebase.

A partir de ahí, los datos ya están listos para ser cargados en los dispositivos móviles de los usuarios, donde tras realizar un sencillo algoritmo, se le aconseja al usuario un medio de transporte para acudir al centro según el tiempo que haya y la distancia a la que viva.

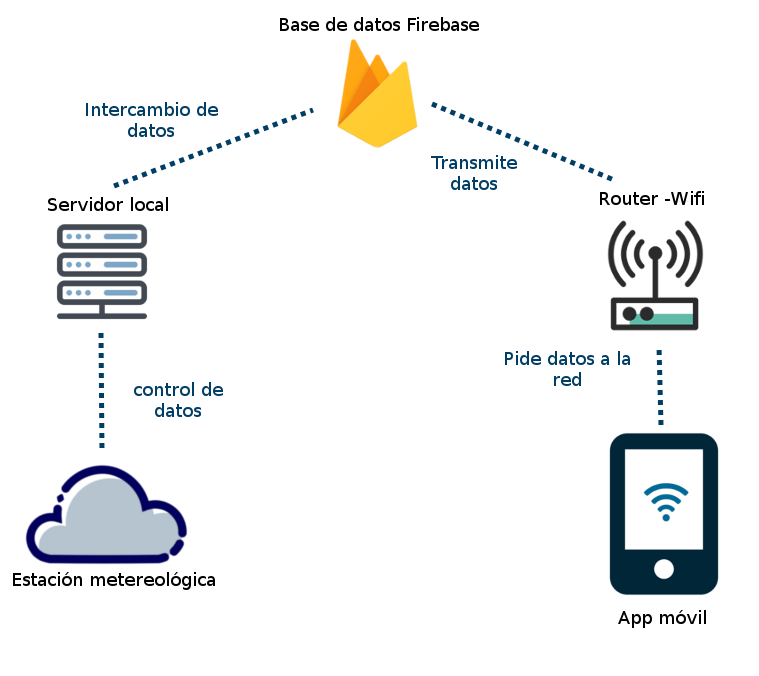
Adicionalmente, el proyecto está complementado por una segunda base de datos en SQL que nos permite almacenar los datos permanentemente para su posterior visualización.

Los motivo tras esta segunda base de datos, que de primeras puede parecer innecesaria son dos; el primero, la comodidad, ya que con nuestros conocimientos es más sencillo obtener los datos de un servidor SQL para crear tablas y gráficas (además de que es más flexible) y por la limitada capacidad de la base de datos de Firebase (al menos la versión gratuita).

Debido a la naturaleza de nuestro proyectos, tenemos la necesidad de almacenar muchos datos en un corto periodo de tiempo en firebase, y para no llenar la base de datos, lo que hacemos, es mantener constantemente en Firebase los datos captados los dos últimos días, y pasar el resto a SQL para poder borrarlas de la nube.

### 2.3.2 DISEÑO DE LAS CONEXIONES

Para que el proyecto funcione correctamente, se deben conectar diversos aparatos entre sí, en este esquema se pretende escenificar dichas conexiones:



2 Imagen estructura conexiones

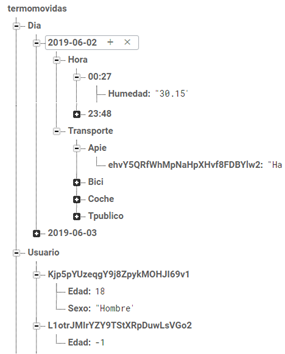
### 2.3.3 DISEÑO DE LA BD

### 2.3.3.1 Firebase

Como ya se ha explicado, la versión gratuita de firebase ofrece un espacio de almacenamiento que no pude cumplir con las necesidades de nuestra aplicación a largo término, pero aún así, su funcionalidad es muy buena y permite tener una base de datos permanentemente en línea.

Y debido a que en la aplicación solo se muestra a los usuarios los datos recopilados la última hora, la utilización de Firebase como base de datos en línea es más que óptima.

Dejando de lado la base de datos en sí, al implementar Firebase en nuestra aplicación también se nos abra acceso a otras herramientas ofrecidas por Google, como autenticación en tiempo real, etc… Funciones que en muchas ocasiones, hacen inclinar la balanza a favor de Firebase frente a otros servicios similares.

En la base de datos se agrupan los datos en dos diferentes apartados, la información recogida por la APP diariamente y el registro de usuarios. su estructura es la siguiente:

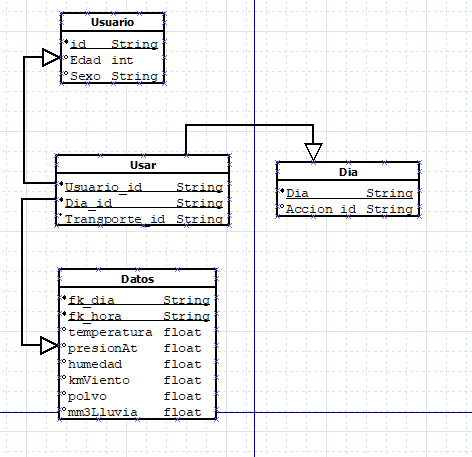
3 Estructura de la Base de datos Firebase

* “Dia” es la raíz e identificador de los datos que se almacenan diariamente en la base de datos.

Presentan un formato YYYY-MM-DD, para poder filtrar con más facilidad los datos.

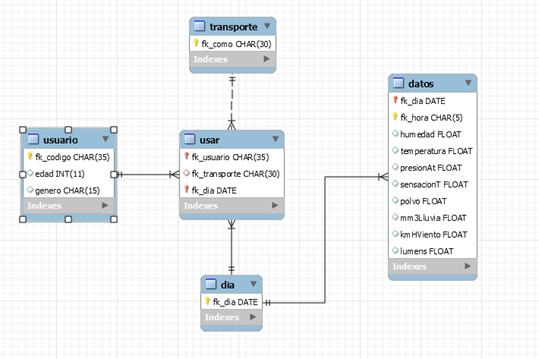
* Por debajo del dia, se encuentra la “Hora”, donde se almacenan los distintos valores captados por la estación meteorológico, como puede ser la temperatura, la humedad o la intensidad luminosa.
* El otro hijo de día es “Transporte”, donde se agrupan los datos sobre los medios de transporte que han utilizados los usuarios ese día.
* “Usuario”, funciona como otra raíz que guarda algunos datos sobre el usuario, que son su edad y género, también cabe la posibilidad de que el usuario no haya accedido a facilitar estos datos, en dicho caso, se puede identificar a los usuarios anónimos igualando su atributo de Edad a menos uno..

#### 2.3.3.2 SQL + subapartados



4 Tabla Relación base de datos

Diseño de los datos en MySQL



5 Estructura de las tablas en SQL

La razón para tener una segunda base de datos es la flexibilidad y control que ofrece sobre el almacenamiento de los datos, ya que no al disponer de esta base secundaria, supone no almacenar la totalidad en Firebase.

Es por ello, en este proyecto se incorpora una segunda base de datos escrita en SQL.

Esta segunda base de datos, se almacena dentro de la Raspberry, la cual además incorpora un conjunto de tareas automatizadas; al encenderse el dispositivo, este ejecuta un programa Java que capta los datos del arduino y los transmite a Firebase y cada cierto tiempo, ejecuta en segundo plano un proceso que elimina los datos de Firebase y los almacena localmente en SQL.

Las tablas más importantes y donde habrá la mayoría de movimiento son;

* “Usar” donde podemos ver los datos del usuario con los diferentes transportes utilizados a lo largo de los días.
* “Datos” que incorpora los datos creados por los sensores de la estación meteorológica.

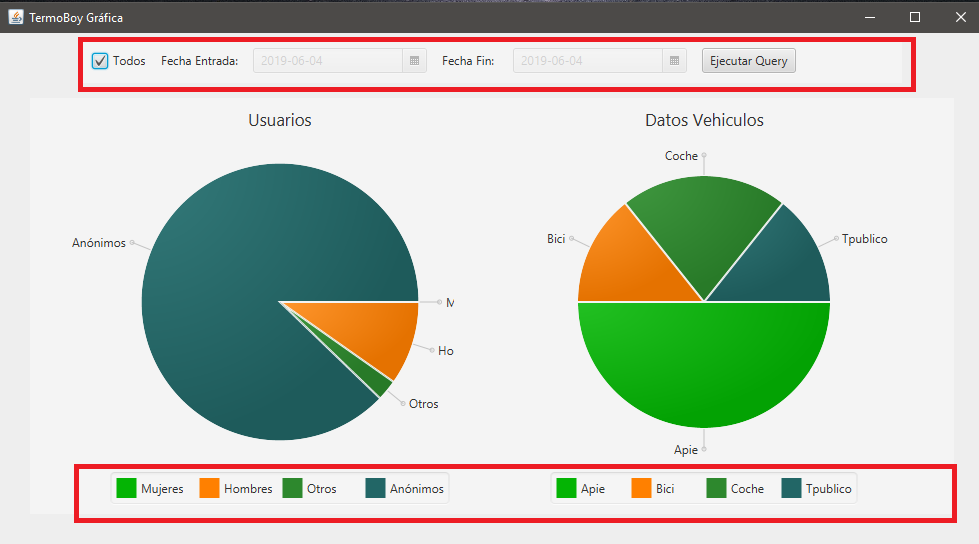
#### 2.3.2.3 App escritorio, análisis de datos

En el proyecto también hay un programa para acceder fácilmente a los datos guardados en la base de datos permanentes, mostrándose en forma de gráficas para el análisis de los usuarios que han utilizados la aplicación móvil y el transporte utilizado.

Modo de búsqueda.

Con el tic de “Todos” localizado en la parte superior puede recuperar todos los datos almacenados y mostrarlas en dos gráficas,

Si desactiva el tic queda activada la búsqueda por fechas, tanto para un solo día o para un rango.

****

6 Imagen del programa para visualizar datos de SQL

### 2.3.4 DISEÑO DE LA APP

#### 2.3.4.1 Interfaces

Para nuestro proyecto queríamos crear una aplicación sencilla, que no tuviese pantallas innecesarias, y que las que tuviese fueran lo más intuitivas posible, por ello, hemos dotado a nuestra app de principalmente 3 pantallas:

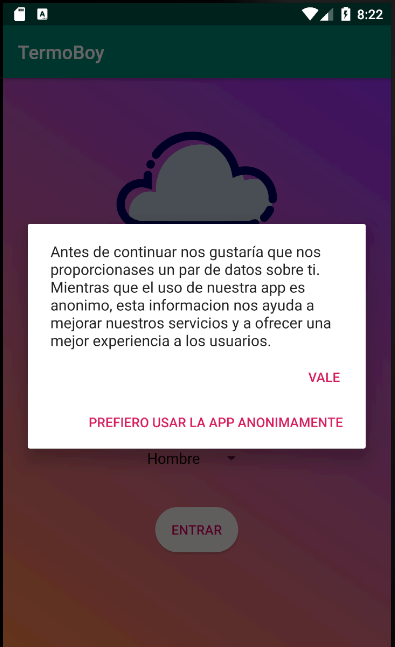
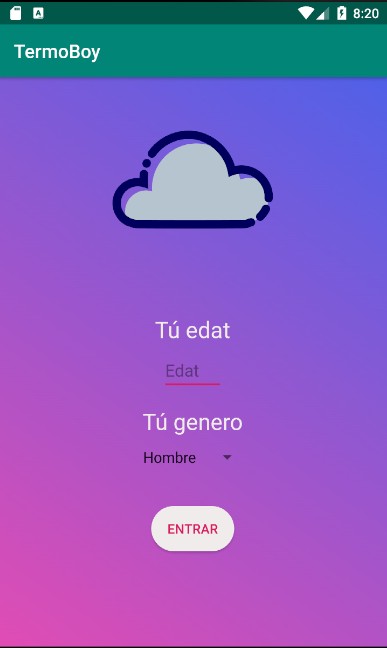
-Una pantalla de Bienvenida, el cual consta de un formulario y de un pop-up preguntandote si quieres rellenar dicho formulario

**MOQUP**

****

7 Modelo Inicio APP 8 Modelo Inicio notificación APP

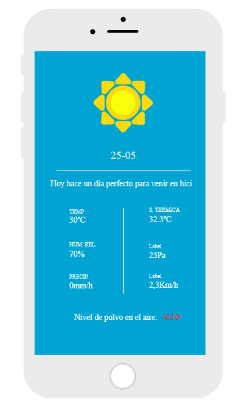
**Resultado Final**

****

9 Imagen Real inicio APP 10 Imagen Real notificación APP

Tras entrar a la app, accedemos a una Tab Activity que consta de dos elementos, una pantalla en la que se muestra información sobre el tiempo y se aconseja como acudir al centro

**Moqup**

****

11 Modelo Principal APP

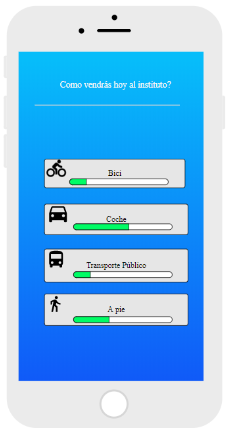
**Resultado final**

****

12 Imagen Real Principal APP 13 Imagen real Principal APP

Y finalmente, una segunda tab view que ofrece un pequeño cuestionario para saber con qué medio de transporte se ha acudido al centro, una vez realizado, se cargarán unas barras que indicarán el porcentaje de alumnos que han acudido con cada medio de transporte.

**MOQUP**

****

14 Modelo Secundario APP 15 Modelo Secundario Animado APP

**Resultado final**

****

16 Imagen real secundario APP 17 Imagen real secundario animado APP

#### 2.3.4.2 Funcionamiento

Una vez vistas las interfaces es hora de explicar su funcionamiento;

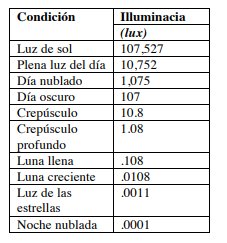
Al iniciar la aplicación se nos mostrará una página inicial, en ella aparecerá un pop-up que nos preguntará si permitimos que la app utilice nuestra ubicación, al aceptar o negar los permisos necesarios, se mostrará aún otro diálogo, en él nos preguntará si deseamos acceder a la aplicación anónimamente o si preferimos rellenar un cuestionario con el pretexto de ayudar a mejorar los servicios ofrecidos por el centro. De elegir esta última opción, pasaremos a la página de bienvenida donde podremos rellenar el formulario.

Seguidamente, entraremos a una tab view, en la cual primeramente veremos una pantalla que muestra información sobre el tiempo actual en el instituto junto a la forma que creemos más aconsejable para acudir al centro. Si deslizamos la pantalla hacia la derecha encontraremos el segundo *fragment* de la *tab activity*, que nos muestra una vez más un cuestionario donde podremos indicar el método utilizado para acceder al centro, al hacerlo, se cargarán unas barras que mostrarán la cantidad (en porcentaje) de usuarios que han utilizado cada medio de transporte.

**Algoritmo para determinar el medio de transporte.**

Una vez leídos los datos de Firebase, toca procesarlos para poder mostrarlos al usuario, en nuestra aplicación, estos datos afectan a 3 componentes, el fondo, el icono que se muestra en la pantalla principal, y el vehículo que se aconseja utilizar.

Los dos primeros son sencillos; el icono se explica solo, si hace sol, se coloca un sol, si llueve, se cambia el icono por el de una nube, etc… Y el fondo solo tiene dos variaciones, fondo oscuro y fondo claro, que se muestre uno o el otro depende de la cantidad de lumens que captemos a través de nuestros sensores.



18 Tabla Condiciones lumínicas

Y finalmente, para determinar qué medio de transporte se recomiendan a los usuarios, hemos creado un sencillo sistema de puntos.

De modo que cada vez que se cargan datos nuevos en la base de datos, la aplicación recorre todas las medidas y según el valor que tengan se les puede restar o sumar puntos a cada medio de transporte.

Al acabar de pasar por todos los datos, el transporte que tenga la mayor cantidad de puntos, será el que determinemos recomendable.

Ejemplo, dadas las siguientes circunstancias hipotéticas:

-Sensación térmica 30ºC

-Velocidad del viento 3.0km/h

-Precipitaciones: 0mm/h

Las puntuaciones quedarían tal que:

1. **Coche** => -8 puntos
2. **Bicicleta** => 8 puntos
3. **A pie** => 6
4. **Transporte Público** => -1

Por lo que se recomienda que los usuarios viniesen en bici.

Aunque antes de realizar cada recomendación se calcula la distancia entre la ubicación del usuario y la del centro (siempre que éste haya dado permisos a la aplicación), si la distancia es mayor a , 5km no se recomendará nunca “venir a pie“, y si es mayor a 10km no se recomendará tampoco “venir en bicicleta”.

Cabe decir que este método no es para nada objetivo ya que depende de la percepción que tengan los desarrolladores de conceptos como “lejos/cerca”, “frío/calor”, etc…

## 2.4 PRUEBAS

### 2.4.1 PRUEBAS ARDUINO

Al realizar el montaje de nuestra estación, antes de conectar cualquier sensor, lo ejecutaremos en un sketch a parte para comprobar su funcionamiento y sólo después se colocaba junto a los otros sensores, es por ello que se ha ido manteniendo un control continuo sobre su funcionalidad, aun así, al finalizar el proyecto, se realizó una prueba unitaria a varios sensores dentro del programa completo, estas pruebas varían según el sensor ya que cada uno de estos tiene una función y funcionalidad distinta

Para probar los sensores realizamos estas pruebas:

#### 2.4.1.1 Sensor de temperatura y Humedad

Primeramente, medimos los datos que nos da a temperatura ambiente, sin ninguna modificación ni variación aplicada al sensor.



19 Imagen muestra prueba 1

**FIGURA Resultados obtenidos sin aplicar ninguna variación al sensor**

Tras esto, realizamos dos pruebas más, una para comprobar la subida de la temperatura y la humedad y otra para comprobar la bajada de temperatura



20 Imagen muestra Prueba 2

**PRIMERA PRUEBA, Echar el aliento a la palca**



21 Imagen muestra Prueba 3

**SEGUNDA PRUEBA, Acercar el sensor a la nevera - Foto resultados**

Estas pruebas se realizaron para ambos sensores de temperatura y humedad (DHT220 y BME280), al final se comprobó el perfecto funcionamiento de ambos sensores, aunque cabe destacar que el DHT220 tarda bastante más en disipar la humedad.

#### 2.4.1.2 Prueba BH1750 (sensor de iluminación)

Debido a que la sensibilidad que muestra ante la luz es la característica principal de este sensor, se realizaron 3 pruebas, la primera, una vez más fue medir sus valores en su estado normal ,en una habitación “moderadamente” iluminada (partiendo de que el sensor devuelve un valor de “0” al taparlo por completo).



22Imagen muestra prueba luz

**RESULTADOS prueba estándar**

y , acercando el flash del movil al sensor.

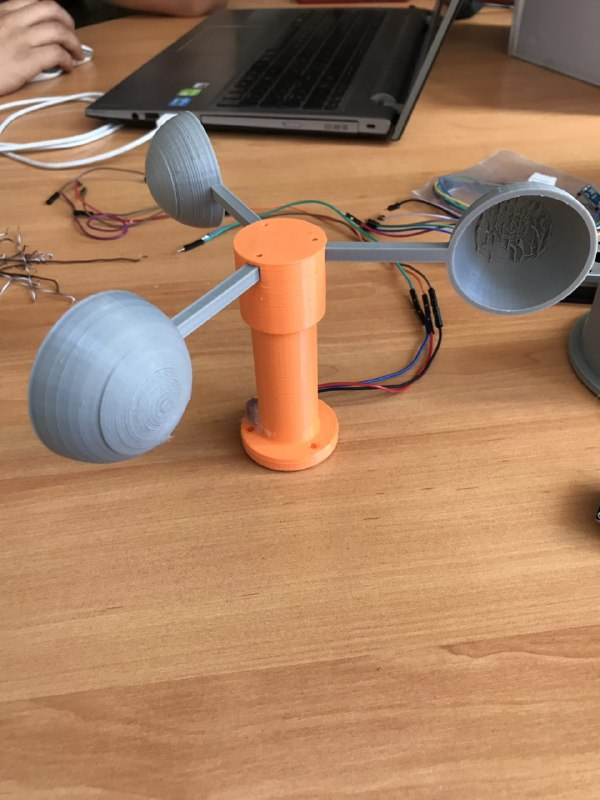


23 Imagen muestra prueba luz con flash

**Resultados de tal**

También intentamos medir los valores que nos daba bajo la luz del sol, desafortunadamente, el valor máximo de luxes al que puede llegar el sensor, es solo la mitad de la que se supone puede captarse bajo luz directa del Sol (r**eferirse a figura 21)**, pero en líneas generales, moviéndonos por puntos con distinta iluminación, el sensor parece funcionar bajo unos valores aceptables.

**Prueba anemómetro y pluviómetro**

****

24 Imagen Anemómetro 25 Imagen Pluviómetro

Por último, hemos juntado las pruebas del anemómetro y el pluviómetro al funcionar ambos aparatos con un sensor HAll y un imán, Realizamos dos pruebas

Prueba de software

Ambos aparatos lo único que hacen es enviar una señal al microprocesador cuando el imán pasa sobre el sensor, por lo que para comprobar que se activase, lo único que hicimos fue pasar un imán continuamente por delante de ambos sensores.



26 iMAN funcionamiento Pluviometro

Pruebas físicas.

Ya comprobado que el software funciona, hay que ver si el sensor puede cumplir su función en las condiciones en las cuales está diseñado a funcionar.

Para simular estos contextos, se llena un vaso de agua y se vierte poco a poco sobre el pluviómetro y para comprobar que funcione el anemómetro, simplemente se acercó un secador a las aspas.

Ambos sensores funcionan correctamente, aunque el anemómetro va un poco suelto y peligra salir disparado ante fuertes vientos.

### 2.4.2 PRUEBAS FIREBASE Y APP

#### 2.4.2.1 Pruebas realizadas en la app

Debido a que los datos se introducen a la app a través del servidor de arduino, el cual ya realiza diversas comprobaciones de los mismos, se descarta la posibilidad de que entren datos malformados o erróneos en la aplicación.

Por lo que simplemente el único testeo que quedan a realizar es el de que todas las iteraciones puedan cumplirse.

La aplicación puede tener 6 iteraciones según el tiempo, 2 según el nivel de luminosidad y 4 según el medio de transporte, para comprobar que fuesen todas posibles, cargamos en firebase datos trucados, seguidamente, se muestran unas medidas de ejemplo para reproducir cada posible iteración.



Iteración de Tiempo 1 - Dia soleado.

* Vientos inferiores a 40km/h.
* Precipitaciones de 0mm/h.
* Nivel de iluminación superior a 10000 lux.

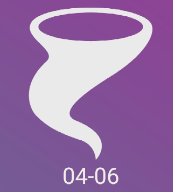


Iteración de Tiempo 2 - Dia lluvioso.

* Cualquier nivel de precipitación superior a 0mm/h.

Iteración de Tiempo 3 - Fuertes vientos.

* Prcipitaciones de 0mm/h.
* Velocidad del viento superior a 40km/h.

Iteración de Tiempo 4 - Vientos Huracanados

* Velocidad del viento superior a 110km/h.

Iteración de Tiempo 5 - Día nublado

* Vientos inferiores a 40km/h
* Precipitaciones de 0mm/h
* Nivel de iluminación inferior a 10000 lux

Iteración de Tiempo 4 - Noche

* Vientos inferiores a 40km/h.
* Precipitaciones de 0mm/h.
* Nivel de iluminación inferior a 40 lux.

**Nivel de luminosidad**

Iteración 1 Fondo claro

Nivel de iluminación superior a 10000 lux.

Iteración 2 Fondo oscuro

Nivel de iluminación inferior a 10000 lux.

**Medios de transporte**

Iteración 1 Coche

* Distancia del centro >10Km
* Día muy lluvioso

Iteración 2 Bici

* Distancia del centro <10Km
* Día no lluvioso
* Sensación Térmica agradable
* Brisa agradable

Iteración 3 A Pie

* Distancia del centro <5Km
* Día no lluvioso
* Sensación Térmica agradable

Iteración 4 transporte Público

* Lluvia moderada .
* Fuertes vientos.
* Sensación Térmica no muy elevada.

## 2.5 PLAN DE TRABAJO

Al finalizar el proyecto, estas han sido las horas que hemos empleado para cada actividad.

|  |  |  |  |
| --- | --- | --- | --- |
| **Tarea** | **Nº Horas** | **Lorenzo** | **Xingang** |
| Planificación del proyecto |  | 9h | 8h |
| Elección y preparación de los recursos |  | 4h | 3h |
| Montaje y conexión de los sensores |  | 2.5h | 6h |
| Programación sketch de arduino |  | 3.5h | 5h |
| Diseño y creación de la base de datos Firebase |  | 4h | 2h |
| Diseño y creación de la base de datos SQL |  | 4.5h | 2.5h |
| Diseño APP móvil |  | 5h | 2.5h |
| Desarrollo App móvil |  | 36h | 10h |
| Instalación SO y software en raspberry |  | 2h | 0h |
| Creación servidor Local Java |  | 3h | 30h |
| Diseño del programa para visualizar datos |  | 0.5h | 3h |
| Programa para visualizar los datos |  | 0h | 15h |
| Creación del programa para migrar datos Firebase - SQL |  | 7h | 5h |
| Corrección errores App android |  | 9.5h | 5h |
| Pruebas y correcciones Arduino |  | 4h | 8h |
| Pruebas y correcciones servidor Local |  | 3.5h | 10h |
| Documentación |  | 9h | 6h |
|  |  |  |  |
| TOTAL |  | 107h | 121h |

# 3 CONCLUSIÓN

Siempre se pueden añadir más componentes a un proyecto y dado el tiempo suficiente, se pueden ir mejorando hasta el infinito, a nuestro entender, con el tiempo dado, ha quedado un proyecto bastante bueno, con sus fallos, sí, pero habiendo cumplido todos los requisitos y objetivos planteados y lo que es más importante ya fuera de la evaluación, es un proyecto con el que nos sentimos a gusto y en el que nos ha gustado trabajar, que al fin y al cabo, es lo que acaba importando.

## 3.1 PROBLEMAS

Afortunadamente, durante la creación de este proyecto, no hemos tenido demasiados problemas, es decir, hemos tenido bastantes problemas, pero no muchos que fuesen complicados o que retrassasen nuestro flujo de trabajo, generalizando, unicamente hemos tenido 4 problemas que nos han llevado más de lo esperado o que nos han hecho replantearnos partes del diseño de nuestro proyecto.

### 3.1.1. Autentificación en la app.

El dato principal que recolecta nuestra aplicación, es el uso de medios de transporte por los alumnos del centro, eso conseguimos hacerlo sin problemas, pero se nos complicó un poco más la cosa cuando decidimos obtener más datos sobre los usuarios, pensamos que podría resultar útil saber datos sobre el sexo y la franja de edad de los usuarios, esto de por sí no es difícil, es simplemente dotar la aplicación de un sistema de registro y login.

Eso es algo que queríamos evitar desde el principio, porque nos resulta un tanto extraño que debas registrarte para utilizar una aplicación que te dice el tiempo que hace en X sitio, aparte de añadir un paso extra de registro podría incomodar a algunos usuarios.

Por lo que se optó por aplicar un registro “Anónimo” usando la función con el mismo nombre que proporciona la base de datos de firebase.

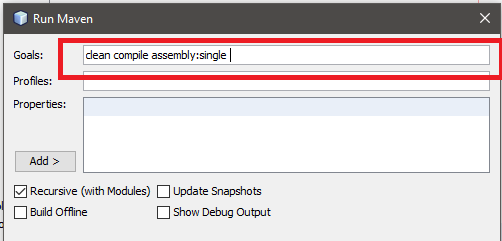
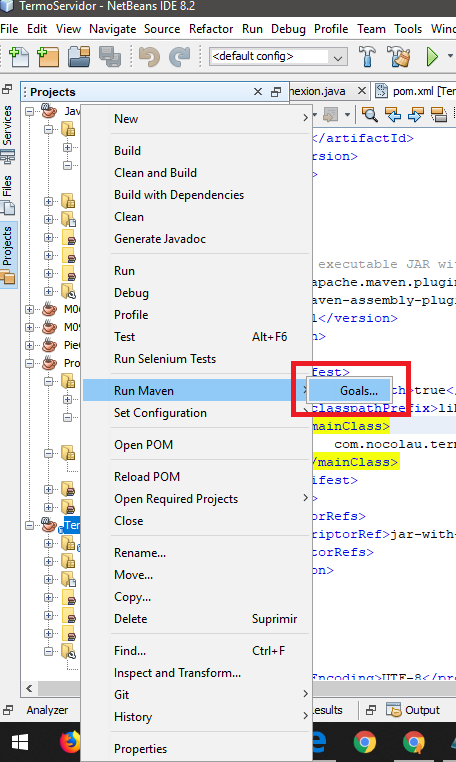
Su funcionamiento es simple, la primera vez que ejecutas la app se te presenta una pantalla de bienvenida con un pop up que te pregunta si deseas rellenar un pequeño formulario para mejorar el funcionamiento de la app o si prefieres entrar totalmente anónimamente, de seleccionar la primera, se te presentará un pequeño formulario tras el cual se te dará acceso a la app con una Id aleatoria a la que se le concatena tu edad y género. De optar por entrar anónimamente, dispondrás de una ID anónima sin género ni edad.

### 3.1.2 Utilización de Firebase en java con Maven

Una de las partes del proyecto que sin duda nos ha llevado más tiempo (no por complejidad sino por falta de conocimiento y documentación), ha sido la programación del servidor local.

Nuestro proyecto consta de un servidor local en java que escucha continuamente los datos mandados por arduino y los traslada a la base de datos en la nube de Firebase, bien, pues para poder conectar Java con firebase es necesario un SDK proporcionado por Google y seguir una serie de pasos documentados en su web, el único problema ,es que hay un pequeño detalle que la documentación, por alguna razón, prefiere obviar, es que todos los métodos de firebase, deben ser ejecutados de forma asíncrona, en hilos.

Una vez entendido esto (gracias a consultar varios foros), el uso del sdk ya se vuelve más sencillo, pero perdimos bastante tiempo siguiendo los ejemplos de la página oficial.



### 3.1.3 Maven en consola

Uno de los primeros componentes del proyecto que desarrollamos fue el servidor local, el cual para realizar testeos, se ejecutaba en el propio ordenador de alguno de los integrantes del grupo, como todo funcionaba bien, lo dejemos un poco olvidado hasta que llegó la hora de trasladarlo al servidor alojado en la raspberry.

Ahí fue cuando saltaron las alarmas al ver la cantidad de errores que surgieron.

*A grosso modo,* el error se producía porque el programa maven no incorporaba los archivos de configuración necesarios para la correcta funcionalidad del programa.

Esto se debe a que al ser un proyecto Maven, al compilar el programa como .jar, todos los recursos que no estén dentro de la carpeta *resources* son ignorados.

(esto incluye las dependencias).

La simple solución fue utilizar los comandos de maven para forzar el empaquetado y ejecutar de forma que acople las extensiones o utilizar la opción “Build with Dependencies” dentro de Netbeans.

### 3.1.4 Base de datos con minúscula/mayúscula

En un caso parecido con el error anterior, tuvimos problemas al migrar la base de datos SQL del ordenador (Windows) a la Raspberry(Raspbian).

Esto se debe a windows no hace distinciones entre mayúsculas y minúsculas a la hora de poner nombre a las tablas y referenciarlas, en cambio Linux si que hace esa distinción.

Archivo a los que afectó el error, todos los archivos creados en este proyecto que conecten con la base de datos a través de java especialmente el programa de copia de seguridad y el programa para mostrar gráficas.

Solución: reescribir el archivo de SQL y todos los afectados descritos anteriormente.

## 3.2 MEJORAS

De llevarse a cabo el proyecto en un futuro, algunas de las mejoras que podrían implementarse son:

### 3.2.1 Multiplataforma

En caso de triunfar, el siguiente paso lógico para cualquier proyecto es expandirse, traduciéndose esto en nuestro ámbito a hacerlo compatible para otras plataformas.

De ser un éxito el proyecto o al crearse cierta demanda, se podría ocupar los sectores de mercado en los que no tenemos ninguna presencia, como por ejemplo el sector IOS, o aún mejor, el sector web, aunque de realizar un porte para esta última plataforma, la aplicación pasaría a ser solo informativa, ya que el formato de cuestionario que posee el proyecto, no funcionará tan bien en una web, otra alternativa es colaborar con el centro y crear un widget para el moodle o la página del IES, ya que usando el formato y estilo de estos, se podrían implementar las votaciones, sin que quedase demasiado forzado.

### 3.2.2 Shield Arduino.

Debido a que la estación meteorológica estará expuesta a los elementos, sería necesario diseñar un shield/contenedor para los sensores y el microprocesador, que fuese impermeable y que estuviera bien sujeto para resistir al viento.

De ser posible, también se le podría añadir una batería y/o un pequeño panel solar para que el módulo fuese eléctricamente independiente.

Esto más que una mejora, se convertirá en requisito si se llega a implantar el proyecto.

### 3.2.3 Pronósticos

Uno de los problemas principales de nuestro proyecto, es por desgracia un problema *lógico.* Y es que solo proporcionamos a lo usuarios información sobre el tiempo actual, con nuestros recursos no tenemos forma alguna de saber qué tiempo hará pasadas unas horas lo que nos impide hacer predicciones sobre el tiempo, una mejora importante, sería utilizar algún servicio online que proporcione información sobre pronósticos del tiempo para complementar nuestra estación.

# 4. ANEXO

## 4.1 PLAN DE EMPRESA

### 4.1.1 Estrategia de marketing

Nuestro proyecto no presentará una estrategia de marketing excesivamente agresiva, debido a que el objetivo es ofrecer un servicio gratuito al instituto, por lo que no buscamos generar beneficio alguno, además, ambos integrantes del grupo somos unos firmes defensores del código abierto y free software por lo que nuestro proyecto, una vez finalizado, estará igualmente disponible en su totalidad en github.

#### 4.1.1.1 Producto

Nuestro producto incluye una estación meteorológica y un servidor que serían entregados al centro y una aplicación, que utilizando los datos recopilados por la estación meteorológica para ofrecer un servicio a los alumnos del centro que a la vez puede proporcionar datos útiles de vuelta al centro.

#### 4.1.1.2 Precio

No planeamos cobrar por nuestro producto, somos unos filántropos, seguiremos el modelo de negocio de otras empresas, como Oracle o SAP, que ofrecen sus productos gratuitamente pero cobran por servicios como personalización, instalación, mantenimiento... etc.

#### 4.1.1.3 Promoción

De todas formas, aun no siendo nuestro objetivo obtener beneficios, está dentro de nuestros intereses que la aplicación se extienda al máximo número de usuarios posibles, y al ser un aplicativo para el instituto, la idea sería colaborar con el centro para darle la máxima exposición posible (evitando ser demasiado intrusivos). Por ejemplo, se podría utilizar la pantalla de anuncios que hay a la entrada del centro para promocionar la aplicación, o si queremos optar por una apuesta más arriesgada, se podrían ofrecer pequeños incentivos a los alumnos para usar la aplicación. Eso ya es trabajo del centro hacer un equilibrio entre lo que se puede ofrecer y lo valiosa que puede ser la información obtenida a través de la App.

#### 4.1.1.4 Distribución

La aplicación se distribuirá inicialmente por la play store de google, de ver que un gran número de usuarios la utiliza y si existe la demanda para ello, se portará la aplicación a IOS.

#### 4.1.1.5 Tabla de costes

|  |  |  |  |
| --- | --- | --- | --- |
| **Recurso** | **Precio Unitario** | **Cantidad** | **Precio total** |
| Raspberry pi 2 | 35.09€ | 1 | 35.09€ |
| NodeMCU(ESP8865) | 5.00€ | 1 | 5.00€ |
| DHT220 | 3.33€ | 1 | 3.33€ |
| BME280 | 4.79€ | 1 | 4.79€ |
| Sensor Hall | 3.32€ | 2 | 6.64€ |
| GP2Y1010 | 10.95€ | 1 | 10.95€ |
| BH1750 | 3.00€ | 1 | 3.00€ |
| Resistencia de 150Ω | 0.10€~ | 1 | 0.10€~ |
| Condensador de 220uF | 0.15€~ | 1 | 0.15€~ |
| Cables varios | 0.15€~ | 20 | 3.00€~ |
| BreadBoar | 3.50€~ | 1 | 3.50€~ |
| **TOTAL** |  |  | **75.55€** |

Como se puede apreciar, el proyecto ha resultado ser bastante asequible, con un mero coste de 72€.

#### 4.1.1.6 Analisis de viabilidad, análisis DAFO

|  |  |
| --- | --- |
| Análisis interno | Análisis externo |
| **DEBILIDAD**  Equipo con poca experiencia desarrollando apps.  Recursos limitados. | **AMENAZA**  El objetivo principal de la app depende de la participación de los usuarios  Hay apps que ofrecen una funcionalidad similar sin pedir nada a cambio. |
| **FORTALEZA**  Buenas aptitudes de programación y diseño.  El tema nos interesa, por lo que le dedicamos muchas horas y con gusto.  Disposición de mucho tiempo para desarrollar y mejorar el proyecto. | **OPORTUNIDAD**  Proyecto que ocupa una necesidad y puede llegar a ser realmente útil.  Coste del proyecto bastante bajo. |

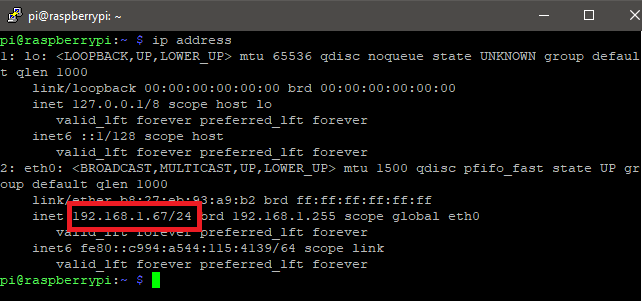
## 4.2 MANUAL DE USUARIO

Pasos a tener en cuenta para configurar los programas necesarios para el correcto funcionamiento del proyecto ( suponiendo que se configure se la misma forma y utilizando los mismos recursos).

Los dispositivos de medición y envió de datos deben conectarse todos a la misma red, junto al programa de muestra de datos.

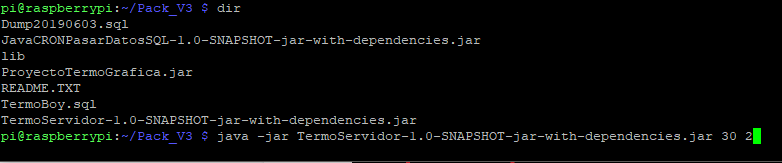
### 4.2.1 Configuración de la Raspberry como servidor Local

* Conexiones, conectar el dispositivo a internet y obtener su dirección IP, en esta caso la dirección es 192.168.1.67



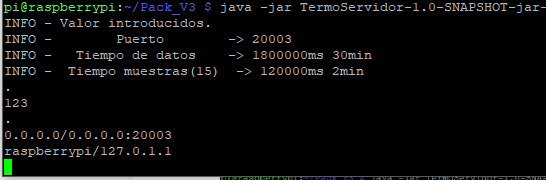
27 Captura de pantalla IP

* Ejecutar el servidor con el nombre de “TermoServidor-1.0…”, los números siguientes son para personalizar el tiempo de espera entre un dato y otro.
  + “30” es el intervalo en minutos que deja entre cada subida de datos a Firebase.
  + “2” es el tiempo en minutos durante el cual el microcontrolador ESP8266 envía datos a la Raspberry.
* NOTA, si no se pone ninguna opción o esta es errónea, el programa se ejecutará con los valores predeterminados.



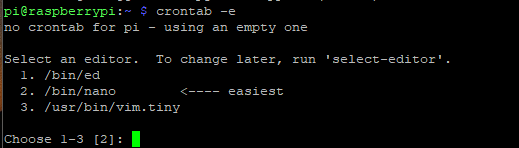
28 Captura pantalla Archivos

* Pre visualización, si el resultado es similar prosigue a la siguiente configuración.



29 Captura pantalla Ejecución

* Ejecutar el servicio de forma automática tras el arranque del sistema: descargamos el plugin de “gnome-schedule” y editamos el archivo “crontab” con el comando que queremos que se ejecute y el intervalo de ejecución del mismo.

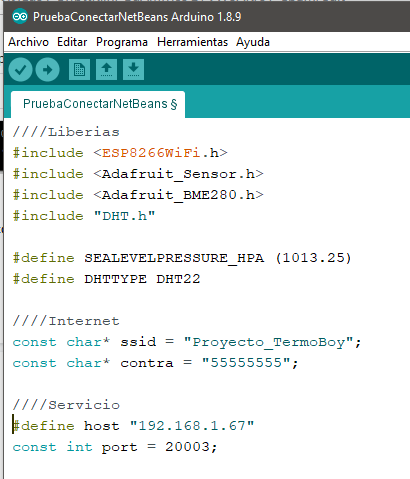


30 Captura pantalla Ejecución automática



### 4.2.2 Configuración del microcontrolador ESP8266

Antes de nada, es preciso el editor de texto Arduino para poder subir el código al microcontrolador.

* Configuración de la conexión: modifica o incluye la wifi a conectar del procesador es importante que sea a la misma red que el servidor.

31 Captura pantalla IDE Arduino

* Servidor, para finalizar la configuración del ESP8366, modifica la dirección de la IP por la dirección de la Raspberry dentro de la misma red..
* Ejecución, para incorporar el fichero al dispositivo clica el botón dirección derecha.

#### 4.2.2.1 Posibles errores

En el sketch de arduino hemos programado el Led incluido en la propia placa para ayudar a debugar el programa señalando rápidamente los posibles fallos.

* El led del dispositivo se mantiene iluminado.
  + - La dirección de la red inalámbrica o la contraseña son erróneas.
* El led parpadea una sola vez.
  + - El servidor no está activado o la dirección del servidor es errónea.
* El dispositivo enciende el led y parpadea dos veces.
  + - Algún sensor no funciona, comprueba las conexiones.
* El dispositivo enciende el led y parpadea continuamente.
  + - El microprocesador tuvo un colapso, para solucionar pulsar el botón “RESET”.

### 4.2.3 Configuración del programa que conecta Firebase con SQL (Raspberry).

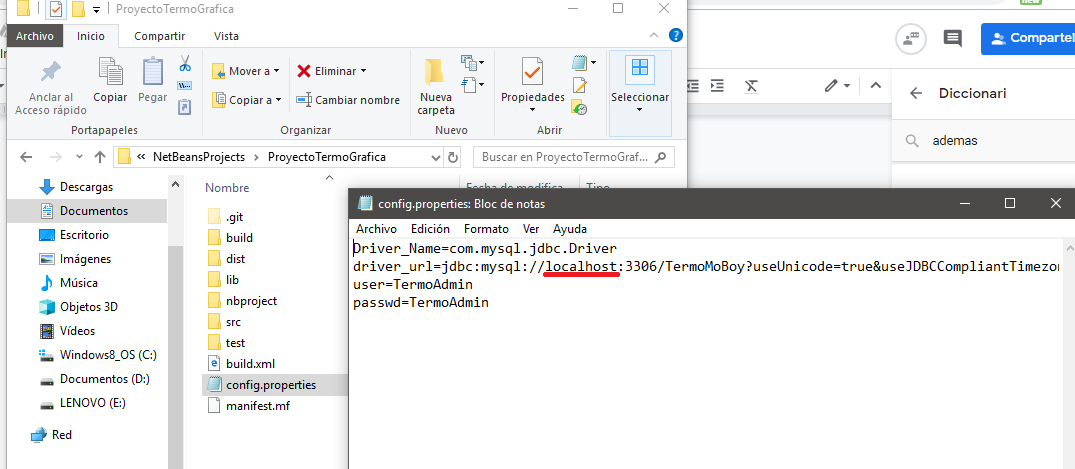
* Para configurarlo, una vez más editamos el archivo “crontab” con el comando que queremos que se ejecute y el intervalo de ejecución del mismo.



32 Captura pantalla Ejecución automática 2

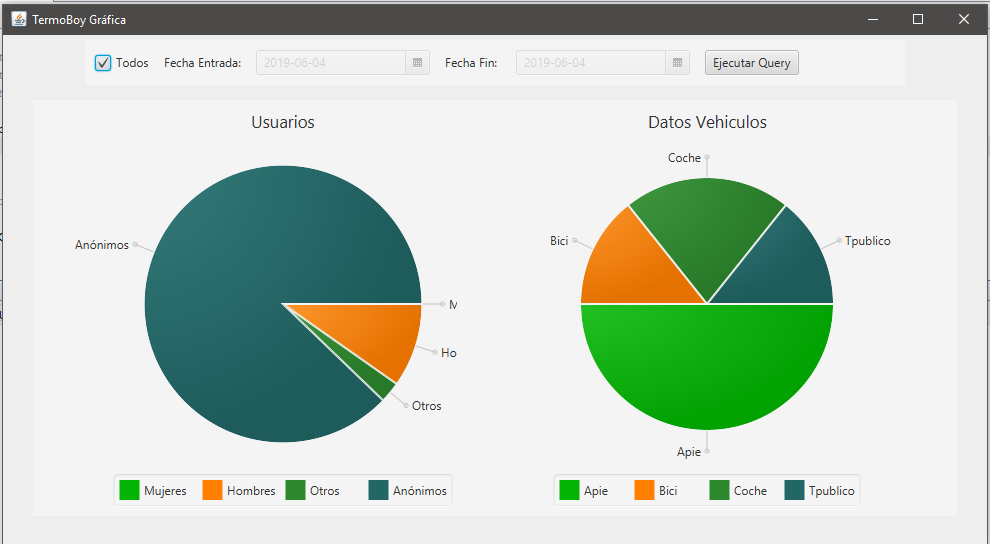
Configurar el programa para ver los datos en una gráfica.

* Conexión, Accede al fichero y modifica el archivo con nombre “config.properties”, en la línea marcada en rojo introducepor la dirección IP de la Rasberry que en esta ocasión es 192.168.1.67



33 Captura pantall configuración APP gráfica

* Ejecutar desde IDE netbeans, clicar el botón “F6” para iniciar el programa.
  + Ejecutar desde terminal, antes de ejecutar es necesario recompilar el proyecto.
  + Visualización, Si las configuraciones son correctas se debe mostrar esta pantalla.



34 Imagen APP gráfica

Configurar la aplicación del usuario.

* Conexión, para la aplicación móvil no es necesario ninguna configuración extra porque está conectada a la base de datos Firebase.
* NOTA, si tiene la aplicación con extensión APK debe ir a configuraciones y aceptar los términos de uso para poder descargar aplicaciones que no sean de google play.

## 4. 3 ENLACES A GITHUB

APP android.

<https://github.com/lscardino/TermoBoy.git>

Controlador para el servidor SQL + sketch de arduino.

<https://github.com/lscardino/ProyectoM14.TermoCosas.git>

Programas para ejecutar con CRON.

<https://github.com/lscardino/termoBoyCRON.git>

Programa para aplicación gráfica.

<https://github.com/5hinko/TermoGraficas.git>

4.4 ENLACE A TRELLO

4.5 WEBGRAFIA

<https://programarfacil.com/blog/arduino-blog/estacion-meteorologica-solar-arduino-sigfox/>

<https://programarfacil.com/>

<https://www.raspberrypi.org/forums/viewtopic.php?t=101982>

<https://www.instructables.com/id/How-to-Interface-With-Optical-Dust-Sensor/>

<http://gidahatari.com/ih-es/tutorial-del-instalacion-y-puesta-en-marcha-de-un-sensor-de-polvo-en-arduino>

<https://roboindia.com/tutorials/nodemcu-MAC-address-arduino>

<https://www.noao.edu/education/QLTkit/es/Safety_Activity_Poster/LightLevels_outdoor+indoor_es.pdf>

<https://techlandia.com/dibujar-linea-punteada-gimp-como_200150/>

<https://www.esp8266.com/wiki/doku.php?id=esp8266_gpio_pin_allocations>

<https://www.esp8266.com/viewtopic.php?f=32&t=8867>

<https://www.raspberrypi.org/forums/viewtopic.php?t=51190>

<http://raspberrywebserver.com/sql-databases/using-mysql-on-a-raspberry-pi.html>

<https://forum.arduino.cc/index.php?topic=420699.0>

<https://www.adafruit.com/product/2652>

<https://www.handsontec.com/pdf_learn/esp8266-V10.pdf>

<https://www.embeddedadventures.com/datasheets/BME280.pdf>

<https://www.mouser.com/ds/2/348/bh1750fvi-e-186247.pdf>

<https://www.miun.se/siteassets/fakulteter/nmt/summer-university/sharp-dust-sensor-and-arduino-_-sensorapppdf>

<https://www.zonamaker.com/electronica/intro-electronica/teoria/resistencias-de-pull-up-y-pull-down>

## 4.6 DATASHEETS

[**https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf**](https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf)