Universidad de Alcalá
Escuela Politécnica Superior

Grado en ingeniería de computadores

Trabajo Fin de Grado

Escuela Politécnica Superior

Sistema *SCADA*-Web comunicado mediante *Proxy* con servidor de sensores por protocolo *CoAP*.

Autor: David Patiño Mochales

Tutor: Juan Manuel Miguel Jiménez

2015

UNIVERSIDAD DE ALCALÁ

Escuela Politécnica Superior

Grado en Ingeniería de Computadores

Trabajo Fin de Grado

Sistema *SCADA*-Web comunicado mediante *Proxy* con servidor de sensores por protocolo *CoAP*.

Autor: David Patiño Mochales
Director: Juan Manuel Miguel Jiménez

Tribunal:

Presidente: Ignacio Bravo Muñoz

Vocal 1º: Julio Pastor Mendoza

Vocal 2º: Juan Manuel Miguel Jiménez

Calificación:_____

Alcalá de Henares a, de Septiembre del 2015

UNIVERSIDAD DE ALCALÁ

Escuela Politécnica Superior

Grado en Ingeniería de Computadores

Trabajo Fin de Grado

Sistema *SCADA*-Web comunicado mediante *Proxy* con servidor de sensores por protocolo *CoAP*.

Autor: David Patiño Mochales
Director: Juan Manuel Miguel Jiménez

Tribunal:

Presidente: Ignacio Bravo Muñoz

Vocal 1º: Julio Pastor Mendoza

Vocal 2º: Juan Manuel Miguel Jiménez

Alcalá de Henares a, de Septiembre del 2015

Sistema *SCADA*-Web comunicado mediante *Proxy* con servidor de sensores por protocolo *COAP*.

Trabajo Fin de Grado

David Patiño Mochales

Tutor: Juan Manuel Miguel Jiménez

Resumen:

Un sistema *SCADA* es la manera de observar de un vistazo una gran cantidad de datos que están siendo generados en un proceso industrial. El conocido como"Internet de la cosas" es una nueva forma de ver la tecnología. Y puesto que todo está conectado a Internet generando datos, esta enorme cantidad de datos debe ser adquirida, supervisada y almacenada.

Este trabajo fin de grado trata de crear un sistema *SCADA*-Web para captar, supervisar y almacenar los datos de una red de sensores atmosféricos. El sistema es capaz de generar históricos y visualizar alarmas de los datos recibidos. Al mismo tiempo, ofrece la posibilidad de administrar la captura de datos y la gestión de los usuarios que los utilizan.

Palabras clave

CoAP, *HTML*, *JavaScript*, *CSS*, *PHP*, *Proxy*

2015

SCADA -Web system connected by a *Proxy* to a sensor server using *CoAP* protocol .

Final Project

David Patiño Mochales

Tutor: Juan Manuel Miguel Jiménez

Abstract:

A SCADA system is the way to see at a glance a lot of data being generated in an industrial process. Internet of things is a new way of looking at technology, everything is connected to the Internet while data is generated. This large amount of data is acquired, monitored and stored.

This final project deals with a SCADA-Web system to capture , monitor and store data from a network of weather sensors. The system is able to generate alarms and display historical data received. It provides the ability to manage data capture and user management.

Keywords

CoAP, *HTML*, *JavaScript*, *CSS*, *PHP*, *Proxy*

2015

Agradecimientos

A mi mujer, por ese apoyo incondicional durante estos duros años en los que han convivido muchas cosas, demasiadas diría yo. Todos esos cambios que hemos sido capaces de aguantar juntos, nos han hecho más fuertes. Si hemos pasado estos años, ya no hay quien nos pare. Por todas esas tardes y fines de semana en los que hemos estado inmersos en este proyecto de estudios. Te quiero, eres la mejor.

A mi hijo, que crece muy rápido, para que siga haciéndolo.

A todos los profesores que me han apoyado durante todos estos años, ayudándome en tantas ocasiones, en esas tutorías a las 20:00 de la tarde por no disponer de más horas en el día, mil gracias.

A Juan Manuel, por ser un magnifico tutor y profesor y por ayudarme con este proyecto.

Índice General

Capítulo 1

Capítulo 2

Capítulo 3

Capítulo 4

Capítulo 5

Capítulo 6

Capítulo 7

Capítulo 8

Capítulo 9

Capítulo 10

Índice de ilustraciones

Índice de código fuente

Índice de tablas

Capítulo 1

Introducción

En este Capítulo de la memoria se describe a modo de introducción el sistema *SCADA*-Web. Se hará una breve presentación global del sistema, explicando las características y funcionalidades principales del mismo. Después, se explicará el proceso de desarrollo seguido para la creación del sistema. Por último, se tratarán los objetivos que se persiguen en este trabajo de fin de grado. Mostramos visión general de aplicación web en la siguiente figura.

Figura 1 - *SCADA*-WEb

Presentación

La necesidad de recopilar datos de las múltiples variables que nos rodean nos lleva a diseñar sistemas que faciliten esta labor. El principal objetivo del proyecto es crear un sistema de supervisión, control y adquisición de datos *SCADA*[1].

El sistema creado para este TFG consiste en una supervisión, control y almacenamiento de datos, implementado mediante una aplicación web que permite dichos cometidos.

El sistema de adquisición de datos, que lee las variables ambientales, se realiza en un proyecto paralelo. Este sistema de adquisición, confecciona una red de sensores, que centraliza la información en un servidor de su propia red. Con dicho servidor, el sistema creado en este TFC se comunica mediante el protocolo *CoAP*, estandarizado para "el Internet de las cosas" (IoT). A raíz de esa comunicación, el sistema *SCADA* alimenta de datos una BBDD y muestra la información de los sensores en la web.

La visualización se puede realizar con cualquier dispositivo que tenga acceso a un navegador web. Mediante *CSS* la web se adapta a cualquier tipo de pantalla donde se visualicen los datos. Además se ha creado una app para *Android* lo que facilita el visionado en este tipo de dispositivos.

El sistema realiza el manejo de los datos, creando históricos, alarmas y estadísticas que pueden verse de manera liviana y rápida. El entorno web está diseñado para observar los datos y para administrar el sistema.

[1]SCADA, acrónimo de Supervisory Control And Data Acquisition (Supervisión, Control y Adquisición de Datos)

Objetivos

El principal objetivo del proyecto es comunicar dos servidores: el servidor web y el servidor de motas (sensores). Para realizar esta comunicación es necesario un elemento intermedio al que llamamos *Proxy.*

Se trata de crear una aplicación web con distintas funcionalidades para la gestión del *Proxy* y de los datos guardados por él. La interface web tiene que ser capaz de visualizar los valores actuales leídos por los sensores, visualizar los históricos de los datos, filtrarlos por fecha, mostrar toda la lista completa de los mismos y ver alarmas cuando los valores leídos se salen de los límites establecidos.

Además, desde la aplicación web se tienen que poder administrar los valores de configuración del *Proxy,* así como indicar a este las direcciones de los sensores, los valores máximos y mínimos y los intervalos de lectura. También se deben de insertar botones con función de arranque y parada del *Proxy* y gestionar la base de datos desde la web. El sistema de alarmas integrado en la aplicación web se actualiza automáticamente mientras el usuario ve la pantalla.

Los datos deben visualizarse divididos por sensor, mostrándolos en modo tabla y en modo gráfico. Estos datos deberán ser editables para modificar, si es necesario, los datos erróneos que dan los sensores en ciertas pruebas.

Otro de los objetivos es diseñar un *Proxy* para la lectura de sensores mediante *CoAP* que permita guardar los datos en la base de datos. Este programa debe de configurarse con los parámetros guardados en la base de datos que hayan configurado en el administrador web.

En definitiva, se trata de crear un sistema de supervisión, control y adquisición de datos que resulte útil y versátil y que cuente con una buena interface para los usuarios.

Proceso de desarrollo

El desarrollo del proyecto se realiza por capas para poder abstraer el sistema y trabajar así de una forma más sencilla. No es otra cosa que aplicar el conocido y popular método del "Divide y vencerás".

A continuación, se indicarán los puntos de desarrollo del sistema:

- Estudio del protocolo *CoAP*. Conocer el protocolo de interconexión entre el servidor de motas y nuestro *Proxy*. Elegir el método de transmisión que resulte más adecuado para la finalidad del proyecto (Get, Post, observe).
- Estudio del *Proxy*. Estudiar qué lenguaje de programación y tecnología son los más útiles para nuestro sistema.
- Estudio del lenguaje del servidor web. Conocer diferentes alternativas y sus características.
- Estudio del servidor del web. Comparar características y elegir el que mejor se adapte al proyecto.
- Estudio de la BBDD. Conocer diferentes servidores de bases de datos *SQL*, así como conocer sus características y elegir la que converja bien con nuestro sistema web y *Proxy*.
- Estudio de los sistemas *Android*.
- Instalación y configuración del servidor base de datos.
- Instalación y configuración del servidor web.
- Instalación de la API de diseño del *Proxy*.
- Instalación de la API para diseño en *Android*.
- Diseño de una base de datos acorde con las especificaciones del proyecto.
- Diseño de la aplicación web.
- Diseño del servidor web, mostrando datos necesarios y administrando el sistema.
- Diseño del *Proxy*.
- Diseño de una aplicación *Android*.

Capítulo 2

Estudio del sistema

El sistema que se propone consta de varias partes:

El *Proxy* es un software que está en constante ejecución, con la finalidad de leer los datos del servidor de sensores en los intervalos configurados y posteriormente guardarlos en una base de datos.

El servidor web lee los valores de la base de datos y los muestra en la web correspondiente. Además, a través del servidor web, se configuraran las características del *Proxy*, los intervalos de lectura, la instalación de sensores nuevos, la dirección de sensores, etc. Es decir, que la administración del sistema se realizará mediante el entorno web.

El diseño de una base de datos potente, que tenga la suficiente capacidad de soportar la gran cantidad de datos que necesitamos para conseguir nuestro objetivo, forma parte del cometido del proyecto.

El increíble crecimiento en los últimos años de los sistemas embebidos móviles bajo el sistema operativo *Android*, nos hace plantearnos la necesidad de crear una app que funcione en este sistema.

Partimos de la base de que la elección del sistema servidor está elegida. Todo el sistema se va a montar sobre un servidor de Apple instalado sobre el sistema operativo Mac Osx 10.9.5.

Otro punto de partida en este proyecto, es seleccionar un software con licenciamiento libre de uso, de manera que no incremente en exceso el coste del proyecto con el pago de licencias propietarias.

Todo el software instalado para este proyecto debe ser multiplataforma incluyendo Mac Osx. Desde estos puntos de inicio empezamos a estudiar el sistema y a elegir los programas necesarios para realizar el conjunto del mismo.

La mejor forma de entender el enfoque global de la aplicación es a partir del esquema que se expone a continuación.

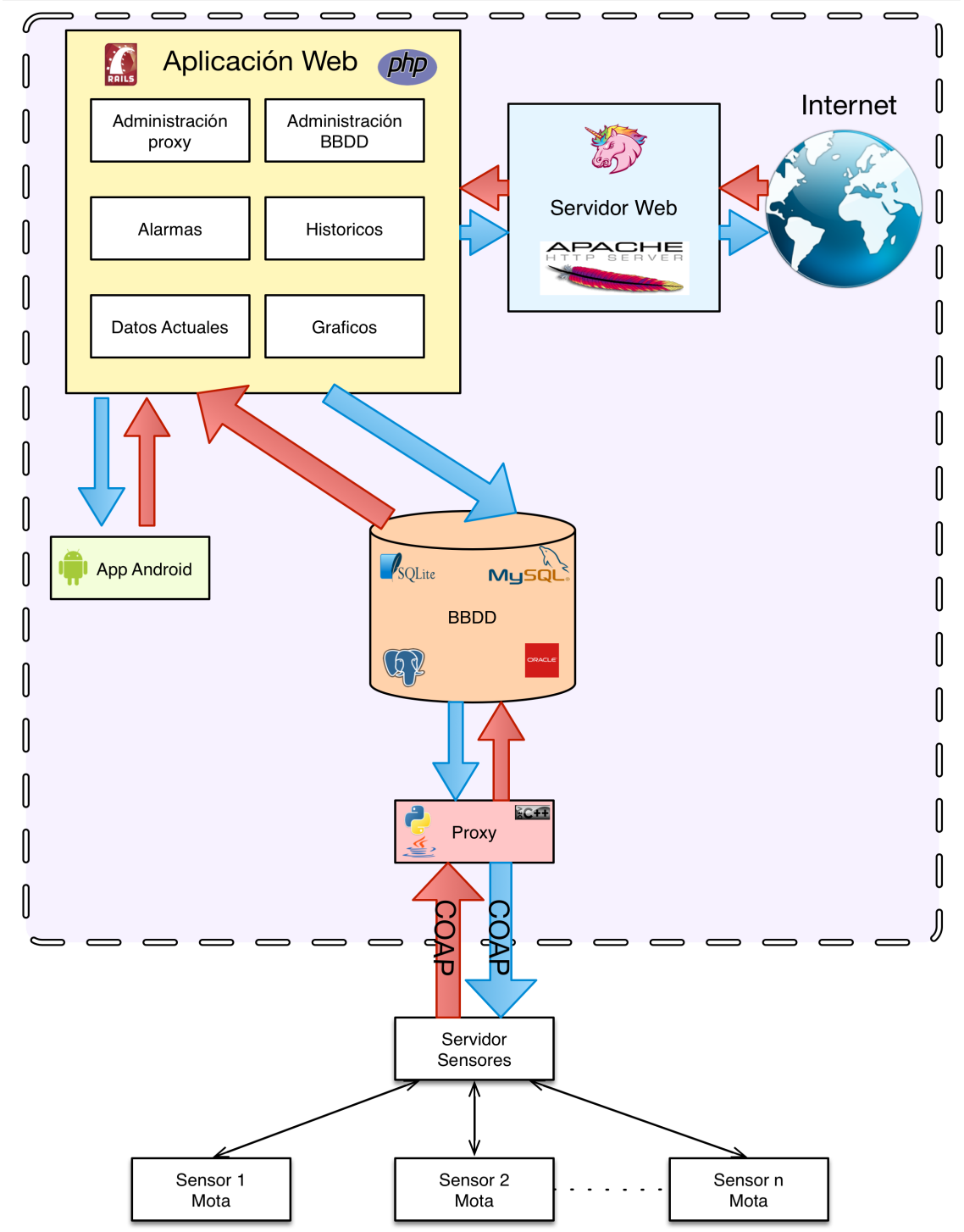


Figura 2 - Esquema general

Estudio del protocolo *CoAP*

CoAP es un protocolo genérico para actuar en sistemas de microcontroladores de bajo consumo y un número limitado de recursos, mejorando así la eficiencia energética de los sistemas y dando más autonomía a los microcontroladores para poder ocupar sus recursos en otros procesos programados.

Está basado en la arquitectura *REST*, pensada para sistemas distribuidos. En esta arquitectura existen clientes y servidores. Los clientes envían peticiones a los servidores y estos responden con una representación del recurso solicitado.

Ofrece una serie de características necesarias para las aplicaciones M2M como el descubrimiento de recursos alojados en los diferentes servidores de la red, un soporte a las comunicaciones multicast y el intercambio asíncrono de mensajes. Las principales **características** del protocolo *CoAP* son:

- Protocolo web con los requerimientos para aplicaciones M2M.
- Conexión UDP con fiabilidad opcional y soporte de transmisiones unicast y multicast.
- Transmisión asíncrona de mensajes.
- Bajo overhead y facilidad para mapear a HTTP.
- Identificación de recursos mediante URI (Uniform Resource Indentifier) y Content-Type.

El modelo de interacción de *CoAP* es parecido al de cliente/servidor del modelo HTTP, con la diferencia de que en el *CoAP* los mensajes se envían de forma asíncrona usando el protocolo UDP*[2]*, mientras que en el *HTTP* se establecen conexiones TCP*[3]* para el envío de peticiones y respuestas.

Una petición en *CoAP* es enviada por un cliente sobre un recurso almacenado en un servidor y la respuesta de este contiene un código que indica si la petición era válida o si, por el contrario, existe algún problema. La respuesta puede contener también una representación del recurso. Vemos estructura de capas OSI en la figura 3.

[2] UDP:User Datagram Protocol
[3] TCP:Transmission Control Protocol

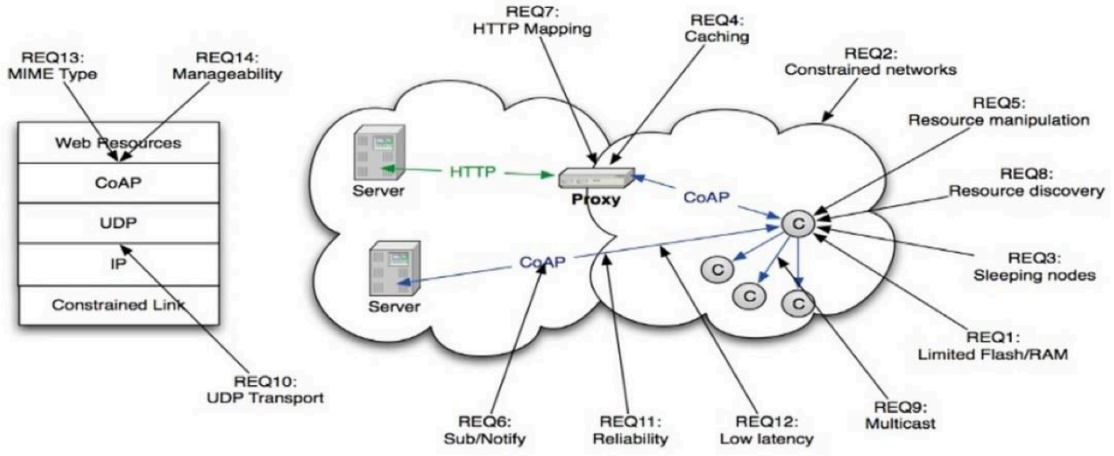


Figura 3 - Capas OSI de *CoAP*

Mensaje

El protocolo *CoAP* define cuatro tipos de mensajes con una estructura igual en todos. Los mensajes están formados por una cabecera de un tamaño fijo, por un número variable de opciones y por un Payload(contenido del mensaje), siendo estos dos últimos opcionales.

Los cuatros mensajes *CoAP* son:

- ***Confirmable* (*CON*):** Mensajes que requieren una confirmación de recepción (mensaje *Acknowledgeme*nt) por parte del destinatario.
- ***Non-Confirmable* (*NON*):** Mensajes que no requieren confirmación.
- ***Acknowledgeme*nt (*ACK*):** Mensaje que se envía para confirmar la recepción de un mensaje de tipo *CON* o bien para responder a una petición de tipo GET.
- **Reset (RST):** Mensaje de respuesta a un mensaje *CON* o *NON* recibido; pero que el receptor es incapaz de procesar, aunque su contenido sea correcto. Esta situación se da cuando el dispositivo receptor se reinicia y pierde información del estado que le permite interpretar el mensaje recibido correctamente.

Cabecera

La cabecera de un mensaje *CoAP* es el único elemento que debe estar siempre presente. En ella es donde se define el tipo de mensaje que se envía. La cabecera ha de ser igual para cualquiera de los tipos de mensaje que veíamos en el apartado anterior. Mostramos la cabecera CoAP en la figura 4.

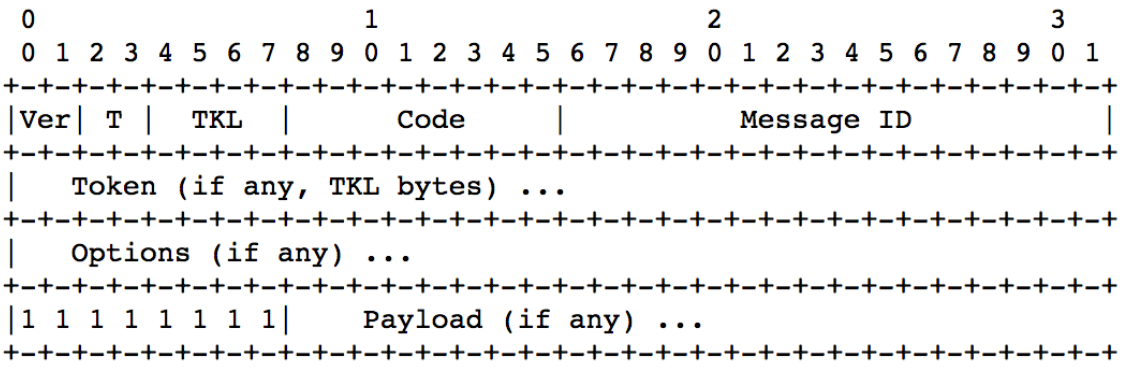


Figura 4 - Cabecera de mensaje *CoAP*

- ***Ver (Version*):** 2 bits que indican la versión de *CoAP*. Las implementaciones de esta especificación deben establecer este campo con el valor 1.
- ***T (Type*):** indican el tipo de mensaje: *CON*, *NON*, *ACK* o RST.
- ***TKL (Token Length*):** indica la longitud del campo *Token*, que está a continuación de la cabecera. La utilidad del *Token* se explicará más adelante.
- ***Code* :** indica si el mensaje es una petición (valores 1-31), una respuesta (64-191) o un vacío (0). Si es una petición, además indicará el método usado (*GET, POST, PUT* o *DELETE)* y si es una respuesta añadirá *Response Code*.
- ***Message ID*:** es un valor que identifica al emisor del mensaje.

Opciones

Las opciones en un paquete *CoAP,* aparecen justo después de la cabecera. Estas deben estar ordenadas mediante su *Option Number.* Los *Option Number* impares son opciones críticas y los pares son opcionales. No se habla de opciones obligatorias. La diferencia entre opciones críticas y opcionales viene dada en cómo se trata el paquete cuando se recibe una opción no reconocida. En otras palabras, el tratamiento en caso de fallo en la recepción. A continuación, observamos estructura de opciones CoAP.

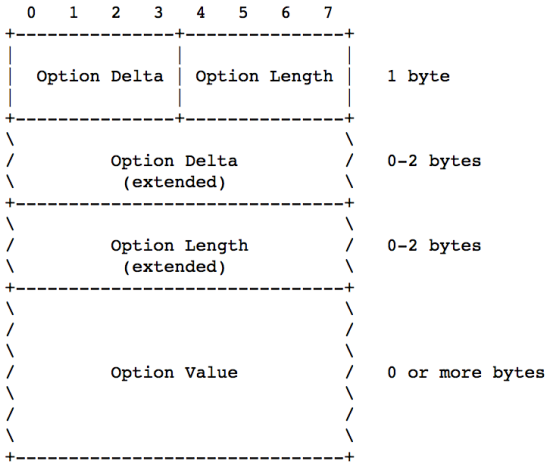


Figura 5 - Formato de Opciones *CoAP*

Posteriormente, se detallan atributos de las opciones de CoAP:

- ***Option Delta*:** Indica la diferencia entre el *Option Number* de la opción actual y la anterior. Para la primera opción insertada, este campo vale 0 y se utilizará en recepción para recuperar el *Option Number* de las sucesivas opciones contenidas en el mensaje.

- ***Length*:** Indica el tamaño en bytes del campo *Option Value*. Normalmente, este campo ocupa 4 bits, pudiendo indicar un valor de 0 a 14 bytes. Si este campo vale 15, se añade otro byte, permitiendo así codificar tamaños del *Option Value* de 15 a 270 bytes.

- ***Option Value***: Contiene el valor de la opción, cuyo significado se muestra en la tabla 1. El tamaño y formato de este campo depende de la opción elegida. Es de tamaño variable.

Tabla 1 - Opciones *CoAP* de IETF 7262

Option Number	Option
1	Content-Type
3	Uri-Host
4	ETag
5	If-None-Match
7	Uri-Port
8	Location-Path
11	Uri-Path
12	Content-Format
14	Max-Age
15	Uri-Query
16	Accept
20	Location-Query
35	*Proxy*-Uri
39	*Proxy*-Scheme

Modelo Request/Response

CoAP consiste en el intercambio de mensajes asíncronos entre dos nodos. Un nodo actúa de cliente y envía una o más peticiones sobre uno o más recursos alojados en un servidor que atiende la petición. El servidor responde a la petición indicando el éxito o fracaso de la petición recibida.

Requests

Una petición (*request) se* envía mediante un mensaje de tipo *CON* o *NON.* Consiste en una petición que ejecuta un método sobre un recurso que viene identificado a través de la opción *URI-Path* contenida en la trama.

Los métodos soportados en *CoAP* son *GET, POST, PUT* y *DELETE*. El método viene indicado en el campo *Code* de la cabecera y tiene las mismas propiedades de seguridad e ídem-potencia que en el HTTP.

Responses

Una respuesta (*response)* viene dada por el código del campo *Code*. Puede ser de tres clases diferentes indicando en cada caso el éxito, el error o en qué parte se ha producido error:

- **2.xx (*Success)*:** La petición fue recibida, entendida y aceptada correctamente.
- **4.xx (*Client Error)*:** La petición contiene una sintaxis errónea o no puede ser correctamente tratada por el servidor.
- **5.xx (*Server Error)*:** El servidor no puede tratar una petición aparentemente correcta. En caso de un error habitual en el campo *payload* de respuesta se añade una breve descripción del error para que el administrador pueda descifrar dónde se encuentra el problema.

Tipos de respuesta

Piggy-backed: El servidor responde inmediatamente a una petición recibida de tipo *CON* y envía un mensaje de tipo *ACK*. Este tipo de respuestas se dan independientemente de si la respuesta indica éxito o fallo al tratar la petición como se muestran en las sucesivas imagenes.

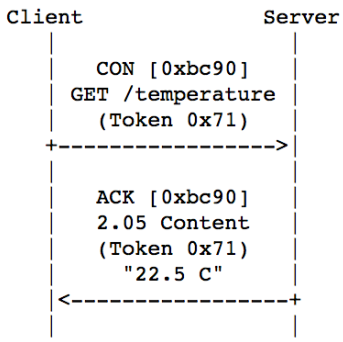


Figura 6 - Respuesta Piggy Backed

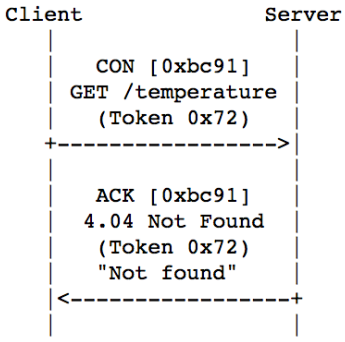


Figura 7 - Respuesta errónea Piggy Backed

Separate: El servidor no es capaz de responder al momento, porque no dispone de acceso temporal al recurso o bien porque está saturado.

Las respuestas a peticiones de tipo *NON* (no confirmarbles) son siempre enviadas mediante respuestas separate, debido a que no existe confirmación (mensaje *ACK)* para estas de peticiones. Seguidamente vemos como actua en la figura 8.

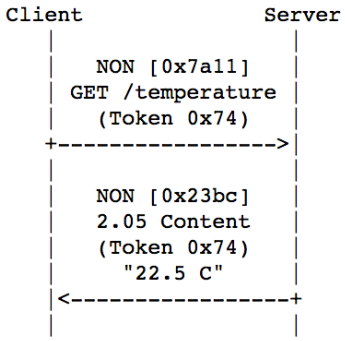


Figura 8 - Respuesta Separate a *NON*

Si la petición se recibe a través de un mensaje *CON* y el servidor no puede enviar la respuesta de forma inmediata, responde con un *ACK* confirmando que ha recibido la petición y que esta será tratada tan pronto como sea posible. Posteriormente, la respuesta con el contenido del recurso se envía con un mensaje de tipo *CON* que deberá ser confirmado por el cliente para asegurar que este último ha recibido correctamente la respuesta. Se aclara funcionamiento en figura 9.

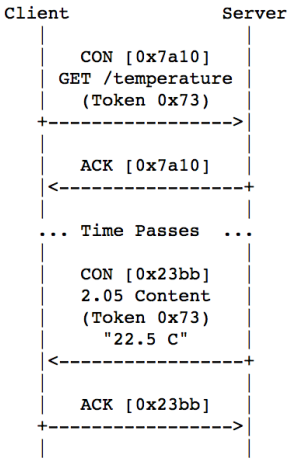


Figura 9 - Respuesta Separate a CON

Request/Response *Matching*, uso del Token

Un nodo puede tener más de un mensaje pendiente de confirmar o, simplemente, interactuar con diferentes nodos a la vez. Es por esto que se requiere un método para poder determinar si un mensaje recibido es la respuesta a un mensaje concreto y no a otro. Esto se consigue mediante el uso del *Token*.

El *Token* es un valor pensado para ser gestionado de forma local en el cliente y así pueda diferenciar de forma concurrente las peticiones que tiene en curso. El valor que toma el *Token* debe ser actualizado para que todas las peticiones que un cliente tenga pendientes de recibir respuesta tengan un valor de *Token* distinto.

Un nodo que recibe un paquete que contiene un *Token* debe responder siempre manteniéndolo y sin alterar el valor del *Token.* Si la respuesta es de tipo *Piggy-backed no modifica el Token ni* altera el valor del *Message ID*. Si la respuesta es de tipo *Separate* se debe de añadir el *Token* de la petición original.

Fiabilidad en *CoAP*

Aunque *CoAP* trabaja con un protocolo de comunicación no fiable como es el UDP, ofrece una fiabilidad muy alta de comunicación entre los clientes y los servidores. Esto se consigue enviando mensajes tipo CON a los que se responde siempre con un *ACK*.

Cuando un cliente envía un mensaje CON y pasado un determinado tiempo no recibe respuesta, volverá a enviar la petición original. El tiempo entre reenvíos sufrirá un *back-off* exponencial hasta recibir el *ACK* si se tuviera que volver a enviar la petición. Agotadas el número máximo de re-transmisiones, el mensaje *CoAP* será borrado y no se volverá a enviar.

Métodos *CoAP*

En este apartado se explican los cuatro posibles métodos que pueden ser aplicados sobre un recurso.

GET

Requiere al servidor una representación de la información correspondiente al recurso, que viene incluido en la opción *URI-Path*. Es un método seguro e ídem-potente al cual se debe responder con un código 2.05 (*Content*) o 2.03 (*Valid*) si se trata de una petición válida.

POST

Requiere que la representación del recurso que aparece incluida con la petición enviada sea procesada. El servidor debe responder con un código 2.01 (*Created*) en caso de haberse creado un nuevo recurso en el servidor y deberá incluir en la respuesta la *URI* donde se aloja este recurso, que será indicada mediante una o más opciones *Location-Path* y/o *Location-Query*.

Si, en cambio, el recurso no se crea porque ya existía y la petición es válida, el servidor contestará con una respuesta 2.04(*Changed*), indicando que el recurso ha sido cambiado según las indicaciones incluidas en la petición. Este método no es ni seguro ni ídem-potente.

PUT

Requiere al servidor que el recurso indicado mediante la *URI* debe ser actualizado o creado con la representación que se incluye en el mensaje.

El servidor debe responder con código 2.01 (*Created*) en caso de haberse creado un nuevo recurso en el servidor. A diferencia del método POST, no incluye su respuesta la *URI* donde se aloja este recurso.

Si, en cambio, el recurso no se crea porque ya existía y la petición es válida, el servidor contesta con una respuesta 2.04(*Changed*), indicando que el recurso ha sido cambiado según las indicaciones incluidas en la petición.

Se trata de un método no seguro pero ídem-potente.

DELETE

Una petición con este método solicita que el recurso identificado mediante la *URI* adjunta sea eliminado. La respuesta del servidor deberá ser con código 2.02 (Deleted), en caso de que se haya eliminado correctamente o si el recurso no existía al recibir la petición.

Caching

CoAP provee un método para que los nodos puedan almacenar en su memoria *caché* respuestas obtenidas a partir de una petición y que podrán ser utilizadas posteriormente bajo ciertas circunstancias.

El principal objetivo que se persigue con esta herramienta es optimizar, todavía más si cabe, el uso de la red y evitar así el envío de paquetes redundantes. Esta respuesta almacenada puede ser re-usada para responder a una petición de igual características que la petición original que causó esta respuesta, sin necesidad de volver a enviar otra petición al servidor que aloja este recurso. El uso de *cache* permite reducir la latencia y el número de mensajes intercambiados en la *WSN.*

Hay ciertas circunstancias que se tienen que producir para que, recibida una petición, el nodo pueda re-utilizar la información que tiene en *cache* y responda con el *payload* de la respuesta almacenada. Las condiciones se exponen a continuación. Que:

- El método recibido y el método usado anteriormente sean el mismo.
- Las opciones contenidas en la petición recibida deben ser las mismas que las que estaban presentes en la original, excepto *Max-Age* o *ETag.*
- La respuesta almacenada debe estar validada o no haber caducado todavía, según los métodos y criterios que se explicarán en el siguiente punto.

Freshness Model

"*fresh*" es una respuesta almacenada en el *cache* que puede ser utilizada por un nodo para responder a una petición recién llegada, siempre que se cumplan las condiciones expuestas en el apartado anterior. Este atributo se determina a través de la opción *Max-Age*. Esta opción indica una "edad" máxima, en segundos, a partir de la cual esta respuesta deberá dejar de considerarse *fresh* y pasará a estar obsoleta. *Max-Age* es añadida por el servidor original y él determina cuando expirará el periodo de validez de la respuesta. Si por la naturaleza cambiante del valor del recurso, o por cualquier otra razón, el servidor quiere evitar que se pueda usar una respuesta guardada en *caché* por parte de otro nodo para responder a una petición sobre el recurso, puede explicitarlo mediante una *Max-Age* con valor 0. A falta de la opción *Max-Age*, se tomará el valor de 60 segundos por defecto, tal y como recomienda el estándar).

Validation Model

Cuando un nodo *CoAP* tiene una respuesta almacenada en el *cache* que ha dejado de ser válida, porque ha expirado su periodo de validez indicado mediante la opción *Max-Age*, puede renovar de nuevo la validez de la respuesta guardada en el caché con el servidor a través del procedimiento conocido como *Validation Model.*

Cuando un servidor recibe una petición sobre un recurso, en la respuesta puede añadir la opción *ETag*. Esta opción se utiliza localmente para identificar y diferenciar representaciones obtenidas mediante varias peticiones de un mismo recurso. Si un nodo quiere actualizar la representación de un determinado recurso del que ya tiene una o más representaciones almacenadas puede enviar una petición de tipo *GET* al servidor añadiendo en esta tantas opciones *ETag* como respuestas almacenadas en caché. El servidor puede validar alguna de las respuestas almacenadas por el cliente indicadas mediante las diferentes opciones *ETag* respondiendo con un código 2.03 (*Valid*) y añadiendo la *ETag* correspondiente a la respuesta que puede ser re-utilizada, marcando esta como "*fresh*" nuevamente. Esta respuesta del servidor solo podrá marcar una de las diferentes respuestas almacenadas como válida, por lo tanto la opción *ETag* únicamente podrá estar presente una vez en la respuesta del servidor.

El nuevo periodo de validez de la respuesta indicada con la opción *ETag* vendrá indicado mediante la inclusión de la opción *Max-Age. E*n su ausencia se le dará el valor por defecto de esta opción.

Resource Discovery

En un contexto de comunicación *M2M*, en el que no intervienen las personas, se hace indispensable que los nodos de una red sean capaces de descubrir qué recursos almacena cada nodo, y con este conocimiento poder dirigir sus peticiones a uno u otro nodo en función de qué recurso están buscando exactamente.

El protocolo *CoAP* ofrece esta posibilidad, que se considera de vital importancia para las aplicaciones *M2M* que sobre él se desarrollarán. El protocolo establece que los nodos deberán soportar el formato *Constrained RESTful Envivorment Link*(*CoRE Link*) de recursos "descubribles".

En dicho protocolo, se establece que una petición de tipo *GET,* dirigida sobre el recurso "*/.well-known/core*", será respondida por el servidor con un mensaje cuyo *Payload* contiene una descripción del recurso siguiendo el formato *CoRE Link*, donde vendrá incluida la dirección de los recursos, es decir, las *URI* únicas que los identifican.

Los servidores pueden hacer uso de colecciones de recursos, como por ejemplo una lista de recursos o un grupo de sensores que midan la temperatura. Core link es capaz de entrar a la lista de recursos del servidor y navegar por ellos por medio de una interfaz de descripción de recursos.

CoRE Link format crea la relación entre los recursos y los servidores a través de un enlace que transmite la URI (dirección del recurso) que describe un recurso alojado en el servidor. Los atributos que se utilizan con el fin de describir la información útil del acceso al enlace destino son:

- Atributo recurso tipo "rt": Especifica el recurso.
- Atributo interfaz "if": proporciona un nombre o la URI que indica una definición de interfaz en concreto para interactuar con el recurso destino.
- Atributo estimación de tamaño "sz": indica el tamaño máximo de la representación de recursos devueltos por medio de un GET al URI destino.

Modelo Observer

Cuando el cliente está interesado en una representación periódica del recurso en lugar de en una puntual, se utiliza el modelo basado en el patrón *observer.* Este provee un método para lidiar con estas situaciones y optimiza los recursos de la red minimizando el número de mensajes intercambiados entre servidor y cliente.

En este modelo, los clientes que reciben notificaciones sobre los cambios en el estado del recurso, son conocidos como *observers*, y deben registrarse en el servidor, llamado *subject*, que se encargará de enviar las notificaciones a los clientes registrados cuando se produzca un cambio en el estado del recurso.

El servidor deberá mantener y actualizar la lista de clientes que quieren recibir notificaciones en función de los mensajes intercambiados con los clientes.

Los clientes envían una petición de tipo *GET* y en esta añaden la opción *observe*. Cuando el servidor recibe esta petición y encuentra en ella esta opción, entiende que es una petición de registro para el recurso indicado en la *URI* especificada. El servidor responderá al cliente con una representación del recurso y además, añadirá a este a la lista de clientes registrados.

La opción *observe* toma el *Option Number* 10. Esta opción debe de valer 0 cuando el cliente la adjunta al mensaje con la intención de registrar su interés por recibir notificaciones de un determinado recurso.

Posteriormente, la opción tiene que estar presente en cada notificación enviada por el servidor, indicando así al cliente que se trata de una notificación; pero con un valor distinto de 0, que el cliente utilizará para ordenar cronológicamente las notificaciones recibidas.

El **funcionamiento** del protocolo *observe* se puede dividir en tres etapas o fases de funcionamiento: registro, notificación y cancelación.

Registro

El cliente envía una petición tipo *GET* que contiene la opción *OBSERVE* sobre un recurso alojado en un servidor. El valor de esta opción vale 0 indicando al servidor que inicie la fase de registro. Este mensaje deberá contener también un *Token,* que el *subject* añadirá en todas las notificaciones que se envíen a este cliente.

El *subject* (servidor) puede responder de dos maneras diferentes:

- **Response con opción *OBSERVE***: Si el servidor tiene los recursos suficientes y está diseñado para soportar este patrón, responderá al cliente con un mensaje de tipo 2.05 (Content) que incluirá la representación del recurso y la opción *OBSERVE* (con un valor distinto

de 0) confirmado así al cliente que su registro ha sido realizado correctamente.

- **Response sin opción *OBSERVE***: Si el servidor es incapaz o no tiene recursos suficientes para añadir a este nuevo cliente a la lista de *observers*, se ignorará la opción *observe* y se tratará la petición como una petición *GET* normal. El hecho de que la *response* no contenga la opción *observe* indicará al cliente que no ha sido posible añadirlo a la lista de *obsevers.* En caso de que el servidor reciba una petición de registro de un cliente que ya está en la lista de *observers*, éste no lo añadirá a la lista de nuevo, pero sí que deberá actualizar la entrada, modificando de esta forma el valor de *Token* asociado al cliente.

Notificaciones

Todas las notificaciones enviadas deberán contener el *Token,* asociado al cliente al que va destinada la notificación, y la opción *observe* con un valor diferente para cada notificación. El valor de la opción *observe* será utilizado por el cliente para ordenar las notificaciones recibidas, dada la posibilidad que estas lleguen desordenadas. Las notificaciones enviadas por el servidor pueden ser *CON o NON* y dependerá del tipo de recurso, la periodicidad o no de sus cambios de valor y del estado de congestión del servidor y/o de la red y el tipo de mensaje usado para cada notificación.

En el caso de que se envíe una notificación a un cliente mediante un mensaje de tipo *CON* y esta sea contestada con el correspondiente *ACK* se le indicará al servidor que el cliente sigue interesado en recibir notificaciones sobre el estado de ese recurso. En el caso de que agotado el número máximo de re-transmisiones para un mensaje *CON*, no se haya recibido un mensaje *ACK* del cliente, se procederá a eliminarlo de la lista de *observers* y no volverá a recibir notificaciones a menos que inicie nuevamente la fase de registro.

La opción que regula este método de *caching,* denominada *Max-Age* desarrolla otra función en este protocolo para el patrón *Observe.* Este valor, servirá también al cliente para determinar si se puede considerar todavía inscrito en la lista de *observers*. Si un cliente no recibe una notificación antes de que expire el periodo marcado en la opción *Max-Age* deberá de asumir que el servidor, por cualesquiera que sean los motivos, lo ha eliminado de su lista de *observers*.

Esto implica que el cliente, para poder seguir recibiendo notificaciones de los cambios de estado del recurso deberá de iniciar de nuevo la fase de registro con el servidor que mantiene el recurso.

Cancelación

Hay dos caminos que un cliente puede tomar de forma activa para que el servidor lo elimine de su lista de *observers* y no reciba más notificaciones:

Respondiendo a un mensaje de notificación (CON o *NON*) con un mensaje de tipo *RST*.

La otra opción es enviar una **petición de tipo *GET*** sobre un recurso del cual ya recibe notificaciones sin la opción *Observe.*

CoAP en nuestro sistema

La comunicación entre la aplicación web y el servidor de sensores se realizará mediante el protocolo *CoAP*. Las especificaciones que se tienen en este momento del servidor de sensores son que implementarán un servidor que debe ser llamado mediante el método POST. Por lo tanto nuestro sistema llevará a cabo la comunicación con ese método.

Sería interesante en próximas revisiones del proyecto encargado del servidor de sensores implementar la opción Observe. De esta manera, cuando cambie un dato de cualquier sensor, el *Proxy* será informado de tal modificación, sin tener que estar llamando constantemente al servidor para conseguir los datos y mantenerlos actualizados. De esta forma, aumentaríamos la eficiencia de las comunicaciones y del sistema *Proxy*.

Estudio del *Proxy*.

El *Proxy* es la parte del sistema que, por un lado, comunicará con el servidor de sensores mediante el protocolo *CoAP*. Por otro lado, guardará la información en la base de datos. De este modo, el *Proxy* hace de puente entre el servidor de sensores y la aplicación web.

Es un servicio que se debe estar ejecutando de manera constante en una máquina servidora que tenga acceso a la base de datos. Este servicio accede a la base de datos y recoge la información de los sensores a leer. La información leída es la dirección de los sensores, los intervalos de lectura, etc. Con estos datos, y de manera concurrente, se accede a los sensores para pedirles la información que están leyendo.

El *Proxy* se debe hacer en un lenguaje de programación en el que exista una librería que cumpla la RFC7252, estándar del protocolo *CoAP*. Los lenguajes con esta librería son múltiples. En el siguiente esquema lo vemos en detalle.

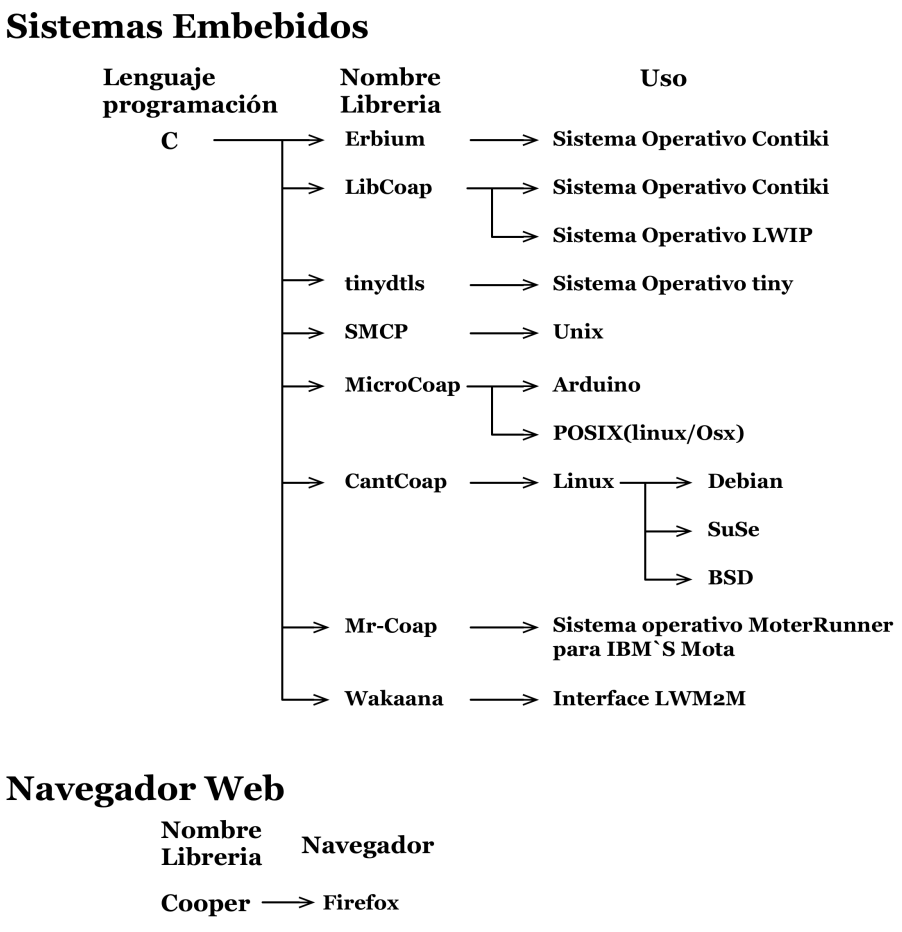


Figura 10 - Esquema Librería *CoAP* 1

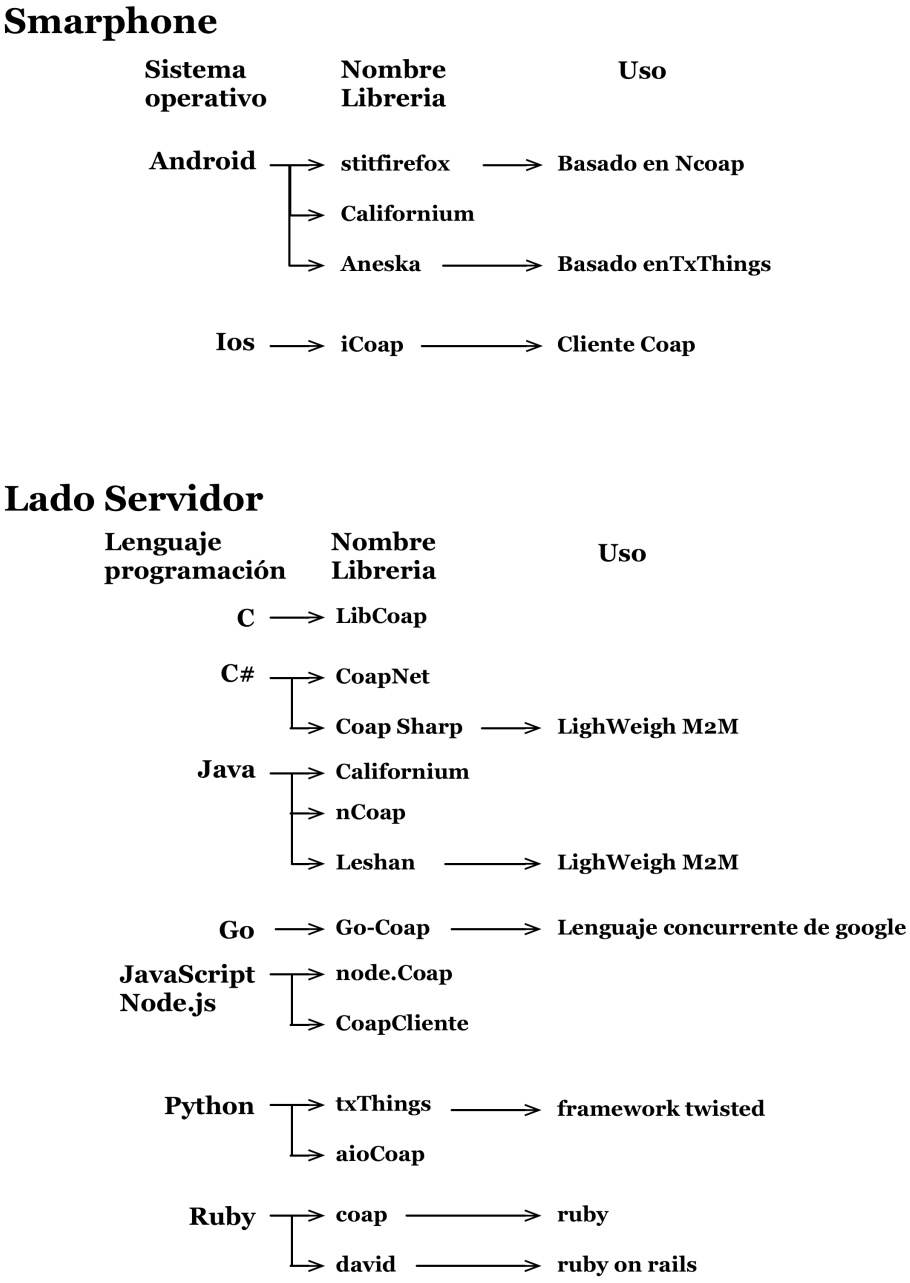


Figura 11 - Esquema librerías *CoAP* 2

En el esquema anterior se observa la diversidad de librerías que existen diseñadas para funcionar con *CoAP* en multitud de lenguajes y sistemas. Las librerías sujetas a estudio para el sistema son las del lado servidor. Puesto que es donde se instalará el *Proxy y hará las funciones de cliente*.

Librerías *CoAP* de Navegador Web

Copper

El Copper (Cu) *CoAP* user-agent es un add-on para el navegador web Firefox. Pertenece al extenso paquete desarrollado por *Californium*. Permite la navegación, los marcadores y la interacción directa con los recursos *CoAP*. Basta con introducir un URI *CoAP* en la barra de direcciones.

Se ha instalado en Firefox para experimentar con distintos servidores de prueba. La instalación se hizo desde la pestaña add-on del propio navegador, se buscó Copper y se instaló. Reiniciamos el navegador y simplemente introduciendo la URI *CoAP* en la barra de direcciones nos muestra, como vemos la figura 12, todas las opciones posibles que pueden realizar el protocolo *CoAP*.

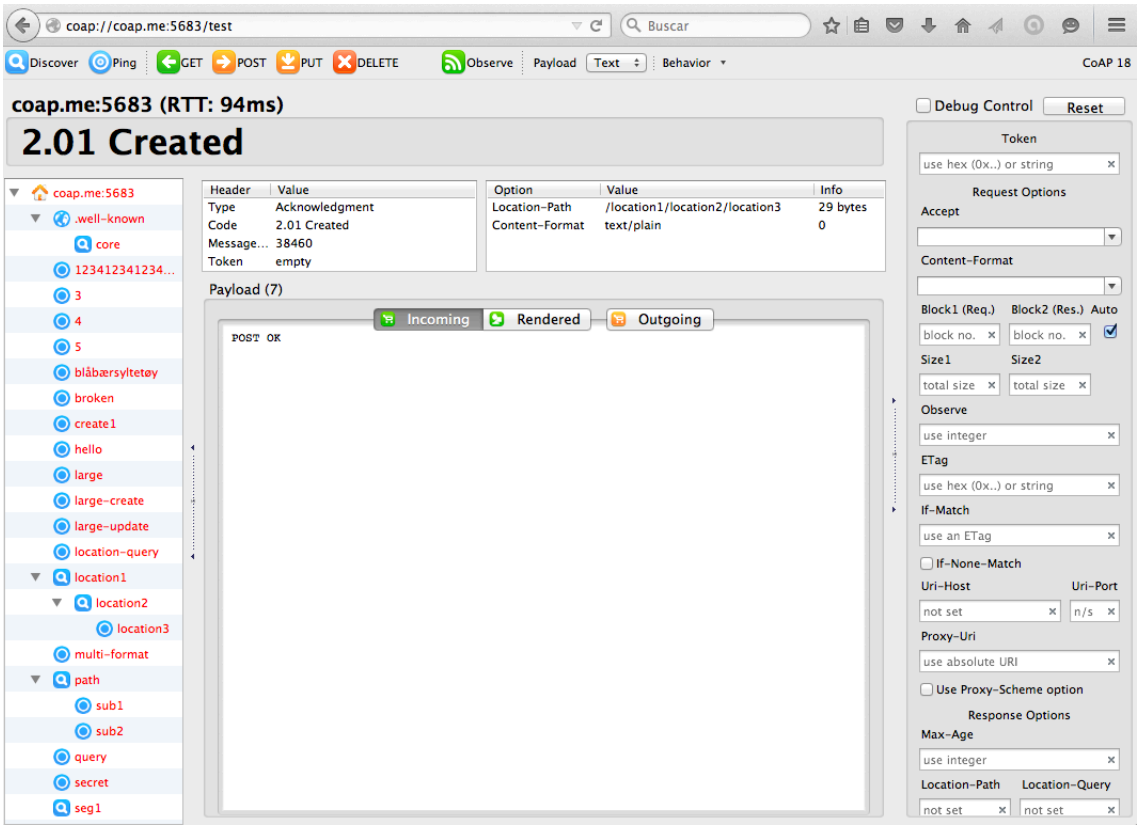


Figura 12 - Copper en Firefox

Librerías *CoAP* Lado Servidor

En cuanto a las librerías del lado servidor, se han probado varias.

CoAP - Ruby

Se empezó con la librería *CoAP* para ruby. Tras muchas pruebas, intentos y diversas modificaciones del código fuente, no se logró ninguna respuesta de los servidores. Saltaban constantemente excepciones y errores. La conclusión a la que se llegó es que la versión del protocolo *CoAP* existente cuando se diseñó la librería era otra distinta a la que actualmente se utiliza, por eso las respuestas que daba el servidor no eran interpretadas por la librería. Por lo tanto, esta línea de estudio se dejó cerrada y sin continuidad.

Este es el código que se utilizó para las pruebas.

```
require 'CoAP'
class CoAP_class
        def initialize
                cliente = CoAP::Client.new #llamar con parametros¿?
                @dato=[]
        end
        #def getCoAP(uri)
    def getCoAP
        #answer = client.get_by_url('CoAP://CoAP.me:5683/hello')
        #@dato = CoAP::Client.new.get_by_uri(uri)
        #@dato = CoAP::Client.new.get_by_uri('CoAP://CoAP.me:5683/hello')
        @dato = CoAP::Client.new.get('CoAP.me', 5683, '/hello')
        #assert_equal('world', answer.payload)
        end
        def Dame_Dato
                @dato
        end
end
client = CoAP::Client.new

puerto=5683
puts puerto.is_a?(Integer)
# validate_arguments!(host, port, path, payload)
# def client(method, path, host = nil, port = nil, payload = nil, options = {},
observe_callback = nil)

answer = client.get_by_uri('CoAP://CoAP.me:5683/hello')
#answer = client.get('/hello','CoAP.me', puerto)
dato=answer.payload
puts dato
```

Código 1 - Ruby y *CoAP*

```
Mac:~ User$ ruby pruebaCoAP.rb

true

/Users/davsel/.rvm/gems/ruby-2.1.0/gems/CoAP-
0.1.1/lib/core/CoAP/transmission.rb:110:in `send': No route to host - sendto(2) for
"CoAP.me" port 5683 (Errno::EHOSTUNREACH)

  from/Users/davsel/.rvm/gems/ruby-2.1.0/gems/CoAP
0.1.1/lib/core/CoAP/transmission.rb:82:in `request'

  from /Users/davsel/.rvm/gems/ruby-2.1.0/gems/CoAP-
0.1.1/lib/core/CoAP/transmission.rb:158:in `invoke'

  from /Users/davsel/.rvm/gems/ruby-2.1.0/gems/CoAP-
0.1.1/lib/core/CoAP/transmission.rb:142:in `request'

  from /Users/davsel/.rvm/gems/ruby-2.1.0/gems/CoAP-
0.1.1/lib/core/CoAP/client.rb:225:in `client'

  from /Users/davsel/.rvm/gems/ruby-2.1.0/gems/CoAP-
0.1.1/lib/core/CoAP/client.rb:43:in `get'

  from /Users/davsel/.rvm/gems/ruby-2.1.0/gems/CoAP-
0.1.1/lib/core/CoAP/client.rb:54:in `get_by_uri'

  from pruebaCoAP.rb:36:in `<main>'
```

Este código devuelve en todo momento errores como los mostrados en el cuadro anterior.

La librería de ruby on rails llamada David, no se probó porque en el momento de la realización del estudio no estaba disponible en la web de *CoAP* tecnology y no se retomó esta línea de estudio por tener ya realizado el *Proxy*.

Node-*CoAP*

Otra librería que se ha estudiado con éxito es la **Node-*CoAP*** que utiliza el sistema node.js para la ejecución de servidores y clientes *CoAP*.

Node.js es un entorno de programación en la capa del servidor (pero no limitándose a ello) basado en el lenguaje de programación **ECMAScript** (*JavaScript*), asíncrono, con I/O de datos en una arquitectura orientada a eventos y basada en el motor *V8* de Google. Fue creado con el objetivo de ser útil en la creación de programas de red altamente escalables, como por ejemplo, los servidores web. El autor fue Ryan Dahl y la fecha de aparición 2009. Su evolución está avalada por la empresa Joyent, que además mantiene contratado a Dahl en plantilla desde entonces.

Node.js es similar en su propósito a *Twisted* de *Python, Perl Object Environment* de Perl, React de *PHP*, libevent o libev de C, *EventMachine* de *Ruby*, *vibe.d* de *D* y de J*ava*, existe *Apache MINA*, *Netty*, *Akka*, *Vert.x, Grizzly o Xsocket*. Y, al contrario que la mayoría del código *JavaScript*, no se ejecuta en un navegador, sino en el servidor. *Node.js* implementa algunas especificaciones de *CommonJS* e incluye un entorno *REPL* para depuración interactiva.

Se instaló *Node.js* siguiendo la instrucciones de la web oficial. Se descargó la librería node-*CoAP*. En esta librería vienen unos ejemplos que sirven como modelo para probar la librería. Se trabajaron los ejemplos y funcionaron a la perfección, llegando a la conclusión de que *Node.js* es un candidato perfecto para realizar el *Proxy*.

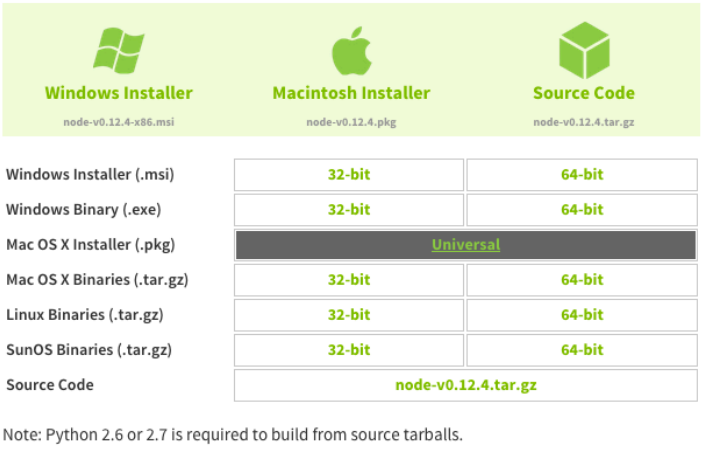


Figura 13 - Descarga de node.js

Una vez instalado node.js, se instala desde consola la librería node-*CoAP* de la siguiente manera:

```
Mac:~ user$ npm install CoAP –save
```

Para que funcione correctamente, también es necesario instalar la librería cliente de *CoAP*, puesto que sin ella saltaban varios errores.

```
Mac:~ user$ npm install CoAP-cli -g
```

Luego creamos un archivo llamado cliente.js.

```
const CoAP  = require('../') // or CoAP
  , req   = CoAP.request('CoAP://CoAP.me:5683/hello')
//Dirección CoAP que devuelve world

req.on('response', function(res) {
 res.pipe(process.stdout)

})

req.end()
```

Código 2 - cliente.js Prueba de librería node-*CoAP*

Ejecutamos cliente.js con node.js

```
Mac:examples Usarl$ node cliente.js
world
```

Nos devuelve la respuesta que esperábamos del servidor de pruebas '*CoAP*://*CoAP*.me:5683/hello'.

TxThings

La librería TxThings se ejecuta sobre el framework *Twisted*. Este es un framework de red para programación dirigida por eventos escrito en *Python* y licenciado bajo la licencia MIT.

Twisted proporciona soporte para varias arquitecturas (*TCP, UDP, SSL/TLS, IP Multicast, Unix domain sockets*), un gran número de protocolos (incluidos *HTTP, XMPP, NNTP, IMAP, SSH, IRC, FTP*), y mucho más. *Twisted* se basa en el paradigma de la programación dirigida por eventos, esto quiere decir que los usuarios de *Twisted* pueden escribir pequeños callbacks predefinidos en el framework para realizar tareas complejas.

Se hicieron pruebas intentando ejecutar la librería. Se instaló python y el framework *Twisted*. Pero esta no se pudo instalar, por desconocimiento de este sistema y debido a la complejidad de instalación y manejo. Por esta razón, se abandonó esta línea de estudio.

aio*CoAP*

Es una librería nativa para python, que se descargó desde git, pero no se consiguieron avances sobre cómo funcionaba la librería y los ejemplos no se pudieron ejecutar.

Se trató de solucionar instalando un framework que se recomendaba en la página web de python llamado kyvy; sin embargo tampoco se realizaron avances, por lo que se dejó a un lado esta línea de estudio.

Librerías basadas en el lenguaje c

La librería existente para el lenguaje c es lib*CoAP*. Se descargó y analizó el código, pero se desestimó por la complejidad y delicadeza al crear hilos mediante el lenguaje C.

Por otro lado, las librerías de C# se desestiman por ser propietarias de Windows, puesto que el objetivo es crear un sistema multiplataforma con librerías y software de licencia libre y C# no cumple estas especificaciones.

Californium

Californium (Cf) es una implementación de código abierto del protocolo *CoAP*. Está escrito en Java y para servicios de back-end (por ejemplo, servidores *Proxy*, directorios de recursos o servicios en la nube) y entornos menos restringidos tales como dispositivos integrados que ejecutan Linux , por ejemplo controladores de casa domótica, controladores de fábricas inteligentes o teléfonos móviles. *Californium* ha participado en estandarización IETF de *CoAP* y fue reimplementada desde cero con toda la experiencia adquirida en la especificación. En particular, Cf se centra ahora en la escalabilidad del servicio de "Internet de las cosas". La nueva aplicación se probó con éxito en la ETSI *CoAP* y OMA LWM2M Plugtests en noviembre de 2013 y marzo de 2014. Cumple con todos los casos de pruebas obligatorias y opcionales.

El marco Californium *CoAP* ofrece las siguientes características:

- Aplicación del CoAP (RFC 7252).
- Implementación del proyecto Observer (draft-ietf-core-observar-16).
- Implementación del proyecto de Transferencias por bloques (draft-ietf-core-bloque-17).
- Implementación del proyecto de Directorio de Recursos (draft-ietf-core-recursos-directory-02).
- Implementación de DTLS 1.2 (RFC 6347).
- CoAP-HTTP apoyo transversal de Proxy a través HttpCore-NIO y Guayaba.
- Marco de recursos Web escalable con un modelo de concurrencia flexible para la implementación de aplicaciones de Internet de las cosas.
- Uso amigable para Internet de las Cosas con JavaScript .
- Envoltorio OSGi para gestión de servidores.

Californium tiene licencia dual bajo EPL y EDL. Esta última es una licencia tipo BSD, lo que significa el marco Cf *CoAP* se puede utilizar junto con código propietario para implementar su producto IoT (Internet of things)

La Licencia Pública Eclipse (EPL) es una licencia de software de código abierto utilizada por la Fundación Eclipse para su software. Sustituye a la Licencia Pública Común (CPL) y elimina ciertas condiciones relativas a los litigios sobre patentes. Pueden utilizar, modificar, copiar y distribuir el trabajo y las versiones modificadas, en algunos casos están obligados a liberar sus propios cambios. La EPL está aprobada por la Open Source Initiative (OSI), y aparece como una licencia de "software libre" por la Free Software Foundation (FSF).

Debido a la amplitud de posibilidades que tiene esta librería y a que las pruebas realizadas fueron satisfactorias, esta es finalmente la librería elegida para realizar el *Proxy*. Y por ello, en el diseño del *Proxy* veremos el código y las pruebas realizadas de la misma.

Tabla 2 - Comparación Librerías *CoAP*

Name	Programming Language	*CoAP* version	Client/Server	Implemented *CoAP* features	License
Californium	Java	*RFC 7252*	Client + Server	Observe, Blockwise Transfers, DTLS	EPL+EDL
cant*CoAP*	C++/C	*RFC 7252*	Client + Server		BSD
Go-*CoAP*	*Go*	*RFC 7252*	Client + Server	Core + Draft Subscribe	MIT
CoAP.NET	C#	*RFC 7252*, *CoAP*-13, *CoAP*-08, *CoAP*-03	Client + Server	Core, Observe, Blockwise Transfers	3-clause BSD
*CoAP*Sharp	C#, .NET	*RFC 7252*	Client + Server	Core, Observe, Block, RD	LGPL
*CoAP*thon	Python	*RFC 7252*	Client + Server + Forward *Proxy* + Reverse *Proxy*	Observe, Multicast server discovery, CoRE Link Format parsing, Block-wise	MIT
Copper	*JavaScript* (Browser Plugin)	*RFC 7252*	Client	Observe, Blockwise Transfers	3-clause BSD
e*CoAP*	C	*RFC 7252*	Client + Server	Core	MIT
CoAP	C	*RFC 7252*	Client + Server	Core, Observe, Block	Comercial
i*CoAP*	Objective-C	*RFC 7252*	Client	Core, Observe, Blockwise Transfers	MIT
j*CoAP*	Java	*RFC 7252*	Client + Server	Observe, Blockwise Transfers	*Apache* License 2.0
lib*CoAP*	C	*RFC 7252*	Client + Server	Observe, Blockwise Transfers	BSD/GPL
micro*CoAP*	C	*RFC 7252*	Client + Server		MIT
n*CoAP*	Java	*RFC 7252*	Client + Server	Observe	BSD
node-*CoAP*	*JavaScript*	*RFC 7252*	Client + Server	Core, Observe, Block	MIT
Ruby *CoAP*	Ruby	*RFC 7252*	Client + Server (david)	Core, Observe, Block, RD	MIT, GPL
Sensinode C Device Library	C	*RFC 7252*	Client + Server	Core, Observe, Block, RD	Comercial
Sensinode Java Device Library	Java SE	*RFC 7252*	Client + Server	Core, Observe, Block, RD	Comercial
Sensinode NanoService Platform	Java SE	*RFC 7252*	Cloud Server	Core, Observe, Block, RD	Comercial
SMCP	C	*RFC 7252*	Client + Server	Core, Observe, Block	MIT
Swift*CoAP*	Swift	*RFC 7252*	Client + Server	Core, Observe, Blockwise Transfers	MIT
TinyOS *CoAP*Blip	nesC/C	*CoAP*-13	Client + Server	Observe, Blockwise Transfers	BSD
txThings	Python (Twisted)	*RFC 7252*	Client + Server	Blockwise Transfers, Observe (partial)	MIT

Estudio del lenguaje del servidor web.

La aplicación web es el corazón del proyecto. Es la interface que se va a tener entre los sensores y los usuarios. La aplicación contendrá diversos apartados con el fin de cumplir todos los objetivos del proyecto.

Se realizará sobre lenguaje de programación del lado servidor para el desarrollo web de contenido dinámico. Entre las alternativas que se barajan están *PHP* y *Ruby on Rails*.

Ruby On Rails

Ruby es un lenguaje de programación totalmente orientado a objetos multiplataforma (lenguaje interpretado y de scripts), en el que Ruby on Rails fue basado para su creación.

Ruby es totalmente software libre y fue creado por Yukihiro Matsumoto también conocido como Matz; la primera versión liberada al público fue en 1995, su sintaxis es muy parecida a lenguajes como Perl y Python.

Rails fue creado en 2003 por David Heinemeier Hansson y desde entonces ha sido extendido por el Rails core team, con más de 2.100 colaboradores y soportado por una extensa y activa comunidad.

Ruby on Rails, también conocido como RoR o Rails, es un framework de aplicaciones web de código abierto escrito en el lenguaje de programación Ruby, siguiendo el paradigma de la arquitectura Modelo Vista Controlador (MVC).

Trata de combinar la simplicidad con la posibilidad de desarrollar aplicaciones del mundo real escribiendo menos código que con otros frameworks y con un mínimo de configuración.

El lenguaje de programación Ruby permite la metaprogramación, de la cual Rails hace uso, lo que resulta en una sintaxis que muchos de sus usuarios encuentran muy legible. Rails se distribuye a través de RubyGems, que es el formato oficial de paquete y canal de distribución de bibliotecas y aplicaciones Ruby.

Los principios fundamentales de Ruby on Rails incluyen *No te repitas* (del inglés Don't repeat yourself, DRY) y *Convención sobre Configuración*.

No te repitas significa que las definiciones deberían hacerse una sola vez. Dado que Ruby on Rails es un framework de pila completa, los componentes están integrados de manera que no hace falta establecer puentes entre ellos. Por ejemplo, en ActiveRecord (Acceso a BBDD), las definiciones de las clases no necesitan especificar los nombres de las columnas; Ruby puede averiguarlos a partir de la propia base de datos, de forma que definirlos tanto en el código como en el programa sería redundante.

Ruby on rails se empezó a probar para realizar la aplicación web en este lenguaje. El framework, conforme vas cogiendo práctica, resulta muy efectivo; pero la curva de aprendizaje es muy alta, motivo por el cual resulta complicado coger soltura en este sistema de programación. Sin embargo, al probar *PHP* se comprobó que en un tiempo relativamente corto se adquiere la soltura suficiente como para realizar programas complejos y, no siendo imperativo el uso de ruby on rails, se eligió *PHP* para el desarrollo de la aplicación web.

PHP

PHP es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico. Fue uno de los primeros lenguajes de programación del lado del servidor que se podían incorporar directamente en el documento *HTML* en lugar de llamar a un archivo externo que procese los datos. El código es interpretado por un servidor web con un módulo de procesador de *PHP* que genera la página Web resultante. *PHP* ha evolucionado por lo que ahora incluye también una interfaz de línea de comandos que puede ser usada en aplicaciones gráficas independientes y utilizado en la mayoría de los servidores web al igual que en casi todos los sistemas operativos y plataformas sin ningún costo.

PHP se considera uno de los lenguajes más flexibles, potentes y de alto rendimiento conocidos hasta el día de hoy lo que ha atraído el interés de innumerables sitios con una gran demanda de tráfico, como es el caso de Facebook, para optar por el mismo como tecnología de servidor.

Fue originalmente diseñado en Perl, con base en la escritura de un grupo de CGI binarios escritos en el lenguaje C por el programador danés-canadiense Rasmus Lerdorf en el año 1994 que lo utilizó para mostrar su currículum vítae y guardar ciertos datos, como por ejemplo la cantidad de tráfico que su página web recibía. El 8 de junio de 1995, fue publicado "Personal Home Page Tools" después de que Lerdorf lo combinara con su propio Form Interpreter para crear *PHP*/FI.

Dos programadores israelíes del Technion, Zeev Suraski y Andi Gutmans, reescribieron el analizador sintáctico (parser en inglés) en el año 1997 y crearon la base del *PHP*3, cambiando el nombre del lenguaje por *PHP*: Hypertext Preprocessor. Inmediatamente comenzaron a realizar pruebas de *PHP*3 y fue publicado oficialmente en junio de 1998. Para 1999, Suraski y Gutmans reescribieron el código de *PHP*, produciendo lo que hoy se conoce como motor Zend. También fundaron Zend Technologies en Ramat Gan, Israel. En mayo de 2000 *PHP* 4 fue lanzado bajo el poder del motor Zend 1.0. Un poco más tarde, el 13 de julio de 2004, fue lanzado *PHP* 5, utilizando el motor Zend Engine 2.0.

Este lenguaje forma parte del software libre publicado bajo la licencia *PHP*, que es incompatible con la Licencia Pública General de GNU debido a las restricciones del uso del término *PHP*

El gran parecido que posee *PHP* con los lenguajes más comunes de programación estructurada, como C y Perl, permiten a la mayoría de los programadores crear aplicaciones complejas con una curva de aprendizaje muy corta. También les permite involucrarse con aplicaciones de contenido dinámico sin tener que aprender todo un nuevo grupo de funciones. Esto ha servido para que el proyecto se realice con una gran velocidad y eficiencia. Por este motivo, se ha elegido *PHP* para el desarrollo de la aplicación web.

Patrón MVC

El MVC (Modelo-Vista-Controlador) es un patrón para diseñar aplicaciones, que nos permite separar nuestro código en tres actores principales. Separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones.

Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información y por otro lado para la interacción del usuario.

Este patrón de arquitectura de software se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones así como su posterior mantenimiento.

MVC fue introducido por Trygve Reenskaug en su web personal, escrita en Smalltalk-76 durante su visita a Xerox Parc en los años 70. Seguidamente, en los años 80, Jim Althoff y otros informáticos implementaron una versión de MVC para la biblioteca de clases de Smalltalk-80. En 1988, MVC se expresó como un concepto general en un artículo sobre Smalltalk-80.

Modelo

Se encarga de la comunicación con la base de datos. Obviando la laboriosa y tediosa tarea de conectar con esta, ya que nos limitamos a indicar la consulta *SQL* y nos devuelve lo que queríamos, sin necesidad de estar pendientes de conectar la base de datos, poner usuario y contraseña, desconectar base de datos, etc. Es una forma muy sencilla de abstraer la obtención de datos. Además, en el Modelo se crean unas funciones que utilizamos para conseguir datos para la aplicación web a través del controlador.

Controlador

El controlador es el actor principal de los tres que entran en juego. Su función es comunicarse con los modelos y las vistas y se encuentra justo en medio de los dos. Es precisamente en el controlador donde va la lógica de la aplicación. La vista llama al controlador para conseguir información requerida por el usuario a través de la vista. El controlador llama a las funciones necesarias del modelo, este devuelve la información requerida y el controlador trata la información y la manda a la vista para que el usuario la visualice.

Vista

Son las plantillas que mostramos a los usuarios, estas tienen código *HTML* y código de lenguajes de programación dinámico además de *JavaScript*. Con ella interactuamos con el usuario. El usuario realiza acciones y cada acción tiene implementada una acción en el controlador para poder responder al usuario de manera adecuada.

En el siguiente esquema se comprende el uso de patrón MVC con las tecnologías utilizadas en el sistema a diseñar.

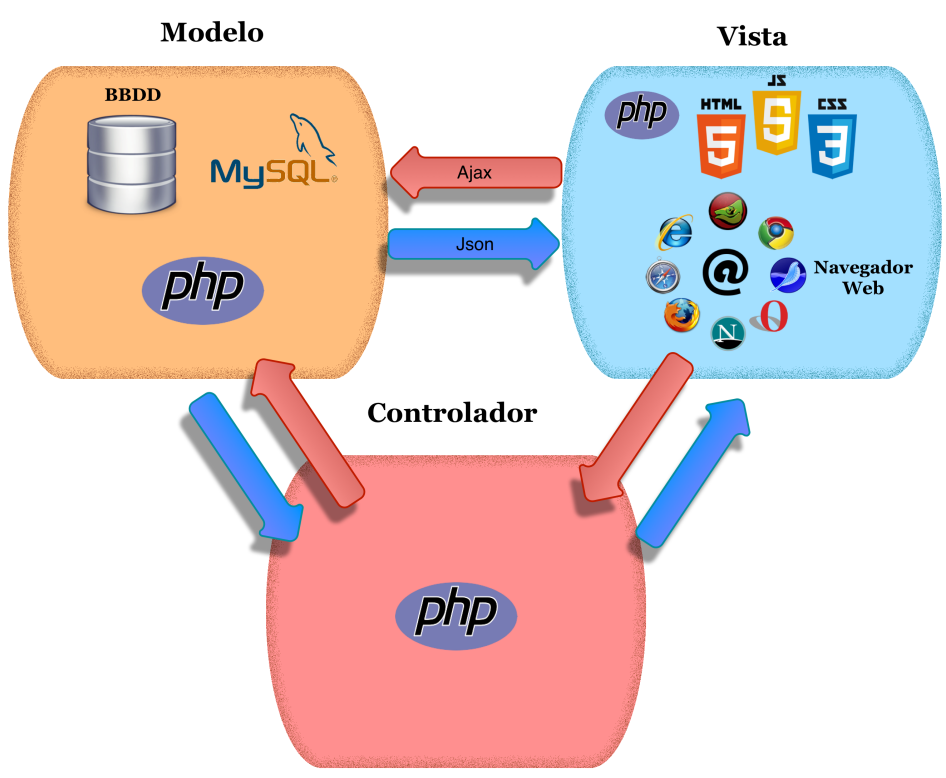


Figura 14 - Modelo-Vista-Controlador

Estudio de la aplicación web.

La aplicación web es el corazón del proyecto. Es la interface que va a existir entre los sensores y los usuarios. La aplicación contendrá diversos apartados con el fin de cumplir todos los objetivos del proyecto.

En una aplicación web, convergen muchas tecnologías. Esto es debido a que *HTML* es un lenguaje muy sencillo, que no mantiene sesiones, no tiene memoria, no sabe qué página acabas de visitar, no puede intercambiar información entre páginas y es estático.

Para solventar estas limitaciones y seguir manteniendo el estándar que durante muchos años estaba funcionando, se empezaron a crear tecnologías anexas para mejorar el lenguaje *HTML* y así hacerlo más atractivo, dinámico y versátil.

JavaScript

JavaScript (abreviado comúnmente "JS") es un lenguaje de programación interpretado. Se trata de un dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

A principios de los años 90, empezaban a desarrollarse las primeras aplicaciones web y por tanto, las páginas web comenzaban a incluir formularios complejos. Con unas aplicaciones web cada vez más complejas y una velocidad de navegación tan lenta, surgió la necesidad de un lenguaje de programación que se ejecutara en el navegador del usuario. De esta forma, si el usuario no rellenaba correctamente un formulario, no se le hacía esperar mucho tiempo hasta que el servidor volviera a mostrar el formulario indicando los errores existentes.

Brendan Eich, un programador que trabajaba en Netscape, pensó que podría solucionar este problema adaptando otras tecnologías existentes (como ScriptEase) al navegador Netscape Navigator 2.0, que iba a lanzarse en 1995. Inicialmente, Eich denominó a su lenguaje LiveScript. Netscape firmó una alianza con Sun Microsystems para el desarrollo del nuevo lenguaje de programación. Además, justo antes del lanzamiento Netscape, decidió cambiar el nombre por el de *JavaScript*.

En la aplicación web que se va a diseñar es necesario el uso de *JavaScript*, para varias causas. Una de ellas es validar los datos introducidos por los usuarios y no llenar la base de datos de datos no válidos. La otra causa es la utilización de *Ajax*.

AJAX

AJAX, acrónimo de Asynchronous *JavaScript* And XML (*JavaScript* asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano y este con la base de datos. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, mejorando la interactividad, velocidad y usabilidad en las aplicaciones.

Ajax es una tecnología asíncrona, en el sentido de que los datos adicionales se solicitan al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página. *JavaScript* es el lenguaje interpretado en el que se realizan las funciones de llamada de *Ajax* mientras que el acceso a los datos se realiza mediante XMLHttpRequest, objeto disponible en los navegadores actuales.

El intercambio de datos se puede hacer en varios formatos, los más utilizados son *XML* y *JSON*[4]. *JSON* es el sistema de intercambio de datos que utilizaremos en nuestra aplicación web, es más ligero que XML y con una notación más simple, por lo que se ha convertido en el formato más utilizado para el intercambio de datos cuando se trabaja con *Ajax*.

La tecnología de *Ajax* se utilizará en nuestra aplicación web en dos ámbitos, por un lado en la aplicación web principal con el fin de poder actualizar tablas de datos sin tener que cargar la web completa. Por otro lado, será necesaria para la creación de la app *Android* que explicaremos más adelante, para poder comunicarnos con el servidor y este con la BBDD.

Workers

La especificación de Web Workers genera secuencias de comandos *JavaScript* en segundo plano de la aplicación web. Los Web *Workers* te permiten realizar acciones con tiempos de ejecución largos para gestionar tareas intensivas de computación, pero sin bloquear la interfaz de usuario u otras secuencias de comandos para gestionar las interacciones del usuario.

Los Web *Workers* utilizan una transferencia de mensajes similar a los hilos para alcanzar el paralelismo. Son perfectos para mantener la interfaz web actualizada, eficiente y receptiva para los usuarios.

Esto se usará para mantener llamadas *Ajax* en segundo plano recuperando los datos guardados en la BBDD de los sensores, de manera que la tabla de información de alarmas y datos esté en constante actualización para tener los datos que está leyendo el *Proxy* en tiempo casi real.

[4] *JSON* acrónimo de JavaScript Object Notation

Jquery

jQuery es una biblioteca de *JavaScript*, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos *HTML*, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica *AJAX* a páginas web. Fue presentada el 14 de enero de 2006 en el BarCamp NYC. En la actualidad, *jQuery* es la biblioteca de *JavaScript* más utilizada. Existen otras librerías como P*rototype* también muy utilizada, pero elegimos Jquery para este proyecto.

jQuery es software libre y de código abierto, posee un doble licenciamiento bajo la Licencia *MIT* y la Licencia Pública General de *GNU* v2, permitiendo su uso en proyectos libres y privados. *jQuery*, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en *JavaScript* que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio.

Jquery-UI

jQuery UI es una biblioteca de componentes para el framework *jQuery* que le añaden un conjunto de plug-ins, widgets y efectos visuales para la creación de aplicaciones web. Cada componente o módulo se desarrolla de acuerdo a la filosofía de *jQuery* (find something, manipulate it: encuentra algo, manipúlalo).

En la aplicación web se utilizará solo un elemento de esta extensa librería. De hecho en la web de Jquery-UI compilamos solo la parte que necesitaremos en él. Se usa el módulo Datepicker para mostrar un calendario gráfico en los filtros de datos de la web, y así hacer más intuitivo para el usuario la selección de fechas.

Librería de Gráficos

La aplicación web tiene como requisito fundamental un sistema de representación de datos mediante gráficos. Para esta labor se utilizará un framework escrito en *JavaScript*, para que, dependiendo de los datos mostrados, dibuje en el navegador una gráfica de la evolución de los datos.

Existen multitud de ellos. En la tabla se muestran algunas de sus características.

Tabla 3 - Comparación de frameworks Gráficos

Framework Nombre	Licencia	*Interactividad*			Tecnología de renderizado			Enlace de datos
		Leyenda	Mouse Over	onClick	*HTML5 Canvas*	*SVG*	*VML*	*Eje XY*
amCharts	Libre y comercial	Si	Si	Si	No	Si	Si	
AnyChart	Propietaria	Si	Si	Si	No	Si	Si	Si
CanvasJS	*CC BY-NC 3.0* o comercial	Si	Si	Si	Si	No	No	
canvasXpress	*GPLv3*,	Si	Si	Si	Si	No	Si	
Chart Builder By Livegap	Libre	Si	Si	Si	Si	No	No	
Chart.js	MIT	Si	Si	Si	Si	No	No	No
Chartist	WTFPL	No	Si	Si	No	Si	No	
ChartJS	Comercial	Si	Si	Si	No	Si	No	
Charts 4 PHP	Libre	Si	Si	Si	Si	No	No	
Cytoscape.js	LGPL	No	Si	Si	Si	No	No	
D3.js, formerly Protovis[17][18]	*BSD-3*	No	Si	No	No	Si	No	Si
dc	*Apache* 2.0	No	Si	Si	No	Si	No	
DHTMLX Charts	GPL o Comercial	Si	Si	Si	Si	No	Si	
Dojo Charting	BSD o AFL	Si	Si	Si	Si	Si	Si	
Dygraphs	MIT	Si	Si	Si	Si	No	No	
Factmint Charts	No-comercial o comercial	Si	Si	Si	No	Si	No	
Flot Charts	MIT	Si	Si	Si	Si	No	No	
Flotr2	MIT	Si	Si	Si	Si	No	No	
FusionCharts	*Propietario*	Si	Si	Si	No	Si	Si	
Google Chart Tools	Google controls API, code samples	Si	Si	Si	No	Si	Si[l]	

Framework Nombre	Licencia		*Interactividad*		Tecnología de renderizado			Enlace de datos
		Leyenda	Mouse Over	onClick	*HTML5 Canvas*	*SVG*	*VML*	*Eje XY*
	Apache 2.0							
gRaphael	MIT	Si	Si	No	No	Si	No	
Highcharts, Highstock	No-comercial o comercial	Si	Si	Si	No	Si	Si	
JenScript	*BSD-3*	Si	Si	Si	No	Si	No	Si
jqPlot	MIT or GPLv2	Si	Si	Si	Si	No	No	
KoolChart	Libre o comercial	Si	Si	Si	Si	No	Si	
MetricsGraphics	Mozilla PLV 2.0	Si	Si	Si	No	Si	Si	
NVD3	*Apache* 2.0	Si	Si	Si	No	Si	No	
OLAPCharts	Free	Si	Si	Si	Si	No	No	
Plotly	Propietaria	Si	Si	Si	No	Si	No	
RGraph	*CC BY-NC 3.0* o comercial	Si	Si	Si	Si	No	No	
rickshaw	MIT	Si	Si	No	No	Si	No	
Shield UI	No-comercial o comercial	Si	Si	Si	No	Si	Si	
TeeChart for JavaScript	No-comercial o comercial	Si	Si	Si	Si	No	No	
VanCharts	cc-by-nc 4.0 o comercial	Si	Si	Si	Si	No	Si	
xcharts	MIT	No	Si	Si	No	Si	No	
YUI Charts	*BSD-3*	Si	Si	No	Si	Si	Si	
ZingChart	Libre o comercial	Si	Si	Si	Si	Si	Si	
ZoomCharts	No-comercial o comercial	Si	Si	Si	Si	No	Si	

Tras probar varios de estos frameworks, nos decantamos por el conocido Highchart, puesto que se adapta a nuestras expectativas y es de libre uso, si su ejecución es en una web no comercial.

CSS

Hoja de estilo en ca*scada* o *CSS* (siglas en inglés de cascading style sheets) es un lenguaje usado para definir y crear la presentación de un documento estructurado escrito en *HTML* X*HTML*. El World Wide Web Consortium (W3C) es el encargado de formular la especificación de las hojas de estilo que servirán de estándar para los agentes de usuario o navegadores. Debido a las limitaciones de presentación de *HTML*, se creó esta tecnología para mejorar el impacto de una web ante el usuario.

Las hojas de estilos aparecieron poco después que el lenguaje de etiquetas SGML, alrededor del año 1970.

La propuesta CHSS fue realizada por Håkon Wium Lie y SSP fue propuesto por Bert Bos. Entre finales de 1994 y 1995 Lie y Bos se unieron para definir un nuevo lenguaje que tomaba lo mejor de cada propuesta y lo llamaron *CSS*

*CSS*1 es la primera especificación oficial de *CSS*, recomendada por la W3C, publicada en diciembre 1996, y abandonada en abril de 2008.

La especificación *CSS*2 fue desarrollada por la W3C y publicada como recomendación en mayo de 1998, y abandonada en abril de 2008.

CSS 2.1, tuvo el estatus de candidato durante varios años, aunque fue rechazada en junio de 2005. Luego en junio de 2007 fue propuesta una nueva versión candidata que se actualizó en 2009; pero, finalmente, en diciembre de 2010 fue de nuevo rechazada.

*CSS*3 está dividida en varios documentos separados, llamados "módulos". Cada módulo añade nuevas funcionalidades a las definidas en *CSS*2, de manera que se preservan las anteriores para mantener la compatibilidad. Debido a esto, se van normalizando módulos de *CSS*. Existen 50 módulos de los cuales 3 se normalizaron en 2011; sin embargo, muchos otros están catalogados como razonablemente estables.

Bootstrap

CSS es un lenguaje difícil y costoso de gestionar, para realizar una página compleja se utilizan frameworks, facilitando la tarea.

Twitter Bootstrap es un framework o conjunto de herramientas de software libre para diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en *HTML* y *CSS*, así como, extensiones de *JavaScript* opcionales.

Bootstrap fue desarrollado por Mark Otto y Jacbod Thornton de Twitter, como un marco de trabajo (framework) para fomentar la consistencia a través de herramientas internas. En agosto del 2011, Twitter liberó a Bootstrap como código abierto. Posteriormente, en febrero del 2012, se convirtió en el proyecto de desarrollo más popular de GitHub.

Por argumentos de peso como los anteriores, esta será la herramienta usada para el diseño de la interface para la aplicación web.

Estudio del servidor web.

El servidor web es el encargado de mostrar al exterior la información de la aplicación web. Este convierte los archivos del programa a archivos *HTML* que puedan entender los navegadores web de los clientes.

En el proyecto que nos ocupa, analizaremos qué servidor web es el más adecuado. Sabiendo ya qué lenguaje de programación se va a usar, se elige el servidor web, debido a que cada lenguaje tiene unos servidores web disponibles. Al decantarnos por *PHP*, exponemos las opciones a continuación.

Los servidores web que disponen de librería para mostrar *PHP* son *Apache* HTTP Server, Microsoft IIS, Netscape e iPlanet.

OmniHTTPd, soporta Internet Server Application Programming Interface (ISAPI), que junto al módulo servidor web de Microsoft's puede ejecutar *PHP*.

Si el servidor web no soporta directamente *PHP*, siempre se puede usar la interface *Common Gateway Interface (CGI*) o procesador *FastCGI*. En este caso, el servidor web tiene que estar configurado para ejecutar *CGI* ejecutando todos los procesos como respuesta a archivos *PHP*. Algunos ejemplos de este tipo de servidores son *Cherokee, lighttpd*, etc.

Debido a que estamos estudiando un sistema nuevo, descartamos las dos últimas casuísticas para centrarnos en servidores web que tengan integrado el módulo decodificador *PHP*. Así evitamos configuraciones complicadas y pérdidas de eficiencia en la decodificación. El la tabla 4 mostramos los servidores web que vamos a estudiar, Netscape se ha descartado por no existir en la actualidad.

Tabla 4 - Plataformas WebServer

Server	Windows	Linux	OS X	BSD	Solaris	eComStation	OpenVMS	AIX	IBM i	z/OS	HP-UX
Apache HTTP Server	Si	Si	Si	Si	Si	Si	Si	Si	Si	Si	Si
Internet Information Services (ISS)	Si	No	No	No	No	No	No	No	No	No	No
Oracle iPlanet Web Server	Si	Si	No	No	Si	No	No	Si	No	No	Si

Apache HTTP Server

El servidor HTTP *Apache* es un servidor web HTTP de código abierto, multiplataforma, extensible y popular que implementa el protocolo HTTP/1.1 y la noción de sitio virtual.

Brian Behlendorf diseñó el servidor HTTP *Apache*. Hasta el año 1995, el servidor *HTTPd 1.3* lo mantenía Robert McCool, de la National *Center for Supercomputing Applications (NCSA*). En ese año Rob abandonó su trabajo y Behlendorf, con un grupo de programadores fundaron el servidor *HTTP Apache*. El nombre viene de una abreviatura de 'a patchy server', que en inglés quiere decir 'un servidor emparchado'.

En 1999 se creó The *Apache* Software Foundation, que se derivaba directamente del *Apache* Group formado en 1995. El servidor *Apache* se desarrolla dentro del proyecto *HTTP* Server (httpd) de la *Apache* Software Foundation.

Apache tiene una amplia aceptación en la red desde 1996. Es el servidor HTTP más usado. Siendo, en 2005 el servidor empleado en el 70% de los sitios web en el mundo. La licencia de software bajo Licencia *Apache* permite la distribución de derivados de código abierto y cerrado a partir de su código fuente original.

Apache server es el único servidor con soporte nativo de *PHP* que cumple con lo que demandamos: ser multiplataforma y de licencia libre. Por lo tanto, este será el servidor que montaremos en nuestro proyecto.

Microsoft IIS

Internet Information Services o IIS1 es un servidor web y un conjunto de servicios para el sistema operativo Microsoft Windows. Originalmente, era parte del Option Pack para Windows NT. Luego, fue integrado en otros sistemas operativos de Microsoft destinados a ofrecer servicios, como Windows 2000 o Windows Server 2003. Windows XP Profesional incluye una versión limitada de IIS. Los servicios que ofrece son*: FTP, SMTP, NNTP y HTTP/HTTPS*.

Debido a que es de licencia propietaria y solo es posible instalarlo en los sistemas operativos Windows, se desestima para ser parte del sistema que se diseñará.

iPlanet

iPlanet era un producto que se utilizó conjuntamente por Sun Microsystems y Netscape. Aquí surgió el servidor http Iplanet. Cuando Sun fue comprada por Oracle, este proyecto fue abandonado y no existen vínculos para ver características ni descargar el servidor web, motivo por el cual se desestima puesto que además está descatalogado.

Estudio de la BBDD.

La base de datos es la parte del sistema que enlaza el servidor *Proxy* con la aplicación web, además de guardar todos los datos de las lecturas realizadas por los sensores con el fin de mostrar históricos de los datos recogidos.

Analizaremos qué sistema servidor de base de datos es el más adecuado para nuestro cometido. En la siguiente tabla veremos una relación de todos los servidores de bases de datos existentes con sus nombres y tipo de licencia.

Tabla 5 - Servidores de BBDD

Nombre	Empresa que mantiene	Primera publicación	Ultima versión	Fecha de ultima actualización	Licencia
4D (4th Dimension)	4D S.A.S.	1984	v14.2	2014-07-10	*Propietario*
ADABAS	*Software AG*	1970	8.1	2013-06	*Propietario*
Adaptive Server Enterprise	*Sybase*	1987	16.0		*Propietario*
Advantage Database Server (ADS)	*Sybase*	1992	11.1	2012	*Propietario*
Altibase	Altibase Corp.	2000	6.1.3	2014-04-18	*Propietario*
Apache Derby	*Apache*	2004	10.11.1.1	2014-08-26	*Apache License*
ClustrixDB	*Clustrix*	2010	v6.0	2015-02-03	*Propietario*
CUBRID	NHN Corporation	2008	9.3.0	2014-05-13	*GPL* v2
dashDB	*IBM*	2013	10.6	Monthly	*Propietario*
Datacom	*CA, Inc.*	1970	14	2012	*Propietario*
DB2	*IBM*	1983	10.5	2013-04-23	*Propietario*
Drizzle	*Brian Aker*	2008	7.1.36	2012-05-23	*GPL* v2 y v3
Empress Embedded Database	Empress Software Inc	1979	10.20	2010-03	*Propietario*
EXASolution	*EXASOL AG*	2004	4.2.8	2014-04-22	*Propietario*
FileMaker	FileMaker, Inc., an Apple subsidiary	1985	14.0v2	2015-05-12	*Propietario*
Firebird	Firebird project	2000	2.5.4	2015-03-30	IPL and IDPL
GPUdb	*GIS Federal*	2014	3.2.5	2015-01-14	*Propietario*
HSQLDB	H*SQL* Development	2001	2.3.2	2014-02-14	*BSD*
H2	H2 Software	2005	1.3.176	2014-04-05	*EPL* and modified *MPL*
Informix Dynamic Server	*IBM*	1980	12.10.xC4	2014-03-14	*Propietario*

Nombre	Empresa que mantiene	Primera publicación	Ultima versión	Fecha de ultima actualización	Licencia
Ingres	*Ingres Corp.*	1974	10.2	2014-09-30	*GPL* y *Propietar*
InterBase	*Embarcadero*	1984	InterBase XE	2010-09-21	*Propietario*
Linter SQL RDBMS	*RELEX Group*	1990	6.x	2013-08-26	*Propietario*
LucidDB	The Eigenbase Project	2007	0.9.3		*GPL* v2
MariaDB	*MariaDB Community*	2010	10.0.19[7]	2015-05-09	*GPL* v2 and *LGPL* for client-libraries
MaxDB	*SAP AG*	2003	7.9.0.8	2014	*Propietario*
***Microsoft Access* (JET)**	*Microsoft*	1992	15 (2013)	2012-10-02	*Propietario*
Microsoft Visual Foxpro	*Microsoft*	1984	9 (2005)	2007-10-11	*Propietario*
Microsoft SQL Server	*Microsoft*	1989	2014 (12)	2014-03-18	*Propietario*
Microsoft SQL Server Compact (Embedded Database)	*Microsoft*	2000	2011 (v4.0)		*Propietario*
MonetDB	The MonetDB Team / *CWI*	2004	11.19.7	2014-11	MonetDB License v1.1
Mongo Db	10gen	2009	3.0.2	2015-04-06	GNU *AGPL* v3.0
mSQL	Hughes Technologies	1994	3.9[9]	2011-02	*Propietario*
MySQL	*Oracle Corporation*	1995	5.6.22	2014-12-01	*GPL* v2 o *Propietario*
MemSQL	*MemSQL*	2012	1.8 (2012)	2012-12	*Propietario*
Nexusdb	Nexus Database Systems Pty Ltd	2003	3.04	2010-05-08	*Propietario*
HP NonStop SQL	*Hewlett-Packard*	1987	*SQL*/MX 2.3		*Propietario*
Omnis Studio	TigerLogic Inc	1982	4.3.1 Release 1no	2008-05	*Propietario*
OpenBase SQL	OpenBase International	1991	11.0.0		*Propietario*
OpenEdge	*Progress Software Corporation*	1984	11.0		*Propietario*
OpenLink Virtuoso	OpenLink Software	1998	7.x	2013-08-05	*GPL* v2 o *Propietario*
Oracle	*Oracle Corporation*	1979	12c Release 1	2013-06-25	*Propietario*

Nombre	Empresa que mantiene	Primera publicación	Ultima versión	Fecha de ultima actualización	Licencia
Oracle Rdb	*Oracle Corporation*	1984	7.3.1.2	2014-10-08[10]	*Propietario*
Paradox	Corel Corporation	1985	11	2003	*Propietario*
Pervasive PSQL	*Pervasive Software*	1982	v11 SP3	2013	*Propietario*
Polyhedra DBMS	*ENEA AB*	1993	8.9	2014-09	*Propietario*
PostgreSQL	Postgre*SQL* Global	1989	9.4.3	2015-06-04	Postgre*SQL* Licence
R:Base	R:BASE Technologies	1982	9.5		*Propietario*
RDM	Raima Inc.	1984	11.0	2012-06-29	*Propietario*
RDM Server	Raima Inc.	1993	8.4	2012-10-31	*Propietario*
SAP HANA	*SAP AG*	2010	1.0		*Propietario*
ScimoreDB	Scimore	2005	3.0	2008-03-03	*Propietario*
SmallSQL	*SmallSQL*	2005	0.20	2008-12	*LGPL*
solidDB	UNICOM Global	1992	7.0.0.10	2014-04-29	*Propietario*
SQL Anywhere	*Sybase*	1992	12.0	2010-07-09	*Propietario*
SQLBase	Unify Corp.	1982	11.5	2008-11	*Propietario*
SQLite	*D. Richard Hipp*	2000	3.8.8.3	2015-02-25[13]	Dominio publico
Superbase	*Superbase*	1984	Scientific (2004)		*Propietario*
Teradata	*Teradata*	1984	15	2014-04	*Propietario*
Tibero	*TmaxData*	1992	5.SP1	2014-08	*Propietario*
UniData	Rocket Software	1988	7.2.12	2011-10	*Propietario*

Lógicamente, no vamos a analizar todos estos sistemas, así que hemos elegido los sistemas más utilizados para realización de aplicaciones web. Los servidores que se van a analizar son *SQL*ite, Oracle, postgre*SQL*, MariaDb, *MySQL*, MongoDb, Firebird y Ms *SQL* server.

Tabla 6 - Servidores BBDD y Plataformas soportadas

	Windows	*OS X*	*Linux*	*BSD*	*UNIX*	*AmigaOS*	*Symbian*	*z/OS*	*iOS*	*Android*	*OpenVMS*
ADABAS	Si	No	Si	No	Si	No	No	Si	No	No	No
Firebird	Si	Si	Si	Si	Si	No	No	Si	No	No	No
MariaDB	Si	No	Si	Si	Si	No	No	No	?	?	No
Microsoft Access	Si	No	No	No	No	No	No	No	No	No	No
Microsoft SQL Server	Si	No	No	No	No	No	No	No	No	No	No
MongoDb	Si	Si	Si	Si	Si	No	No	No	No	No	No
MySQL	Si	Si	Si	Si	Si	Si	Si	Si	?	Si	No
Oracle	Si	Si	Si	No	Si	No	No	Si	No	No	Si
PostgreSQL	Si	Si	Si	Si	Si	No	No	Bajo *Linux*	No	Si	No
SQLite	Si	Si	Si	Si	Si	Si	Si	Si	Si	Si	No

*SQL*lite

*SQL*ite es un sistema de gestión de bases de datos relacional compatible con ACID, contenida en una pequeña (~275 kiB) biblioteca escrita en C. Debido a su pequeño tamaño, *SQL*ite es muy adecuado para los sistemas integrados. En su versión 3, *SQL*ite permite bases de datos de hasta 2 Terabytes de tamaño. *SQL*ite es un proyecto de dominio público creado por D. Richard Hipp.

Tabla 7 - Tipos de dato *SQL*Lite

Tipo de sistema	Dinámica
Integer	INTEGER (64-bit)
Float	REAL (aka FLOAT, DOUBLE) (64-bit)
Decimal	N/A
String	TEXT (aka CHAR, CLOB)
Binario	BLOB
Date/Time	N/A
Boolean	N/A
Otros	N/A

Oracle

Oracle Database es un sistema de gestión de base de datos objeto-relacional desarrollado por Oracle Corporation. Se considera a Oracle Database como uno de los sistemas de bases de datos más completos, soporte de transacciones, estabilidad, escalabilidad, soporte multiplataforma.

Su dominio en el mercado de servidores empresariales ha sido casi total hasta hace poco. Recientemente, sufre la competencia del Microsoft *SQL* Server de Microsoft y de la oferta de otros RDBMS con licencia libre como Postgre*SQL*, *MySQL* o Firebird. Las últimas versiones de Oracle han sido certificadas para poder trabajar bajo GNU/Linux.

Oracle surge en 1977 bajo el nombre de SDL (Software Development Laboratories); luego, en 1979, SDL cambia su nombre por Relational Software, Inc. (RSI). La fundación de Software Development Laboratories (SDL) fue motivada principalmente a partir de un estudio sobre los SGBD (Sistemas Gestores de Base de Datos) de George Koch.

La última versión de Oracle es la versión 11g, liberada en el mes de julio de 2009 y tiene la capacidad de hasta 4 peta bytes de información.

Al ser un sistema de licenciamiento propietario se desestima para nuestro sistema.

Tabla 8 - Tipos de datos Oracle

Tipo de sistema	Estatica + Dinamica
Integer	NUMBER
Float	BINARY_FLOAT, BINARY_DOUBLE
Decimal	NUMBER
String	CHAR, VARCHAR2, CLOB, NCLOB, NVARCHAR2, NCHAR, LONG (desuso)
Binario	BLOB, RAW, LONG RAW (desuso)
Date/Time	DATE, TIMESTAMP
Boolean	N/A
Otros	SPATIAL, IMAGE, AUDIO, VIDEO, DICOM,

postgre*SQL*

Postgre*SQL* es un Sistema de gestión de bases de datos relacional orientado a objetos y libre, publicado bajo la licencia BSD.

Postgre*SQL* se inicia en 1982 con el proyecto Ingres en la Universidad de Berkeley. Este proyecto, liderado por Michael Stonebraker, fue uno de los primeros intentos en implementar un motor de base de datos relacional. En 1985 Michael empieza nuevo proyecto sobre la experiencia de Ingres, dicho proyecto fue llamado post-ingres o simplemente POSTGRES.

Post-ingres pretendía resolver los problemas con el modelo de base de datos relacional y comprender cómo manejar "tipos" de datos. Actualmente estos son llamados objetos. Postgres usó muchas ideas de Ingres; pero no su código.

Andrew Yu y Jolly Chen, comenzaron a trabajar sobre el código de POSTGRES, lo primero que hicieron fue añadir soporte para el lenguaje *SQL* anteriormente utilizaba el lenguaje de consultas QUEL (basado en Ingres), creando así el sistema al cual denominaron Postgres95. A partir de aquí, continuo la evolución hasta nuestro tiempo.

Como ventajas, cabe destacar su alto nivel de concurrencia y su amplia variedad de tipos, los cuales son mostrados en la tabla 9.

Es una sistema de gestión de base de datos muy potente, con licencia libre y soporte para Mac Osx. Podría ser un buen candidato para poder ser usado en el sistema; pero finalmente se decide utilizar *MySQL*.

Tabla 9 - Tipo de datos PostGre*SQL*

Tipo de sistema	Estático
Integer	SMALLINT (16-bit), INTEGER (32-bit), BIGINT (64-bit)
Float	REAL (32-bit), DOUBLE PRECISION (64-bit)
Decimal	DECIMAL, NUMERIC
String	CHAR, VARCHAR, TEXT
Binario	BYTEA
Date/Time	DATE, TIME, TIMESTAMP,INTERVAL
Boolean	BOOLEAN
Otros	ENUM, POINT, LINE, LSEG, BOX, PATH, POLYGON, CIRCLE, CIDR, INET, MACADDR, BIT, UUID, XML, *JSON*, arrays, composites, ranges, custom

MySQL

MySQL es un sistema de gestión de bases de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones. *MySQL* AB fue comprada en enero de 2008 por Sun Microsystems y esta, a su vez, por Oracle Corporation en abril de 2009 que desarrolla *MySQL* como software libre en un esquema de licenciamiento dual.

Por un lado se ofrece bajo la GNU GPL para cualquier uso compatible con esta licencia, pero para aquellas empresas que quieran incorporarlo en productos privativos deben comprar a la empresa una licencia específica que les permita este uso. Está desarrollado en su mayor parte en ANSI C. *MySQL* AB fue fundado por David Axmark, Allan Larsson y Michael Widenius.

MySQL es muy utilizado en aplicaciones web, como Drupal o *PHP*BB, en plataformas (Linux/Windows-*Apache*-*MySQL*-*PHP*/Perl/Python), y por herramientas de seguimiento de errores como Bugzilla. Su popularidad como aplicación web está muy ligada a *PHP*, que, a menudo, aparece en combinación con *MySQL*. Es una base de datos muy rápida en la lectura cuando utiliza el motor no transaccional MyISAM. Vemos los tipos que soporta MySQL en la tabla 10.

Debido a que *MySQL* es especialista en aplicaciones web y a que tiene una licencia de libre uso, se toma la decisión de realizar nuestro sistema con las cualidades de este sistema de gestión de base de datos.

Tabla 10 - Tipos de datos *MySQL*

Tipo de sistema	Estático
Integer	TINYINT (8-bit), SMALLINT (16-bit), MEDIUMINT (24bit), INT (32-bit), BIGINT (64-bit)
Float	FLOAT(32-bit), DOUBLE (REAL) (64-bit)
Decimal	DECIMAL
String	CHAR, BINARY, VARCHAR, VARBINARY, TEXT, TINYTEXT, MEDIUMTEXT, LONGTEXT
Binario	TINYBLOB, BLOB, MEDIUMBLOB, LONGBLOB
Date/Time	DATETIME, DATE, TIMESTAMP, YEAR
Boolean	BIT(1), BOOLEAN = TINYINT
Otros	ENUM, SET, GIS data types (Geometry, Point, Curve, LineString, Surface, Polygon, GeometryCollection, MultiPoint, MultiCurve, MultiLineString, MultiSurface, MultiPolygon)

MariaDb

MariaDB es un sistema de gestión de bases de datos derivado de *MySQL* con licencia GPL. Es desarrollado por Michael (Monty) Widenius (fundador de *MySQL*) y la comunidad de desarrolladores de software libre. Introduce dos motores de almacenamiento nuevos, uno llamado Aria que reemplaza con ventajas a MyISAM- y otro llamado XtraDB, en sustitución de InnoDB. Tiene una alta compatibilidad con *MySQL*, ya que posee las mismas órdenes, interfaces, APIs y bibliotecas, siendo su objetivo poder cambiar un servidor por otro directamente.

Este SGBD surge a raíz de la compra de Sun Microsystems por parte de Oracle. MariaDB es un fork directo de *MySQL* que asegura licencia GPL. Monty decidió crear esta variante porque estaba convencido de que el único interés de Oracle en *MySQL* era reducir la competencia que *MySQL* daba al mayor vendedor de bases de datos relacionales del mundo que es Oracle.

Al ser un sistema relativamente nuevo, todavía no tiene soporte para Mac Osx. Por este motivo no se elige este servidor para el sistema a desarrollar. A continuación, se muestran los tipos de datos que puede utilizar MariaDB, se puede observar la simulitud a los tipos de MySQL.

Tabla 11 - Tipos de datos MariaDb

Tipo de sistema	Estático
Integer	TINYINT (8-bit), SMALLINT (16-bit), MEDIUMINT (24bit), INT (32-bit), BIGINT (64-bit)
Float	FLOAT(32-bit), DOUBLE (REAL) (64-bit)
Decimal	DECIMAL
String	CHAR, BINARY, VARCHAR, VARBINARY, TEXT, TINYTEXT, MEDIUMTEXT, LONGTEXT
Binario	TINYBLOB, BLOB, MEDIUMBLOB, LONGBLOB
Date/Time	DATETIME, DATE, TIMESTAMP, YEAR
Boolean	BIT(1), BOOLEAN = TINYINT
Otros	ENUM, SET, GIS data types (Geometry, Point, Curve, LineString, Surface, Polygon, GeometryCollection, MultiPoint, MultiCurve, MultiLineString, MultiSurface, MultiPolygon)

MongoDb

MongoDB es un sistema de base de datos No*SQL* orientado a documentos, desarrollado bajo el concepto de código abierto. En vez de guardar los datos en tablas, como se hace en las base de datos relacionales, MongoDB guarda estructuras de datos en documentos tipo *JSON* con un esquema dinámico (MongoDB llama ese formato BSON), haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida.

El desarrollo de MongoDB empezó en octubre de 2007 por la compañía de software 10gen. El código binario está disponible para los sistemas operativos Windows, Linux, OS X y Solaris.

Esta base de datos tiene muchas ventajas, pero vamos a fijarnos en los inconvenientes: tiene ciertos problemas de consistencia, no tiene concurrencia cuando se está escribiendo un dato y cuando supera los 100GB de almacenamiento tiene problemas de rendimiento. Con estos inconvenientes vemos que no es apto para nuestro sistema, debido a que nuestro *Proxy* está constantemente escribiendo y los usuarios leyendo. Por no hablar de la cantidad de datos que vamos a almacenar, 100GB no es una cifra tan grande como parece. Por lo tanto, no es el servidor de bases de datos elegido. La tabla 12 muestra los tipos de datos soportados por MongoDb

Tabla 12 - Tipos de datos MongoDb

Tipo de sistema	No*SQL*
Integer	Int, 32-bit integer , 64-bit integer
Float	NumberLong, Double
Decimal	
String	Symbol, String
Binario	Binary
Date/Time	Date, Timestamp, Binary data
Boolean	Boolean
Otros	Regular Expression, OID, DB Reference, Undefined Type, Object, Array, *JavaScript*, MinKey, MaxKey

Firebird

Firebird es un sistema de administración de base de datos relacional (o RDBMS) (Lenguaje consultas: *SQL*) de código abierto, basado en la versión 6 de Interbase, cuyo código fue liberado por Borland en 2000. Su código fue reescrito de C a C++. El proyecto se desarrolla activamente, la versión 2.5.2 es la más reciente del proyecto, fue liberada el 24 de Marzo de 2013.

Es multiplataforma, y, actualmente, puede ejecutarse en los sistemas operativos: Linux, HP-UX, FreeBSD, Mac OS, Solaris y Microsoft Windows. Ejecutable pequeño, con requerimientos de hardware bajos. Arquitectura Cliente/Servidor sobre protocolo TCP/IP. Soporte de transacciones ACID y claves foráneas. Es medianamente escalable. Buena seguridad basada en usuarios/roles. Pleno soporte del estándar *SQL*-92, tanto de sintaxis como de tipos de datos. Versión autoejecutable, sin instalación para la creación de catálogos en CD-Rom y para crear versiones de evaluación de algunas aplicaciones.

Tabla 13 - Tipo de datos de Firebird

Tipo de sistema	Estático
Integer	INT64, INTEGER, SMALLINT
Float	DOUBLE, FLOAT
NUMERIC, DECIMAL, SMALLMONEY, MONEY	DECIMAL, NUMERIC, DECIMAL
String	BLOB, CHAR, CHAR(x) CHARACTER SET UNICODE_FSS, VARCHAR(x) CHARACTER SET UNICODE_FSS, VARCHAR
Binario	BLOB SUB_TYPE TEXT, BLOB
Date/Time	TIME -STAMP
Boolean	BI CHAR(1), INTEGER
Otros	

Las características de este SDGB son muy interesantes y se adapta a la perfección a nuestras exigencias. Vemos en la tabla 13 la variedad de tipo de datos soportada. Además es de licenciamiento libre y multiplataforma con soporte a Mac Osx , lo que hace que sea candidato para el sistema a desarrollar.

Ms *SQL* server

Microsoft *SQL* Server es un sistema de manejo de bases de datos del modelo relacional, desarrollado por la empresa Microsoft.

El lenguaje de desarrollo utilizado es Transact-*SQL* (T*SQL*), una implementación del estándar ANSI del lenguaje *SQL*, utilizado para manipular y recuperar datos (DML), crear tablas y definir relaciones entre ellas (DDL). *SQL* Server solo está disponible para sistemas operativos Windows de Microsoft.

Tabla 14 - Tipos de datos de Ms *SQL* server

Tipo de sistema	Estático
Integer	TINYINT, SMALLINT, INT, BIGINT
Float	FLOAT, REAL
NUMERIC, DECIMAL, SMALLMONEY, MONEY	NUMERIC, DECIMAL, SMALLMONEY, MONEY
String	CHAR, VARCHAR, TEXT, NCHAR, NVARCHAR, NTEXT
Binario	BINARY, VARBINARY, IMAGE, FILESTREAM
Date/Time	DATE, DATETIMEOFFSET, DATETIME2, SMALLDATETIME, DATETIME, TIME
Boolean	BIT
Otros	CURSOR, TIMESTAMP, HIERARCHYID, UNIQUEIDENTIFIER, *SQL*_VARIANT, XML, TABLE, Geometry, Geography

La tabla 14 muestra los tipos de datos que soporta este sistema de gestión de BBDD. Debido a que este servidor es del universo Windows, no existe posibilidad de instalarlo en ninguna otra plataforma y, además, es de licencia propietaria; por lo tanto, se desestima para el uso en el sistema a desarrollar.

Estudio de sistemas *Android*.

Android es un sistema operativo basado en el núcleo Linux. Fue diseñado principalmente para dispositivos móviles con pantalla táctil, como teléfonos inteligentes o tablets. También se utiliza para relojes inteligentes, televisores y automóviles. Inicialmente fue desarrollado por *Android* Inc., empresa que Google respaldó económicamente y que más tarde, en 2005, compró. *Android* fue presentado en 2007.

La versión básica de *Android* es conocida como *Android* Open Source Project (AOSP).

La aplicación *Android* pretende ofrecer las mismas funcionalidades de visionado de datos que la aplicación web. Debido a que los smartphones y tabletas tienen reducidas dimensiones, estas no se suelen adaptar bien a las aplicaciones web. Si creamos una aplicación dedicada a este sistema operativo, mejoraremos el visionado de datos en estos dispositivos.

Para crear una aplicación en una plataforma móvil tenemos tres alternativas. La primera es diseñar la aplicación de manera nativa, es decir utilizando el lenguaje de programación y las API del fabricante, cada plataforma móvil tiene su lenguaje y su forma de programar. Si se quiere hacer una aplicación multiplataforma con este método, resulta muy costoso.

La segunda forma de llevarlo a cabo es mediante un framework que con un único lenguaje nos compila como si programáramos nativamente en las diferentes plataformas. La interface se tiene que programar independientemente. Como ventaja la app tendrá el mismo rendimiento que si se programara nativamente, como desventaja tenemos que aprender un nuevo lenguaje. El sistema de este tipo más conocido es Xamarin.

La tercera forma es realizar una aplicación *HTML*5+*JavaScript*+*CSS* y convertirla mediante un framework que trabaje a base de *Apache* Cordova.

Y teniendo en cuenta que crear una aplicación de *Android* de manera nativa es costoso y tiene una curva de aprendizaje alta, se toma la decisión de diseñar mediante un framework que convierta una aplicación *HTML*5+*JavaScript*+*CSS* en una aplicación para móvil. Como ventajas de este método de diseño podemos decir que aumentamos la versatilidad, porque con este tipo de frameworks podremos convertir la app a cualquier plataforma móvil del mercado. Además, utilizamos los mismos lenguajes de programación que los usados en la aplicación web, y como ya tenemos la curva de aprendizaje resuelta, así ahorraremos tiempo de diseño. Como desventaja está la pérdida de rendimiento, pero como es una app que no requiere muchos recursos, no es problema.

La aplicación web creada no nos valdrá para hacer la conversión, debido a que utiliza un lenguaje web servidor dinámico y estos framework no los soportan. Por lo tanto, habrá que crear otra aplicación web más sencilla que

se comunique con la base de datos mediante *Ajax* y nos muestre los datos y alarmas.

Se han buscado diferentes alternativas de frameworks que conviertan de html5+*JavaScript*+*CSS* a aplicación *Android*. Todos estos sistemas se basan en el motor *Apache* Cordova.

Titanium

Appcelerator Titanium es un framework libre y open source para el desarrollo de aplicaciones nativas para dispositivos móviles y aplicaciones de escritorio basadas en tecnología web, de una forma sencilla. Este framework proporciona al usuario más de 100 controles totalmente personalizables como pueden ser tablas, botones, listas, soporte para la geolocalización, redes sociales y multimedia. Pese a ello, se desestima por tener licencia propietaria.

Sencha Touch

Sencha Touch es un framework para desarrollar aplicaciones para dispositivos móviles utilizando *HTML* 5, que permite la creación de aplicaciones como si fueran nativas de sistemas operativos *Android* o Apple iOS. Este framework soporta *HTML* 5, *CSS* 3 y *JavaScript* que proporciona un alto nivel de poder, flexibilidad y optimización en las aplicaciones que se desarrollan. Tiene licencia propietaria, así que no cuadra tampoco con nuestras premisas.

Sproutcore Touch

Sproutcore Touch es el framework para el desarrollo de aplicaciones web basadas en *HTML* 5 que incluye un completo soporte para eventos táctiles y aceleración de hardware en el iPad y iPhone. No cumple cometido, solo está diseñado para Apple y queremos realizar la aplicación en *Android*.

PhoneGap

PhoneGap es el origen de todos los frameworks de los que estamos hablando para construir aplicaciones web para dispositivos móviles utilizando los estándares *HTML* 5, *CSS* 3 y *JavaScript*.

En el año 2009 una desconocida empresa llamada Nitobi crea un framework para desarrollo móvil multiplataforma llamado PhoneGap. La idea es crear aplicaciones orientadas a móviles con *HTML*5 y dotarlas de una capa *JavaScript* que permita acceder a las funciones nativas de cada sistema, así como de un entorno de ejecución que permita ejecutarlas en cualquier sistema operativo móvil. Para dar a conocer la aplicación, Nitobi decidió donar el código a *Apache* fundation. Más tarde, Nitobi fue comprada por Adobe. Tras ese evento, *Apache* modifico el nombre de PhoneGap y lo llamo Cordova. En realidad, son el mismo software con distinto nombre.

El framework soporta geolocalización, vibración, acelerómetro, cámara, cambio de orientación, magnetómetro y otras interesantes

características para iPhone, *Android*, Blackberry, Symbia y Palm. Se instala con base a node.js. La interface de uso es mediante consola. La aplicación gráfica está en fase beta.

El sistema por comando es muy engorroso, la visualización del proyecto se hace sobre el navegador web, por lo que no se ve cómo queda en la pantalla del teléfono. No resulta fácil de trabajar, por eso se dejó esta línea de estudio.

IUI

iUI es un framework consistente en una librería *JavaScript*, *CSS* e imágenes para la creación de aplicaciones webs avanzadas para iPhone y dispositivos compatibles. Al ser solo para Iphone, no vale para nuestro cometido.

Iwebkit

iWebkit 5 es la nueva versión de este ultraligero framework para la creación de forma sencilla de aplicaciones táctiles para iPhone y iPod touch. La versión actual cuenta con nuevas características mejoradas y es muy fácil de entender para poder desarrollar en pocos minutos sus aplicaciones web. Está orientado a dispositivos Apple, se desestima al querer desarrollar para *Android*.

XUI

XUI es otro framework *JavaScript* para construir simples aplicaciones web para dispositivos móviles. Tiene la desventaja de que no está muy documentada, pero vale la pena intentarlo si lo que vas a crear no es muy complejo. Es complejo y no se consigue que funcione se desetima.

JQPad

jQPad es un framework *jQuery* para el desarrollo de aplicaciones para iPad, que permitirá la creación de aplicaciones sencillas. Se desestima al estar orientado solo a Ipad.

Intel XDK

Intel XDK es una herramienta para desarrollar apps cross-plataform utilizando *HTML*5+*CSS*+*JavaScript*. Con XDK se puede programar usando tecnologías estándar como *HTML*5 y desde una misma base de código generar apps para distintas plataformas, es posible construir apps para las siguientes plataformas:

- Aplicaciones móviles: iOS, *Android* (nativo, Cordova, Crosswalk), Windows 8 Store, Windows Phone 8, Tizen, y Nook.
- Aplicaciones web: Web, Chrome App, Fracebook App.

El XDK cuenta con un IDE que permite emular apps en dispositivos virtuales para darse cuenta de cómo se verá su app en distintos dispositivos (iPhone, Microsoft Surface, Google Nexus, entre otros). También ofrece la capacidad de que los desarrolladores puedan almacenar su código en la nube de manera gratuita.

La licencia de Intel XDK permite el uso de la aplicación, aunque el código no es libre, pues tiene patente. Es multiplataforma se puede utilizar en Windows, Mac y Linux.

Intel XDK es el framework elegido para el desarrollo de la aplicación *Android* contenida en este proyecto.

Jquery Mobile

jQuery Mobile es el framework *jQuery* orientado a dispositivos móviles. El framework soporta iOS, *Android*, Windows Phone, BlackBerry, Symbian, Palm webOS y más dispositivos. Ofrece interface *JavaScript* para poder luego utilizarla en un framework de los que hemos hablado hasta ahora y convertirlo en una aplicación móvil. Solo trata la interface gráfica.

Capítulo 3

Instalación de las Herramienta para el desarrollo

Este Capítulo trata de mostrar todos los pasos necesarios para realizar el TFG y dejar operativo al completo el sistema. Se van a ir desglosando las diferentes partes del proyecto y mostrando los archivos necesarios para descargas, así como los pasos imprescindibles para su instalación.

Instalación y configuración del lenguaje para la aplicación web .

PHP

Las librerías necesarias para ejecutar *PHP* vienen instaladas en el sistema Mac Osx por defecto, no es necesario realizar ninguna acción. Para ver la versión que tenemos instalada:

```
Mac:~ david$ PHP –v

    PHP 5.4.38 (cli) (built: Mar 19 2015 14:49:15)
    Copyright (c) 1997-2014 The PHP Group
    Zend Engine v2.4.0, Copyright (c) 1998-2014 Zend Technologies
```

Instalación y configuración del servidor de la base de datos.

En el estudio de la base de datos determinamos que la elegida para el proyecto es el servidor *MySQL*. La máquina servidora que utilizaremos corre bajo el sistema operativo MacOsx. Para instalar *MySQL* entramos en su página: *http://dev.MySQL.com/downloads/MySQL/*

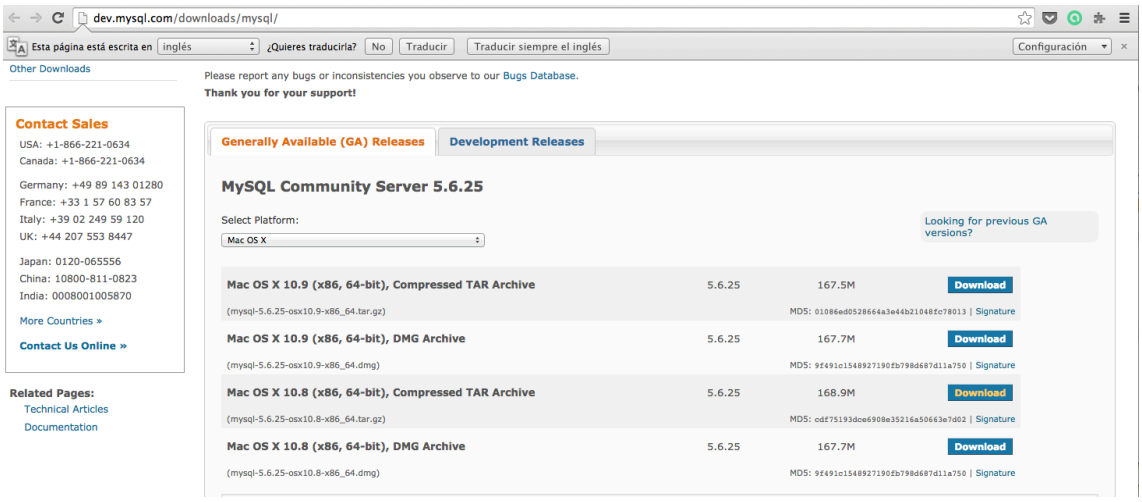
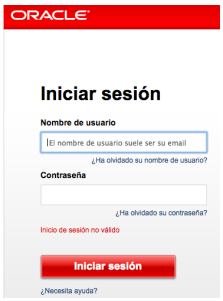


Figura 15 - Web *MySQL*

Una vez entramos en la web y estamos en la pestaña download, elegimos la versión y formato de descarga, observamos estos pasos en figura 15. Cuando se descargó la ultima versión era la 5.6. El formato elegido más efectivo para la instalación fue la imagen Dmg. Al pulsar descargar en la pagina de *MySQL* debes de registrarte y autentificarte, según vemos en la figura 16.

Una vez descargado lo instalamos como cualquier programa de Mac Osx. Vemos el proceso de instalación en la figura 17. El instalador requiere crear el usuario *MySQL* en la maquina de Mac, pero Apple piensa en todo y ya lo trae por defecto creado.



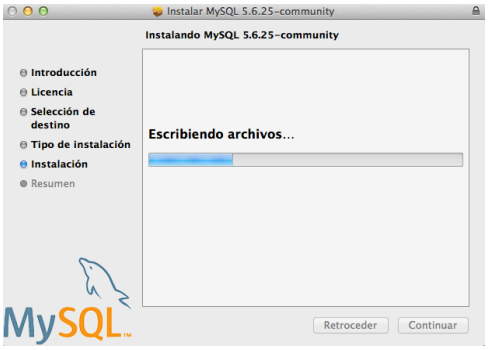


Figura 16 – Inicio sesión Oracle

Figura 17 - Proceso de la instalación de *MySQL*

Una vez instalado tenemos que iniciar el servidor de bases de datos. Para iniciar se hace gráficamente, aunque hay opción también de iniciar desde consola. Gráficamente desde el panel de preferencias del sistema pulsamos *MySQL* y nos muestra un cuadro de dialogo donde podemos iniciar, parar e indicar qué hacer al arrancar el sistema, en la siguiente figura queda claro lo explicado en este párrafo.

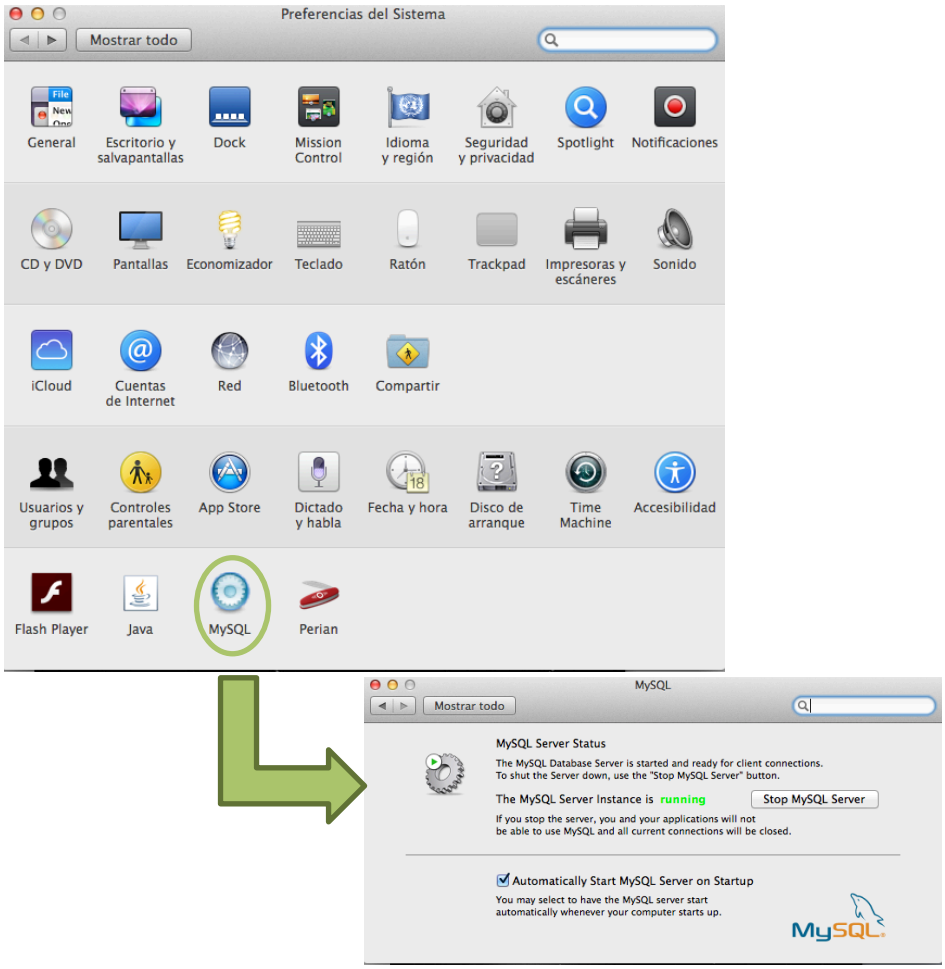


Figura 18 - Iniciar *MySQL*

Una vez instalado e iniciado el servidor está listo para funcionar. Lo que tenemos que configurar son los usuarios, puesto que vienen por defecto unos usuarios y contraseñas predeterminados. Es necesario cambiarlos por seguridad. Para configurar *MySQL* se realiza por la consola del sistema operativo. Para entrar abrimos la consola y escribimos :

```
MySQL -u root
MySQL>
MySQL> SELECT Host,User FROM MySQL.user ORDER BY User DESC;
    +---------------------+--------------------+
    | Host                  | User             |
    +---------------------+--------------------+
    | localhost           | root               |
    | mac.local | root  |                    |
    | 127.0.0.1           | root               |
    | ::1                 | root               |
    | localhost           |                    |
    +-----------+----------------------+
    5 rows in set (0,06 sec)
```

Eliminamos usuarios que no usamos, si no queremos eliminar al menos se debe de asignar una contraseña a cada usuario y dominio.

```
MySQL>  drop user 'root@imac','root@127.0.0.1';
MySQL> SELECT Host,User FROM MySQL.user ORDER BY User DESC;

    +-----------+------------------------+
    | Host       | User                  |
    +-----------+------------------------+
    | localhost | root                   |
    | localhost | davsel                 |
    +-----------+------------------------+
    2 rows in set (0,02 sec)
```

Modificamos las contraseñas por defecto. Se hace así:

```
MySQL> SET PASSWORD FOR 'root'@'localhost' =
PASSWORD('contraseña_root');
```

Lo hacemos con todos los usuarios para que no queden las contraseñas por defecto.

A partir de este momento para entrar en la consola *MySQL* debemos de añadir un parámetro, debido a que ahora el usuario root tiene contraseña.

```
MySQL -u root –p
```

Para crear usuarios se hace así:

```
CREATE USER 'david'@'localhost' IDENTIFIED BY 'pass';
```

Desde la consola *MySQL* se pueden gestionar todas las bases de datos. Con gestionar me refiero a crear (CREATE) , eliminar (DROP) ,alterar, (ALTER) insertar (INSERT), consultar (SELECT * FROM tabla WHERE Nombre=David), actualizar (UPDATE). Además de configurar parámetros de *MySQL*. Todo tipo de consultas *SQL*.

Desde *PHP*MyAdmin, que a continuación veremos como se instala, también se pueden gestionar los usuarios:

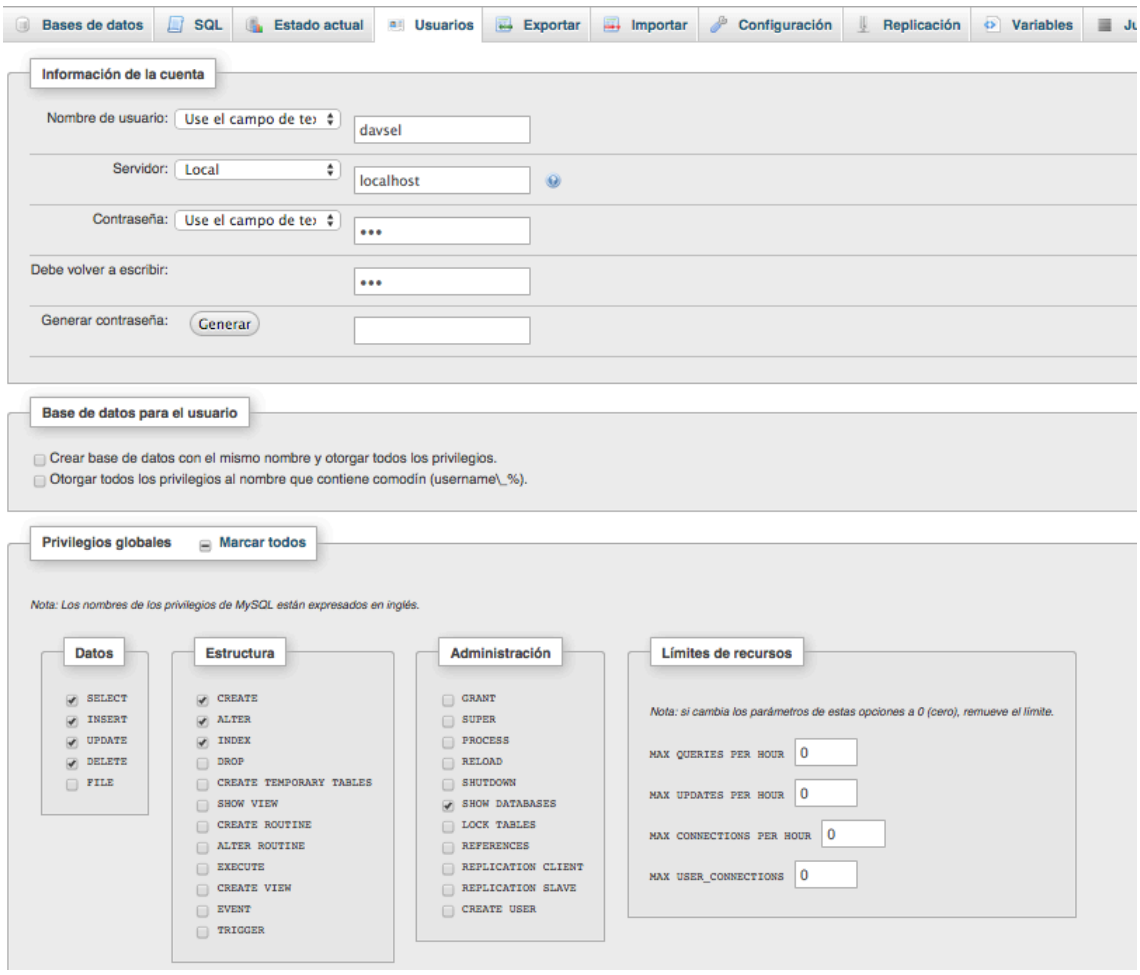


Figura 19 - Administrar Usuarios *PHP*MyAdmin

Instalación de *PHP*MyAdmin

Para facilitar la labor de administrar el servidor *MySQL* existen diversas aplicaciones web que muestran gráficamente las acciones realizadas en el servidor de BBDD. Se ha elegido la aplicación web de *PHP*MyAdmin.

*PHP*MyAdmin es una herramienta escrita en *PHP* con la intención de manejar la administración de *MySQL* a través de páginas web, utilizando Internet. Se pueden crear y eliminar Bases de Datos, crear, eliminar y alterar tablas, borrar, editar y añadir campos, ejecutar cualquier sentencia *SQL*, administrar claves en campos, administrar privilegios, exportar datos en varios formatos y está disponible en 62 idiomas. Se encuentra disponible bajo la licencia GPL Versión 2.

Tobias Ratschiller, consultor IT y fundador de Maguma, una compañía de software, comenzó a trabajar en la elaboración de una red administrativa basada en *PHP* cliente-servidor de *MySQL* en 1998, fue inspirado por Peter Kuppelwieser creador de *MySQL*-Webadmin. Cuando Ratschiller dejó el proyecto por falta de tiempo, el *PHP*MyAdmin se había convertido en una de las aplicaciones *PHP* más populares y las herramientas de administración *MySQL* constituían una gran comunidad de usuarios y administradores.

Para coordinar el creciente número de parches, tres desarrolladores de software, Olivier Müller, Marc Delisle y Loïc Chapeaux, registraron el proyecto *PHP*MyAdmin en SourceForge.net y continuó su crecimiento en 2001. La instalación de *PHP*MyAdmin se hace como cualquier aplicación web: se copia el código que descargamos de su página web, lo descomprimimos y copiamos en la carpeta de salida del servidor web, el directorio raíz de *Apache*. Todo se explica en el siguiente epígrafe.

Una vez copiado en la carpeta Sites, caso del servidor del proyecto, abrimos el navegador y escribimos en la barra de direcciones:

```
http://localhost/~username/PHPmyadmin/setup/
```

En la figura 20 podemos ver el aspecto del instalador de PHPMyAdmin.

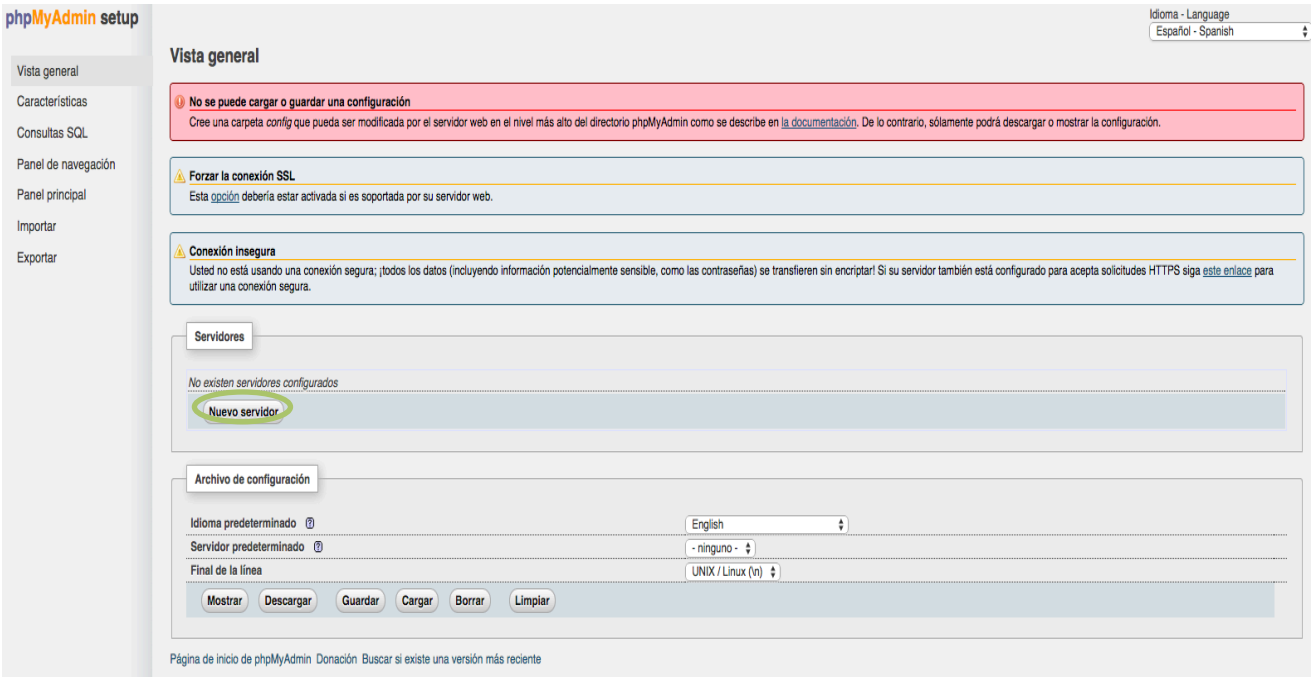


Figura 20 - Instalación de *PHP*MyAdmin

Pulsamos el botón crear nuevo servidor, generaremos un nuevo servicio de administración de *MySQL, vemos aspecto en figura 21*.

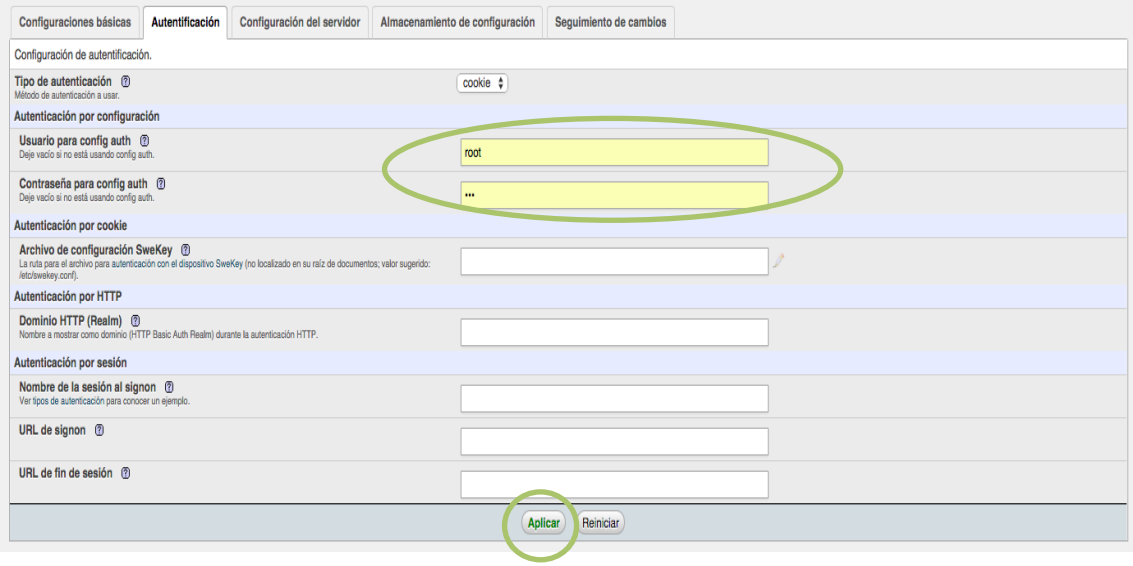


Figura 21 - Crear usuario *PHP*MyAdmin

Se pulsa aplicar y en la siguiente pantalla guardar. Ya tenemos creado la administración de *MySQL*. Para acceder tenemos que poner en la barra de direcciones del navegador, en figura 22 avistamos primera pantalla de la aplicación:

http://localhost/~username/PHPmyadmin/

Figura 22 - Autentificación *PHP*MyAdmin

Una vez autentificados, vemos el aspecto del gestor de *MySQL* en la figura23.

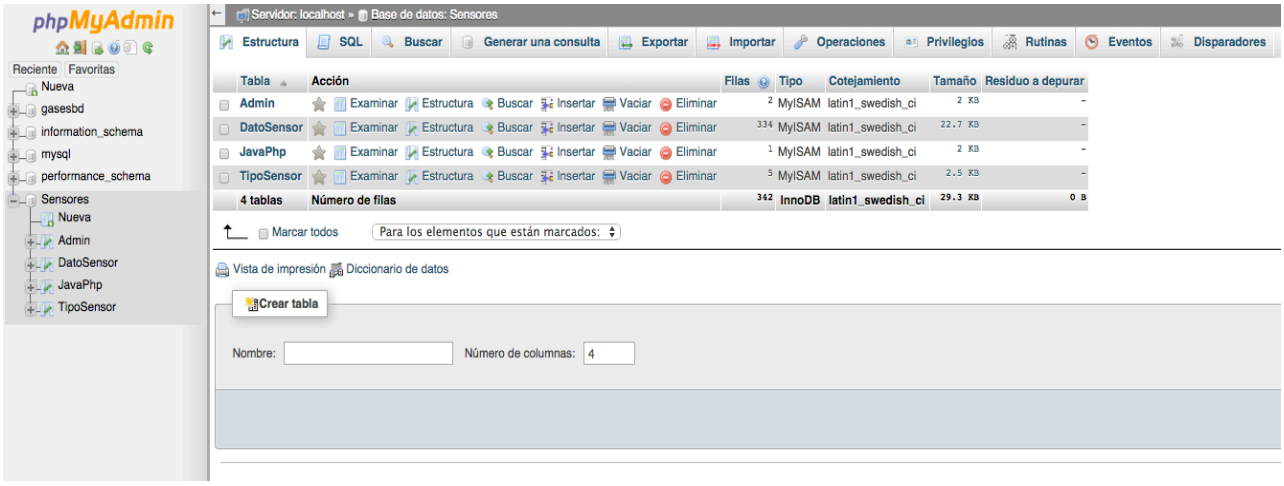


Figura 23 - Vista de *PHP*MyAdmin

Desde esta pantalla podemos ver las bases de datos creadas en *MySQL*, ver las tablas creadas, modificarlas, eliminar y crear nuevas tablas y atributos. Desde la pestaña *SQL* podemos hacer cualquier consultar a la base de datos.

Tiene una pestaña muy importante que es el *consejero,* aquí nos muestra los déficit que tiene nuestra base de datos. Las conexiones no cerradas, si los buffers son del tamaño necesario, si tenemos muchas consultas de ordenación nos da consejos de cómo acelerarlas, etc. Todo tipo de consejos útiles para dar más rendimiento a nuestra base de datos, vemos un ejemplo en figura 24.

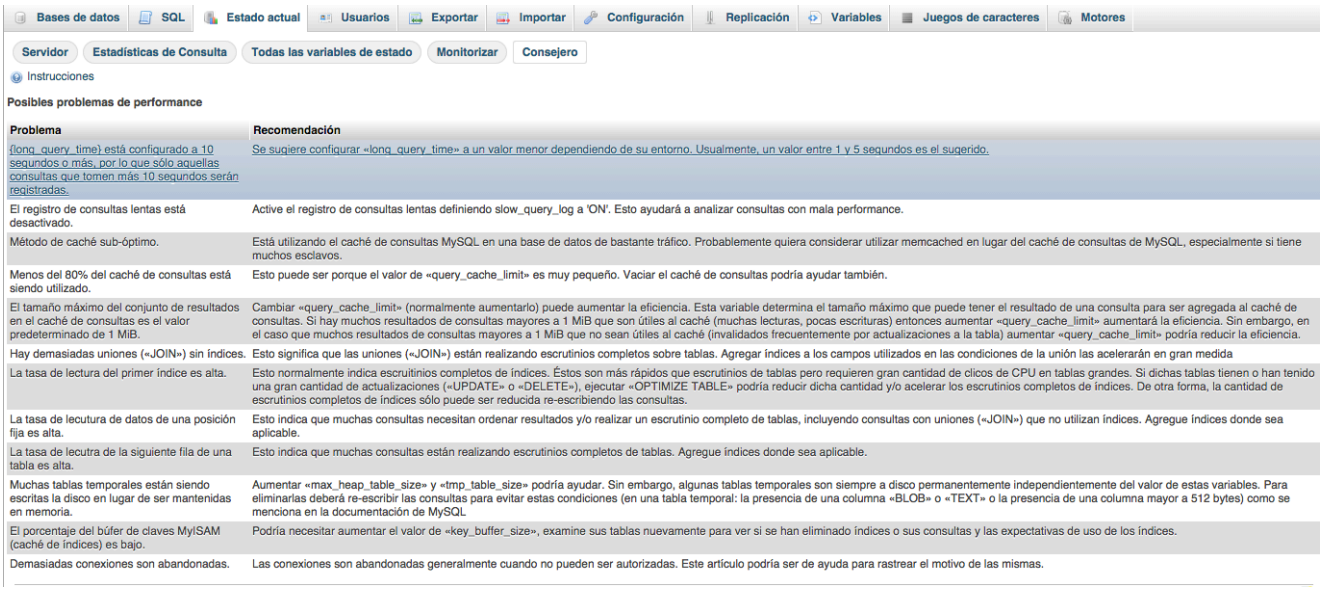


Figura 24 - Detalle de Consejero *PHP*MyAdmin

Instalación y configuración del servidor web.

El servidor web elegido para este proyecto es *Apache*. El servidor que se va a utilizar para correr *Apache* tiene un sistema operativo Mac Osx. *Apache* está preinstalado en el sistema operativo y solamente le falta ser activado a través de la línea de comandos desde la consola.

Para iniciar *Apache*:

```
Mac:~ user$  sudo Apachectl start
```

Para reiniciar:

```
Mac:~ user$  sudo Apachectl restart
```

Si queremos parar:

```
Mac:~ user$  sudo Apachectl stop
```

La versión que viene instalada es:

```
Mac:~ user$  httpd -v

    Server version: Apache/2.2.29 (Unix)
```

Para probar que se está ejecutando, abrimos el navegador web y escribimos localhost en la barra de direcciones, en figura 25 vemos el resultado que debe mostrarnos.

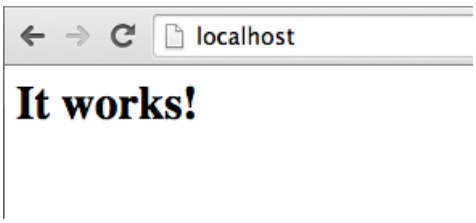


Figura 25 - Prueba de *Apache*

La configuración de *Apache* consta de indicar cuál es el directorio raíz, este directorio es donde se copian las web para su conversión y visionado.

Por defecto vienen dos carpetas configuradas como directorio raíz, una perteneciente al usuario y otra al sistema. La carpeta del sistema es /Library/WebServer/Documents/

La que vamos a usar es la carpeta de usuario, viene sin crear, en nuestro usuario creamos la carpeta Sites. Ese es nuestro directorio raíz. Aquí es donde copiaremos la estructura de carpetas de nuestra web y *Apache* la verá para convertirla al lenguaje *HTML* y que nuestro navegador web la entienda.

Por defecto *Apache* tiene desahibitada la posibilidad de usar nuestros .htaccess para sobreescribir las configuraciones por defecto. Para corregir esto vamos a editar el archivo httpd.conf ubicado en: /etc/*Apache*2/httpd.conf, lo abrimos y buscamos la línea 217 que dice AllowOverride none y lo cambiamos por AllowOverride all. Con el fin de aclarar este párrafo, se muestra en figura 26 una captura de cómo tiene que quedar el archivo de configuración de *Apache.*

```
Mac:~ user$ cd /etc/Apache2
Mac:~ user$ open httpd.conf
```

```
212     #
213     # AllowOverride controls what directives may be placed in .htaccess files.
214     # It can be "All", "None", or any combination of the keywords:
215     #   Options FileInfo AuthConfig Limit
216     #
217     AllowOverride All
218
116   LoadModule rewrite_module libexec/apache2/mod_rewrite.so
117   #LoadModule perl_module libexec/apache2/mod_perl.so
118   LoadModule php5_module libexec/apache2/libphp5.so
119   LoadModule hfs_apple_module libexec/apache2/mod_hfs_apple.so
```

Figura 26 - Detalle de httpd.conf

Descomentamos la línea 118 para activar la compatibilidad con *PHP*. De esta manera es capaz de convertir los archivos *PHP* a *HTML*. Luego reiniciamos *Apache* para hacer efectivos los cambios.

```
Mac:~ user$ sudo Apachectl restart
```

Instalación del IDE de diseño del *Proxy*.

La IDE a utilizar para el diseño del *Proxy* ha sido eclipse. Para instalar este IDE es necesario descargar de la página web de eclipse, *https://Eclipse.org/* el archivo de imagencon extensión Dmg para la instalación en MacOsx 64bits. Ejecutamos la imagen e instalamos el programa. Se muestra proceso en figura 27.

Eclipse fue desarrollado originalmente por IBM como el sucesor de su familia de herramientas para VisualAge. Eclipse es ahora desarrollado por la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios. El 2 de febrero de 2004 se creó la fundación eclipse.

Eclipse fue liberado originalmente bajo la Common Public License; pero después fue re-licenciado bajo la Eclipse Public License. La Free Software Foundation ha dicho que ambas licencias son licencias de software libre, sin embargo son incompatibles con Licencia pública general de GNU (GNU GPL).

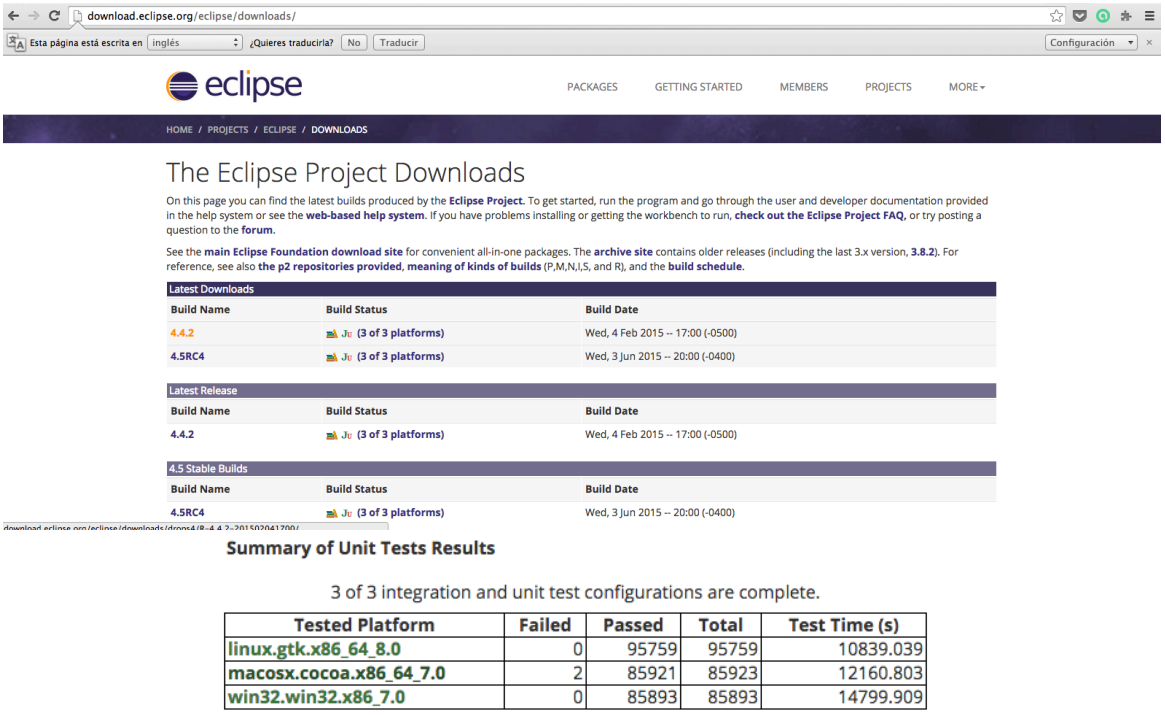


Figura 27 - Descarga de eclipse

Para el diseño del *Proxy* se utiliza, según comentábamos antes, la librería de *CoAP* llamada *Californium*. Para utilizar esta librería y crear nuestros archivos lo que hacemos desde eclipse es importar los archivos de californium.

Primero importamos como vemos en la figura 28, el proyecto californium. Para ello vamos a file/import. Aquí elegimos general/File System y buscamos donde tenemos guardado la librería *Californium*, previamente descargada de internet.

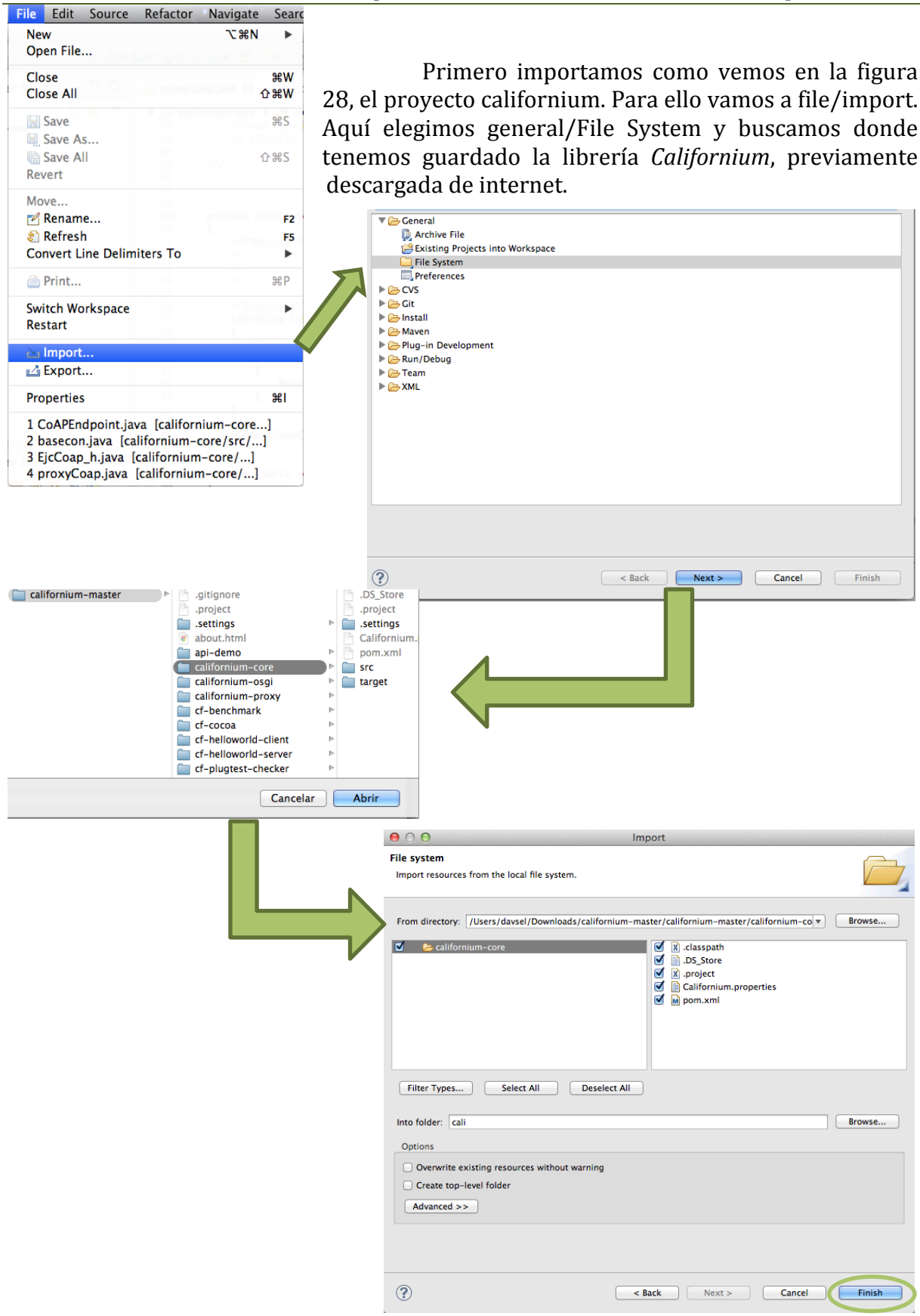


Figura 28 - Importar proyecto *Californium*

Después tenemos que crear el path de eclipse para que al compilar recoja todos los archivos de la librería californium y el driver *MySQL*. En la figura 29 vemos el proceso. De la librería *Californium* sólo nos hace falta la carpeta core, las demás son librerías que no necesitamos para la implementación del *Proxy*.

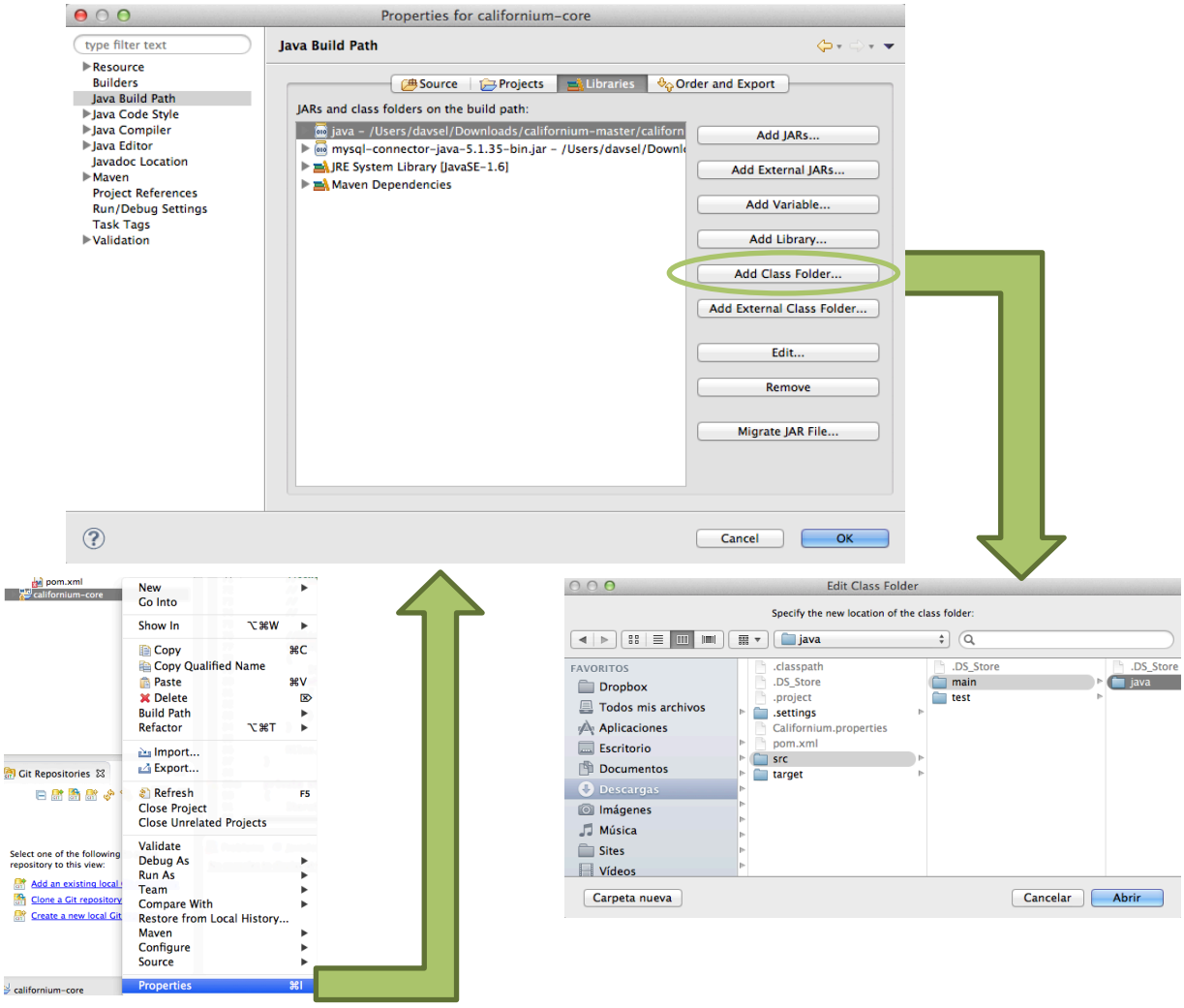


Figura 29 - Agregar path a eclipse

Con este último paso decimos a Eclipse donde están los archivos java.

Es necesario hacer la misma operación insertando el driver *MySQL* a eclipse para que podamos leer y escribir en la base de datos. Este driver tenemos que descargarlo de la página de *MySQL*, autentificarnos con una cuenta Oracle como explicábamos en el apartado de instalar *MySQL*. Una vez descargado el archivo .jar, lo copiamos al directorio donde tenemos el proyecto y hacemos la misma operación en propiedades del proyecto eclipse, java build path/libraries add external jar. Elegimos el .jar descargado y ya podemos utilizar funciones *MySQL* para acceder a la BBDD.

Instalación del IDE para diseño en *Android*.

La aplicación elegida para la creación de la apk para *Android* es XDK de Intel. Con esta aplicación convertimos una aplicación web con ciertas características en una apk. Intel® XDK *HTML*5 la herramienta de desarrollo de plataformas cruzadas ofrece un flujo de trabajo simplificado para permitir que los desarrolladores diseñen, depuren, desarrollen e implementen fácilmente aplicaciones web *HTML*5 e híbridas en múltiples tiendas de aplicaciones, así como dispositivos de formato.

Para su instalación, primero la descargamos desde la página oficial como vemos en la figura sucesiva. Luego la instalamos como un software normal.

Figura 30 - Instalación de Intel XDK

Capítulo 4

Diseño del sistema

Diseño de la base de datos

La base de datos es el centro del sistema, razón por la cual tiene que estar bien estructurada para que todo funcione a la perfección.

Primero se diseña el diagrama relacional.

Con las especificaciones del proyecto claras se empieza viendo las necesidades de almacenamiento de datos.

Especificaciones

Tenemos sensores con unas determinadas características. Tienen direcciones de red a las que tiene que acceder el *Proxy* para recoger los datos, estas pueden cambiar en un momento dado y tenemos que modificarlas fácilmente desde el administrado web. Requerimos de unas alarmas que tienen un límite superior y otro inferior, estos deben ser modificados dependiendo de la situación, de las pruebas y ajustes que se vayan dando a los sensores. Necesitamos un nombre legible para que aparezca en los botones dinámicos de la web. Tenemos diferentes sensores con múltiples unidades de medida, guardaremos el nombre de la unidad y el diminutivo por si es necesario en algún momento en el diseño. Dependiendo del sensor se tienen unos refrescos mayores o menores. No es lo mismo medir la velocidad del viento, que cambia cada segundo, que la humedad que tiene una inercia de media hora, por lo tanto cada uno tendrá un tiempo de lectura determinado. También necesitamos saber el valor actual de cada sensor de donde leer de manera rápida y eficiente.

Los datos generados por los sensores tienen que almacenarse de manera que se puedan crear históricos y realizar una consulta de fechas anteriores. Con estos datos podemos tomar decisiones precisas para solucionar problemas de calidad del aire. La forma de almacenar los datos es guardar el dato junto con el timestamp del momento que se tomó. Timestamp es una formato de fecha de UNIX que toma de referencia el día 1 de enero de 1970, desde ese día mide los segundos pasados hasta la toma del dato. *SQL* soporta este formato y sus múltiples conversiones. Los datos de todos los sensores se almacenaran juntos y mediante consultas *SQL* se resolverá qué dato mostrar en cada momento.

Necesitamos tener una gestión de usuarios para la administración del *Proxy* desde la aplicación web. Para este menester es necesario almacenar los usuarios, sus claves y los privilegios que tiene este usuario. Con los privilegios damos distintos permisos dependiendo del usuario que sea. La aplicación web tendrá más versatilidad de esta manera y podremos hacer ampliaciones sin tocar la base de datos.

Necesitamos comunicar el *Proxy* con la aplicación web para poder pararlo cuando queramos en remoto. Además necesitamos reiniciar los hilos creados en el *Proxy* cuando modifiquemos algún valor de los sensores en la administración web. Para este cometido el *Proxy* necesita leer constantemente el valor de dos variables, una que indique si está encendido o apagado el *Proxy* desde la web y otro para saber si se ha modificado algún valor de los sensores. Imaginemos que cambiamos la dirección del sensor con el *Proxy* funcionando, cuando modifiquemos el valor cambiará el estado del campo que llamaremos modifica, de esta manera el *Proxy* lo verá, parará los hilos de ejecución y los iniciará con los nuevos valores modificados.

Como ampliación se prevé una asociación de sensores, si pusiéramos varias estaciones de sensores para medir en varias localizaciones, podríamos saber los datos por estación.

Modelo relacional

Con las especificaciones anteriores se realiza el diagrama relacional mostrado en la figura 31, para más tarde generar las tablas necesarias de la BBDD así como planificar las consultas que se realizarán.

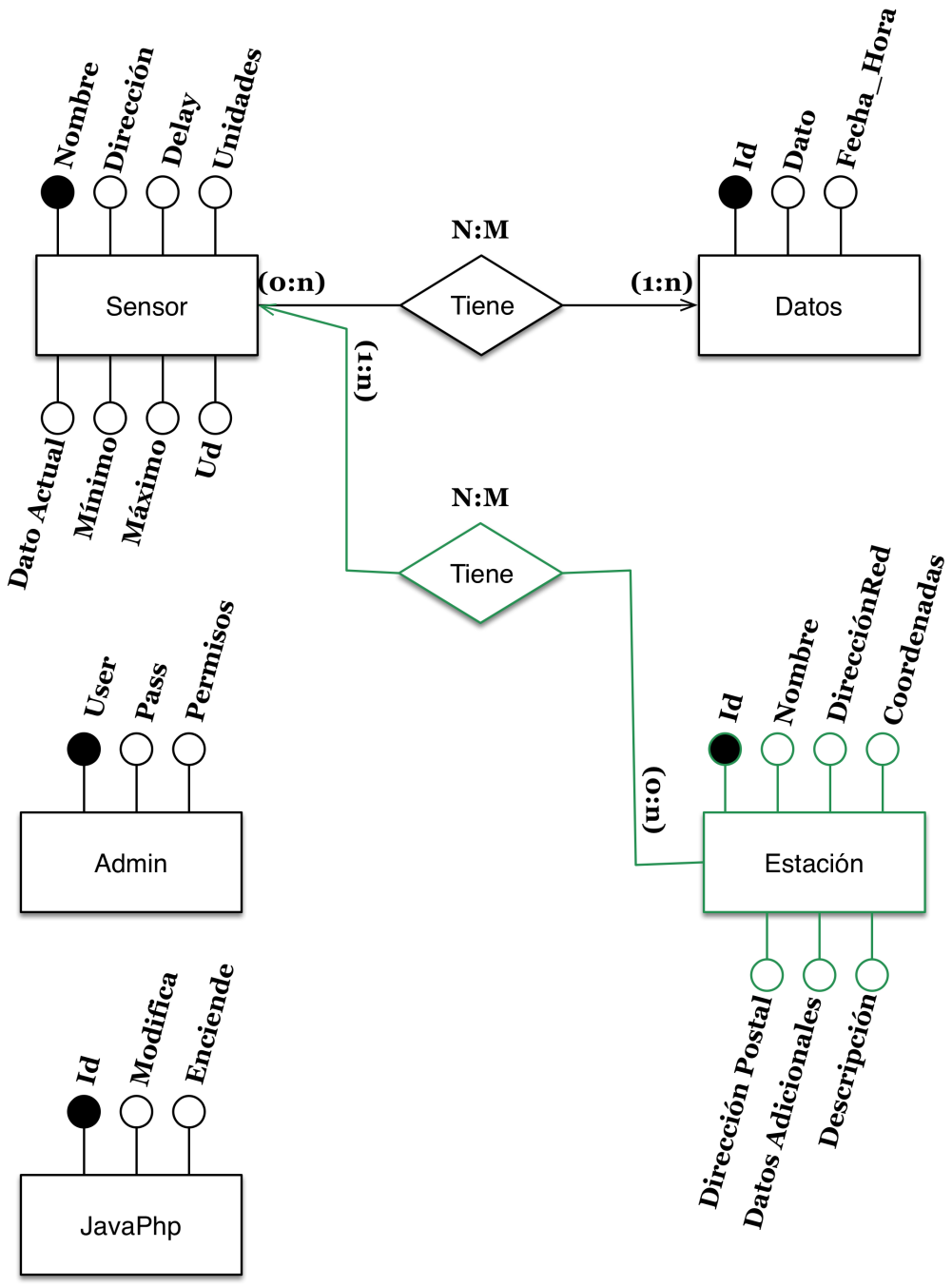


Figura 31 - Diagrama Relacional BBDD

Tablas de la BBDD

A partir del diagrama de modelos relacional, se crean las tablas necesarias para llevar a cabo la funcionalidad diseñada y ejecutarlo en el servidor de bases de datos. A continuación, en la figura, se muestran las tablas creadas.

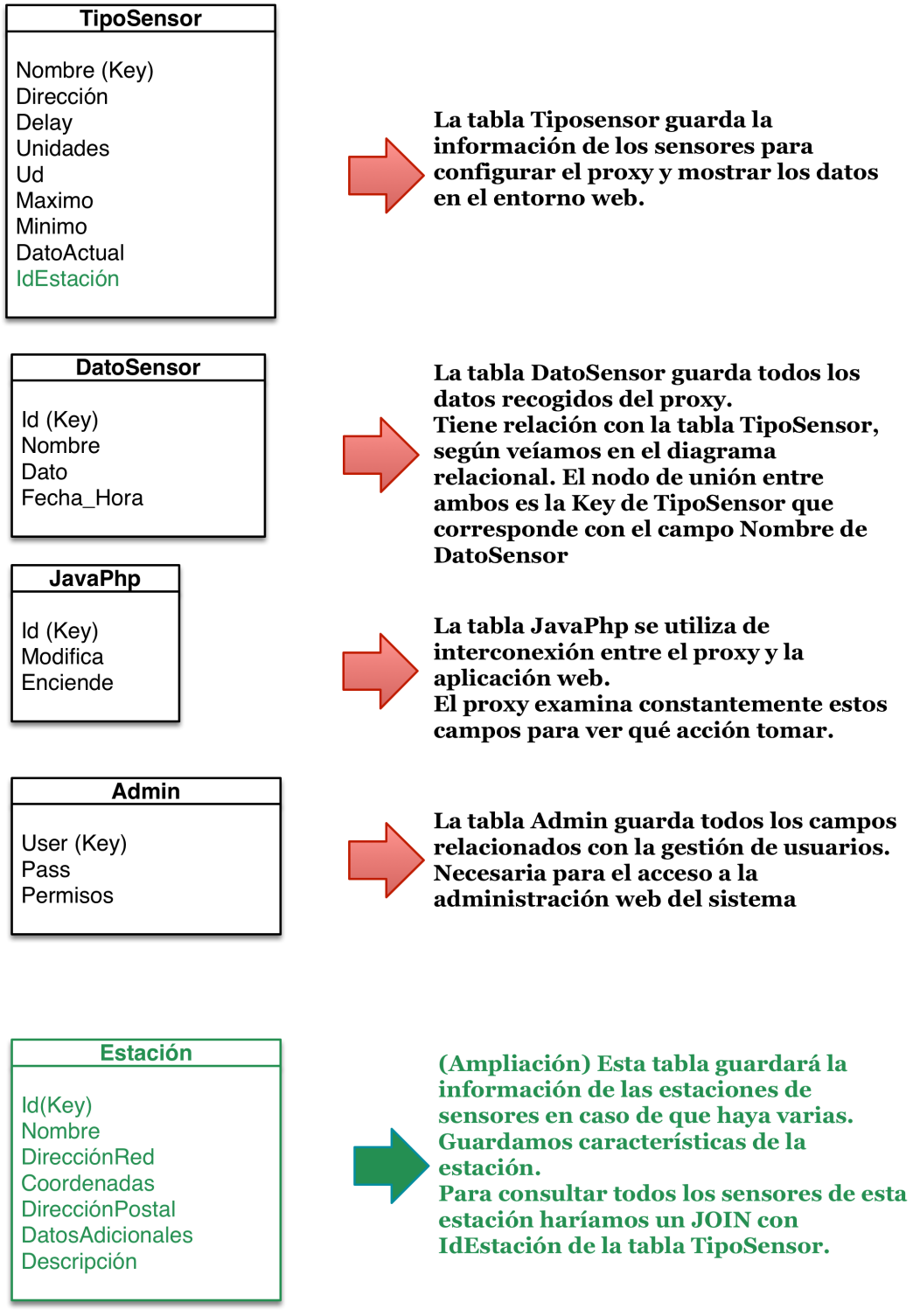


Figura 32 - Uso de tablas de BBDD

Se van a describir para qué se usarán los atributos en cada tabla.

Empezamos por la tabla **TipoSensor** cuya estructura se muestra en la figura 33**:**

- **Nombre:** En este atributo se especifica el nombre del sensor a crear, este nombre ha de ser único debido a que es la primarykey de la tabla. De esta forma no se crearán sensores repetidos. Aunque se intente y la aplicación web no lo contemple el servidor de BBDD desechará las peticiones de insertar un nuevo sensor con un nombre ya existente.

TipoSensor		
Nombre (Key)	VARCHAR(20)	NOTNULL
Dirección	VARCHAR(100)	NOTNULL
Delay	DOUBLE	NOTNULL
Unidades	VARCHAR(20)	NOTNULL
Ud	VARCHAR(2)	NOTNULL
Maximo	VARCHAR(20)	NOTNULL
Minimo	VARCHAR(100)	NOTNULL
DatoActual	VARCHAR(20)	NOTNULL
IdEstación	INTEGER	NOTNULL

Figura 33 – Tabla TipoSensor

- **Dirección:** Este atributo guarda la dirección *CoAP* del sensor donde debemos pedir el dato que actualmente está leyendo el sensor. Esta dirección debe de tener el siguiente formato:

  CoAP://iot.Eclipse.org:5683/large-create/3. Este atributo lo lee el *Proxy* para realizar las llamadas al sensor mediante *CoAP*, si la dirección no existe se genera una excepción java en el *Proxy* no guardando ningún dato en la BBDD.

- **Delay:** Se tiene que expresar en segundos, este atributo refleja el espacio de tiempo entre lecturas del *Proxy*. Dependiendo de este dato se generaran más o menos datos de ese sensor en la BBDD. El *Proxy* lee este atributo y mediante un ***Sleep*** espera en tiempo indicado hasta la siguiente llamada *CoAP* al sensor indicado.

- **Unidades:** Atributo que guarda el tipo de valor que devuelve el sensor configurado. Se utiliza para mostrar en la aplicación las unidades leídas por el sensor.

- **Ud:** Es la abreviatura del campo Unidades, para utilizar en el visionado de datos de los sensores.

- **Máximo:** Este campo se utiliza para generar alarmas al ser sobrepasado por encima el valor configurado en este atributo. El valor depende de la variable que lea el sensor y de los valores que determinemos peligrosos.

- **Mínimo:** Este campo se utiliza para generar alarmas al ser sobrepasado por debajo el valor configurado en este atributo. El valor depende de la variable que lea el sensor y de los valores que determinemos peligrosos.

- **DatoActual:** Campo donde se guarda el último dato recogido por el *Proxy*. Se utiliza para mostrar las alarmas y el dato más cercano en el tiempo que tenemos de un sensor. A la vez que se guarda este campo se guarda en la tabla Datosensor para completar el histórico.

Tabla **DatoSensor,** se muestra en la figura 33 la estructura de esta tabla**:**

- **Id:** Campo auto incrementable que identifica unívocamente un dato. Es la primaryKey de la tabla. Utilizado para poder identificar un elemento y poder eliminarlo, actualizarlo, etc.
- **Nombre:** Atributo que sirve de interconexión con la tabla tipoSensor, es el nexo de unión entre ambas tablas. Indica a qué sensor pertenece el valor guardado en esa línea.

DatoSensor		
Id (Key)	INTERGER AUT.INC.	NOTNULL
Nombre	VARCHAR(20)	NOTNULL
Dato	DOUBLE	NOTNULL
Fecha_Hora	TIMESTAMP	NOTNULL

Figura 34 – Tabla DatoSensor

- **Dato:** Es el valor recuperado del sensor por medio del *Proxy* mediante el protocolo *CoAP*.
- **Fecha:** Campo autogenerado que guarda la fecha y hora en la que se generó el dato. Guarda el timeStamp de manera automática mediante este parámetro:

```
Fecha_hora TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
```

Tabla **Java*PHP***, la figura 35 muestra la estructura creada para esta tabla.

- **Id:** Se utiliza de primaryKey de la tabla.

JavaPhp		
Id (Key)	INTEGER AUT.INC.	NOTNULL
Modifica	TINYINT(1)	NOTNULL
Enciende	TINYINT(1)	NOTNULL

Figura 35 – Tabla Java*PHP*

- **Modifica:** Es un atributo para la interconexión con el *Proxy*. Cuando se modifica algún valor en la administración web y se guarda el valor, este atributo cambia su valor a 1. De este modo el *Proxy*, que consulta constantemente este campo, lo detecta y reinicia los hilos creados para ejecutarlos con los parámetros modificados, una vez ha ocurrido esto vuelve a modificar el atributo en cuestión a 0.
- **Enciende:** Campo con finalidad parecida al anterior atributo. En este caso se utiliza para dejar de leer los datos desde el Proxy, es decir, que para la ejecución de los hilos del Proxy. El campo es modificado desde los botones apaga y enciende Proxy situados en el administrador de la aplicación web.

Tabla **Estación**

Esta tabla está sin determinar totalmente, los atributos no se han desarrollando, depende de un proyecto paralelo que se está llevando a cabo y es este el que está desarrollando el hardware de los sensores. Por lo tanto se deja como ampliación y se muestra en verde en el esquema general mostrado en la figura 36.

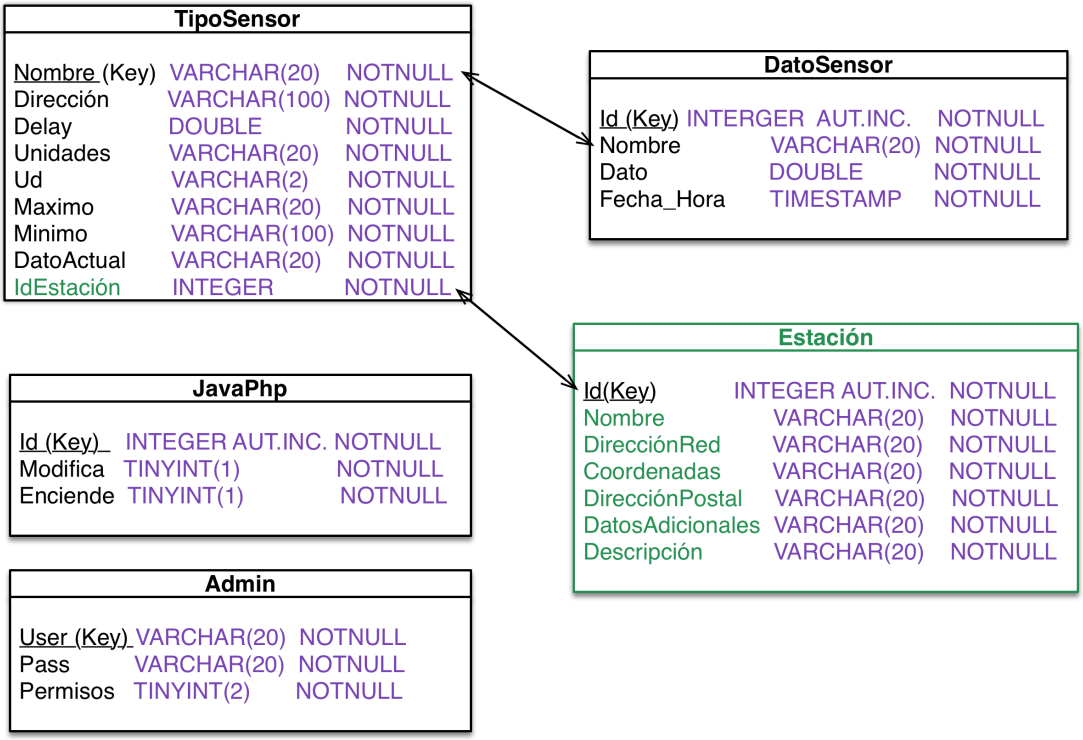


Figura 36 - Diagramas de tablas y tipos de BBDD

Código de creación BBDD

En este punto mostramos el código de creación de la base de datos que se utiliza en la aplicación web para crear la base de datos y empezar con ella en blanco.

Este primer archivo se separa del código por limpieza y seguridad. De esta manera, si se modifica una contraseña, sabes a qué archivo ir directamente sin tener que estar repasando líneas y líneas de código.

```
<?PHP
define('MYSQL_HOST','localhost');
define('MYSQL_USER','davsel');
define('MYSQL_PASS','contraseña');
define('MYSQL_USERR','root');
define('MYSQL_PASSR','ContraseñaRoot');
define('MYSQL_DB','Sensores');
?>
```

Código 3 - db.ini.*PHP*

El siguiente código se corresponde con el archivo CreaDb.*PHP* al que llamamos desde el controlador de la aplicación web para crear la BBDD. Aquí se pueden ver cómo se han configurado todos y cada uno de los atributos y tablas que diseñábamos en apartados anteriores.

```
<?PHP
    require 'db.ini.PHP';
    require_once("./class/PaqClass.PHP");

    //Creamos un objeto de la clase paquete
    $paquete = new Paq();

    $query="CREATE DATABASE IF NOT EXISTS Sensores";
    $paquete->conectarCrearBD($query);

    //Da permisos a BBDD
    $User=MYSQL_USER;
    $Host=MYSQL_HOST;
    $query="GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, ALTER ON
                                                    Sensores.* TO `$User`@`$Host`;";
    $paquete->ejeConsultaR ($query);

    //Crea Tabla admin para guardar user
    $query="CREATE TABLE IF NOT EXISTS Admin (
                        User VARCHAR(20) NOT NULL,
                        Pass VARCHAR(20) NOT NULL ,
                        Permisos TINYINT(2) NOT NULL ,
                        PRIMARY KEY (User)
```

```
                    )
                    ENGINE=MyISAM;";
$paquete->ejeConsulta($query);
//MySQL_query($query,$con) or die(MySQL_error($con));


$query="INSERT IGNORE INTO Sensores.Admin (User,Pass,Permisos)
                                                VALUES ('Admin', 'Pass','99')";
$paquete->ejeConsulta($query);
$query="INSERT IGNORE INTO Sensores.Admin (User,Pass,Permisos)
                                                    VALUES ('User', 'Pass','0')";
$paquete->ejeConsulta($query);


$query="CREATE TABLE IF NOT EXISTS TipoSensor (
                    Nombre VARCHAR(20) NOT NULL,
                    Direccion VARCHAR(100) NOT NULL ,
                    Delay DOUBLE NOT NULL ,
                    Unidades VARCHAR(20) NOT NULL ,
                    Ud VARCHAR(2) NOT NULL ,
                    Maximo VARCHAR(20) NOT NULL,
                    Minimo VARCHAR(20) NOT NULL,
                    DatoActual VARCHAR(20) ,
                    PRIMARY KEY (Nombre)
                    )
                    ENGINE=MyISAM;";
$paquete->ejeConsulta($query);


//Revisar current_Timestamp
$query="CREATE TABLE IF NOT EXISTS DatoSensor (
                    Id INTEGER UNSIGNED AUTO_INCREMENT,
                    Nombre VARCHAR(20) NOT NULL,
                    Dato DOUBLE NOT NULL,
                    Fecha_hora TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON
                                                UPDATE CURRENT_TIMESTAMP ,
                    PRIMARY KEY (Id)
                    )
                    ENGINE=MyISAM;";
$paquete->ejeConsulta($query);

$query = "ALTER TABLE Sensores.DatoSensor ADD INDEX (`Nombre`);";
$paquete->ejeConsulta($query);

$query = "ALTER TABLE Sensores.DatoSensor ADD INDEX (`Fecha_hora`);";
$paquete->ejeConsulta($query);
```

```
    $query="CREATE TABLE IF NOT EXISTS JavaPHP (
                        Id INTEGER UNSIGNED AUTO_INCREMENT,
                        Modifica TINYINT(1) NOT NULL,
                        Enciende TINYINT(1) NOT NULL,
                        PRIMARY KEY (Id)
                        )
                        ENGINE=MyISAM;";
    $paquete->ejeConsulta($query);

    $query="INSERT IGNORE INTO Sensores.JavaPHP (Modifica,Enciende)
                                                        VALUES ('0', '1')";
    $paquete->ejeConsulta($query);

    header('Location:admin.PHP');
?>
```

Código 4 - Creación BBDD *PHP*

Diseño de la aplicación web.

La aplicación web cumple con el patrón de diseño de software Modelo-Vista-Controlador. En la siguiente figura vemos un esquema genérico de cómo se realiza el diseño cumpliendo este patrón.

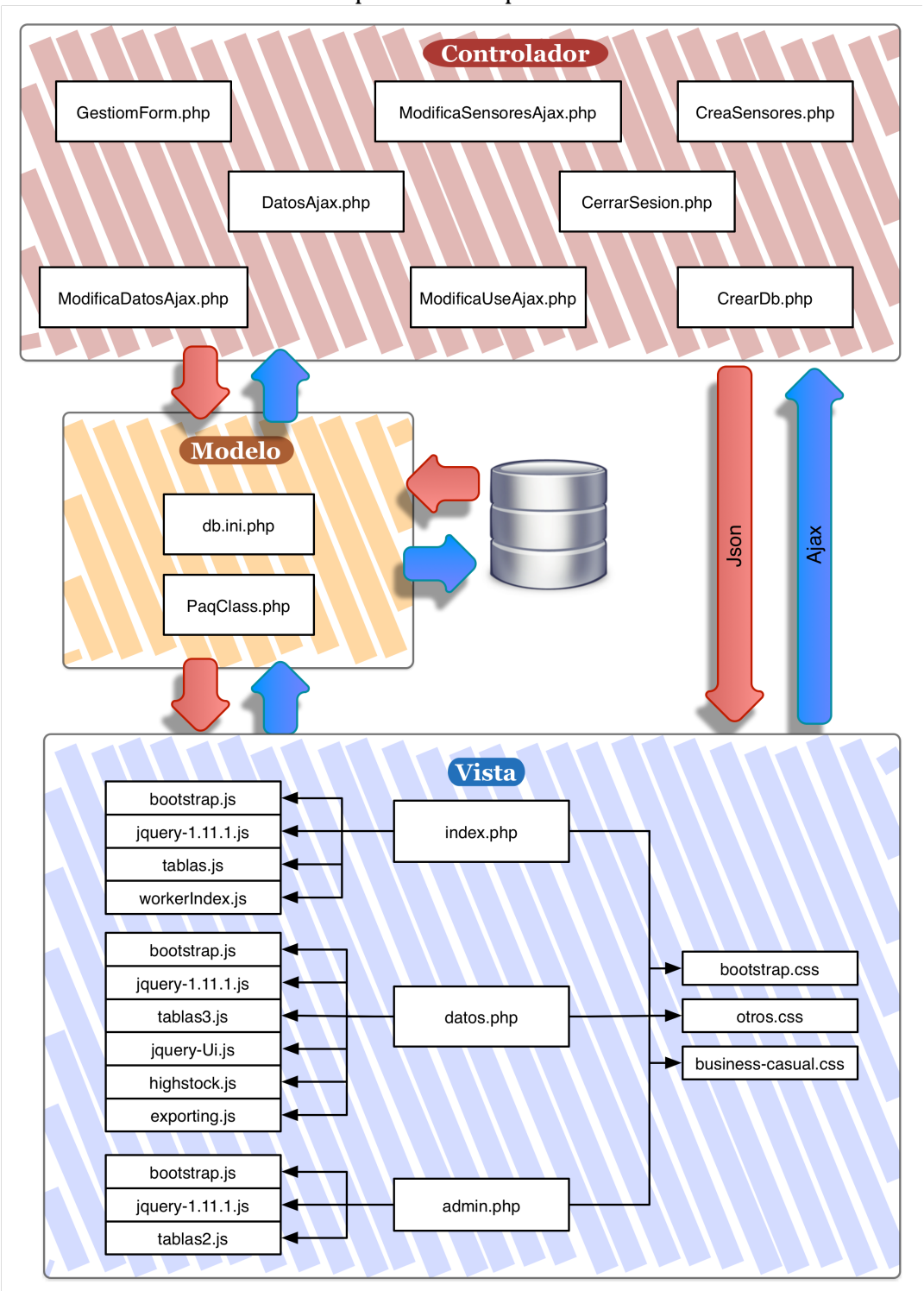


Figura 37 - Modelo Vista Controlador

El modelo es el encargado de hablar con la base de datos, conecta con ella y trae los datos que necesitan las demás partes del diseño. En el modelo tenemos dos archivos, PaqClass.*PHP* y db.ini.*PHP*. El primero es una clase que contiene todas las funciones necesarias para conectar con la BBDD. El segundo contiene las contraseñas de acceso a la base de datos, así como el nombre de la BBDD y modo de acceso.

En la vista tenemos todos los archivos *HTML* con código *PHP*, además de los modificadores de estilo *CSS* y el tratamiento dinámico de usuario *JavaScript*. Estos son los archivos que se ejecutan en el navegador del usuario. Son la interface con la que interactúan los clientes para administrar, visualizar y modificar datos de sensores.

En la vista se utilizan muchas librerías, tanto para *CSS* como para *JavaScript*. Para *CSS* utilizamos bootstrap con el fin de facilitar el diseño de la interface web. En *JavaScript* se utiliza Jquery, jquery-UI para mejorar y facilitar el mantenimiento del código. Hightchart realiza la función de generar gráficos de los datos obtenidos.

Los archivos tablas.js, son necesarios para realizar las conexiones *Ajax* con el controlador y así no tener que cargar constantemente la página para actualizar datos.

Todas las vistas de la aplicación web se adaptan a los distintos tamaños de pantallas. A través de bootstrap, se han diseñado las tablas y los div para que se adapten de manera correcta a cuatro tipos de pantalla. Pantallas Xs de menos de 768 px, pantallas Sm entre 768 y 992 px, pantallas Md entre 992 y 1200 px y pantallas Lg mayores de 1200 px. Estas pantallas se corresponden con móviles, Tablet, pequeños ordenadores y ordenadores de sobremesa respectivamente. La pantalla se divide en 12 columnas, al diseñar ajustamos el contenido a esas columnas. Se pueden añadir las filas que se crean convenientes. Así nos aseguramos que el contenido de la web se verá de forma correcta en todos los dispositivos. Un ejemplo de configuración de un div con autoajuste es el siguiente :

```
<div class="col-xs-6 col-sm-offset-1 col-sm-3 col-md-2 col-lg-2">
```

El controlador contiene todos los archivos que interactúan entre el modelo y las vistas. Las vistas mediante *Ajax* o formularios comunican con los archivos que forman el controlador, este en la parte servidora, consigue los datos a través del modelo, los trata y se los devuelve a la vista para que el usuario vea el contenido. Hace de intermediario entre vista y modelo. En ocasiones, las vistas llaman directamente al modelo para conseguir un dato puntual que no requiere tratamiento.

Index.*PHP*

En este apartado vamos a ver el diseño de los archivos que interactúan directamente con la vista index.*PHP*.

En la siguiente figura se ve el patrón MVC específico de esta parte de la aplicación web. Solo se visualizan los archivos con los que se toma contacto desde index.*PHP*.

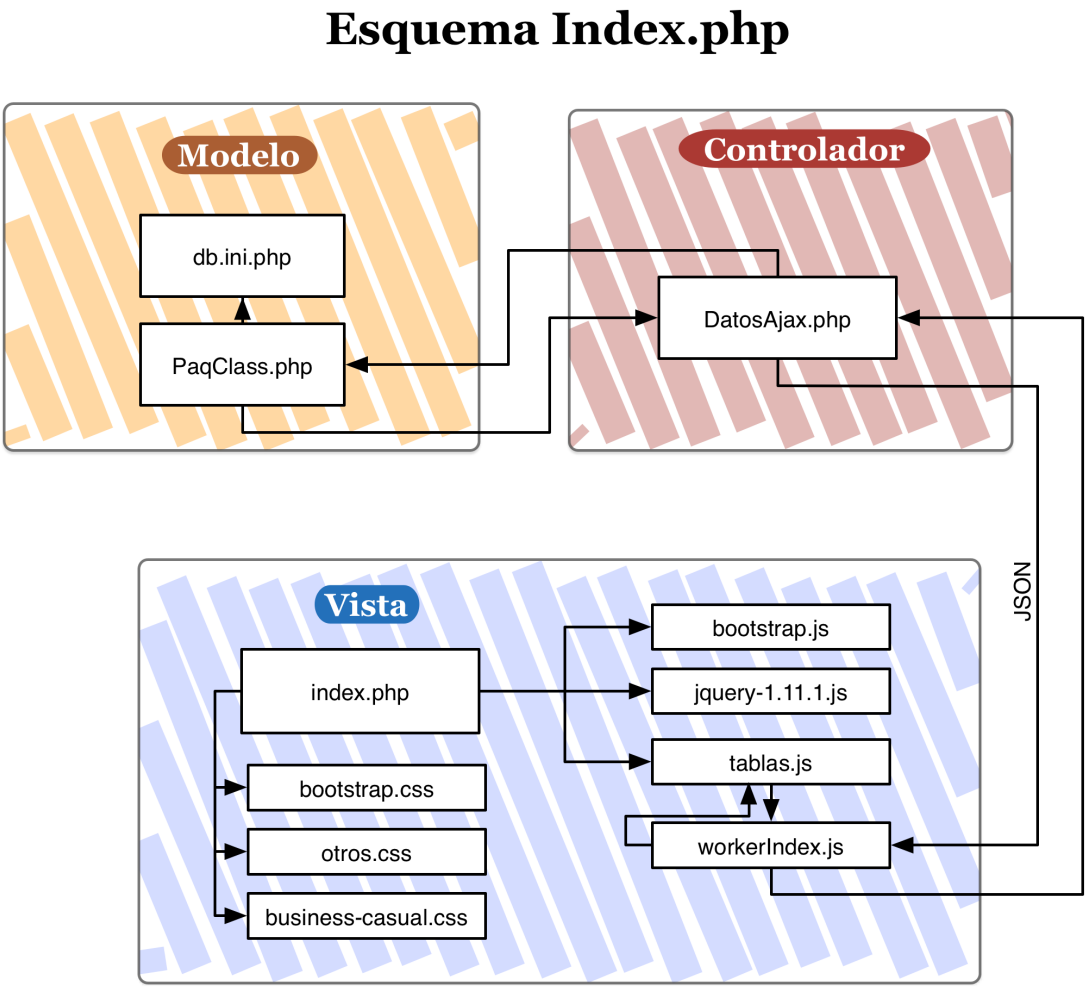


Figura 38 - Esquema MVC de Index.*PHP*

En la vista cabe destacar el uso de wokers. Se usa con el fin de realizar llamadas reiterativas a la BBDD mediante *Ajax*. Es el símil a los hilos, la programación concurrente que se utiliza en todos los lenguajes de programación modernos. En *HTML*5 se incluye este sistema de ejecución concurrente de código, aunque mucho más limitado en comparación de la oferta de otros lenguajes de programación.

Cada cierto tiempo el worker, por detrás de lo que esté haciendo el usuario, recupera los datos cada segundo conectando con el controlador Datos*Ajax.PHP* mediante *Ajax*, este habla con el modelo y devuelve los datos

codificados en *JSON*. El worker ofrece estos datos a tablas.js los decodifica e introduce dinámicamente en la tabla de datos de index.*PHP*.

En los siguientes flujogramas vemos cómo se realizan estos procesos que se acaban de explicar.

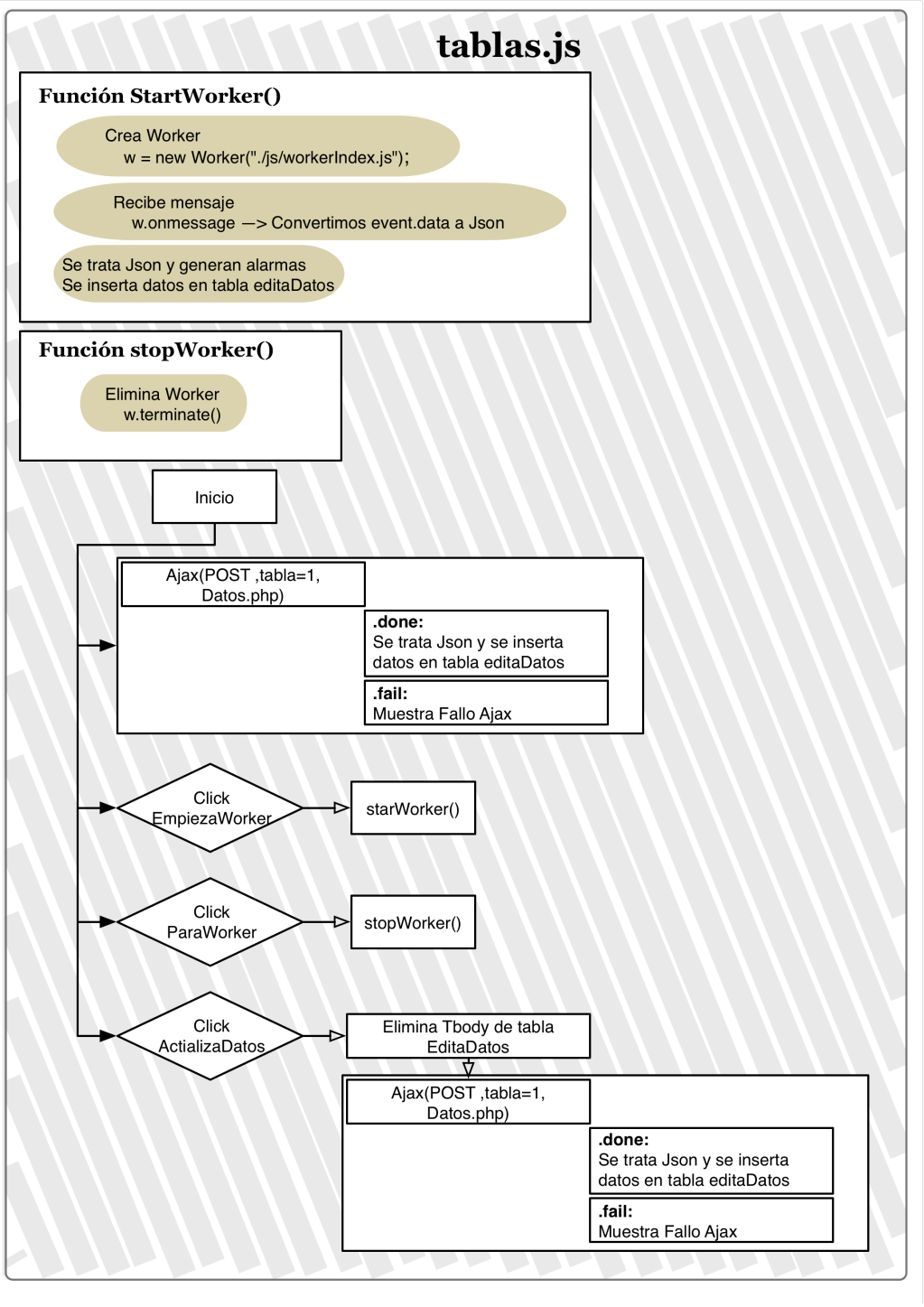


Figura 39 - Flujograma tablas.js

Figura 40 – Flujograma WorkerIndex.js

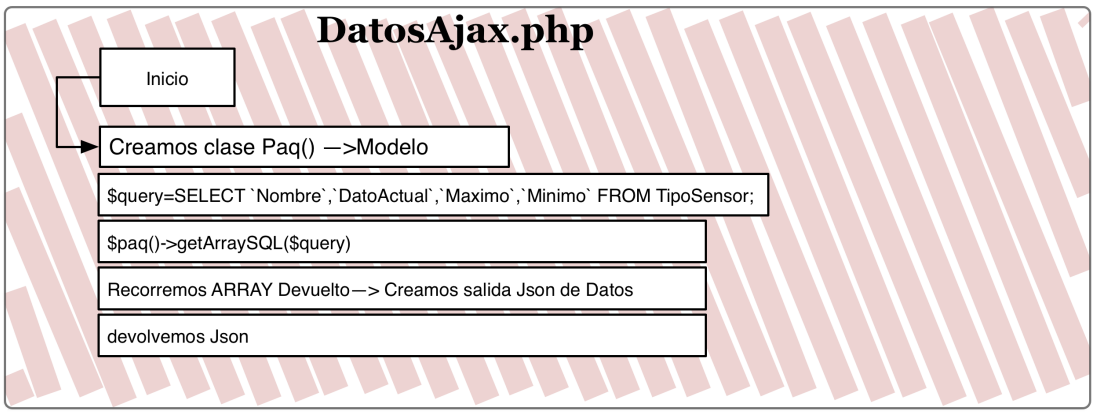


Figura 41 - Flujograma Datos*Ajax.PHP*

Datos.*PHP*

La vista datos.*PHP* tiene diversas formas dependiendo de la elección del usuario. Primero nos muestra un listado con los sensores dados de alta. Una vez elegimos un sensor, nos muestra un cuadro de diálogo para elegir el filtro de datos, o mostramos todo o filtramos por rango de fechas. Una vez se elige un filtro se muestra una tabla con los datos encontrados según filtro y el gráfico con esos datos. Dependiendo de si estamos autentificados con permisos de usuario o de administrador nos deja editar y eliminar datos de la tabla o no.

La tabla se actualiza, edita y elimina sin recargar la página completa, solo se actualizan los datos de esa tabla mediante *Ajax*.

En los flujogramas que se muestran a continuación, queda claro el funcionamiento de esta parte del sistema. Para conseguir estas especificaciones se sigue este esquema MVC especifico para esta vista.

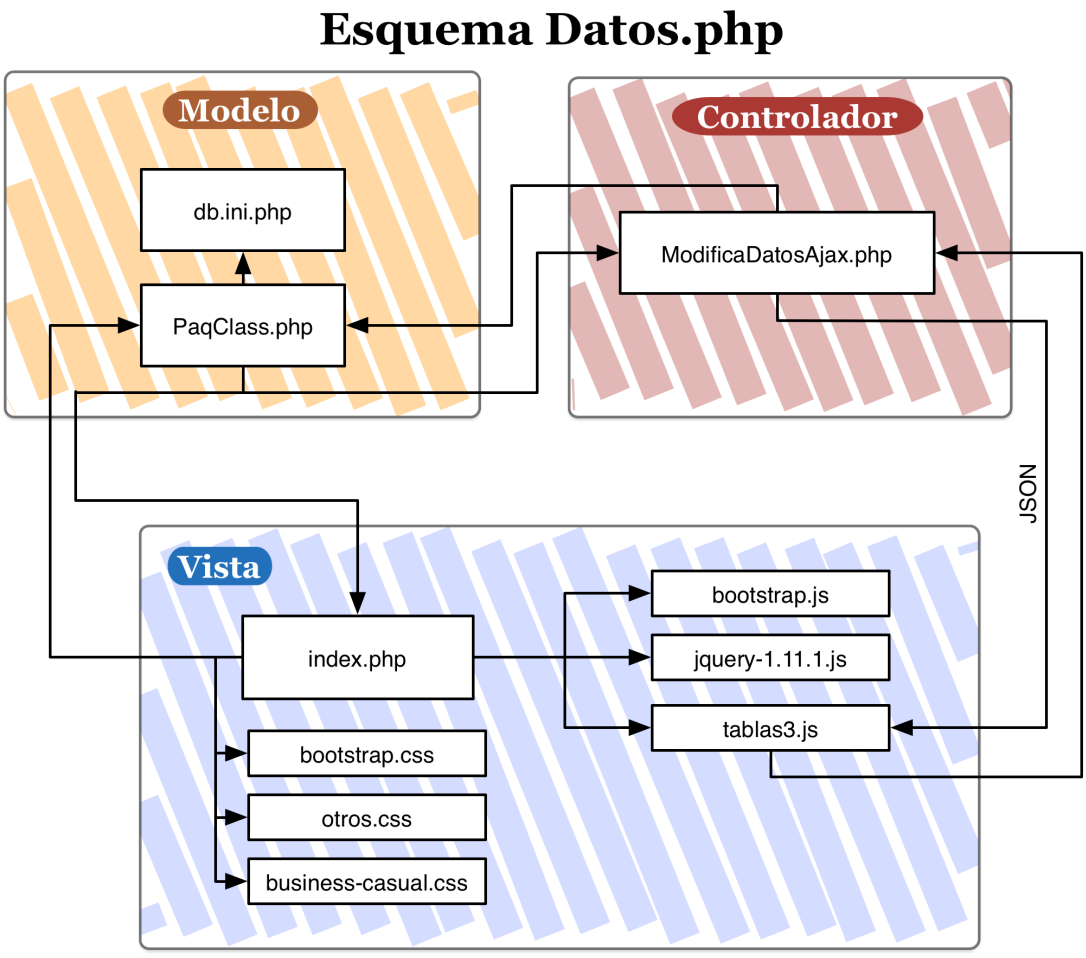


Figura 42 - Esquema MVC de Datos.*PHP*

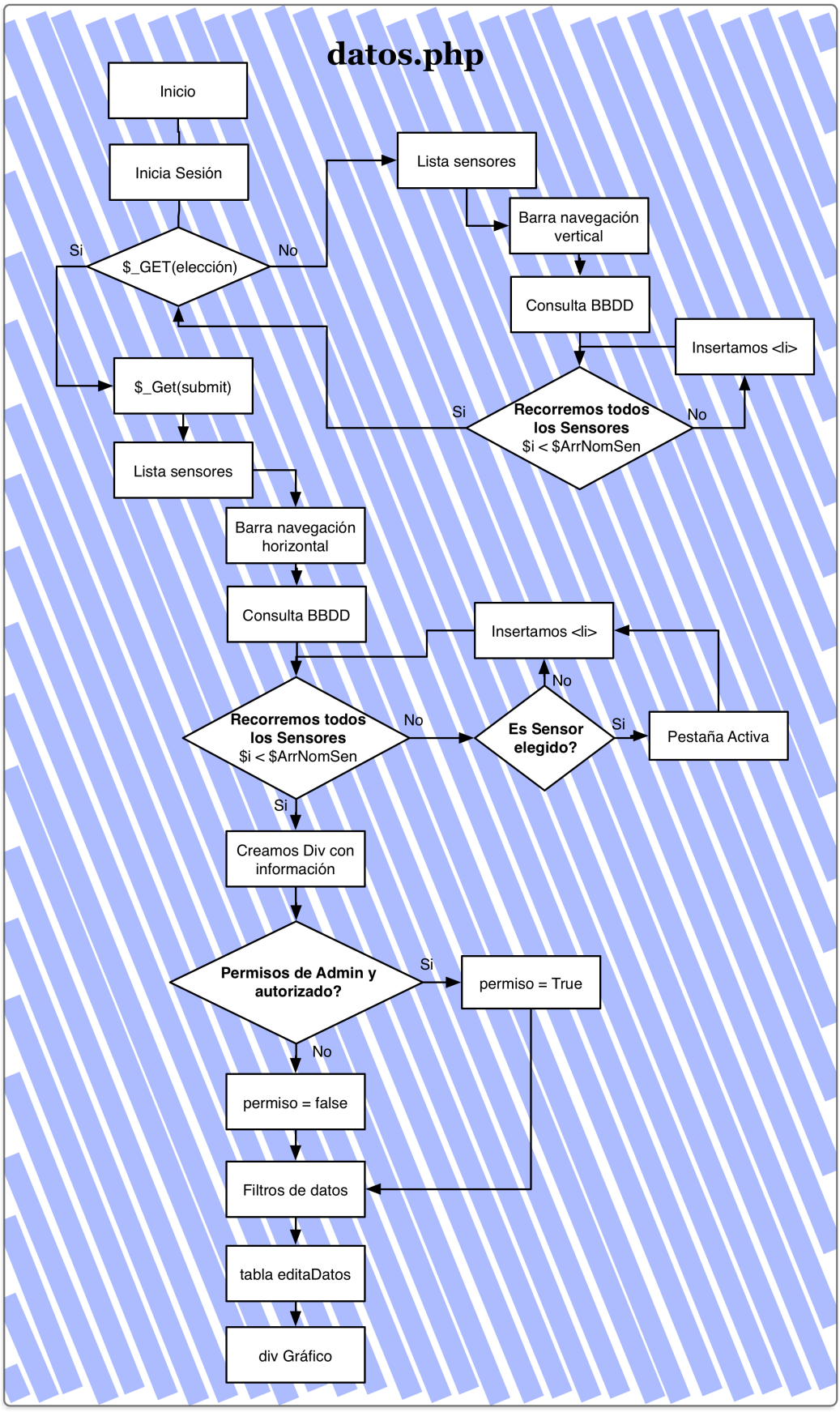
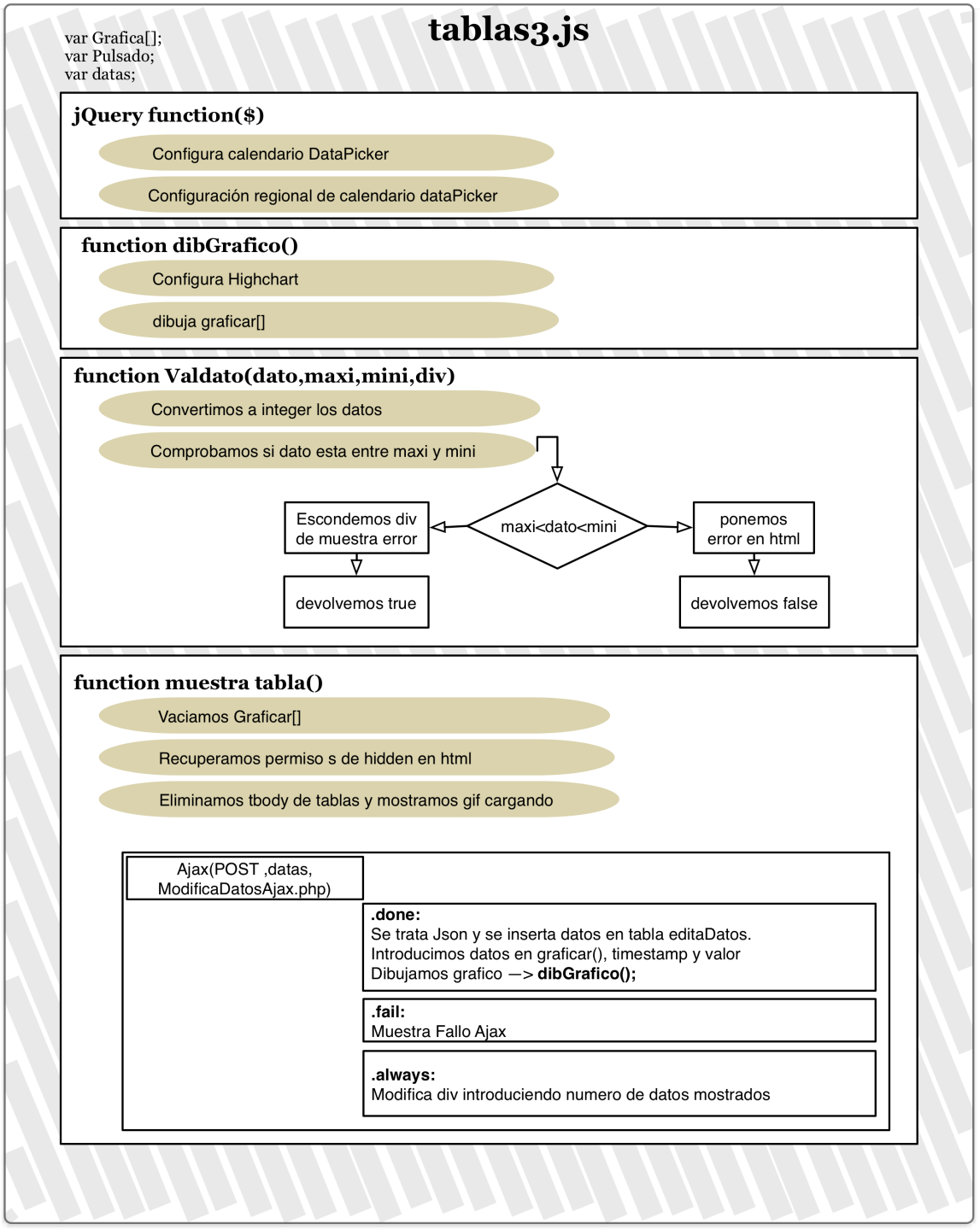


Figura 43 - Flujograma Datos.*PHP*


tablas3.js
var Grafica[];
var Pulsado;
var datas;
jQuery function($)
Configura calendario DataPicker
Configuración regional de calendario dataPicker
function dibGrafico()
Configura Highchart
dibuja graficar[]
function Valdato(dato,maxi,mini,div)
Convertimos a integer los datos
Comprobamos si dato esta entre maxi y mini
maxi<dato<mini
Escondemos div de muestra error
ponemos error en html
devolvemos true
devolvemos false
function muestra tabla()
Vaciamos Graficar[]
Recuperamos permiso s de hidden en html
Eliminamos tbody de tablas y mostramos gif cargando
Ajax(POST ,datas, ModificaDatosAjax.php)
.done:
Se trata Json y se inserta datos en tabla editaDatos.
Introducimos datos en graficar(), timestamp y valor
Dibujamos grafico —> dibGrafico();
.fail:
Muestra Fallo Ajax
.always:
Modifica div introduciendo numero de datos mostrados

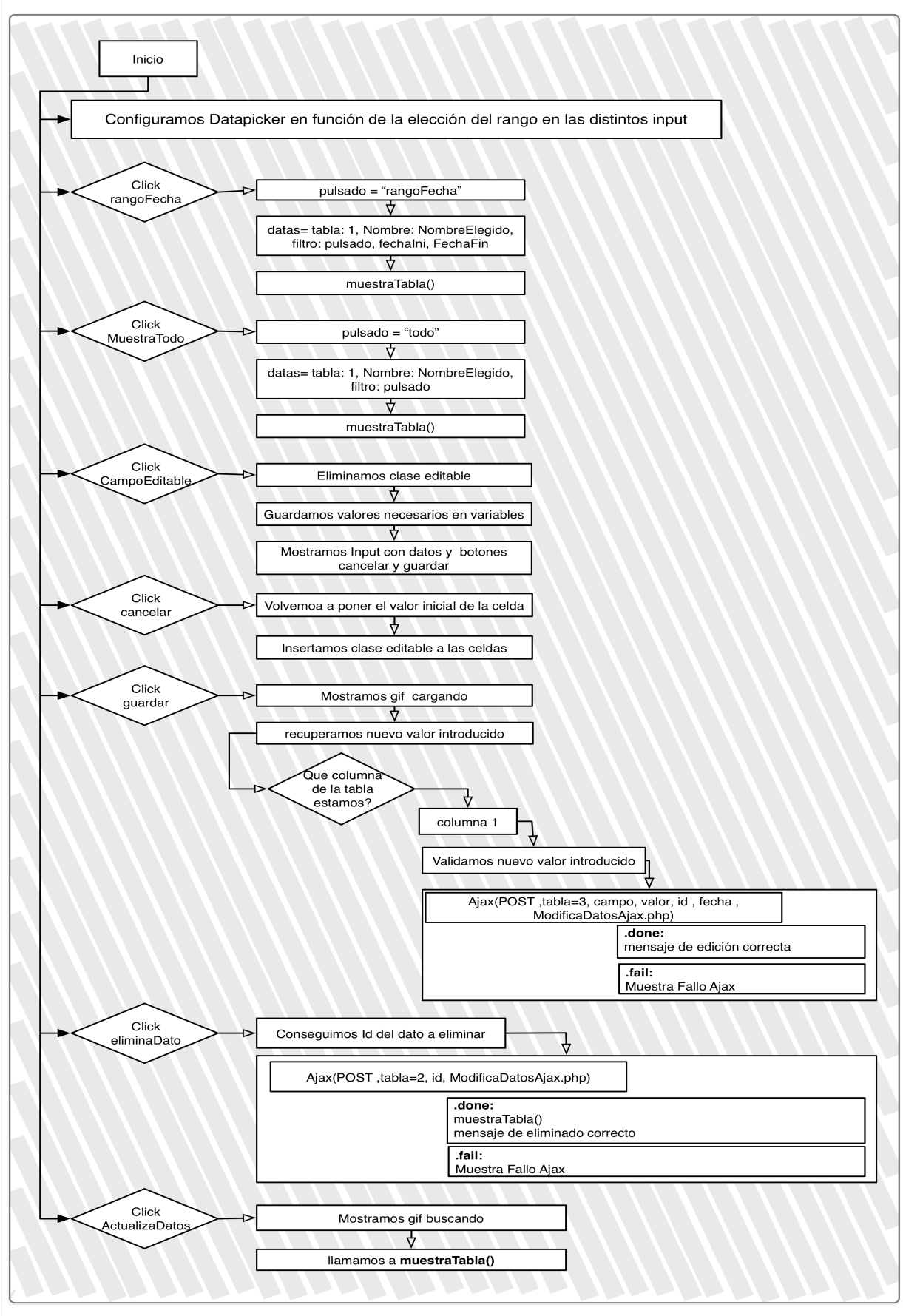
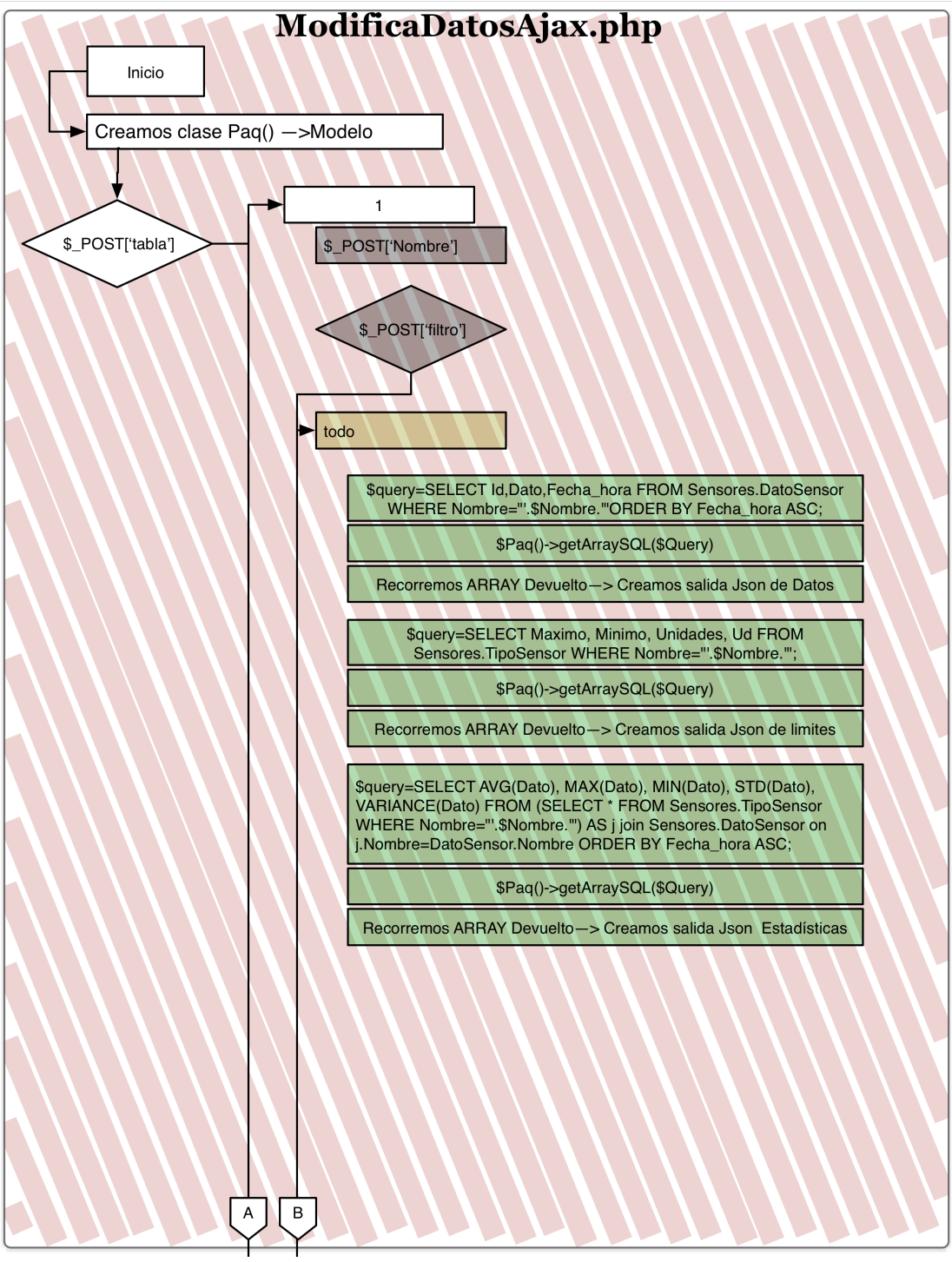


Figura 44 - tablas3.js


ModificaDatosAjax.php
Inicio
Creamos clase Paq() —>Modelo
$_POST['tabla']
1
$_POST['Nombre']
$_POST['filtro']
todo
$query=SELECT Id,Dato,Fecha_hora FROM Sensores.DatoSensor WHERE Nombre="'.$Nombre.'"ORDER BY Fecha_hora ASC;
$Paq()->getArraySQL($Query)
Recorremos ARRAY Devuelto—> Creamos salida Json de Datos
$query=SELECT Maximo, Minimo, Unidades, Ud FROM Sensores.TipoSensor WHERE Nombre="'.$Nombre.'";
$Paq()->getArraySQL($Query)
Recorremos ARRAY Devuelto—> Creamos salida Json de limites
$query=SELECT AVG(Dato), MAX(Dato), MIN(Dato), STD(Dato), VARIANCE(Dato) FROM (SELECT * FROM Sensores.TipoSensor WHERE Nombre="'.$Nombre.'") AS j join Sensores.DatoSensor on j.Nombre=DatoSensor.Nombre ORDER BY Fecha_hora ASC;
$Paq()->getArraySQL($Query)
Recorremos ARRAY Devuelto—> Creamos salida Json Estadísticas
A
B

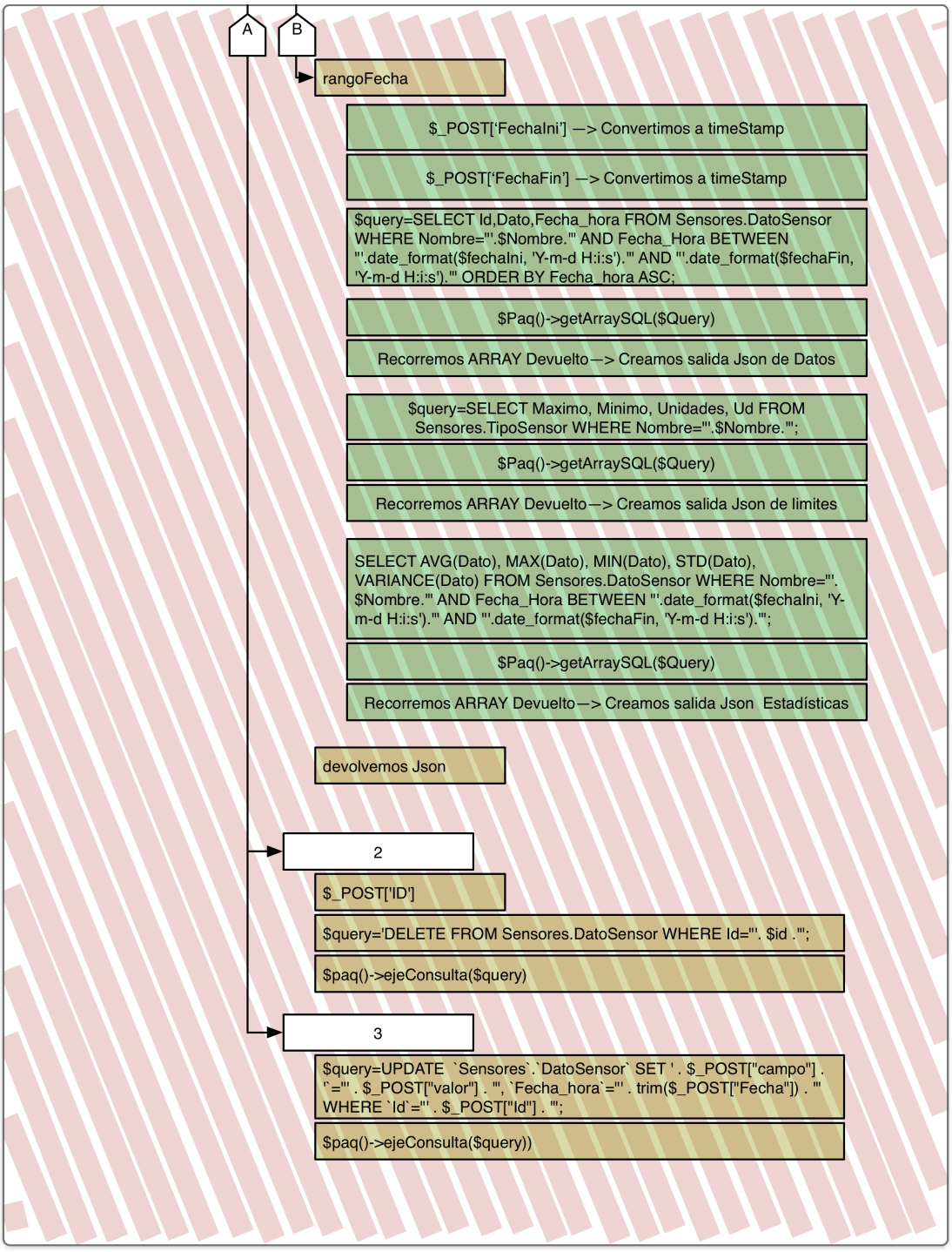


Figura 45 - Flujograma ModificaDatos.*PHP*

Admin.PHP

La vista Admin.*PHP* pretende ser el sistema de administración del sistema completo, desde aquí se configuran los sensores, los usuarios, el *Proxy* y la BBDD.

La información mostrada en esta vista es distinta dependiendo de si estamos autentificados con permisos de usuario o de administrador. Si somos usuarios solo podemos visualizar datos de sensores, si somos administradores podemos editar, crear y eliminar datos de usuario y sensores. Además de parar *Proxy*, encender *Proxy* y administrar BBDD.

Las tablas de sensores y usuarios se editan, eliminan y crean sin cargar la página por completo, solo se modifican los datos de la tabla afectada por la modificación. Esto se consigue mediante una comunicación *Ajax* con el controlador, este a su vez llama al modelo. El controlador devuelve los datos que generó el modelo mediante *JSON*, tablas.js decodifican estos datos y actualizan la tabla en cuestión.

En el siguiente esquema se muestra la estructura MVC de Admin.*PHP*. Las imágenes que siguen al esquema MVC representan el funcionamiento de esta parte de la aplicación web.

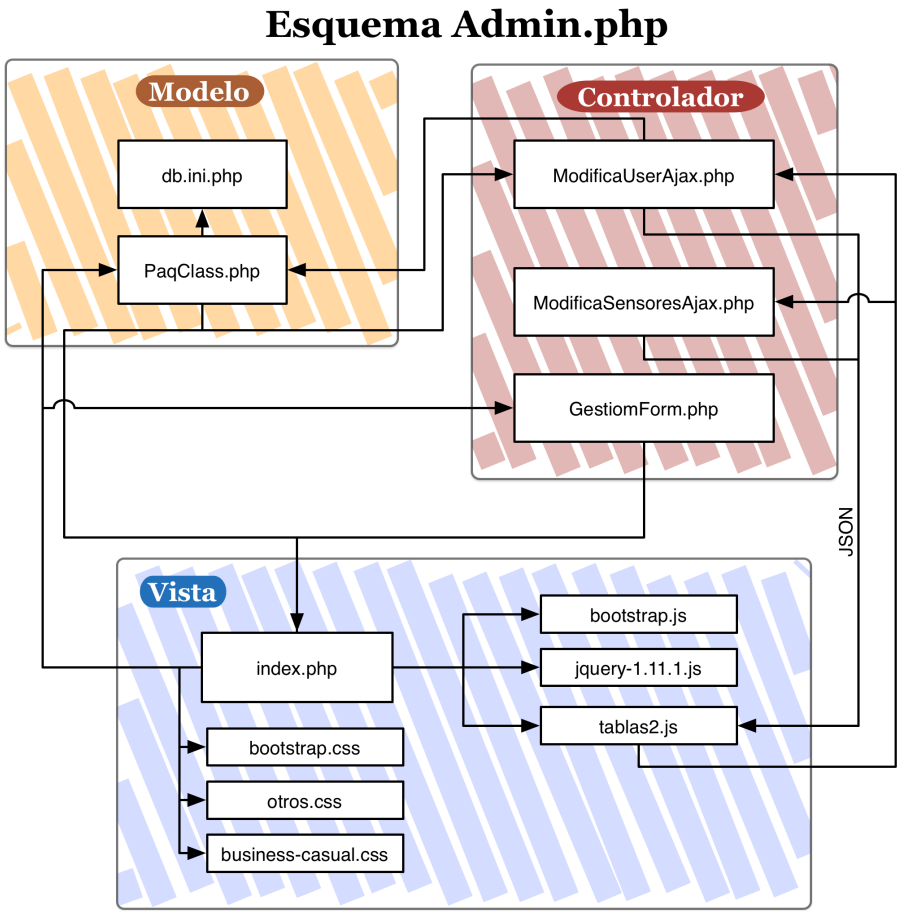


Figura 46 - Esquema MVC de Admin.*PHP*

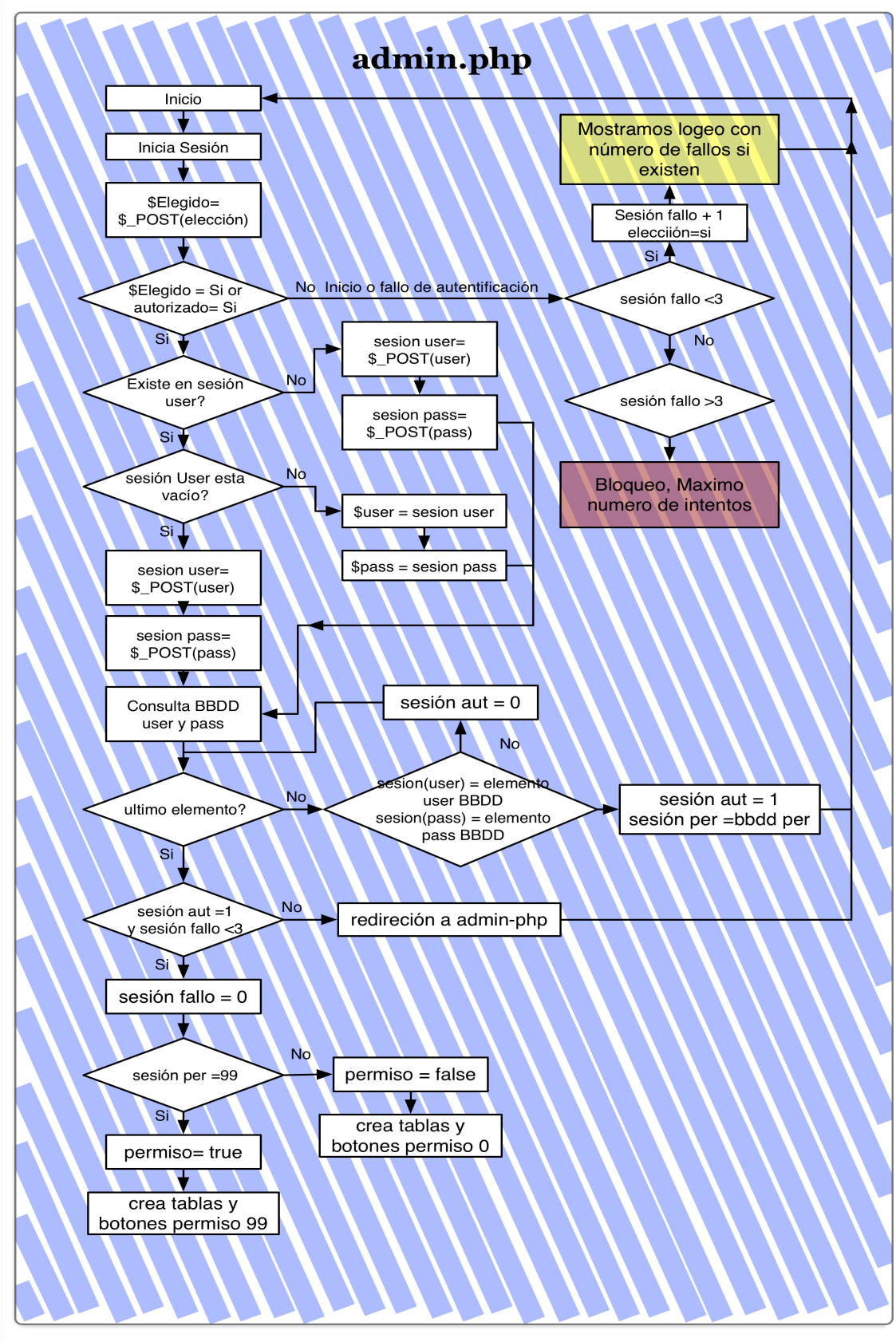
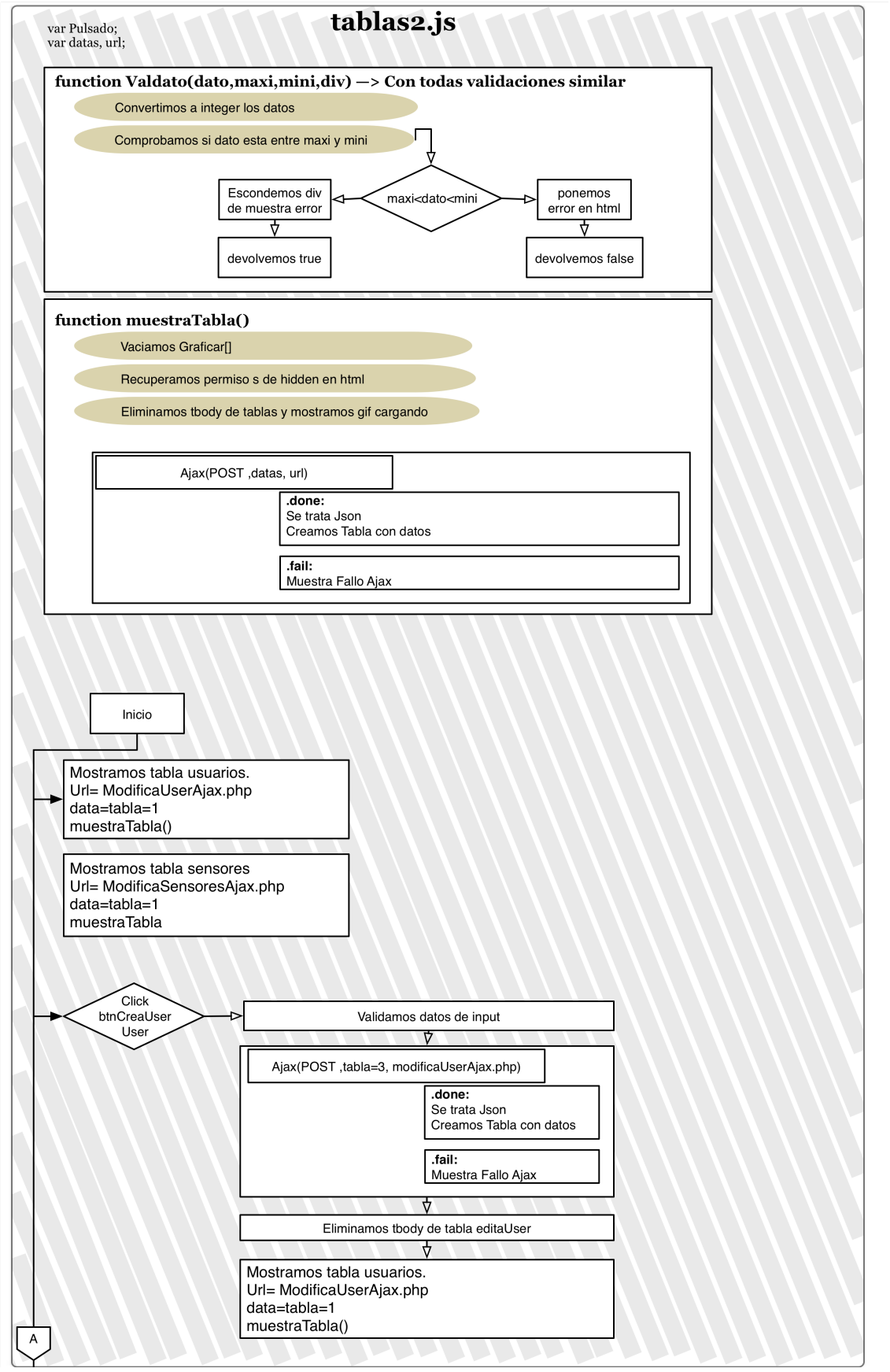
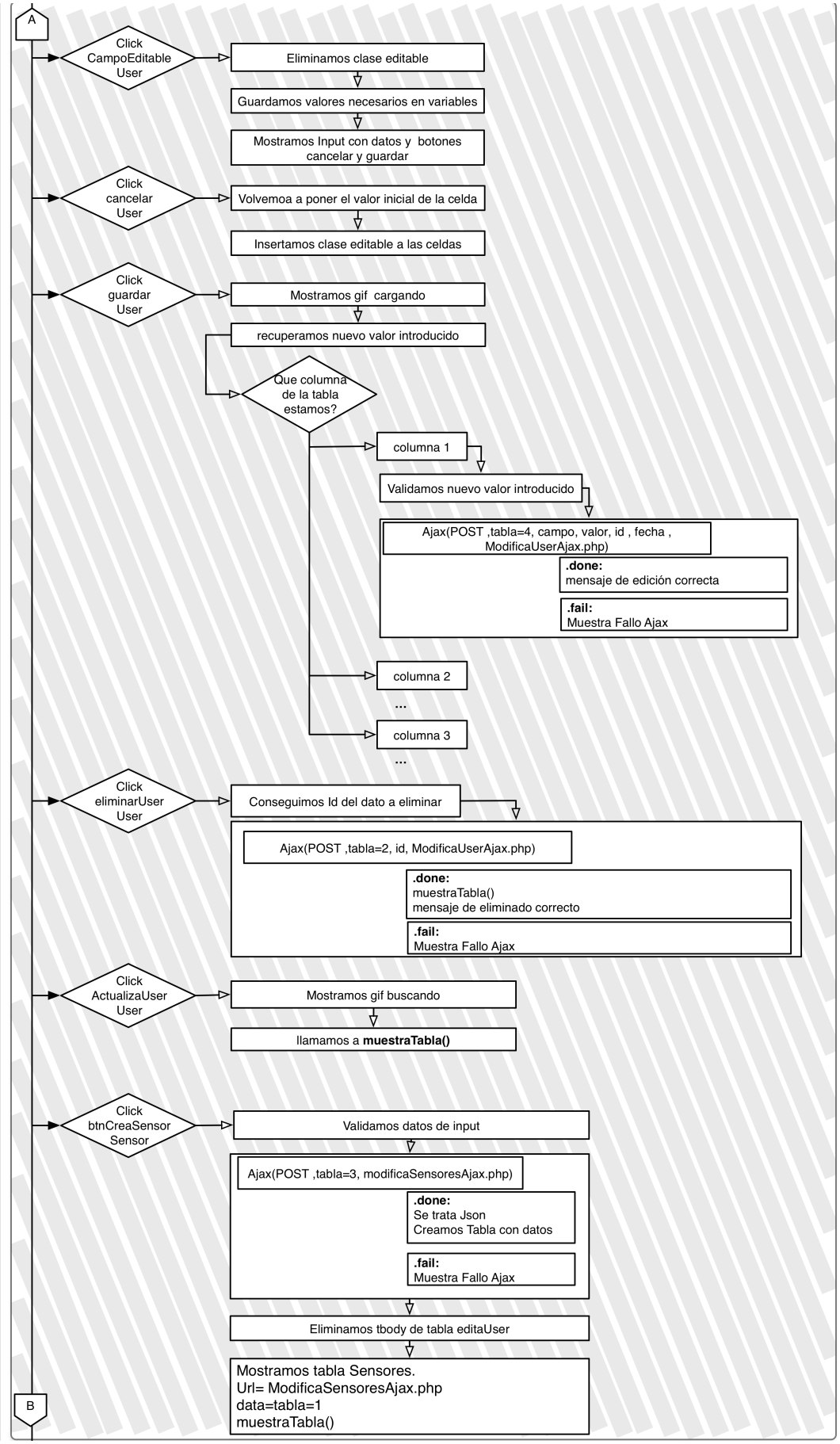


Figura 47 - Flujograma admin.*PHP*


tablas2.js
var Pulsado;
var datas, url;
function Valdato(dato,maxi,mini,div) —> Con todas validaciones similar
Convertimos a integer los datos
Comprobamos si dato esta entre maxi y mini
maxi<dato<mini
Escondemos div de muestra error
ponemos error en html
devolvemos true
devolvemos false
function muestraTabla()
Vaciamos Graficar[]
Recuperamos permiso s de hidden en html
Eliminamos tbody de tablas y mostramos gif cargando
Ajax(POST ,datas, url)
.done:
Se trata Json
Creamos Tabla con datos
.fail:
Muestra Fallo Ajax
Inicio
Mostramos tabla usuarios.
Url= ModificaUserAjax.php
data=tabla=1
muestraTabla()
Mostramos tabla sensores
Url= ModificaSensoresAjax.php
data=tabla=1
muestraTabla
Click btnCreaUser User
Validamos datos de input
Ajax(POST ,tabla=3, modificaUserAjax.php)
.done:
Se trata Json
Creamos Tabla con datos
.fail:
Muestra Fallo Ajax
Eliminamos tbody de tabla editaUser
Mostramos tabla usuarios.
Url= ModificaUserAjax.php
data=tabla=1
muestraTabla()
A


A
Click CampoEditable User
Eliminamos clase editable
Guardamos valores necesarios en variables
Mostramos Input con datos y botones cancelar y guardar
Click cancelar User
Volvemoa a poner el valor inicial de la celda
Insertamos clase editable a las celdas
Click guardar User
Mostramos gif cargando
recuperamos nuevo valor introducido
Que columna de la tabla estamos?
columna 1
Validamos nuevo valor introducido
Ajax(POST ,tabla=4, campo, valor, id , fecha , ModificaUserAjax.php)
.done:
mensaje de edición correcta
.fail:
Muestra Fallo Ajax
columna 2
...
columna 3
...
Click eliminarUser User
Conseguimos Id del dato a eliminar
Ajax(POST ,tabla=2, id, ModificaUserAjax.php)
.done:
muestraTabla()
mensaje de eliminado correcto
.fail:
Muestra Fallo Ajax
Click ActualizaUser User
Mostramos gif buscando
llamamos a muestraTabla()
Click btnCreaSensor Sensor
Validamos datos de input
Ajax(POST ,tabla=3, modificaSensoresAjax.php)
.done:
Se trata Json
Creamos Tabla con datos
.fail:
Muestra Fallo Ajax
Eliminamos tbody de tabla editaUser
Mostramos tabla Sensores.
Url= ModificaSensoresAjax.php
data=tabla=1
muestraTabla()
B

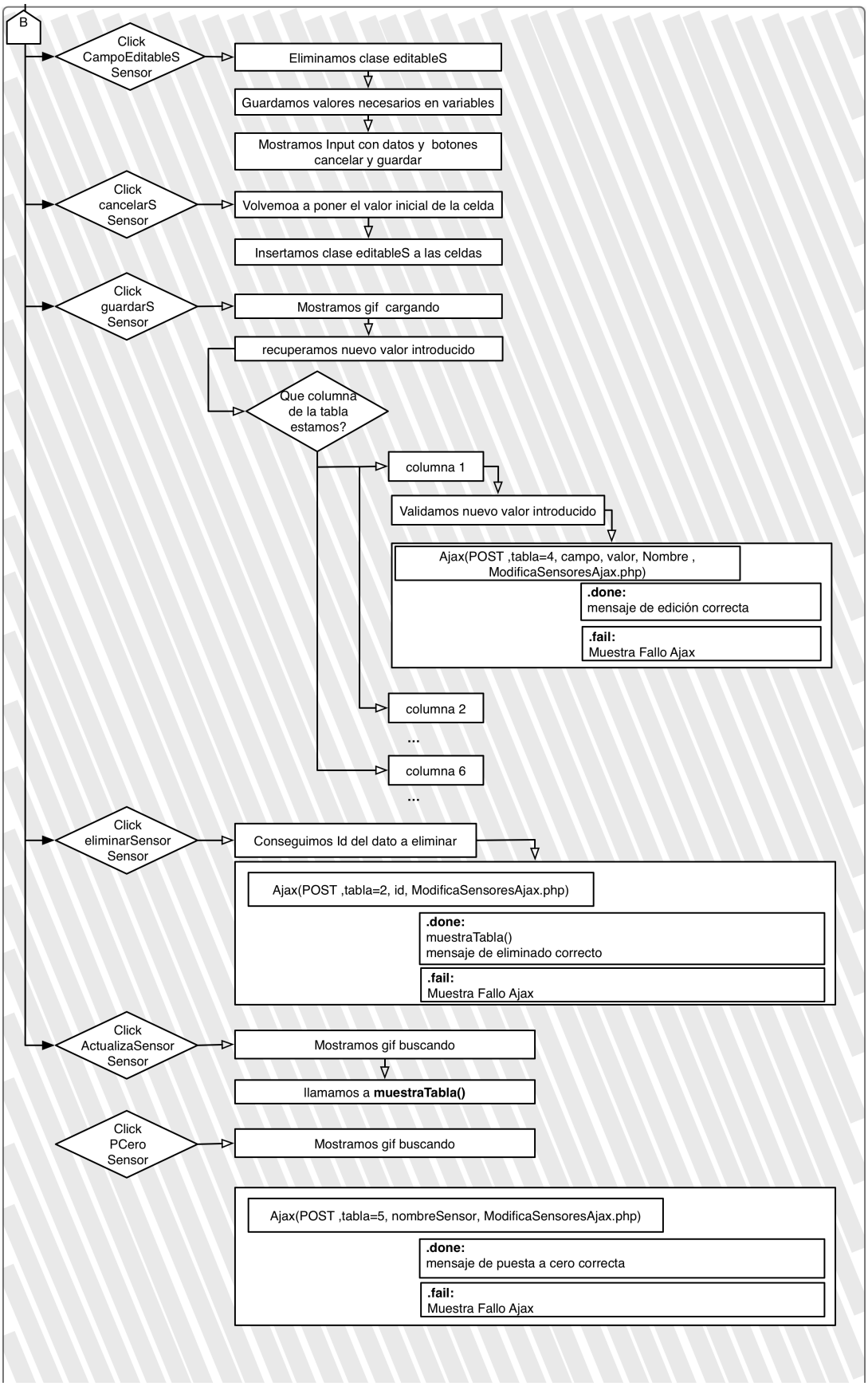


Figura 48 - Flujograma tabla2.js

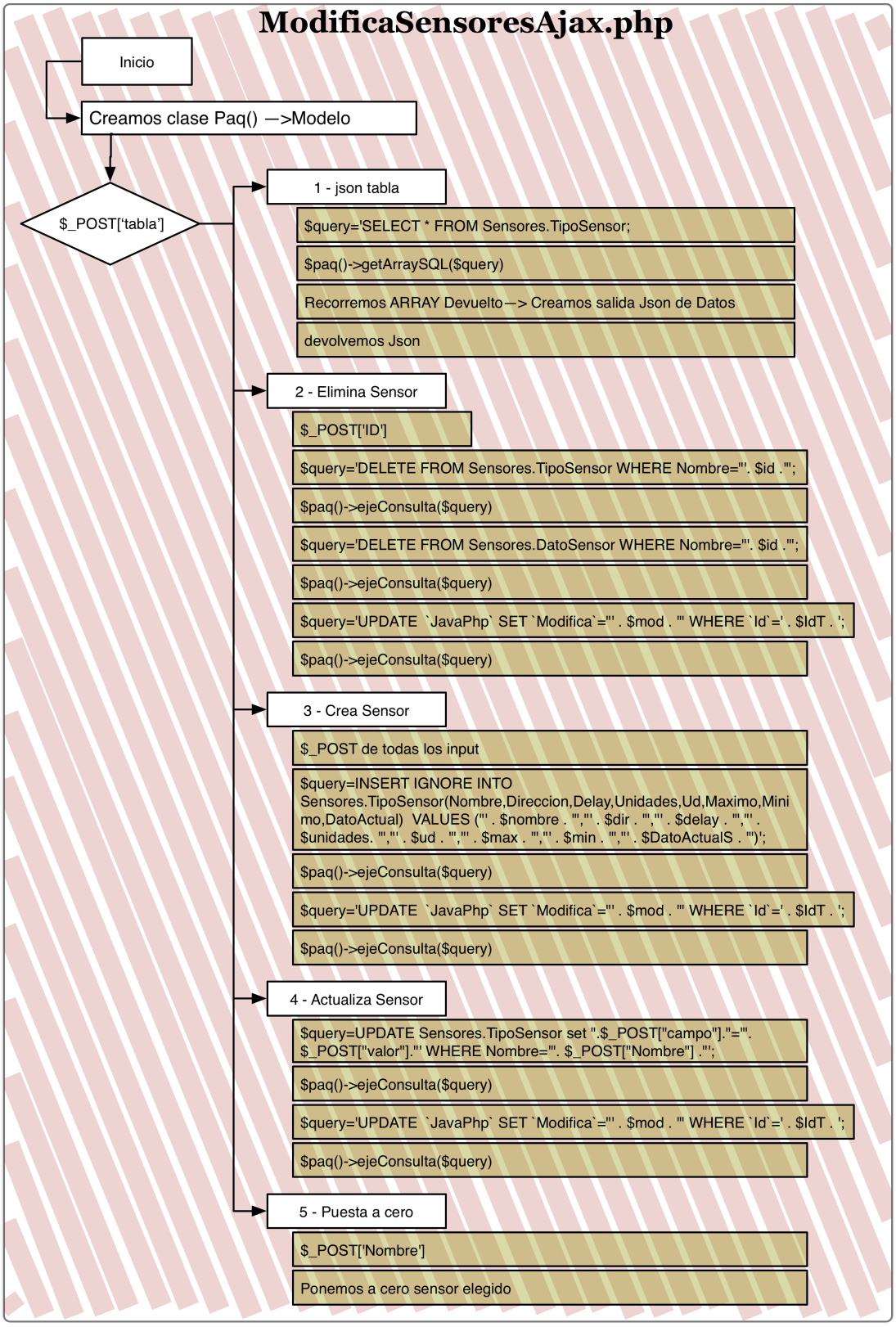


Figura 49 - Flujograma ModificaSensores*Ajax.PHP*

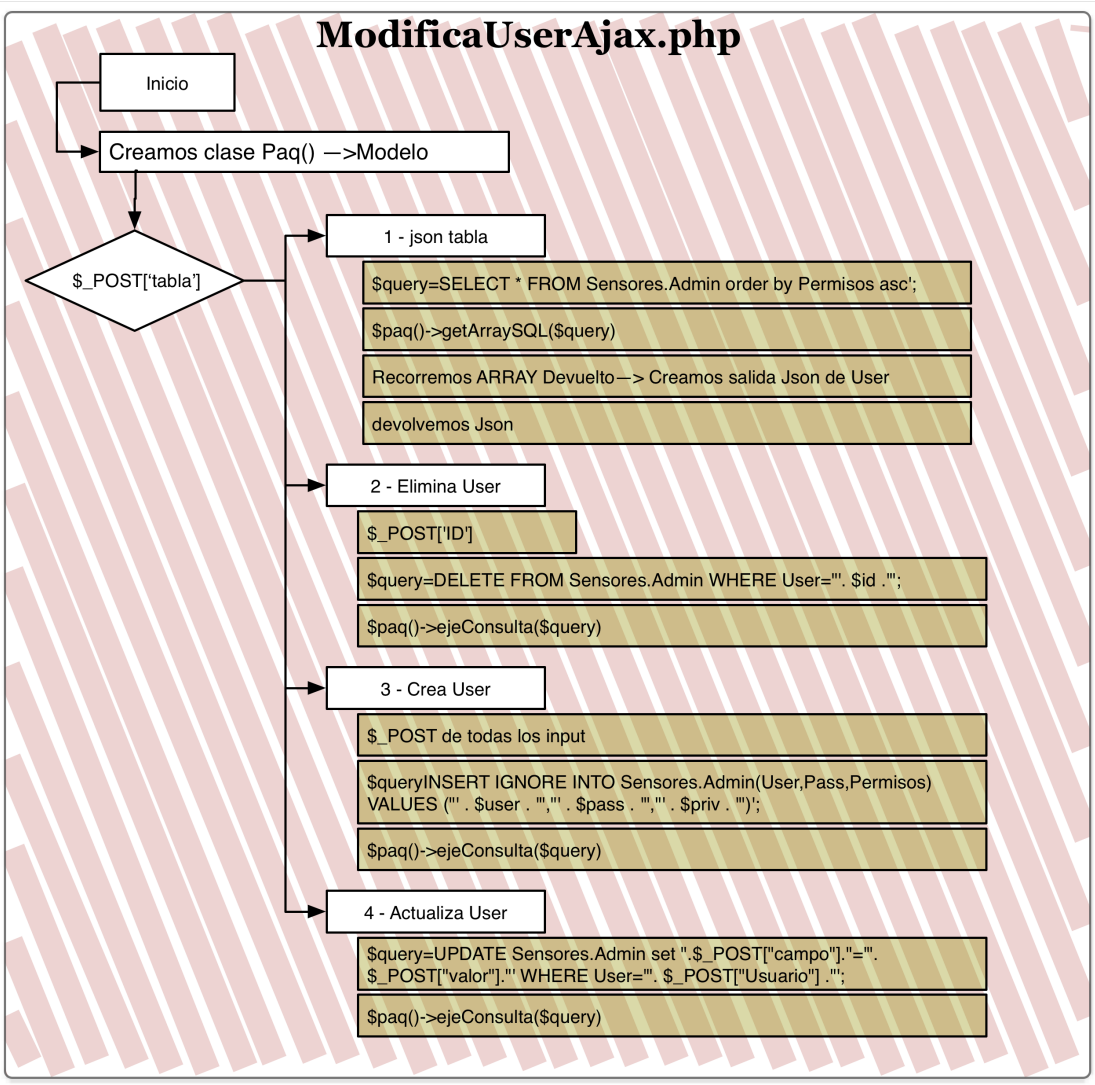


Figura 50 – Flujograma ModificaUser*Ajax.PHP*

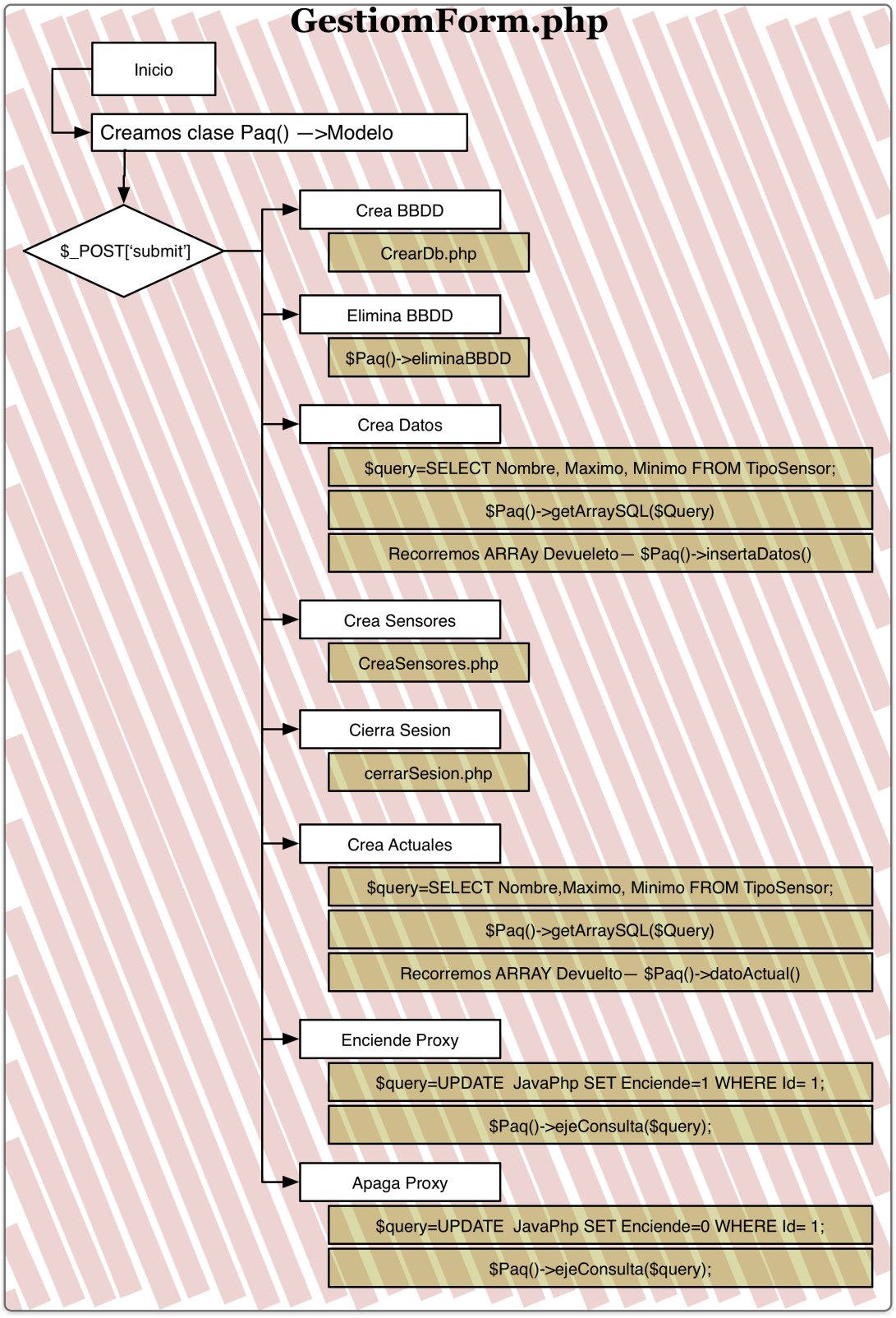
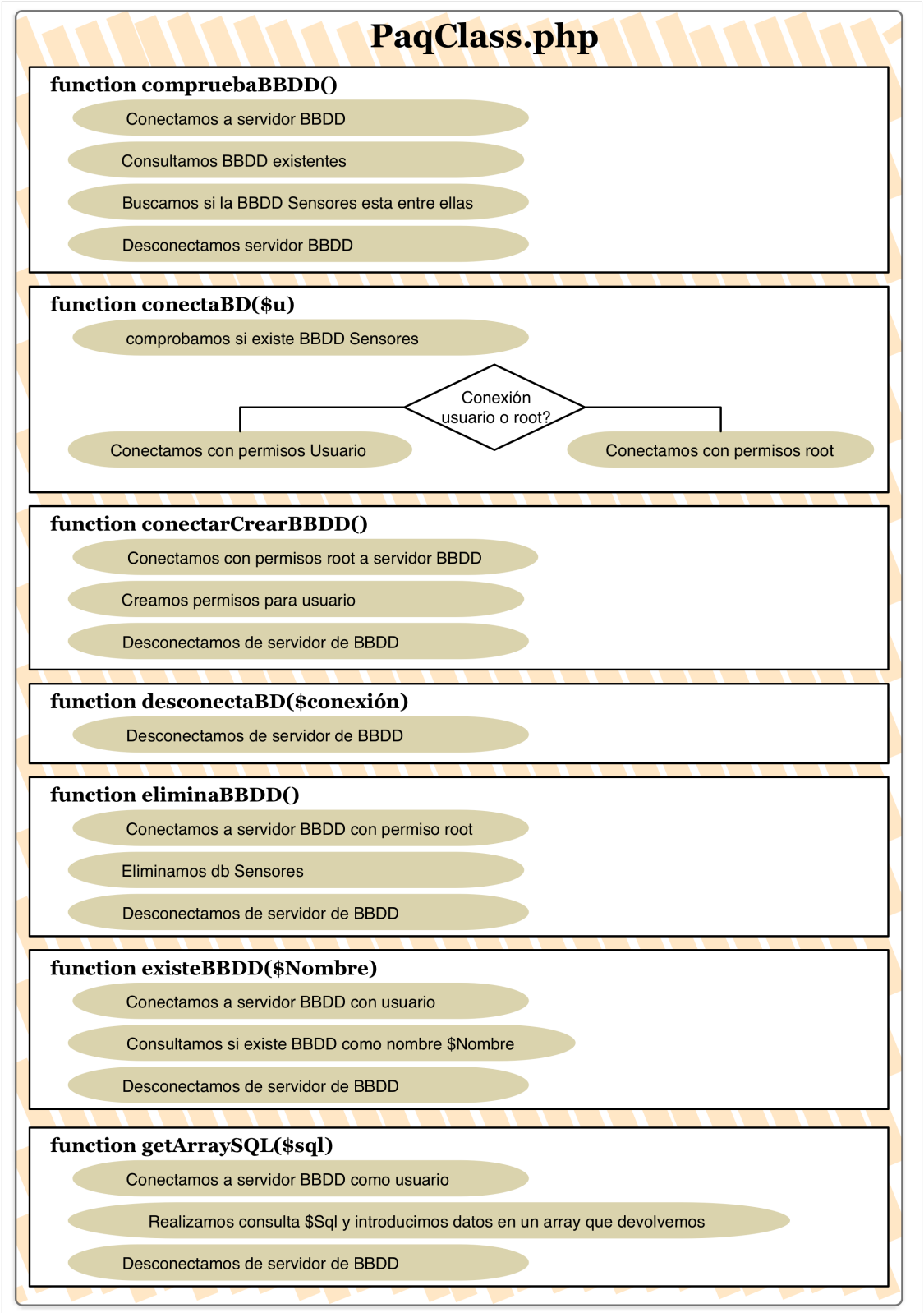


Figura 51 - Flujograma gestiomForm.*PHP*

Modelo

En este flujograma vemos el funcionamiento interno del archivo que realiza las funciones de modelo en la aplicación web, este archivo es común para todos los controladores y vistas.



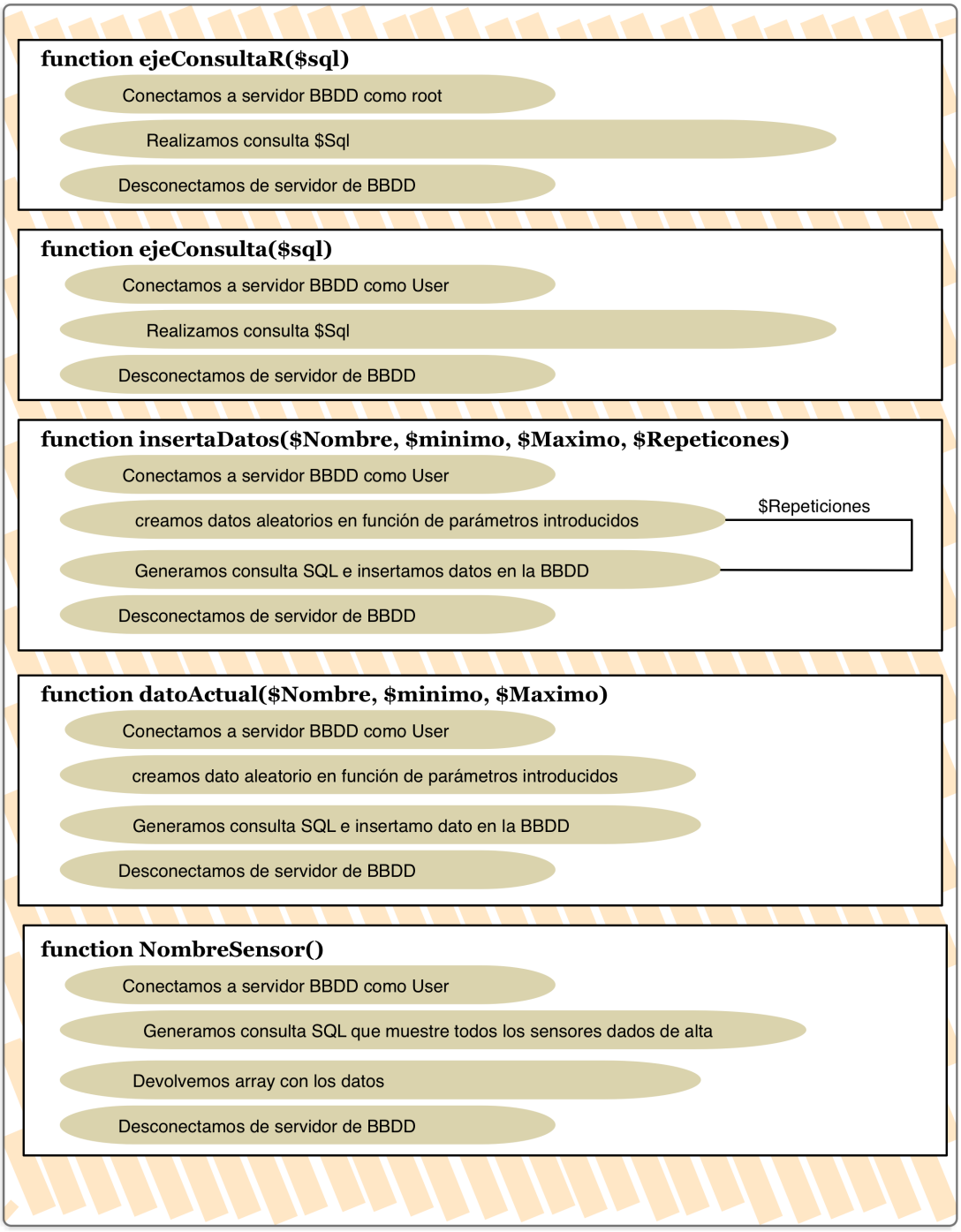


Figura 52 - Flujograma PaqClass.*PHP*

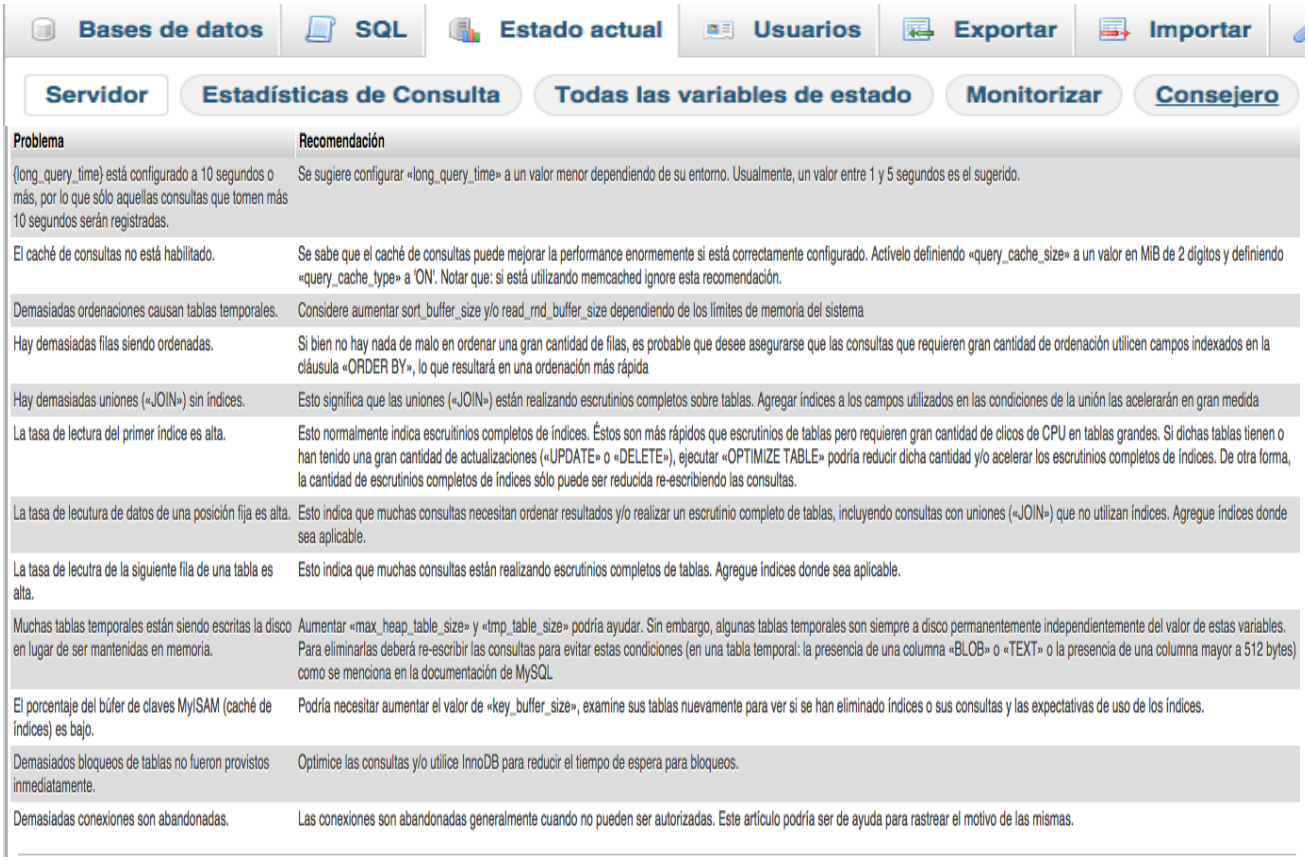
Vemos como el modelo tiene todas las funciones necesarias para interactuar con la base de datos y devolver los datos a los controladores para que traten los datos y se los envíen a las vistas.

El modelo se encarga de conectar y desconectar el servidor de BBDD, haciendo transparente este proceso al programador.

Diseño del servidor BBDD

El servidor BBDD como tal está diseñado. Lo que se pretende en este apartado es afinar el funcionamiento del servidor. Mejorando el rendimiento de las consultas y ampliando o disminuyendo el tamaño de los buffer con los que trabaja el servidor.

En el gestor de *MySQL* llamado *PHP*Myadmin tenemos una herramienta llamada consejero, que nos ayuda a localizar problemas en las transacciones de la base de datos. Nos muestra los errores y la posibles mejoras que podemos ejecutar para mejorar el rendimiento de las consultas a la BBDD. Dentro de la web de administración en la pestaña estado actual, encontramos el botón de la herramienta consejero como vemos en la siguiente figura.



Problema	Recomendación
{long_query_time} está configurado a 10 segundos o más, por lo que sólo aquellas consultas que tomen más 10 segundos serán registradas.	Se sugiere configurar «long_query_time» a un valor menor dependiendo de su entorno. Usualmente, un valor entre 1 y 5 segundos es el sugerido.
El caché de consultas no está habilitado.	Se sabe que el caché de consultas puede mejorar la performance enormemente si está correctamente configurado. Actívelo definiendo «query_cache_size» a un valor en MiB de 2 dígitos y definiendo «query_cache_type» a 'ON'. Notar que: si está utilizando memcached ignore esta recomendación.
Demasiadas ordenaciones causan tablas temporales.	Considere aumentar sort_buffer_size y/o read_rnd_buffer_size dependiendo de los límites de memoria del sistema
Hay demasiadas filas siendo ordenadas.	Si bien no hay nada de malo en ordenar una gran cantidad de filas, es probable que desee asegurarse que las consultas que requieren gran cantidad de ordenación utilicen campos indexados en la cláusula «ORDER BY», lo que resultará en una ordenación más rápida
Hay demasiadas uniones («JOIN») sin índices.	Esto significa que las uniones («JOIN») están realizando escrutinios completos sobre tablas. Agregar índices a los campos utilizados en las condiciones de la unión las acelerarán en gran medida
La tasa de lectura del primer índice es alta.	Esto normalmente indica escrutinios completos de índices. Éstos son más rápidos que escrutinios de tablas pero requieren gran cantidad de clicos de CPU en tablas grandes. Si dichas tablas tienen o han tenido una gran cantidad de actualizaciones («UPDATE» o «DELETE»), ejecutar «OPTIMIZE TABLE» podría reducir dicha cantidad y/o acelerar los escrutinios completos de índices. De otra forma, la cantidad de escrutinios completos de índices sólo puede ser reducida re-escribiendo las consultas.
La tasa de lecutura de datos de una posición fija es alta.	Esto indica que muchas consultas necesitan ordenar resultados y/o realizar un escrutinio completo de tablas, incluyendo consultas con uniones («JOIN») que no utilizan índices. Agregue índices donde sea aplicable.
La tasa de lecutra de la siguiente fila de una tabla es alta.	Esto indica que muchas consultas están realizando escrutinios completos de tablas. Agregue índices donde sea aplicable.
Muchas tablas temporales están siendo escritas la disco en lugar de ser mantenidas en memoria.	Aumentar «max_heap_table_size» y «tmp_table_size» podría ayudar. Sin embargo, algunas tablas temporales son siempre a disco permanentemente independientemente del valor de estas variables. Para eliminarlas deberá re-escribir las consultas para evitar estas condiciones (en una tabla temporal: la presencia de una columna «BLOB» o «TEXT» o la presencia de una columna mayor a 512 bytes) como se menciona en la documentación de MySQL
El porcentaje del búfer de claves MyISAM (caché de índices) es bajo.	Podría necesitar aumentar el valor de «key_buffer_size», examine sus tablas nuevamente para ver si se han eliminado índices o sus consultas y las expectativas de uso de los índices.
Demasiados bloqueos de tablas no fueron provistos inmediatamente.	Optimice las consultas y/o utilice InnoDB para reducir el tiempo de espera para bloqueos.
Demasiadas conexiones son abandonadas.	Las conexiones son abandonadas generalmente cuando no pueden ser autorizadas. Este artículo podría ser de ayuda para rastrear el motivo de las mismas.

Figura 53 - Consejero de *PHP*MyAdmin

Vemos diversos problemas que se mejoran aumentando los buffers o activando las caches del servidor *MySQL*

Activar caché

Una de las acciones que se va a tomar es activar la cache de consultas. Primero entramos en la consola *MySQL*:

```
iMac-de-David:~ davsel$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 666160060
Server version: 5.7.4-m14 MySQL Community Server (GPL)

Copyright (c) 2000, 2014, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

Figura 54 - Entrar en consola *MySQL*

El consejero recomienda configurar el tiempo de consulta entre 1 y 5 segundos. Este es el tiempo por el cual el servidor da por finalizada la consulta si no se han obtenido datos.

```
mysql> set long_query_time=5;
Query OK, 0 rows affected (0,00 sec)
```

Figura 55 - Activar long_query_time entre 1 y 5

Ahora comprobamos el estado de la cache mediante esta consulta:

```
mysql> SHOW VARIABLES LIKE '%query_cache%';
+------------------------------+---------+
| Variable_name                | Value   |
+------------------------------+---------+
| have_query_cache             | YES     |
| query_cache_limit            | 1048576 |
| query_cache_min_res_unit     | 4096    |
| query_cache_size             | 1048576 |
| query_cache_type             | OFF     |
| query_cache_wlock_invalidate | OFF     |
+------------------------------+---------+
6 rows in set (0,00 sec)
```

Figura 56 - Comprobamos qué cache está desactivada y su pequeño tamaño

Para modificar las variables de *MySQL* en Mac OS Lion, necesitamos un fichero que no existe por defecto /etc/my.cnf. *MySQL* utiliza la configuración por defecto que trae instalada Mac.

La solución es añadir un enlace simbólico que apunte a la ubicación del fichero de configuración my-default.cnf.

Para crear el enlace simbólico se utiliza el siguiente comando:

```
sudo ln -s /usr/local/MySQL/support-files/my-huge.cnf /etc/my.cnf
```

Modificamos el archivo my.cnf introduciendo las siguientes líneas:

```
query-cache-type = 1
query-cache-size = 20M
```

Reiniciamos servidor *MySQL* y ya tenemos resuelto el problema de la cache que nos decía *PHP*MyAdmin.

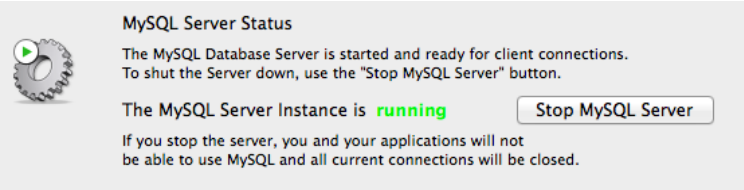


Figura 57 - Reinicio *MySQL*

```
mysql> SHOW VARIABLES LIKE '%query_cache%';
No connection. Trying to reconnect...
Connection id:    3094
Current database: *** NONE ***

+------------------------------+----------+
| Variable_name                | Value    |
+------------------------------+----------+
| have_query_cache             | YES      |
| query_cache_limit            | 1048576  |
| query_cache_min_res_unit     | 4096     |
| query_cache_size             | 20971520 |
| query_cache_type             | ON       |
| query_cache_wlock_invalidate | OFF      |
+------------------------------+----------+
6 rows in set (0,01 sec)
```

Figura 58 - Comprobación de cache activada y si tamaño aumentado

Buffer read_rdn y sort_buffer

Ahora se va a solucionar el problema con el buffer read_rnd y sort_buffer. Estos buffers son los encargados de soportar las tablas temporales creadas por las consultas. El gestor de bases de datos nos recomienda subir este valor para no ralentizar la visión de los datos en las aplicaciones cliente.

Con esta consulta vemos el estado de las variables a tratar:

```
MySQL> SHOW VARIABLES LIKE '%buffer%';
```

El estado en el que nos encontramos los buffers es el siguiente:

```
| read_rnd_buffer_size            | 2097152     |
| sort_buffer_size                | 2097152     |
```

Figura 59 – Tamaño inicial de buffer *MySQL*

Como vemos tiene un valor pequeño, solo 2 megas, vamos a modificar este valor para mejorar la interacción con este buffer.

Introducimos inicialización de variables en my.cnf:

read_rnd_buffer_size= 40M
sort_buffer_size = 40M

Reiniciamos servidor *MySQL* y vemos estado actual:

```
| read_rnd_buffer_size            | 41943040    |
| sort_buffer_size                | 41943040    |
```

Figura 60 - Tamaño modificado de buffer *MySQL*

Indexado de tablas

Vemos en el consejero que las consultas de ordenación de columnas se utilizan masivamente, cuanto más tiempo tenemos activado el *Proxy* más y más datos se generan el la base de datos. Existe una solución y es indexar los campos que más veces se ordenan. De esta manera se realizaran las operaciones más rápido.

Esta es la consulta más costosa que tenemos en nuestro diseño:

```
SELECT Id,Dato,Fecha_hora FROM Sensores.DatoSensor WHERE
Nombre='Direccion_Viento' ORDER BY Fecha_hora ASC;,
```

Es la que vamos a analizar.

Sin ninguna modificación, se realiza la consulta desde *PHP*myAdmin y ésta tardaba 0.121seg en producirse y mostrar los datos.

Indexamos las dos columnas más usadas y que con más datos trabajan en nuestra web. Son la columna Fecha_Hora, que se ordena en cada consulta y la columna nombre que se utiliza mucho en clausulas SQL tipo WHERE.

Para realizarlo ejecutamos las siguientes modificaciones de la BBDD desde *PHP*MyAdmin:

```
ALTER TABLE DatoSensores ADD INDEX (Fecha_Hora);
ALTER TABLE DatoSensores ADD INDEX (Nombre)
```

Código 5 - Modificación de índices en BBDD

Una vez indexado, volvemos a realizar la consulta en las mismas condiciones que anteriormente. El tiempo de consulta es 0,0502seg, como observamos se realiza la consulta en la mitad de tiempo que la consulta sin indexar campo. Un mejora considerable.

¿Qué está pasando?, ¿Por qué mejora el tiempo?

La base de datos, para ordenar todos los registros de la columna Fecha_Hora, tiene que consultar fila por fila. No tienen ningún orden.

Sin ningún orden en nuestros datos, *MySQL* debe leer todos los registros de la tabla "DatosSensor" y efectuar una comparación entre el campo "nombre" y la variable $Nombre para encontrar coincidencias. A medida que esta base de datos sufra modificaciones, como un incremento en el número de registros, dicha consulta irá requiriendo un mayor el esfuerzo de la CPU y el uso de memoria del servidor para poder ejecutarse.

Si tuviéramos una guía telefónica a mano localizaríamos fácilmente a cualquiera con nombre, por ejemplo "Direccion" iríamos al principio de la guía, a la letra "D". El método en sí está dado en función de cómo están ordenados los

datos y en el conocimiento de los mismos. En otras palabras, localizamos rápidamente a "Direccion" porque está ordenado por Nombre y porque conocemos el abecedario.

Viendo estadísticas en *PHP*MyAdmin, nos damos cuenta de la mejora realizada es muy notable. Antes de indexar la velocidad de ordenación del parámetro ORDER By Fecha_hora era de 2,2 filas por segundo. Indexando la Columna Fecha_Hora, se ha mejorado a 16,1 filas por segundo ordenadas.

Configuración de usuario davsel

Por seguridad el usuario que utilizamos para acceder a nuestra BBDD desde la aplicación web debe de estar restringido a usar solo la base de datos Sensores. De esta forma un ataque externo con ese usuario solo modificaría valores de esa base de datos, pero no podría acceder a las bases de datos de configuración del servidor, a las cuales se accede como a cualquier otra base de datos creada en *MySQL*. Si se accede a estas bases de datos de configuración, pueden ser capazes de cambiar la contraseña y modificar configuraciones que dejarían sin servicio la aplicación.

Para cambiar privilegios de usuario, entramos en *PHP*myAdmin. Pulsamos la pestaña Usuarios, elegimos usuario Davsel y pulsamos editar privilegios. Una vez dentro desmarcamos todos los ticks y damos a continuar. Después solo daremos acceso a este usuario a una cierta db.

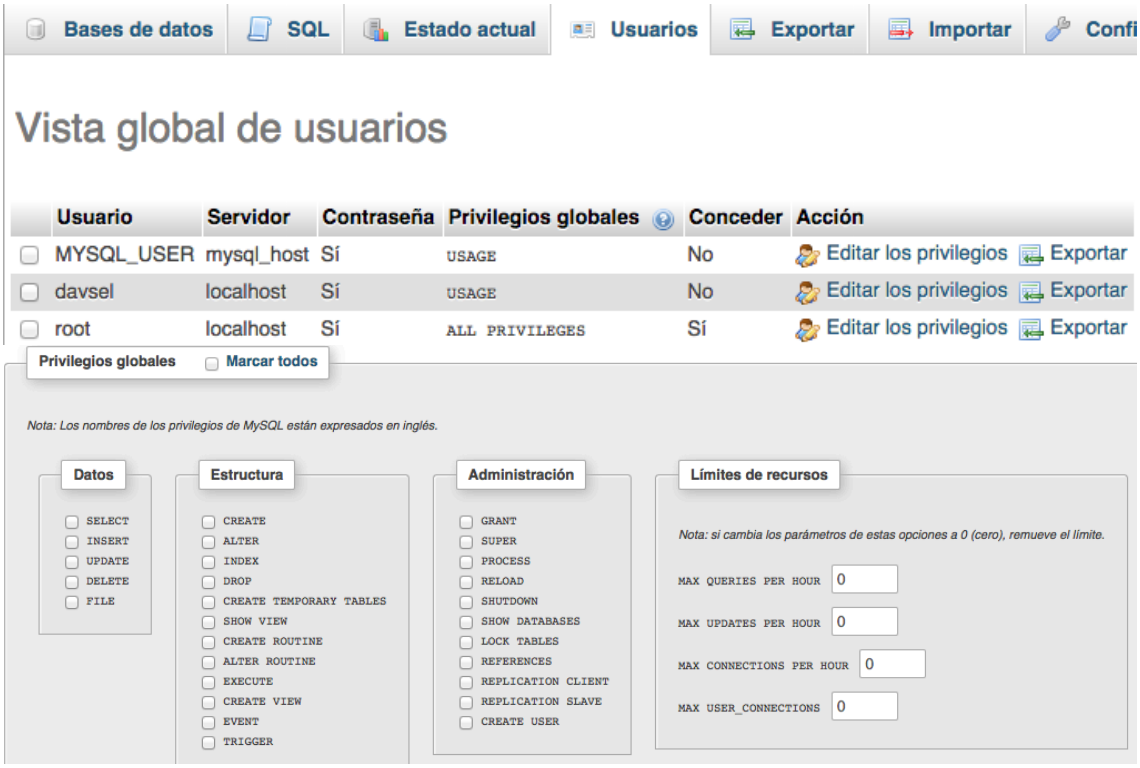


Figura 61 - Editar privilegio de usuario *PHP*MyAdmin.

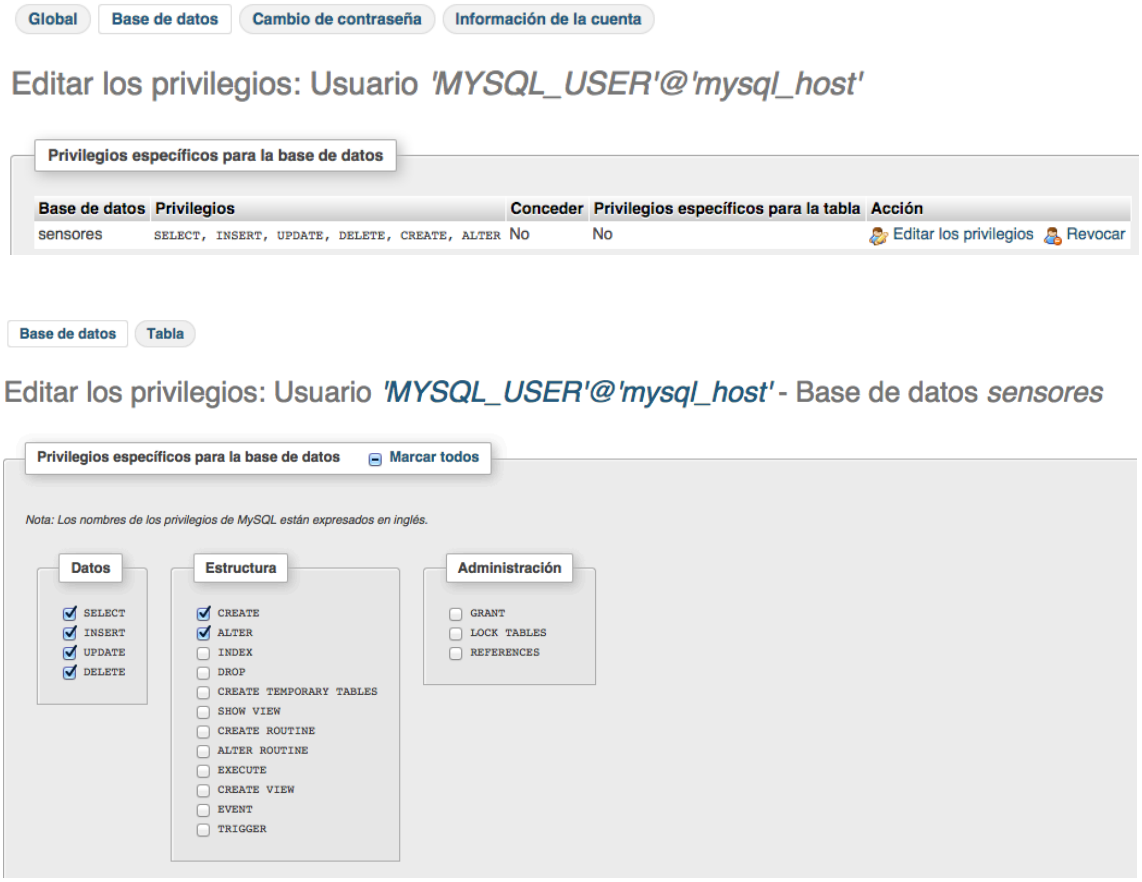


Figura 62 - Añadir privilegio de BBDD a un usuario

Pulsamos sobre el botón bases de datos, donde se configuran los privilegios de un usuario a una cierta base de datos. Pinchamos sobre la base de datos Sensores y pulsamos editar privilegios. Marcamos los ticks que vemos en pantalla y pulsamos continuar para que se guarde la configuración. Nuestro usuario ya puede usar la base de datos Sensores, pero no otra.

Diseño del *Proxy*.

El *Proxy* es un servicio que debe estar ejecutándose en un servidor que tenga acceso a la base de datos Sensores. Este programa lo que pretende es hacer de intermediario entre los sensores y la aplicación web. El *Proxy* lee mediante el protocolo *CoAP* los datos de los sensores según intervalos configurados en la aplicación web. Estos datos son guardados en la base de datos. Los datos son leídos por la aplicación web desde la base de datos.

Desde la aplicación web, como veíamos, se configuran los datos de los sensores, su dirección *CoAP*, el intervalo de lectura, etc. El *Proxy* lee esta información de la base de datos para saber qué leer y cada cuánto tiempo. Para que el *Proxy* esté activo y cuando se modifique un dato la aplicación web, este empiece a funcionar con este cambio, la web cambia el estado del campo Modifica. El *Proxy* constantemente está leyendo este campo para actuar en cuanto se modifique y así ver cómo cambia la actuación del sistema.

Desde la web podemos apagar y encender el *Proxy*, es decir que esté leyendo datos o esté parado. Cuando se pulsa el botón Apagar o Encender *Proxy* de la aplicación web, esta cambia de estado el campo Enciende de la BBDD. El *Proxy*, al igual que el campo modifica, está leyendo constantemente la BBDD para ver el estado y actuar rápidamente parando la captura de datos.

Para conseguir tales efectos se creará un bucle infinito y dentro de este otro bucle que se parará o iniciará dependiendo del estado de un botón en la aplicación web. Dentro de este último bucle se crearán tantos hilos como sensores estén configurados en la base de datos. Cada hilo tendrá su intervalo de acceso a los sensores y escribirá en la BBDD según vaya recogiendo las medidas.

Ejc*CoAP*_h.java es el código encargado de crear la estructura de los hilos. Muestra lo que tienen que hacer los hilos cuando se ejecuten. Básicamente es un bucle infinito que realiza una llamada *CoAP* en cada iteración. El dato recogido es guardado en la BBDD, después hace una espera pasiva del tiempo que se configura en la aplicación web.

Como los servidores *CoAP* reales no están activos aún, se genera un número aleatorio respetando los máximos y mínimos programados en la web, con un error para que salgan alarmas. Este dato es enviado mediante POST a un servidor *CoAP* de prueba, el dato se recoge mediante el método GET de *CoAP* en la misma dirección Uri. De este modo probamos que la conexión *CoAP* funciona como esperábamos.

El hilo se para mediante la función rompeHilo(), esta función para el bucle de ejecución del hilo, espera 0,1 segundos para que se terminen las operaciones en curso y se destruye el hilo. Cada vez que se cambia el campo modifica, se destruyen todos los hilos y se vuelven a crear con los datos modificados desde la web.

Vemos el flujograma que genera las funciones de los hilos, realizado mediante el código Ejc*CoAP*_h.java:

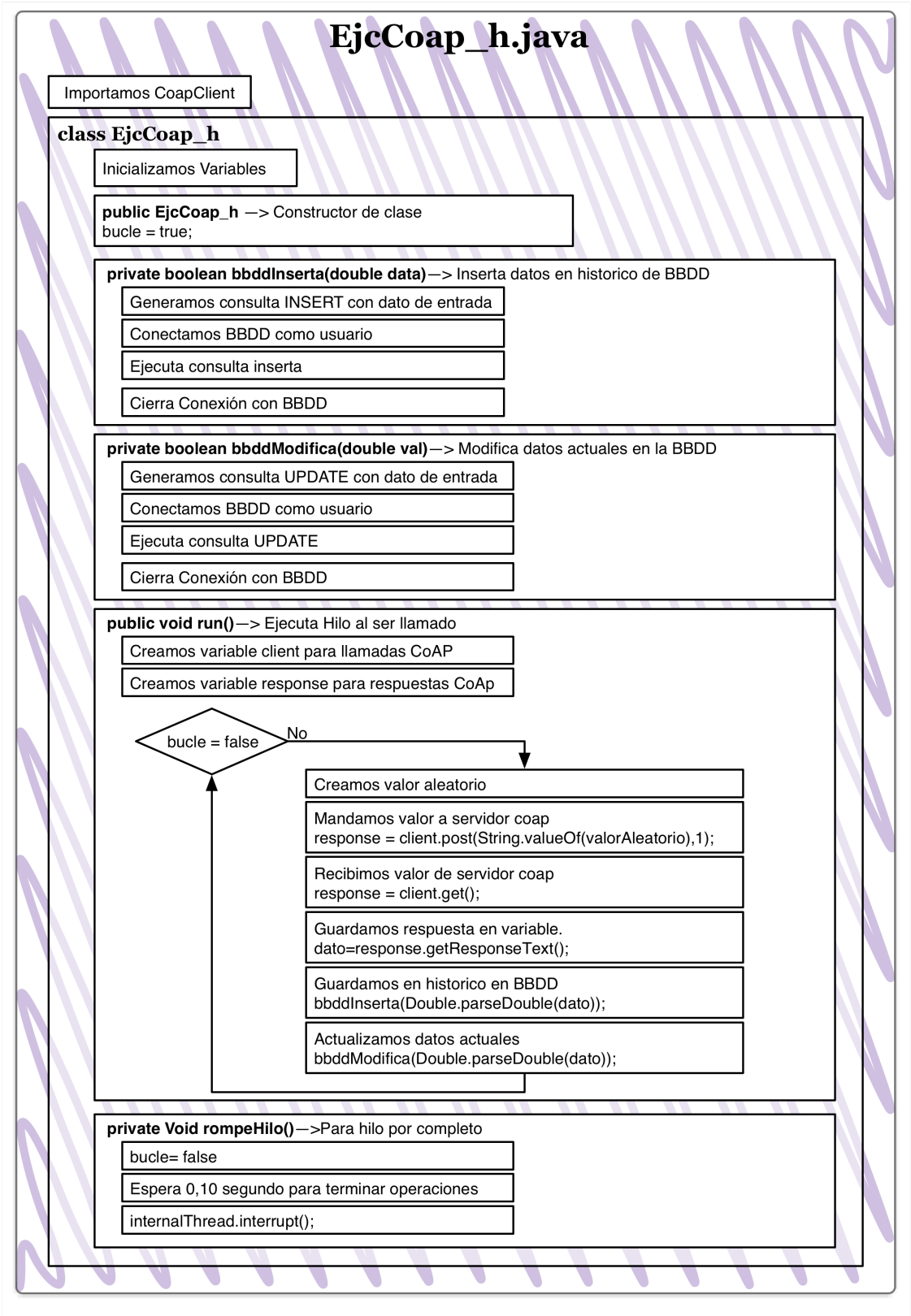


Figura 63 - Flujograma Ejc*CoAP*_H

La clase basecon hace la misma función que el modelo de la aplicación web. Abstrae al programador de las operaciones con la base de datos. Contiene todas las funciones necesarias para conectar, realizar consultas y desconectar con la base de datos.

Tiene dos modos de conexión dependiendo del parámetro que le pasemos a la función conectaBbdd(). Una en modo usuario para realizar las consultas básicas y otra que se conecta como root para realizar cambios en la configuración de la BBDD.

ConsultaBBDD(*SQL*) es la función que pasando una consulta como parámetro nos devuelve un resulSet con los datos que tiene la BBDD.

ActualizaBbdd(*SQL*) hace la misma función que consultaBbdd, lo único que no devuelve ningún valor, está pensada para la ejecución de consultas tipo update o insert.

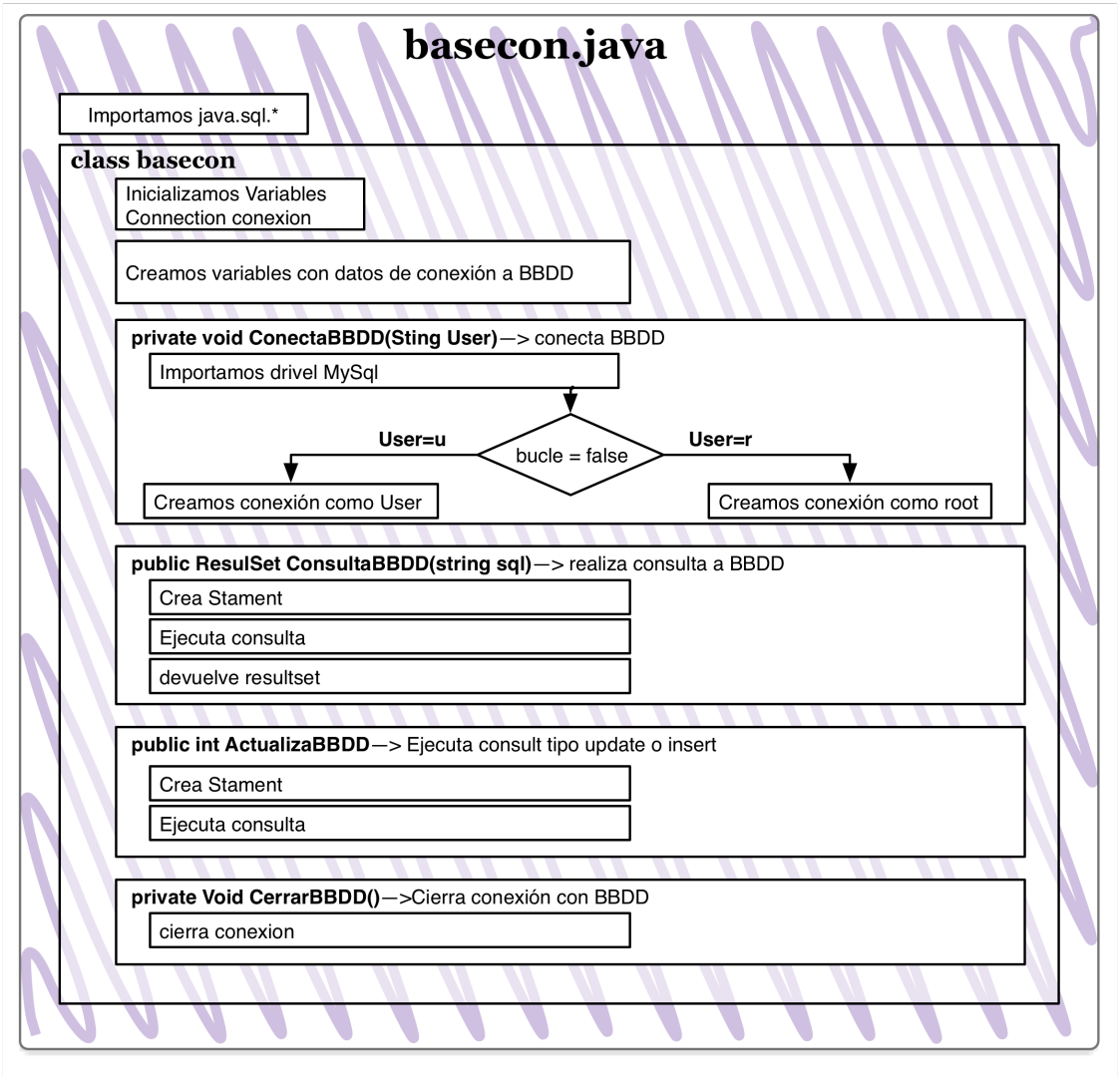


Figura 64 - Flujograma basecon.java

ProxyCoAP.java es el programa principal del *Proxy*. El código ejecuta un bucle infinito y dentro de este otro bucle que compueba constantemente el estado de los campos modifica y enciende.

En el arranque se generan tantos hilos como sensores hay configurados en la BBDD. Esto hace que se empiecen a ejecutar todas las lecturas de manera concurrente, cada una con su tiempo de espera entre lecturas. Cada hilo ejecuta la clase Ejc*CoAP*_h.

Cuando se cambia el estado de modifica, se eliminan los hilos y se vuelven a crear con los datos nuevos. Este cambio se realiza cuando un dato es modificado o es creado un sensor nuevo en la administración de sensores de la aplicación web. Una vez se han iniciado de nuevo todos los hilos, *ProxyCoAP* vuelve a cambiar el valor del campo modifica a 0, quedado así hasta que se modifique de nuevo desde la web.

Si se pulsa el botón Apaga *Proxy* en la web, se detecta en el bucle interno y se sale de él. Se eliminan todos los hilos y se queda iterando en el bucle externo comprobando el estado del campo enciende. Cuando vuelva a cambiar de estado en campo enciende, entra de nuevo en el bucle interno inicializando todo el proceso de nuevo.

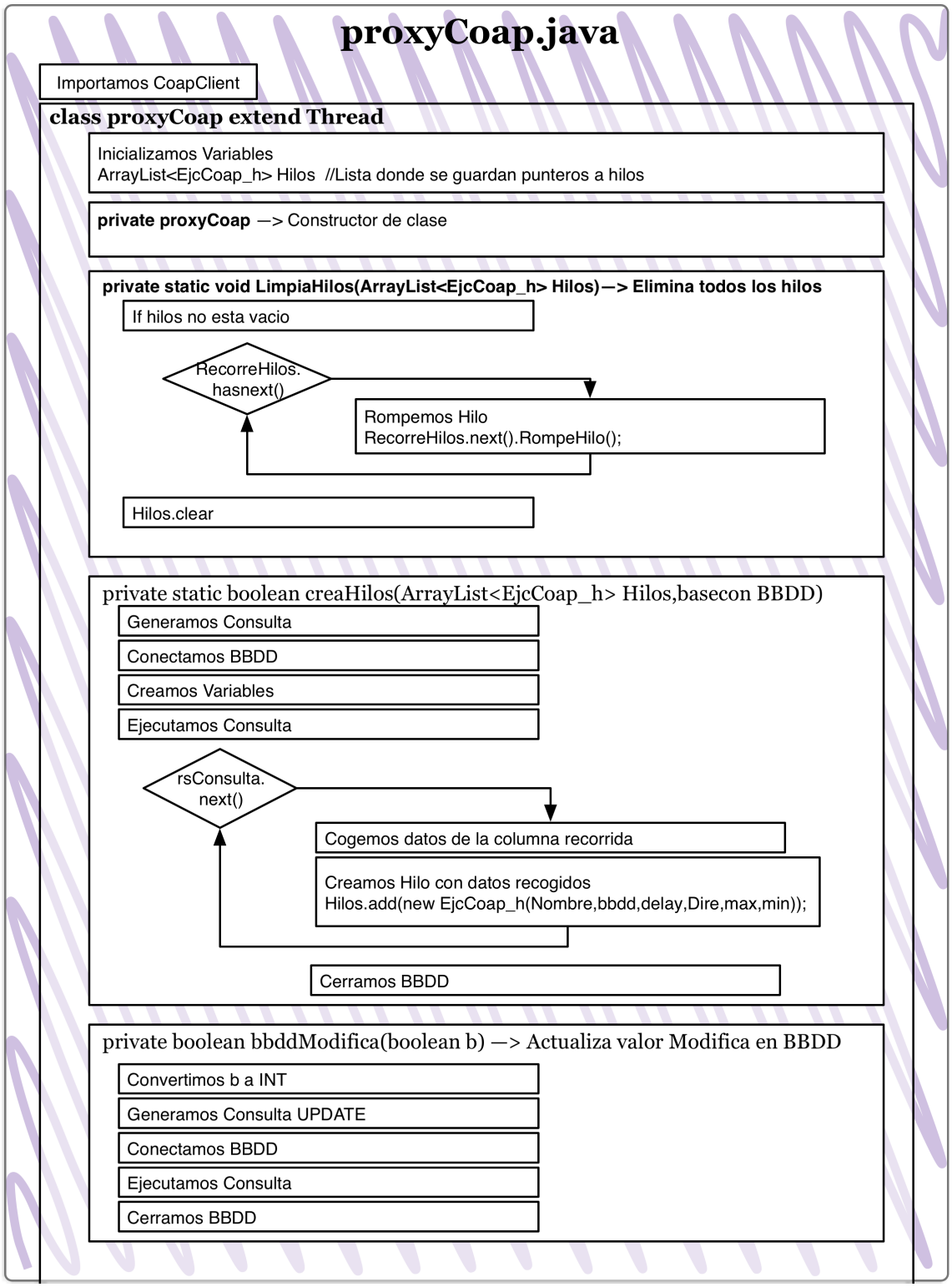
Un hilo es un subprograma que se ejecuta de manera concurrente al programa principal. Así se puede estar ejecutando unos procesos en el programa principal y muchos otros en hilos. En nuestro caso lo utilizamos para conseguir realizar lecturas con diferentes intervalos de tiempo. Mientras que en el programa principal vigilamos el estado de las variable modifica y enciende.

Cuando creamos un hilo se crea un puntero para tener localizado este subprograma y poder acceder a él cuando sea necesario. En este caso se necesita un sistema dinámico de almacenamiento de punteros. Para ello se utiliza una lista dinámica. Es una estructura de datos en la que podemos guardar cualquier tipo de dato y añadir o quitar datos según se necesite. Estos datos se guardan en memoria RAM, como cualquier variable. Los punteros de los hilos se guardan en esta estructura de datos, para poder llamarles con el fin de destruir el hilo cuando sea necesario.

Al crear los hilos vamos llamando al constructor de clase Ejc*CoAP*_h con los datos necesarios sacados de la base de datos y guardando el puntero de memoria del hilo en la lista dinámica.

Para destruir todos los hilos recorremos la lista dinámica, ejecutando el puntero.rompeHilo(), internamente se destruye el hilo y desaparece de memoria, luego vaciamos la lista dinámica y la dejamos preparada para una nueva creación de hilos.

Vemos el flujograma para entender en conjunto lo explicado:


proxyCoap.java
Importamos CoapClient
class proxyCoap extend Thread
Inicializamos Variables
ArrayList<EjcCoap_h> Hilos //Lista donde se guardan punteros a hilos
private proxyCoap —> Constructor de clase
private static void LimpiaHilos(ArrayList<EjcCoap_h> Hilos)—> Elimina todos los hilos
If hilos no esta vacio
RecorreHilos.hasnext()
Rompemos Hilo
RecorreHilos.next().RompeHilo();
Hilos.clear
private static boolean creaHilos(ArrayList<EjcCoap_h> Hilos,basecon BBDD)
Generamos Consulta
Conectamos BBDD
Creamos Variables
Ejecutamos Consulta
rsConsulta.next()
Cogemos datos de la columna recorrida
Creamos Hilo con datos recogidos
Hilos.add(new EjcCoap_h(Nombre,bbdd,delay,Dire,max,min));
Cerramos BBDD
private boolean bbddModifica(boolean b) —> Actualiza valor Modifica en BBDD
Convertimos b a INT
Generamos Consulta UPDATE
Conectamos BBDD
Ejecutamos Consulta
Cerramos BBDD

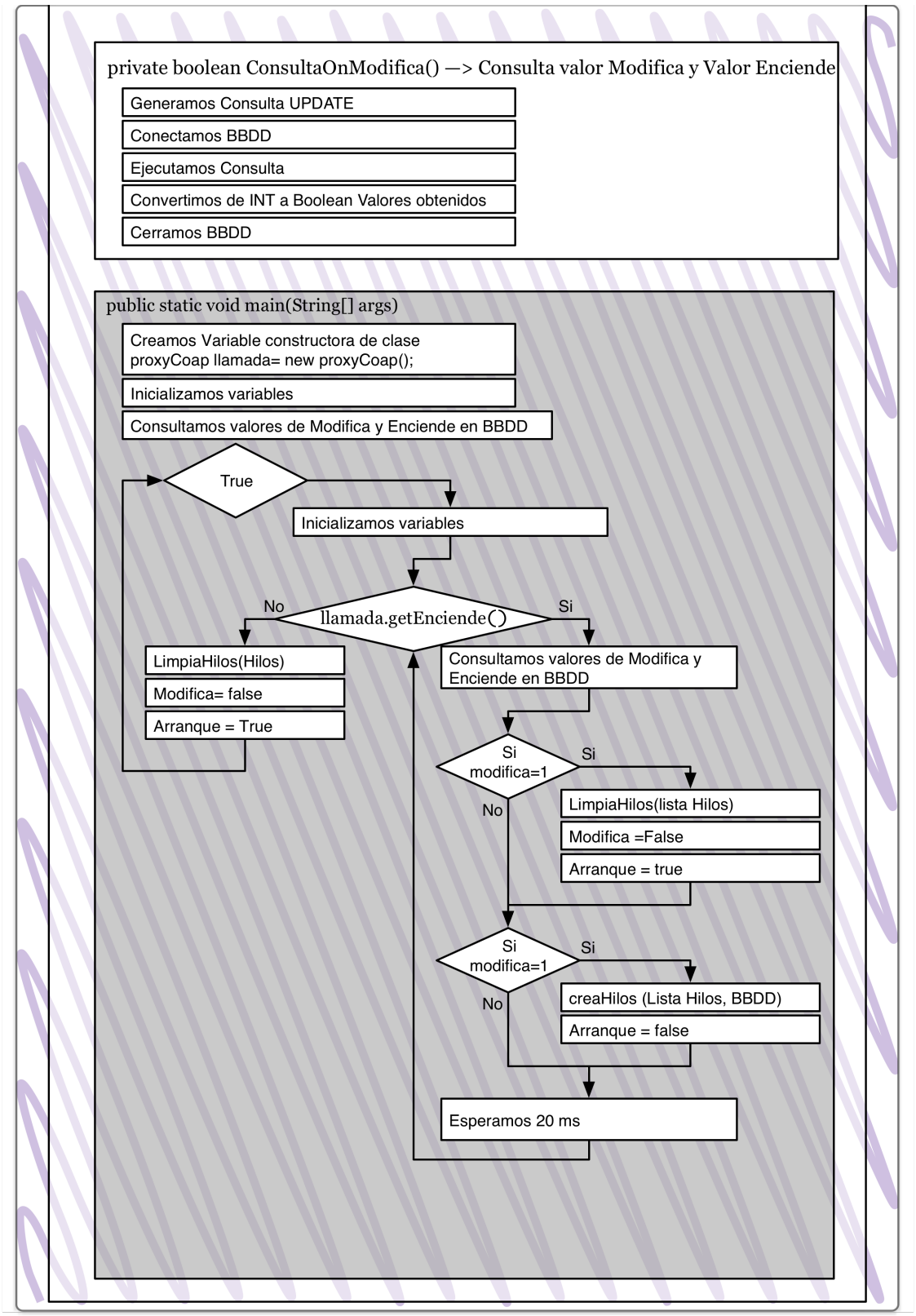


Figura 65 - *ProxyCoAP*.java

Para la creación del código se utiliza el IDE eclipse. La ejecución del comando *CoAP* se hace a través de la librería californium. Una vez seguidos los pasos explicados en el apartado instalación de *Proxy* y creados los archivos .java explicados anteriormente, tenemos que generar un archivo .jar del proyecto eclipse. De esta manera podemos ejecutar el *Proxy* en cualquier sistema operativo que tenga instalado java. A continuación, se explica cómo realizar esta conversión:

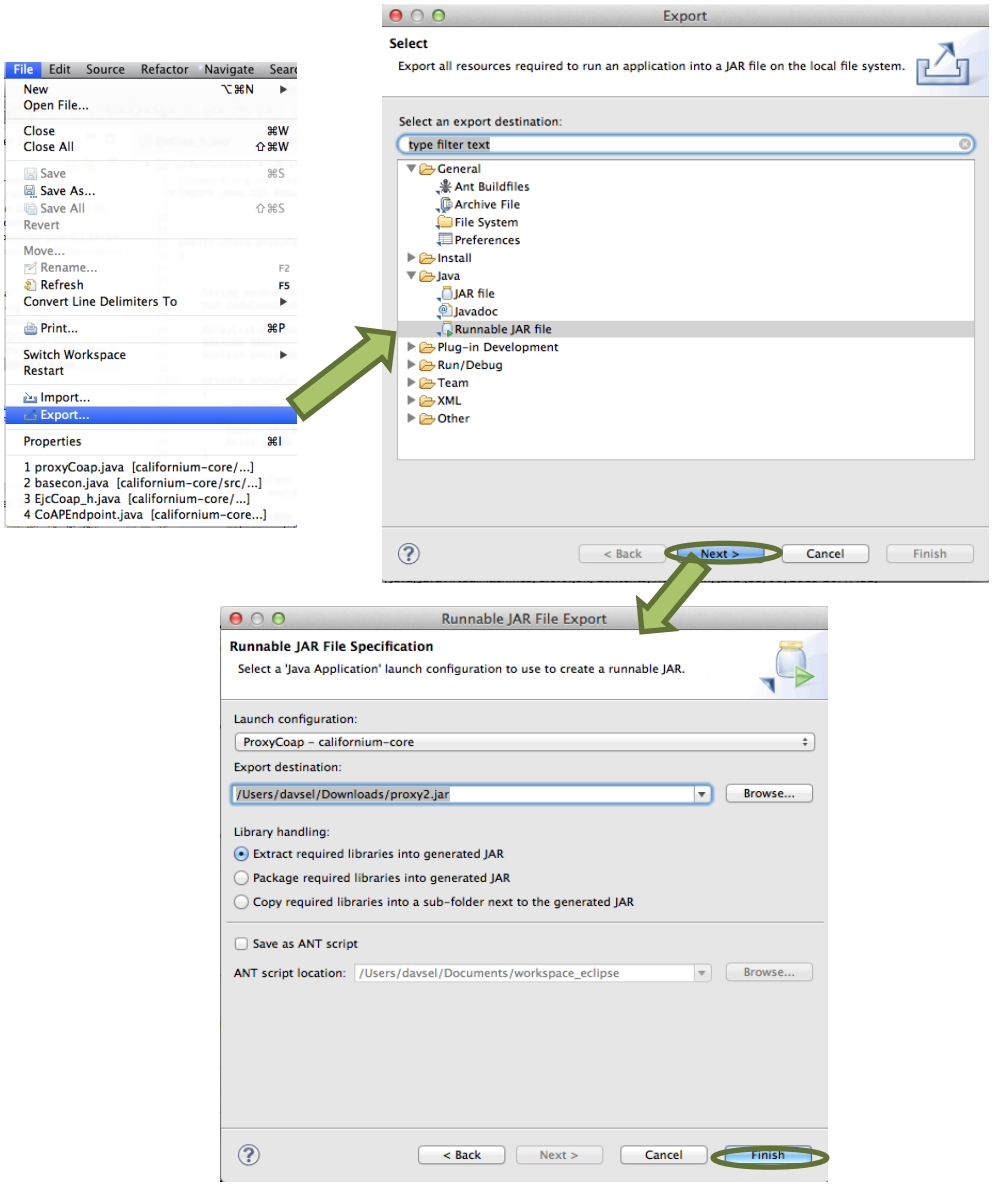


Figura 66 - Exportar JAR de *Proxy*

Una vez creado, para ejecutar el *Proxy* desde consola se realiza del siguiente modo:

```
Mac: User$ java -jar Proxy2.jar
```

Diseño de la aplicación *Android*.

La aplicación *Android*, como se explicó en el apartado de estudio de aplicación *Android*, se va a realizar a partir de una aplicación web para después convertirla con el programa Intel XDK a APK.

El diseño de la aplicación web para la apk sigue el patrón Modelo-Vista-Controlador. La característica fundamental es que la vista se convierte en apk mientras el controlador y el modelo están en el servidor. Cuando la apk necesite datos llamará mediante *Ajax* al controlador. Este con los datos que le provea el modelo, devolverá a la apk codificados en *JSON* los datos. La apk decodifica y muestra los datos en la aplicación *Android*. A continuación, vemos el esquema MVC:

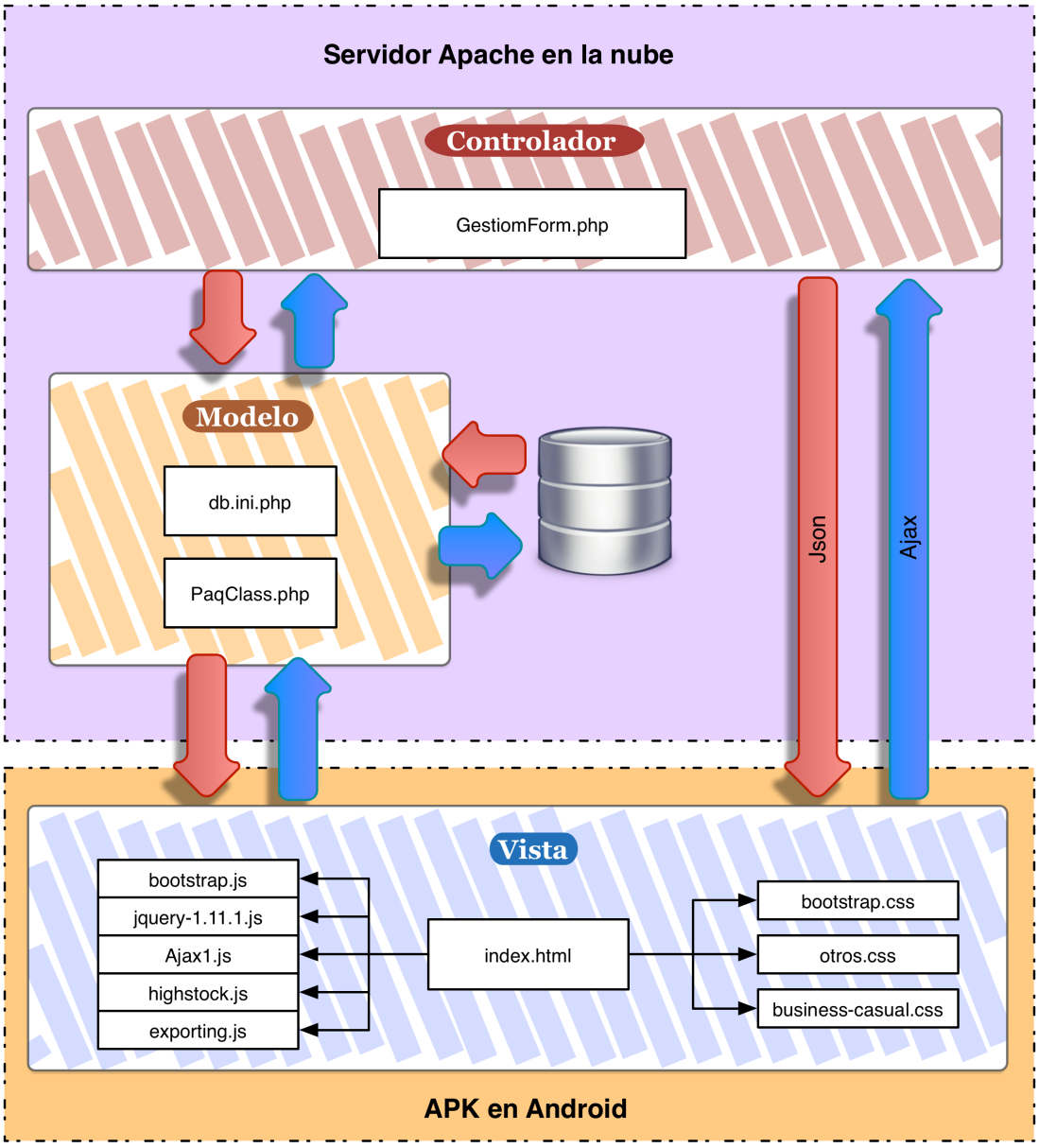


Figura 67 - Esquema MVC APK *Android*

Parte Servidor

La parte alojada en el servidor sirve de interlocutor entre la apk y el modelo. GestiomForm.*PHP* tiene tres funcionalidades, una es mostrar la lista de sensores existente en la BBDD, la segunda mostrar los datos actuales de los sensores dados de alta y la tercera devolver todos los datos que tenga un sensor.

A continuación vemos el flujograma de funcionamiento del controlador gestiomForm.*php*.

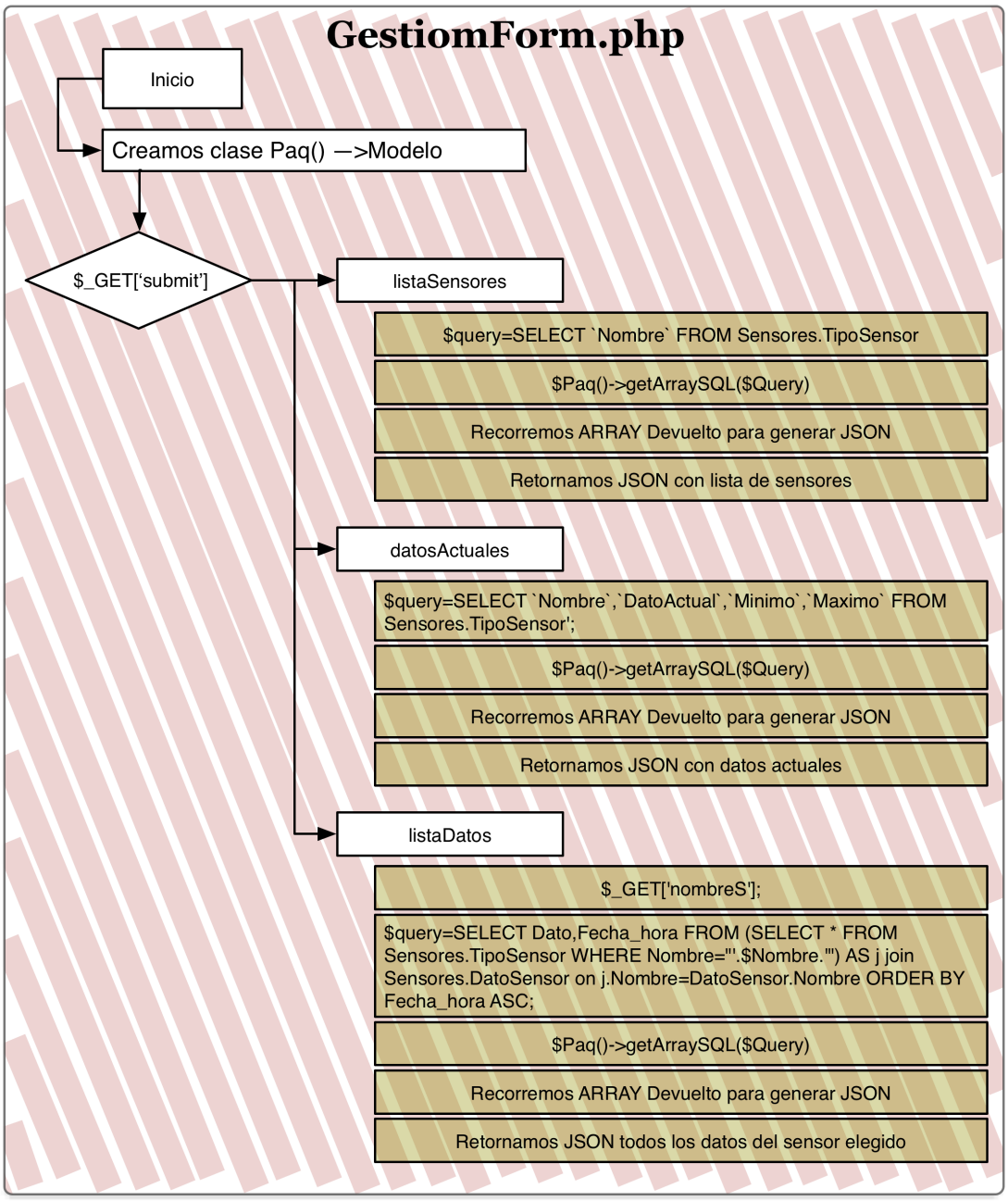
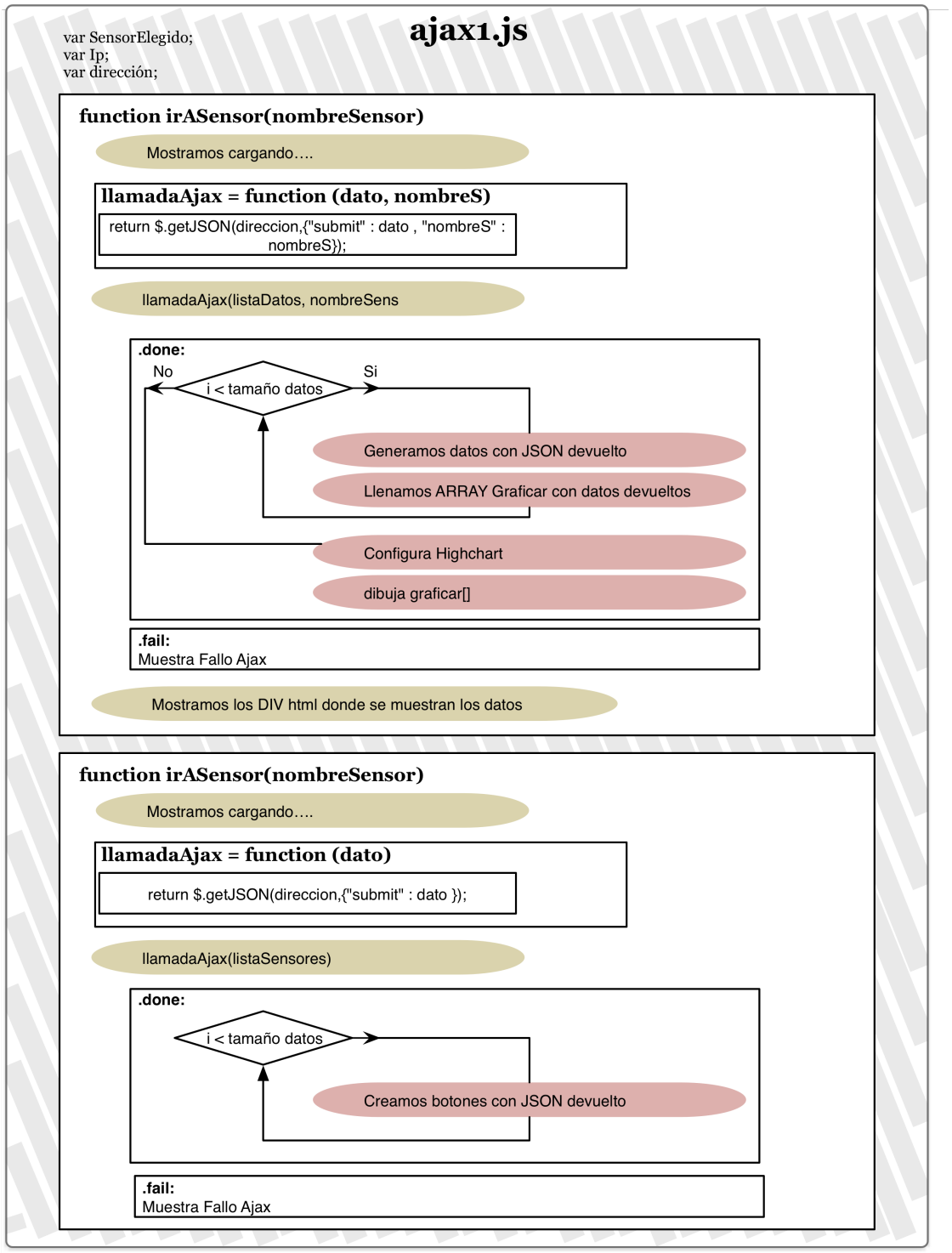


Figura 68 - Flujograma gestiomForm APK

Parte Apk *Android*

La parte que se ejecuta en *Android* es la vista del patrón MVC. El archivo *HTML* solo contiene contenedores div para mostrar la información que se vierte desde *JavaScript*. El archivo *Ajax*1.js realiza llamadas *Ajax* al controlador, este devuelve *JSON* con los datos, desde *Ajax*1.js decodificamos datos e insertamos en el archivo *HTML*. Vemos flujograma de funcionamiento:



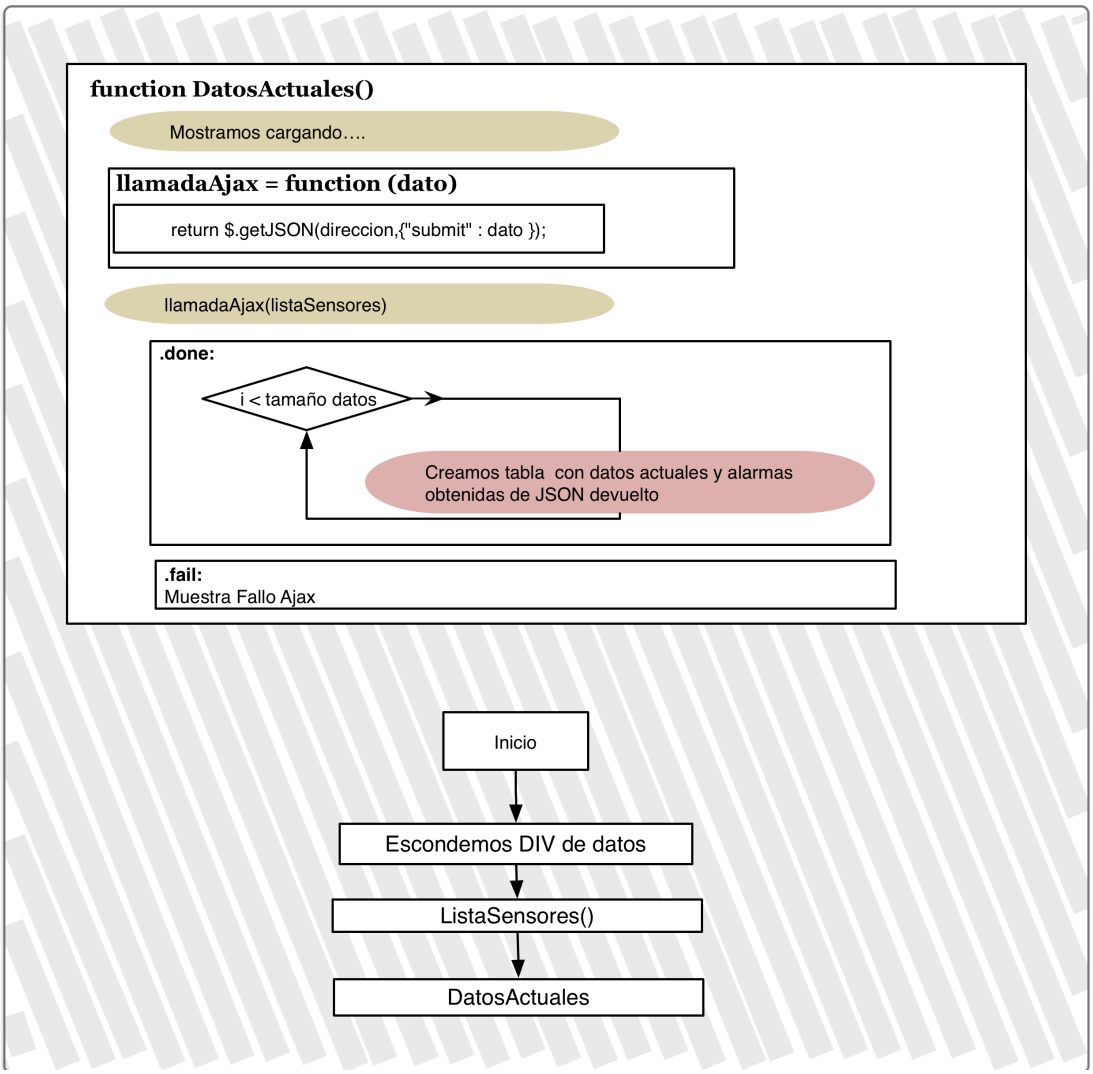


Figura 69 - Flujograma *Ajax*1.js APK *Android*

Una vez tenemos probado y generado el código de la aplicación web, abrimos Intel XDK para convertir de *HTML-CSS-JavaScript* a apk. Primero tenemos que identificarnos o crear una cuenta en XDK, según vemos en la figura 70.

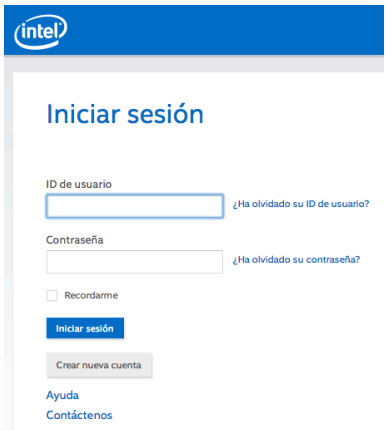


Figura 70 - Inicio de sesión XDK

Una vez dentro pulsamos nuevo proyecto. Ahora Pulsamos Import Your *HTML*5 Code Base. Nos pide carpeta raíz del sistema de archivos de la aplicación web. Una vez elegido nos pide nombre del proyecto, introducimos uno. Después configuramos preferencias del proyecto e introducimos imágenes de icono y fondo de pantalla con los diferentes tamaños pedidos como vemos en la figura 71.

Antes de cargar el proyecto tenemos que asegurarnos de tener configurada correctamente la IP del servidor donde tenemos los archivos que hacen la función de controlador y modelo.

En el archivo *JavaScript Ajax*1.js tenemos que poner en la variable llamada IP la IP del servidor donde tengamos alojado los archivos antes citados. Si estamos en la misma red local y conectamos el wifi del terminal en esa misma red, tendremos que poner la IP local del servidor, en nuestro caso 192.168.1.34. Si los archivos están colgados en internet, pondremos la IP fija o el nombre de dominio.

```
var ip="192.168.1.34";
var direccion="http://" + ip + "/~davsel/PHPAjax/GestiomForm.PHP";
```

Código 6 - Configuración de IP servidor en APK, *Ajax*1.js

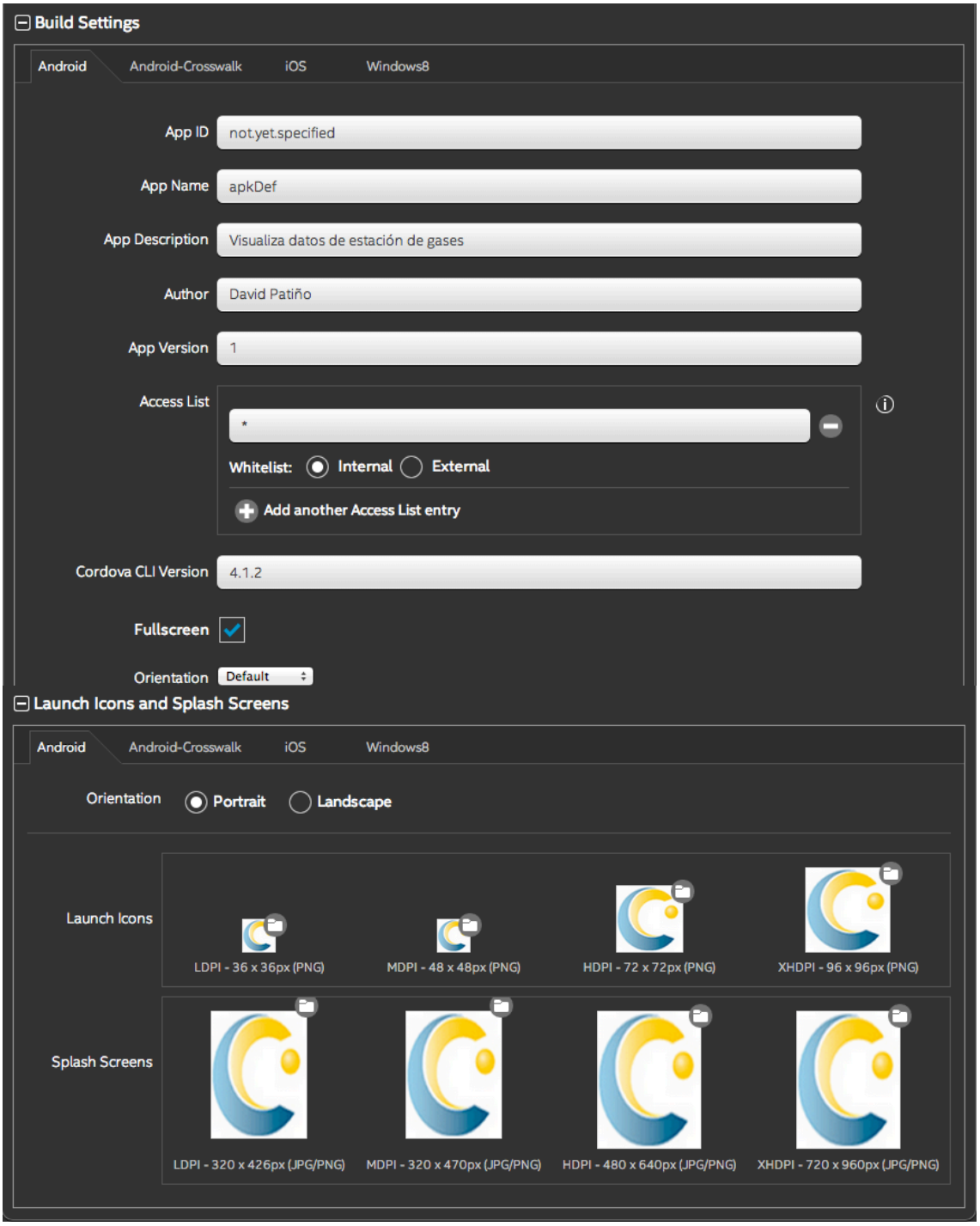


Figura 71 - Configuración de proyecto XDK

Acto seguido, podemos ver el código, probarlo, simularlo con el teléfono conectado mediante usb, etc. Para crear la apk debemos ir a la pestaña build, y elegir *Android*.

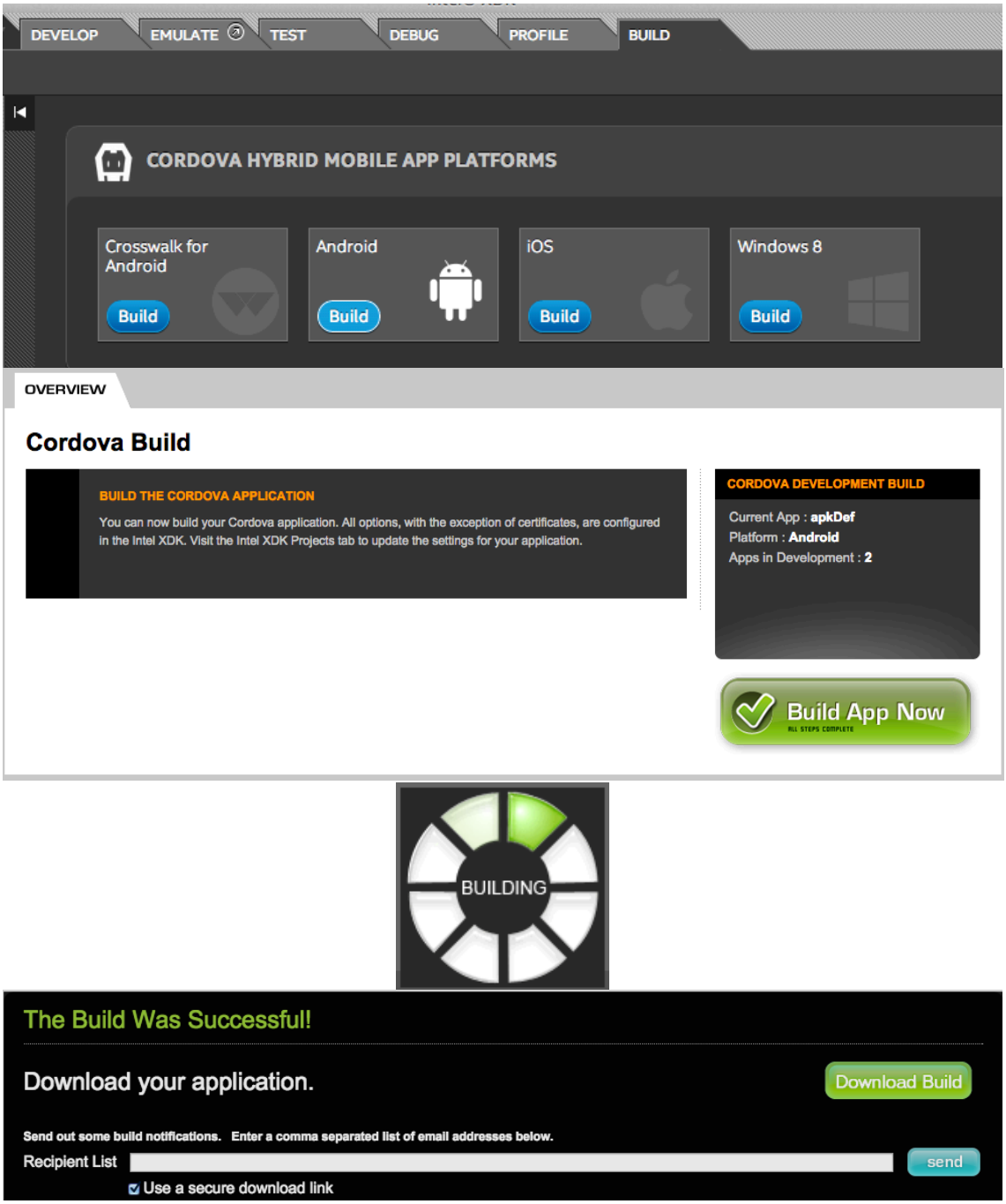


Figura 72 - Creación de Apk

Siguiendo los pasos que vemos en la figura anterior, nos crea la apk desde la aplicación web. Los archivos son guardados en el sistema cloud de Intel, puedes descargar la apk desde cualquier navegador incluso acceder a la aplicación web en versiones antiguas. Cuando termina de construir la aplicación, puedes enviar el enlace mediande email o descargar la apk desde el botón verde que se observa el la figura.

A continuación se muestran unas imágenes de cómo se ve la aplicación funcionando en *Android*.

Figura 73 - Detalle de aplicación *Android*

Para mejorar el funcionamiento de la aplicación cuando tenemos muchos datos en la base de datos, es necesario incluir worker *JavaScript* para la ejecución en segundo plano de la decodificación *JSON*. Se queda pendiente como mejora de la apk *Android*.

Capítulo 5

Código del sistema

El Capítulo 5 desarrolla el código completo de todas las partes del sistema desarrollado. Primero mostraremos el código de la aplicación web con sus diferentes archivos.

Se mostrará, seguidamente, el código de los archivos creados para crear el *Proxy*, con las tres clases creadas para tal función.

A continuación se mostrará el código para la creación de la aplicación *Android*. Que luego convertiremos mediante XDK como explicábamos en anteriores capítulos.

Código de la Aplicación web

Una de las cosas más importantes para el diseño de una aplicación web es la organización de archivos. Debido a la multitud de tecnologías que interactúan en la aplicación, todos los archivos tiene que estar lo más compartimentados posible. Es decir que los archivos *CSS* estén en su carpeta y fuera del archivo *HTML*, lo mismo pasa con los archivos Javascript. Es mucho más operativo sacar todos los script fuera del archivo *HTML*. Se muestra a continuación un esquema de cómo queda el sistema de archivos:

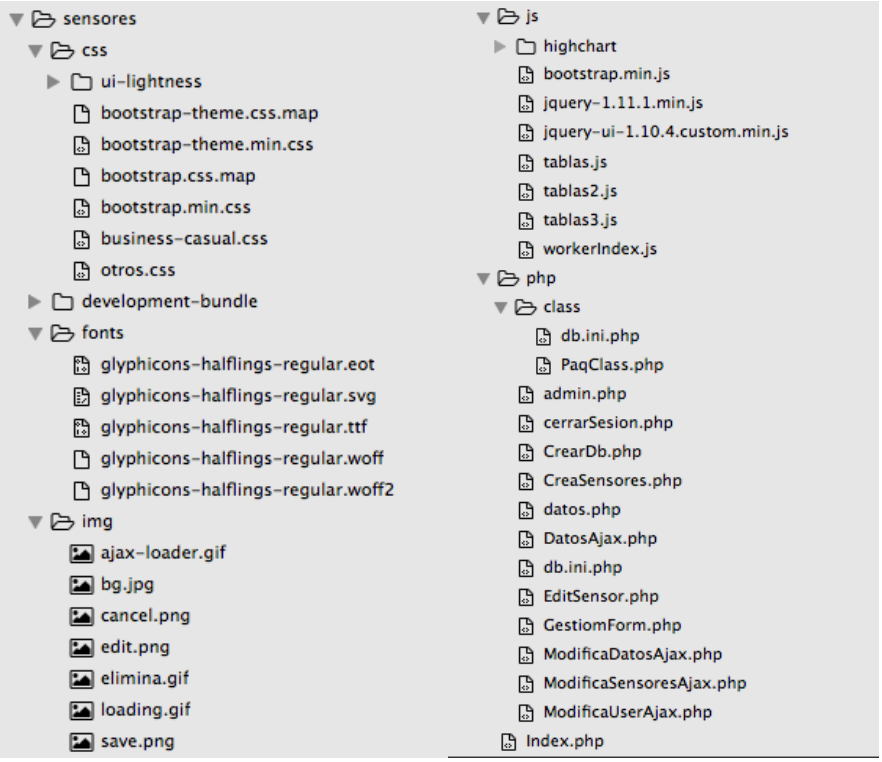


Figura 74 - Estructura de archivos de la aplicación web

CSS

Se muestra el código *CSS* utilizado en la aplicación web. Se utiliza de manera general en todas las páginas creadas. Solo se ponen los códigos *CSS* que se han creado aparte. La librerías externas *Bootstrap* no se reflejan aquí.

Aparte de la librería externa, utilizamos dos archivos más, uno llamado *Business-casual.css* que modifica el aspecto de algunos elementos de *Bootstrap*, como por ejemplo la barra de navegación y las fuentes de texto. El segundo archivo css utilizado se llama *Otros.css*, en éste se definen los iconos y aspecto de las tablas de modificación de datos así como los mensajes de error y cambio realizado correctamente. También se define el aspecto de las alarmas que se muestran de los últimos datos leídos.

Business-casual.*CSS*

```
/*!
 * Start Bootstrap - Business Casual Bootstrap Theme (http://startbootstrap.com)
 * Code licensed under the Apache License v2.0.
 * For details, see http://www.Apache.org/licenses/LICENSE-2.0.
 */

body {
    font-family: "Open Sans","Helvetica Neue",Helvetica,Arial,sans-serif;
    background: url('../img/bg.jpg') no-repeat center center fixed;
    -webkit-background-size: cover;
    -moz-background-size: cover;
    background-size: cover;
    -o-background-size: cover;
}

h1,h2,h3,h4,h5,h6 {
    text-transform: uppercase;
    font-family: "Josefin Slab","Helvetica Neue",Helvetica,Arial,sans-serif;
    font-weight: 700;
    letter-spacing: 1px;
}

p {
    font-size: 1.25em;
    line-height: 1.6;
    color: #000;
}

hr {
    max-width: 400px;
    border-color: #999999;
}

.brand,
.address-bar {
    display: none;
}

.navbar-brand {
    text-transform: uppercase;
    font-weight: 900;
    letter-spacing: 2px;
}

.navbar-nav {
    text-transform: uppercase;
    font-weight: 400;
    letter-spacing: 3px;
}

.img-full {
    min-width: 100%;
}

.brand-before,
.brand-name {
```

```
    text-transform: capitalize;
}

.brand-before {
    margin: 15px 0;
}

.brand-name {
    margin: 0;
    font-size: 4em;
}

.tagline-divider {
    margin: 15px auto 3px;
    max-width: 250px;
    border-color: #999999;
}

.box {
    margin-bottom: 20px;
    padding: 30px 15px;
    background: #fff;
    background: rgba(255,255,255,0.9);
}

.intro-text {
    text-transform: uppercase;
    font-size: 1.25em;
    font-weight: 400;
    letter-spacing: 1px;
}

.img-border {
    float: none;
    margin: 0 auto 0;
    border: #999999 solid 1px;
}

.img-left {
    float: none;
    margin: 0 auto 0;
}

footer {
    background: #fff;
    background: rgba(255,255,255,0.9);
}

footer p {
    margin: 0;
    padding: 50px 0;
}

@media screen and (min-width:768px) {
    .brand {
        display: inherit;
        margin: 0;
        padding: 30px 0 10px;
```

```
    text-align: center;
    text-shadow: 1px 1px 2px rgba(0,0,0,0.5);
    font-family: "Josefin Slab","Helvetica Neue",Helvetica,Arial,sans-serif;
    font-size: 5em;
    font-weight: 700;
    line-height: normal;
    color: #fff;
}

.top-divider {
    margin-top: 0;
}

.img-left {
    float: left;
    margin-right: 25px;
}

.address-bar {
    display: inherit;
    margin: 0;
    padding: 0 0 40px;
    text-align: center;
    text-shadow: 1px 1px 2px rgba(0,0,0,0.5);
    text-transform: uppercase;
    font-size: 1.25em;
    font-weight: 400;
    letter-spacing: 3px;
    color: #fff;
}

.navbar {
    border-radius: 0;
}

.navbar-header {
    display: none;
}

.navbar {
    min-height: 0;
}

.navbar-default {
    border: none;
    background: #fff;
    background: rgba(255,255,255,0.9);
}

.nav>li>a {
    padding: 35px;
}

.navbar-nav>li>a {
    line-height: normal;
}

.navbar-nav {
```

```
    display: table;
    float: none;
    margin: 0 auto;
    table-layout: fixed;
    font-size: 1.25em;
  }
}

@media screen and (min-width:1200px) {
  .box:after {
    content: '';
    display: table;
    clear: both;
  }
}
```

Código 7 - Business-casual.CSS

Otros.*CSS*

```
.elimina{background:url(../img/elimina.gif) 0 0 no-repeat}
.editable span{display:block;}
.editable span:hover {background:url(../img/edit.png) 90% 50% no-repeat;cursor:pointer}
.editableS span{display:block;}
.editableS span:hover {background:url(../img/edit.png) 90% 50% no-repeat;cursor:pointer}
td input{height:24px;width:200px;border:1px solid #ddd;padding:0 5px;margin:0;border-
        radius:6px;vertical-align:middle}
a.enlace{display:inline-block;width:24px;height:24px;margin:0 0 0 5px;overflow:hidden;text-indent:-
        999em;vertical-align:middle}
.guardar{background:url(../img/save.png) 0 0 no-repeat}
.cancelar{background:url(../img/cancel.png) 0 0 no-repeat}
.guardarS{background:url(../img/save.png) 0 0 no-repeat}
.cancelarS{background:url(../img/cancel.png) 0 0 no-repeat}
.mensaje{display:block;text-align:center;margin:0 0 20px 0}
.ok{display:block;padding:10px;text-align:center;background:green;color:#fff}
.ko{display:block;padding:10px;text-align:center;background:red;color:#fff}

.three-columns {
  -webkit-column-count: 3; /* Chrome, Safari, Opera */
  -moz-column-count: 3; /* Firefox */
  column-count: 3;
}

.CirculoRojo {
 background: red;
  border-radius: 0.8em;
 -moz-border-radius: 0.8em;
 -webkit-border-radius: 0.8em;
 color: #ffffff;
 display: inline-block;
 font-weight: bold;
 line-height: 1.6em;
 margin-right: 15px;
 text-align: center;
 width: 1.6em;
}
```

```
.CirculoAmarillo {
  background: #FFFF00;
   border-radius: 0.8em;
  -moz-border-radius: 0.8em;
  -webkit-border-radius: 0.8em;
  color: #ffffff;
  display: inline-block;
  font-weight: bold;
  line-height: 1.6em;
  margin-right: 15px;
  text-align: center;
  width: 1.6em;
}

.CirculoNaranja {
  background: #FF9900;
   border-radius: 0.8em;
  -moz-border-radius: 0.8em;
  -webkit-border-radius: 0.8em;
  color: #ffffff;
  display: inline-block;
  font-weight: bold;
  line-height: 1.6em;
  margin-right: 15px;
  text-align: center;
  width: 1.6em;
}

.CirculoVerde {
  background: #5EA226;
  border-radius: 0.8em;
  -moz-border-radius: 0.8em;
  -webkit-border-radius: 0.8em;
  color: #ffffff;
  display: inline-block;
  font-weight: bold;
  line-height: 1.6em;
  margin-right: 15px;
  text-align: center;
  width: 1.6em;
}

.scroll{
  border-color:blue;
  width:100%;
  height:100px;
  overflow:auto;
  margin: 0 auto;


}
```

Código 8 - otros.*CSS*

JavaScript

Se muestra el código *javascript* utilizado en las vistas de la aplicación. Se muestra solo el código creado para este fin, pero no se muestran librerías externas que se han utilizado.

Tablas.js

```
/*****************************************************************
        Función para crear worker(hilo)
*****************************************************************/

var w; //variable para crear worker

function startWorker() {
   if(typeof(Worker) !== "undefined") {
      if(typeof(w) == "undefined") {
         w = new Worker("./js/workerIndex.js");
      }
      //Cuando responde con datos, los mete en la tabla.
      w.onmessage = function(event) {
        // document.getElementById("result").innerHTML = event.data;

        var JSON = JSON.parse(event.data);
        $('#editaDatos tbody').remove();
        salida = "<tbody>"
        $.each(JSON.datos, function(key,valor){
         max=parseInt(valor.Max);
         min=parseInt(valor.Min);
         dato=parseInt(valor.Dato);
         var d = dato.toFixed(2);
         salida+= "<tr> <td><span>"+valor.Sensor+"</span></td><td><span>"+d+' '+ valor.ud
         +"</span></td>";
         if(max>=(dato) && min<=(dato))
         salida+= "<td> <span class='CirculoVerde'> </span>  </td></tr>";
                        else
                            salida+="<td> <span class='CirculoRojo'> </span>  </td></tr>";
                        });
         salida += "</tbody>"
         $('#editaDatos').append(salida);

      };
   } else {
      $("#errorWorker").HTML("<span class='ko'>Lo sentimos, Tu navegador no soporta workers, acualiza
         su navegador</span>");
   }
}

function stopWorker() {
   w.terminate();
   w = undefined;
}



$(document).ready(function()
```

```
{
	//Iniciamos worker
	//startWorker();
	/*****************************************************************
			OBTENEMOS TABLA DATOS con Conexión Ajax
	*****************************************************************/
		$.Ajax({
			type: "POST",
			url: "./PHP/DatosAjax.PHP",
			data: { tabla: 1}
		    })
		    .done(function(JSON) {
			//console.log(JSON);
			if (JSON.vacio=="si")
			{
			    //console.log("no datos");
			    $("#errorAjaxDatos").HTML("<span class='ko'>No existen datos</span>");
			}
			else
			{
			    //console.log(JSON);
			    //Iniciamos worker
			    startWorker();
			    salida="";
			    $.each(JSON.datos, function(key,valor){
				salida="";
				max=parseInt(valor.Max);
				min=parseInt(valor.Min);
				dato=parseInt(valor.Dato);
				salida+= '<tr>
	<td><span>'+valor.Sensor+'</span></td><td><span>'+valor.Dato+ ' '+ valor.ud +
	'</span></td>';
				if(max>=(dato) && min<=(dato))
				salida+= '<td> <span class="CirculoVerde"> </span></td></tr>';
			    else
				salida+= '<td> <span class="CirculoRojo"> </span></td></tr>';
			    $('#editaDatos tbody').append(salida);
			    });
			}
		    })
		    .fail(function(jqXHR, textStaus, errorThorown){
			$("#errorAjaxDatos").HTML("<span class='ko'>Fallo en conexion de datos: " +textStaus
	+ "</span>");
		    });

		$(document).on("click","button.EmpiezaWorker",function(e)
		{
		    e.preventDefault();
		    startWorker();
		});

		$(document).on("click","button.ParaWorker",function(e)
		{
		    e.preventDefault();
		    stopWorker();
		});

		$(document).on("click","button.ActualizaDatos",function(e)
```

```
        {
            e.preventDefault();
            $('#editaDatos tbody').remove();
            $.Ajax({
                type: "POST",
                url: "./PHP/DatosAjax.PHP",
                data: { tabla: 1}
            })
            .done(function(JSON) {
            if (JSON.vacio=="si")
                {
                    //console.log("no datos");
                    $("#errorAjaxDatos").HTML("<span class='ko'>No existen datos</span>");
                }
                else
                {
                    salida="<tbody>";
                    $.each(JSON.datos, function(key,valor){
                        max=parseInt(valor.Max);
                        min=parseInt(valor.Min);
                        dato=parseInt(valor.Dato);
                        salida+= "<tr>
    <td><span>"+valor.Sensor+"</span></td><td><span>"+valor.Dato+"</span></td>";
                        if(max>=(dato) && min<=(dato))
                        salida+= "<td> <span class='CirculoVerde'> </span>  </td></tr>";
                    else
                        salida+="<td> <span class='CirculoRojo'> </span>  </td></tr>";
                    });
                    salida+="</tbody>";
                $('#editaDatos').append(salida);
              }
            })
            .fail(function(jqXHR, textStaus, errorThorown){
                $("#errorAjaxDatos").HTML("<span class='ko'>Fallo en conexion de datos: " +textStaus
    + "</span>");
            });
        });
});
```

Código 9 - tablas.js

WorkerIndex.js

```
var i = 0;
var xhr;

//funciones Ajax Puro JavaScript
 function load(url, callback) {
        if(typeof XMLHttpRequest !== 'undefined') xhr = new XMLHttpRequest();
        else {
                var versions = ["MSXML2.XmlHttp.5.0",
                        "MSXML2.XmlHttp.4.0",
                        "MSXML2.XmlHttp.3.0",
                        "MSXML2.XmlHttp.2.0",
                        "Microsoft.XmlHttp"]

                for(var i = 0, len = versions.length; i < len; i++) {
                try {
                    xhr = new ActiveXObject(versions[i]);
                    break;
                }
                    catch(e){}
                } // end for
        }
        xhr.onreadystatechange = ensureReadiness;

        function ensureReadiness() {
                if(xhr.readyState < 4) {
                    return;
                }

                if(xhr.status !== 200) {
                    return;
                }

                // all is well
                if(xhr.readyState === 4) {
                    callback(xhr);
                }
        }
        xhr.open('POST', url, true);
        xhr.send('');
}
function buSCADAtosAjax() {
//Llamada a Ajax puro javascrip
        load('../PHP/DatosAjax.PHP', function(xhr) {
                var result = xhr;//.responseText;
                //console.log(result.response);
                postMessage(result.response);
        });
        //Tiempo de actualización en milisegundos
   setTimeout("buSCADAtosAjax()",1000);
}
buSCADAtosAjax();
```

Código 10 - workerIndex.js

Tablas2.js

```
/*****************************************************************
        Valida Password de entrada
        _____________________
            IN: Dato cualquier tipo, div para mostrar el error
            OUT: Boolean true o false.
*****************************************************************/
function ValPass(dato,div){

    if(dato.length < 3){
            $(div).fadeIn();
        $(div).HTML("<span class='ko'>Contrase&ntilde;a no puede estar vacio ni tener menos de 3
    caracteres</span>");
        return false;
      }
      //en otro caso, el mensaje no se muestra
    else{
        $(div).fadeOut();
        return true;
    }
}

/*****************************************************************
        Valida User de entrada
        _____________________
            IN: Dato cualquier tipo, div para mostrar el error
            OUT: Boolean true o false.
*****************************************************************/
function ValUser(dato,div){
    if(dato.length < 3){
                    $(div).fadeIn();
        $(div).HTML("<span class='ko'>Usuario no puede estar vacio ni tener menos de 3
    caracteres</span>");
        return false;
      }
      //en otro caso, el mensaje no se muestra
    else{
        $(div).fadeOut();
        return true;
    }
}

/*****************************************************************
        Valida Privilegios de entrada
        _____________________
            IN: Dato cualquier tipo, div para mostrar el error
            OUT: Boolean true o false.
*****************************************************************/
function ValPriv(dato,div){
    if((dato == "99")||(dato == "0")){
            $(div).fadeOut();
            return true;
      }
      //en otro caso, el mensaje no se muestra
    else{
    $(div).fadeIn();
        $(div).HTML("<span class='ko'>Privilegios no puede estar vacio, debe ser 0 o 99</span>");
```

```
                return false;
        }
}

/****************** Valida TAbla Sensores ************************/


/*****************************************************************
        Valida Nombre Sensor de entrada
        _____________________
            IN: Dato cualquier tipo, div para mostrar el error
            OUT: Boolean true o false.
*****************************************************************/
function ValNom(dato,div){

        if(dato.length < 3){
            $(div).fadeIn();
            $(div).HTML("<span class='ko'>Nombre no puede estar vacio ni tener menos de 3
        caracteres</span>");
            return false;
          }
          //en otro caso, el mensaje no se muestra
        else{
            $(div).fadeOut();
            return true;
        }
}

/*****************************************************************
        Valida Dirección de Sensor de entrada
        _____________________
            IN: Dato cualquier tipo, div para mostrar el error
            OUT: Boolean true o false.
*****************************************************************/
function ValDir(dato,div){

        if(dato.length < 6){
            $(div).fadeIn();
            $(div).HTML("<span class='ko'>Direcci&oacute;n no puede estar vacio ni tener menos de 6
        caracteres</span>");
            return false;
          }
          //en otro caso, el mensaje no se muestra
        else{
            $(div).fadeOut();
            return true;
        }
}
```

```
/*****************************************************************
                Valida Delay de Sensor de entrada
                _____________________
                    IN: Dato cualquier tipo, div para mostrar el error
                    OUT: Boolean true o false.
*****************************************************************/
function ValDel(dato,div){
        minDel=0;
        maxDel=1441;
        datoInt=parseInt(dato);
        if(!((datoInt > minDel) && (datoInt < maxDel))){
                $(div).fadeIn();
            $(div).HTML("<span class='ko'>Delay no puede estar vacio, debe estar entre " + (minDel+1) +  "
        y " + (maxDel-1) + " segundos</span>");
            return false;
          }
          //en otro caso, el mensaje no se muestra
        else{
            $(div).fadeOut();
            return true;
        }
}

/*****************************************************************
                Valida Unidades de Sensor de entrada
                _____________________
                    IN: Dato cualquier tipo, div para mostrar el error
                    OUT: Boolean true o false.
*****************************************************************/
function ValUnidad(dato,div){
        if(dato.length < 3){
                $(div).fadeIn();
            $(div).HTML("<span class='ko'>Unidades no puede estar vacio debe ser mayor de 3
        caracteres</span>");
            return false;
          }
          //en otro caso, el mensaje no se muestra
        else{
            $(div).fadeOut();
            return true;
        }
}
/*****************************************************************
                Valida Ud de Sensor de entrada
                _____________________
                    IN: Dato cualquier tipo, div para mostrar el error
                    OUT: Boolean true o false.
*****************************************************************/
function ValUd(dato,div){
        if(dato.length < 1){
                $(div).fadeIn();
            $(div).HTML("<span class='ko'>Ud no puede estar vacio debe ser mayor de 1
        caracteres</span>");
            return false;
          }
          //en otro caso, el mensaje no se muestra
        else{
```

```
            $(div).fadeOut();
            return true;
        }

/*****************************************************************
                Valida Ud de Maximo de entrada
                _____________________
                    IN: Dato cualquier tipo, div para mostrar el error
                    OUT: Boolean true o false.
*****************************************************************/
function ValMax(dato,div){
        mini=0;
        datoInt=parseInt(dato);
        if(!(datoInt > mini)){
        $(div).fadeIn();
     $(div).HTML("<span class='ko'>M&aacute;ximo no puede estar vacio, debe ser mayor que " + (mini+1)
        + " unidades </span>");
     return false;
    }
    //en otro caso, el mensaje no se muestra
        else{
     $(div).fadeOut();
     return true;
  }
}

/*****************************************************************
                Valida Ud de Mínimo de entrada
                _____________________
                    IN: Dato cualquier tipo, div para mostrar el error
                    OUT: Boolean true o false.
*****************************************************************/
function ValMin(dato,max,div){
        maxInt=parseInt(max);
        datoInt=parseInt(dato);
        if(!(datoInt < maxInt)){
                $(div).fadeIn();
            $(div).HTML("<span class='ko'>M&iacute;nimo no puede estar vacio, debe ser menor que " +
        maxInt + " unidades </span>");
            return false;
          }
          //en otro caso, el mensaje no se muestra
        else{
            $(div).fadeOut();
            return true;
        }
}


$(document).ready(function()
        {
        /*****************************************************************
                        OBTENEMOS TABLA USER con Conexión Ajax
        *****************************************************************/
                $.Ajax({
                        type: "POST",
                        url: "ModificaUserAjax.PHP",
                        data: { tabla: 1}
```

```
            })
        //$.getJSON("ModificaUserAjax.PHP",{"tabla":1})
        .done(function(JSON) {
            $.each(JSON.datos, function(key,valor){
                $('#editaUser tbody').append(
                    "<tr title='Edita dato'><td class='editable' id='id' data-
campo='User'><span>"+valor.User+"</span></td><td class='editable' data-
campo='Pass'><span>"+valor.Pass+"</span></td><td class='editable' data-
campo='Permisos'><span>"+valor.Priv+"</span></td><td><button type='button' id='del "+ key+ " '
title='Elimina Usuario' class='eliminarUser btn btn-danger'>x</button></td></tr>");
            });
        })
        .fail(function(jqXHR, textStaus, errorThorown){
                $("#errorAjaxUser").HTML("<span class='ko'>Fallo en conexion de datos: " +textStaus
+ "</span>");
        });


/*****************************************************************
                *Editamos Valores unitarios de tabla USER
*****************************************************************/
        var td,campo,valor,Usuario;
        $(document).on("click","td.editable span",function(e)
        {
            e.preventDefault();
            $("td").removeClass("editable");
            td=$(this).closest("td");
            campo=$(this).closest("td").data("campo");
            valor=$(this).text();
            Usuario=$(this).closest("tr").find("#id").text();
            td.text("").HTML("<input title='Edita dato' type='text' name='"+campo+"'
value='"+valor+"'><a class='enlace guardar' href='#'>Guardar</a><a class='enlace cancelar'
href='#'>Cancelar</a>");
        });


/*****************************************************************
            *Acciones al pulsar icono cancelar*
*****************************************************************/
        $(document).on("click",".cancelar",function(e)
        {
            e.preventDefault();
            td.HTML("<span>"+valor+"</span>");
            $("td").addClass("editable");
        });

/*****************************************************************
                *Acciones al pulsar icono guardar*
*****************************************************************/
        $(document).on("click",".guardar",function(e)
        {
            $("#aciertoAjaxUser").HTML("<img src='../img/loading.gif'>");
            e.preventDefault();
            nuevovalor=$(this).closest("td").find("input").val();

            //console.log($(this).closest("td").prevAll().length);
            /*********************************************
                    VALIDAMOS DATOS INTRODUCIDOS
```

```
            ***********************************************/
            //Sacamos la columna en la que estamos para saber que dato validamos
            switch ($(this).closest("td").prevAll().length)
            {
case 0: //Usuarios
  {
        console.log("cero");
        if (ValUser(nuevovalor,"#errorAjaxUser")){}
        else
        {
                setTimeout(function() {$('.ok,.ko').fadeOut('fast');}, 3000);
                return false ;
            }
        break;
  }
case 1: //Password
  {
        if (ValPass(nuevovalor,"#errorAjaxUser")){}
            else
            {
                setTimeout(function() {$('.ok,.ko').fadeOut('fast');}, 3000);
                return false ;
            }
        break;
  }
case 2: //Privilegios
  {
        if (ValPriv(nuevovalor,"#errorAjaxUser")){}
            else
            {
                setTimeout(function() {$('.ok,.ko').fadeOut('fast');}, 3000);
                return false ;
            }
        break;
  }
}
            $.Ajax({
                type: "POST",
                url: "ModificaUserAjax.PHP",
                data: { tabla: 4, campo: campo, valor: nuevovalor, Usuario: Usuario }
            })
            .done(function( msg ) {
                $("#aciertoAjaxUser").HTML(msg);
                td.HTML("<span>"+nuevovalor+"</span>");
                $("td").addClass("editable");
                setTimeout(function() {$('.ok,.ko').fadeOut('fast');}, 3000);
            })
            .fail(function(jqXHR, textStaus, errorThorown){
                $("#errorAjaxUser").HTML("<span class='ko'>Fallo al guardar datos: " +textStaus +
"</span>");
            });
        });
```

```
/*****************************************************************
            *Acciones al pulsar el boton eliminar USER*
*****************************************************************/
    $(document).on("click","button.eliminarUser",function(e)
    {
        e.preventDefault();
        //console.log($(this).closest("tr").find("#id").text());
        id=$(this).closest("tr").find("#id").text();
        $.Ajax({
            type: "POST",
            url: "ModificaUserAjax.PHP",
            data: { tabla: 2 , ID: id }
        })
        .done(function( msg ) {

            $('#editaUser tbody').remove();
            salida="<tbody>";
            $.Ajax({
                type: "POST",
                url: "ModificaUserAjax.PHP",
                data: { tabla: 1 }
            })
                .done(function(JSON) {
                    $.each(JSON.datos, function(key,valor){
                        salida+="<tr title='Edita dato'><td class='editable' id='id' data-
campo='User'><span>"+valor.User+"</span></td><td class='editable' data-
campo='Pass'><span>"+valor.Pass+"</span></td><td class='editable' data-
campo='Permisos'><span>"+valor.Priv+"</span></td><td><button type='button' id='del "+ key+ " '
title='Elimina Usuario' class='eliminarUser btn btn-danger'>x</button></td></tr>";
                    });
                    salida+="</tbody>"
                    $('#editaUser').append(salida);
                });
            $("#aciertoAjaxUser").HTML("<span class='ok'>El Usuario "+id+" ha sido eliminado
correctamente "+ msg +"</span>");
            //td.HTML("<span>"+id+"</span>");
            setTimeout(function() {$('.ok,.ko').fadeOut('fast');}, 3000);
        })
        .fail(function(jqXHR, textStaus, errorThorown){
            $("#errorAjaxUser").HTML("<span class='ko'>Fallo en Eliminacion de usuario: "
+textStaus + "</span>");
        });
    });
/*****************************************************************
            *Acciones al pulsar boton crear usuario*
*****************************************************************/
    $(document).on("click","#btnCreaUser",function(e)
    {
        e.preventDefault();
        User= $("#NuevoUser").val();
        Pass= $("#NuevoPass").val();
        Priv= $("#NuevoPriv").val();

        if (ValUser(User,"#errorAjaxUser")){}
        else
        {
            setTimeout(function() {$('.ok,.ko').fadeOut('fast');}, 3000);
            return false ;
```

```
            }

if (ValPass(Pass,"#errorAjaxUser")){}
            else
            {
                setTimeout(function() {$('.ok,.ko').fadeOut('fast');}, 3000);
                return false ;
            }

            if (ValPriv(Priv,"#errorAjaxUser")){}
            else
            {
                setTimeout(function() {$('.ok,.ko').fadeOut('fast');}, 3000);
                return false ;
            }
            $.Ajax({
                type: "POST",
                url: "ModificaUserAjax.PHP",
                data: { tabla: 3, U:User, Pa:Pass, p:Priv }
            })
            .done(function( msg ) {

                $('#editaUser tbody').remove();
                salida="<tbody>";
                $.Ajax({
                    type: "POST",
                    url: "ModificaUserAjax.PHP",
                    data: { tabla: 1 }
                })
                .done(function(JSON) {
                    $.each(JSON.datos, function(key,valor){
                        salida+="<tr title='Edita dato'><td class='editable' id='id' data-
campo='User'><span>"+valor.User+"</span></td><td class='editable' data-
campo='Pass'><span>"+valor.Pass+"</span></td><td class='editable' data-
campo='Permisos'><span>"+valor.Priv+"</span></td><td><button type='button' id='del "+ key+ " '
title='Elimina Usuario' class='eliminarUser btn btn-danger'>x</button></td></tr>";
                    });
                    salida+="</tbody>"
                    $('#editaUser').append(salida);
                });
                $("#NuevoUser").val("");
                $("#NuevoPass").val("");
                $("#NuevoPriv").val("");

                $("#aciertoAjaxUser").HTML("<span class='ok'>El Usuario "+User+" ha sido creado
correctamente "+ msg +"</span>");
                setTimeout(function() {$('.ok,.ko').fadeOut('fast');}, 3000);
            })
            .fail(function(jqXHR, textStaus, errorThorown){
                $("#errorAjaxUser").HTML("<span class='ko'>Fallo en creaci&oacute;n de usuario: "
+textStaus + "</span>");
            });
        });
```

```
/*****************************************************************
                    *Acciones al pulsar boton Actualizar tabla*
*****************************************************************/
$(document).on("click","button.ActualizaUser",function(e)
            {
                $('#editaUser tbody').remove();
                salida="<tbody>"
                $.Ajax({
                    type: "POST",
                    url: "ModificaUserAjax.PHP",
                    data: { tabla: 1 }
                    })
                    .done(function(JSON) {
                        $.each(JSON.datos, function(key,valor){
                            salida+="<tr title='Edita dato'><td class='editable' id='id' data-
    campo='User'><span>"+valor.User+"</span></td><td class='editable' data-
    campo='Pass'><span>"+valor.Pass+"</span></td><td class='editable' data-
    campo='Permisos'><span>"+valor.Priv+"</span></td><td><button type='button' id='del "+ key+ " '
    title='Elimina Usuario' class='eliminarUser btn btn-danger'>x</button></td></tr>";
                        });
                        salida+="</tbody>"
                        $('#editaUser').append(salida);
                    })
                .fail(function(jqXHR, textStaus, errorThorown){
                    $("#errorAjaxUser").HTML("<span class='ko'>Fallo en actuañización de tabla: "
    +textStaus + "</span>");
                });
});
/*****************************************************************
                    OBTENEMOS TABLA SENSORES con Conexión Ajax
    *****************************************************************/
            per=$('#permiso').val();
            $.Ajax({
                    type: "POST",
                    url: "ModificaSensoresAjax.PHP",
                    data: { tabla: 1}
                })
            .done(function(JSON) {
                //console.log(JSON.vac);

                if (JSON.vacio=="no")
                {
                    $.each(JSON.datos, function(key,valor){
                        $('#editaSensor tbody').append(
                            "<tr title='Edita dato'><td class='editableS' id='idS' data-
    campo='Nombre'><span>"+valor.NombreS+"</span></td><td class='editableS' data-
    campo='Direccion'><span>"+valor.DirS+"</span></td><td class='editableS' data-
    campo='Delay'><span>"+valor.DelS+"</span></td><td class='editableS' data-
    campo='Unidades'><span>"+valor.UnidadS+"</span></td><td class='editableS' data-
    campo='Ud'><span>"+valor.UdS+"</span></td><td class='editableS' data-campo='Maximo'><span
    id='MaxS'>"+valor.MaxS+"</span></td><td class='editableS' data-
    campo='Minimo'><span>"+valor.MinS+"</span></td><td data-
    campo='DatoActual'><span>"+valor.Act+"</span></td><td><button type='button' id='delSensor
    "+ key+ " ' title='Elimina Sensor' class='eliminarSensor btn btn-danger'>x</button></td></tr>");
                    });
```

```
                if (per=="false")
                        {
                            $("#editaSensor td").removeClass("editableS");
                            $("#editaSensor button.eliminarSensor").attr("disabled", "disabled");
                        }
            }
            else
            {
                $("#errorAjaxSensor").HTML("<span class='ko'>No hay datos en esta tabla </span>");
            }

        })
        .fail(function(jqXHR, textStaus, errorThorown){
                $("#errorAjaxSensor").HTML("<span class='ko'>Fallo en conexion de datos: "
+textStaus + "</span>");
        });

/*****************************************************************
                *Editamos Valores unitarios de tabla SENSORES
*****************************************************************/
        var tdS,campoS,valorS,Nombre;
        $(document).on("click","td.editableS span",function(e)
        {
            e.preventDefault();
            $("td").removeClass("editableS");
            tdS=$(this).closest("td");
            campoS=$(this).closest("td").data("campo");
            valorS=$(this).text();
            Nombre=$(this).closest("tr").find("#idS").text();
            tdS.text("").HTML("<input title='Edita dato' type='text' name='"+campoS+"'
value='"+valorS+"'><a class='enlace guardarS' href='#'>Guardar</a><a class='enlace cancelarS'
href='#'>Cancelar</a>");
        });

/*****************************************************************
            *Acciones al pulsar icono cancelar Sensor*
*****************************************************************/
        $(document).on("click",".cancelarS",function(e)
        {
            e.preventDefault();
            tdS.HTML("<span>"+valorS+"</span>");
            $("td").addClass("editableS");
        });

/*****************************************************************
                *Acciones al pulsar icono guardar Sensor*
*****************************************************************/
        $(document).on("click",".guardarS",function(e)
        {
            $("#aciertoAjaxSensor").HTML("<img src='../img/loading.gif'>");
            e.preventDefault();
            nuevovalorS=$(this).closest("td").find("input").val();

            console.log($(this).closest("td").prevAll().length);
            /*********************************************
                    VALIDAMOS DATOS INTRODUCIDOS
            *********************************************/
            //Sacamos la columna en la que estamos para saber que dato validamos
```

```
            switch ($(this).closest("td").prevAll().length)
            {
case 0: //Nombre
  {
        if (ValNom(nuevovalorS,"#errorAjaxSensor")){}
        else
        {
                setTimeout(function() {$('.ok,.ko').fadeOut('fast');}, 3000);
                return false ;
            }
        break;
  }
case 1: //Direccion
  {
        if (ValDir(nuevovalorS,"#errorAjaxSensor")){}
            else
            {
                setTimeout(function() {$('.ok,.ko').fadeOut('fast');}, 3000);
                return false ;
            }
        break;
  }
case 2: //Delay
  {
        if (ValDel(nuevovalorS,"#errorAjaxSensor")){}
            else
            {
                setTimeout(function() {$('.ok,.ko').fadeOut('fast');}, 3000);
                return false ;
            }
        break;
  }
case 3: //Unidades
  {
        if (ValUnidad(nuevovalorS,"#errorAjaxSensor")){}
            else
            {
                setTimeout(function() {$('.ok,.ko').fadeOut('fast');}, 3000);
                return false ;
            }
        break;
  }
case 4: //Ud
  {
        if (ValUd(nuevovalorS,"#errorAjaxSensor")){}
            else
            {
                setTimeout(function() {$('.ok,.ko').fadeOut('fast');}, 3000);
                return false ;
            }
        break;
  }
case 5: //Max
  {
        if (ValMax(nuevovalorS,"#errorAjaxSensor")){}
            else
            {
                setTimeout(function() {$('.ok,.ko').fadeOut('fast');}, 3000);
```

```
                        return false ;
                    }
            break;
        }
    case 6: //Min
        {
            max=$(this).closest("tr").find("#MaxS").text();
            if (ValMin(nuevovalorS,max,"#errorAjaxSensor")){}
                    else
                    {
                        setTimeout(function() {$('.ok,.ko').fadeOut('fast');}, 3000);
                        return false ;
                    }
            break;
        }
    }

                $.Ajax({
                    type: "POST",
                    url: "ModificaSensoresAjax.PHP",
                    data: { tabla: 4, campo: campoS, valor: nuevovalorS, Nombre: Nombre }
                })
                .done(function( msg ) {
                    $("#aciertoAjaxSensor").HTML(msg);
                    tdS.HTML("<span>"+nuevovalorS+"</span>");
                    $("td").addClass("editableS");
                    setTimeout(function() {$('.ok,.ko').fadeOut('fast');}, 3000);
                })
                .fail(function(jqXHR, textStaus, errorThorown){
                    $("#errorAjaxSensor").HTML("<span class='ko'>Fallo al guardar datos: " +textStaus +
"</span>");
                });
            });

    /*****************************************************************
                    *Acciones al pulsar el boton eliminar Sensor*
    *****************************************************************/
            $(document).on("click","button.eliminarSensor",function(e)
            {
                e.preventDefault();
                //console.log($(this).closest("tr").find("#idS").text());
                idS=$(this).closest("tr").find("#idS").text();
                $.Ajax({
                    type: "POST",
                    url: "ModificaSensoresAjax.PHP",
                    data: { tabla: 2 , ID: idS }
                })
                .done(function( msg ) {

                    $('#editaSensor tbody').remove();
                    salida1="<tbody>";
                    $.Ajax({
                        type: "POST",
                        url: "ModificaSensoresAjax.PHP",
                        data: { tabla: 1 }
                    })
                        .done(function(JSON) {
                            $.each(JSON.datos, function(key,valor){
```

```
                                salida1+="<tr title='Edita dato'><td class='editableS' id='idS' data-
campo='Nombre'><span>"+valor.NombreS+"</span></td><td class='editableS' data-
campo='Direccion'><span>"+valor.DirS+"</span></td><td class='editableS' data-
campo='Delay'><span>"+valor.DelS+"</span></td><td class='editableS' data-
campo='Unidades'><span>"+valor.UnidadS+"</span></td><td class='editableS' data-
campo='Ud'><span>"+valor.UdS+"</span></td><td class='editableS' data-campo='Maximo'><span
id='MaxS'>"+valor.MaxS+"</span></td><td class='editableS' data-
campo='Minimo'><span>"+valor.MinS+"</span></td><td data-
campo='DatoActual'><span>"+valor.Act+"</span></td><td><button type='button' id='delSensor
"+ key+ " ' title='Elimina Sensor' class='eliminarSensor btn btn-danger'>x</button></td></tr>";
                            });
                            salida1+="</tbody>"
                            $('#editaSensor').append(salida1);
                        });
                    $("#aciertoAjaxSensor").HTML("<span class='ok'>El Sensor "+idS+" ha sido eliminado
correctamente "+ msg +"</span>");
                    setTimeout(function() {$('.ok,.ko').fadeOut('fast');}, 3000);
                })
                .fail(function(jqXHR, textStaus, errorThorown){
                    $("#errorAjaxSensor").HTML("<span class='ko'>Fallo en Eliminacion de sensor:  "
+textStaus + "</span>");
                });
            });
/*****************************************************************
                *Acciones al pulsar boton crear Sensor*
*****************************************************************/
            $(document).on("click",".CreaSensor",function(e)
            {
                e.preventDefault();
                nom=$("#NombreS").val();
                dir=$("#DirS").val();
                del=$("#DelS").val();
                uni=$("#UnidadS").val();
                ud=$("#UdS").val();
                max=$("#MaxS").val();
                min=$("#MinS").val();
                console.log("max=" +max+ " min="+min);
                if (ValNom(nom,"#errorAjaxSensor")){}
else
{
                    setTimeout(function() {$('.ok,.ko').fadeOut('fast');}, 3000);
                    return false ;
                }
                if (ValDir(dir,"#errorAjaxSensor")){}
                else
                {
                    setTimeout(function() {$('.ok,.ko').fadeOut('fast');}, 3000);
                    return false ;
                }
                if (ValDel(del,"#errorAjaxSensor")){}
                else
                {
                    setTimeout(function() {$('.ok,.ko').fadeOut('fast');}, 3000);
                    return false ;
                }
                if (ValUnidad(uni,"#errorAjaxSensor")){}
                else
                {
```

```
                setTimeout(function() {$('.ok,.ko').fadeOut('fast');}, 3000);
                return false ;
            }
            if (ValUd(ud,"#errorAjaxSensor")){}
                    else
            {
                setTimeout(function() {$('.ok,.ko').fadeOut('fast');}, 3000);
                return false ;
            }
            if (ValMax(max,"#errorAjaxSensor")){}
            else
            {
                setTimeout(function() {$('.ok,.ko').fadeOut('fast');}, 3000);
                return false ;
            }

    if (ValMin(min,max,"#errorAjaxSensor")){}
            else
            {
                setTimeout(function() {$('.ok,.ko').fadeOut('fast');}, 3000);
                return false ;
            }
            $.Ajax({
                type: "POST",
                url: "ModificaSensoresAjax.PHP",
                data: { tabla: 3, N:nom, D:dir, Del:del, UN:uni, U:ud, M:max, MI:min }
            })
            .done(function( msg ) {

                $('#editaSensor tbody').remove();
                salida2="<tbody>"
                $.Ajax({
                    type: "POST",
                    url: "ModificaSensoresAjax.PHP",
                    data: { tabla: 1 }
                })
                    .done(function(JSON) {
                        $.each(JSON.datos, function(key,valor){
                            salida2+="<tr title='Edita dato'><td class='editableS' id='idS' data-
    campo='Nombre'><span>"+valor.NombreS+"</span></td><td class='editableS' data-
    campo='Direccion'><span>"+valor.DirS+"</span></td><td class='editableS' data-
    campo='Delay'><span>"+valor.DelS+"</span></td><td class='editableS' data-
    campo='Unidades'><span>"+valor.UnidadS+"</span></td><td class='editableS' data-
    campo='Ud'><span>"+valor.UdS+"</span></td><td class='editableS' data-campo='Maximo'><span
    id='MaxS'>"+valor.MaxS+"</span></td><td class='editableS' data-
    campo='Minimo'><span>"+valor.MinS+"</span></td><td data-
    campo='DatoActual'><span>"+valor.Act+"</span></td><td><button type='button' id='delSensor
    "+ key+ " ' title='Elimina Sensor' class='eliminarSensor btn btn-danger'>x</button></td></tr>";
                        });
                        salida2+="</tbody>"
                        $('#editaSensor').append(salida2);
                        if (per=="false")
                        {
                            $("#editaSensor td").removeClass("editableS");
                            $("#editaSensor button.eliminarSensor").attr("disabled", "disabled");
                        }
                    });
```

```
                $("#NombreS").val("");
                $("#DirS").val("");
                $("#DelS").val("");
                $("#UnidadS").val("");
                $("#UdS").val("");
                $("#MaxS").val("");
                $("#MinS").val("");

                $("#aciertoAjaxSensor").HTML("<span class='ok'>El Sensor "+nom+" ha sido creado
    correctamente "+ msg +"</span>");
                //td.HTML("<span>"+id+"</span>");
                setTimeout(function() {$('.ok,.ko').fadeOut('fast');}, 3000);
            })
            .fail(function(jqXHR, textStaus, errorThorown){
                $("#errorAjaxSensor").HTML("<span class='ko'>Fallo en creaci&oacute;n de sensor: "
    +textStaus + "</span>");
            });
        });

/*****************************************************************
                *Acciones al pulsar boton Actualizar tabla*
*****************************************************************/
    $(document).on("click","button.ActualizaSensor",function(e)
        {
            $('#editaSensor tbody').remove();
            salida3="<tbody>"
            $.Ajax({
                type: "POST",
                url: "ModificaSensoresAjax.PHP",
                data: { tabla: 1 }
            })
            .done(function(JSON) {
                if (JSON.vacio=="no")
                {

                    $.each(JSON.datos, function(key,valor){
                        salida3+="<tr title='Edita dato'><td class='editableS' id='idS' data-
    campo='Nombre'><span>"+valor.NombreS+"</span></td><td class='editableS' data-
    campo='Direccion'><span>"+valor.DirS+"</span></td><td class='editableS' data-
    campo='Delay'><span>"+valor.DelS+"</span></td><td class='editableS' data-
    campo='Unidades'><span>"+valor.UnidadS+"</span></td><td class='editableS' data-
    campo='Ud'><span>"+valor.UdS+"</span></td><td class='editableS' data-campo='Maximo'><span
    id='MaxS'>"+valor.MaxS+"</span></td><td class='editableS' data-
    campo='Minimo'><span>"+valor.MinS+"</span></td><td data-
    campo='DatoActual'><span>"+valor.Act+"</span></td><td><button type='button' id='delSensor
    "+ key+ " ' title='Elimina Sensor' class='eliminarSensor btn btn-danger'>x</button></td></tr>";
                    });
                    salida3+="</tbody>"
                    $('#editaSensor').append(salida3);
                    if (per=="false")
                        {
                            $("#editaSensor td").removeClass("editableS");
                            $("#editaSensor button.eliminarSensor").attr("disabled", "disabled");
                        }
                }
                else
                {
                    $("#errorAjaxSensor").HTML("<span class='ko'>No hay datos en esta tabla
```

```
        </span>");
                    }
                })
                .fail(function(jqXHR, textStaus, errorThorown){
                    $("#errorAjaxSensor").HTML("<span class='ko'>Fallo en actualizacion tabla:  "
        +textStaus + "</span>");
                });
            });

/*****************************************************************
            *Acciones al pulsar boton Puesta a cero de sensores*
*****************************************************************/

        $(document).on("click","button.Pcero",function(e)
            {
                Bpulsado=$(this).attr("id");
                //console.log(Bpulsado);
                $.Ajax({
                    type: "POST",
                    url: "ModificaSensoresAjax.PHP",
                    data: { tabla: 5, Nombre:Bpulsado}
                })
                .done(function(JSON) {
                    $("#aciertoCero").HTML("<span class='ok'>Sensor "+Bpulsado+" puesto a cero
        correctamente</span>");
                })
                .fail(function(jqXHR, textStaus, errorThorown){
                    $("#errorCero").HTML("<span class='ko'>Fallo en al poner a cero sensor :  " +textStaus
        + "</span>");
                });
                setTimeout(function() {$('.ok,.ko').fadeOut('fast');}, 3000);
                //Acciones de libreria CoAP Javascrip
            });
});
```

Código 11 - tablas2.js

tablas3.js

```
var max, min, NombreElegido;
var graficar=[]; ///array para dibujar grafica
var pulsado,datas; //datas: objeto para configurar Ajax según filtro

jQuery(function($){
   $.datepicker.regional['es'] =
                {
        closeText: 'Cerrar',
                    numberOfMonths:1,
                    minDate: new Date(2010, 5, 1),
                maxDate: new Date(2020, 12, 01),
                    showAnim: 'fadeIn',
                    showButtonPanel: true,
                    changeMonth: true,
                    changeYear: true,
        prevText: '&#x3c;Ant',
        nextText: 'Sig&#x3e;',
        currentText: 'Hoy',
        monthNames: ['Enero','Febrero','Marzo','Abril','Mayo','Junio',
        'Julio','Agosto','Septiembre','Octubre','Noviembre','Diciembre'],
        monthNamesShort: ['Ene','Feb','Mar','Abr','May','Jun',
        'Jul','Ago','Sep','Oct','Nov','Dic'],
        dayNames: ['Domingo','Lunes','Martes','Mi&eacute;rcoles','Jueves','Viernes','S&aacute;bado'],
        dayNamesShort: ['Dom','Lun','Mar','Mi&eacute;','Juv','Vie','S&aacute;b'],
        dayNamesMin: ['Do','Lu','Ma','Mi','Ju','Vi','S&aacute;'],
        weekHeader: 'Sm',
        dateFormat: 'dd-mm-yy',
        firstDay: 1,
        isRTL: false,
        showMonthAfterYear: false,
        yearSuffix: ''
                };
   $.datepicker.setDefaults($.datepicker.regional['es']);
});

/*
*/
function dibGrafico(){
        chartCPU = new Highcharts.StockChart({
                chart: {
                   renderTo: 'contenedor'
                   //defaultSeriesType: 'spline'
                   //Se puede usar tipo de grafico activo con llamada Ajax en el interior Spline updating each second
                },
             rangeSelector : {
                enabled: false
             },
             title: {
                text: 'Grafica'
             },
             xAxis: {
                type: 'datetime',
                labels: {
                   formatter: function () {
                      return Highcharts.dateFormat('%d %m %Y', this.value);
                   },
```

```
                dateTimeLabelFormats: {
                  minute: '%H:%M',
                  hour: '%H:%M',
                  day: '%e. %b',
                  week: '%e. %b',
                  month: '%b \'%y',
                  year: '%Y'
                }
              }

            },
            yAxis: {
              minPadding: 0.2,
              maxPadding: 0.2,
              title: {
                text: 'Valores',
                margin: 10
              }
            },
            series: [{
              name: 'valor',
              data: (function() {
                var data = graficar;
                return data;
              })()
            }],
            tooltip: {
              formatter: function () {
                return '<b>Dato: </b>'+Highcharts.numberFormat(this.y, 2)+'<br/>' +
                '<b>Fecha: </b>'+ Highcharts.dateFormat('%d/%m/%Y %H:%M:%S', this.x) + '<br/>'
                ;
              }
            },
          credits: {
            enabled: false
          }
          });
}
/*****************************************************************
                Valida dato de entrada
                _____________________
                    IN: Dato cualquier tipo, div para mostrar el error
                    OUT: Boolean true o false.
*****************************************************************/
function Valdato(dato,maxi,mini,div){

        maxInt=parseInt(maxi);
        minInt=parseInt(mini);
        datoInt=parseInt(dato);
        if(!((datoInt <= maxInt)&&(datoInt>=minInt))){
                $(div).fadeIn();
            $(div).HTML("<span class='ko'>M&iacute;nimo no puede estar vacio, debe estar comprendido" +
        maxInt + " y " + minInt + " </span>");
            return false;
          }


          //en otro caso, el mensaje no se muestra
```

```
        else{
            $(div).fadeOut();
            return true;
        }
}

function muestraTabla(){
                    graficar=[];
                    per=$('#permiso').val();
                    $("#buscando").HTML("<spam><img src='../img/Ajax-loader.gif'></spam>");
                    $('#editaDatos tbody').remove();
                    $('#static tbody').remove();

                    $.Ajax({
                        type: "POST",
                        url: "ModificaDatosAjax.PHP",
                        data: datas
                    })
                    .done(function(JSON) {
                        if (JSON.vacio=="si")
                        {
                            $("#Quefiltro p").HTML("No existen datos para el filtro elegido");
                        }
                        else
                        {
                            $('#contDatos').fadeIn();
                            $('#contCabe').fadeIn();
                            $('#static').fadeIn();
                            $('#contGraf').fadeIn();
                            max=JSON.max;
                            min=JSON.min;
                            Ud=JSON.ud;
                            Unidades=JSON.uni;
                            $("#Unidades").HTML("<span> Unidades del sensor expresadas en
        "+Unidades+"</span>")
                            salida="<tbody>";
                            $.each(JSON.datos, function(key,valor){
                                date= new Date(valor.FechaZulu);
                                var anio=parseInt(date.getFullYear())

                graficar.push([(Date.UTC(date.getUTCFullYear(),date.getUTCMonth(),date.getUTCDate(),date.ge
        tUTCHours(),date.getUTCMinutes(),date.getUTCSeconds())),parseInt(valor.Valor)]);
                                salida+="<tr title='Edita dato'><td style='display:none;' id='id' data-
        campo='Id'><span>"+valor.Id+"</span></td><td class='editable' data-
        campo='Dato'><span>"+valor.Valor+"</span></td><td ><span>"+Ud+"</span></td><td id='fecha'
        data-campo='Fecha_hora'><span>"+valor.Fecha+"</span></td><td><a class='eliminarDato enlace
        elimina' id='"+ $('#NombreElegido').val() +"' title='Elimina Dato' href='#'>Eliminar</a></td></tr>";

                            });
                            salida+="</tbody>"
                            $('#editaDatos').append(salida);
                            if (per=="false")
                            {
                                $("#editaDatos td").removeClass("editable");
                                $("#editaDatos a.eliminarDato").hide();
                            }

                            dibGrafico();
```

```
                        $.each(JSON.estatic, function(key,valor){
                            $('#static').append(

                "<tbody><tr><th>Media</th><td>"+valor.AVG+"</td></tr><tr><th>M&aacute;ximo</th><td>"
            +valor.MAX+"</td></tr><tr><th>M&iacute;nimo</th><td>"+valor.MIN+"</td></tr><tr><th>Desviaci
            &oacute;n
            tipica</th><td>"+valor.DES+"</td></tr><tr><th>Varianza</th><td>"+valor.VAR+"</td></tr></tbody
            >");
                        });
                    }
                    $("#buscando").HTML("");
                    tamano=parseInt(JSON.tam);
                })
                .fail(function(jqXHR, textStaus, errorThorown){
                    $("#falloDatoAjax").HTML("<span class='ko'>Fallo en actualización de tabla: " +textStaus +
            "</span>");
                })
                .always(function() { //Cuando termina el proceso de mostrar datos, devuelve numero de datos
            mostrados

                $("#Quefiltro p span").HTML(tamano);
                });

}

$(document).ready(function()
{
        $('#contDatos').hide();
        $('#contCabe').hide();
        $('#static').hide();
        $('#contGraf').hide();


        NombreElegido = $('#NombreElegido').val();


        $("#FechaIni").datepicker({
                onClose: function (selectedDate) {
                    $("#FechaFin").datepicker("option", "minDate", selectedDate);
                    $("#FechaFin").datepicker("option", "defaultDate", selectedDate);
                }
        });

        $("#FechaFin").datepicker({
                onClose: function (selectedDate) {
                    $("#FechaIni").datepicker("option", "maxDate", selectedDate);
                }
        });


        $(document).on("click","#rangoFecha",function(e)
                {

                    pulsado="rangoFecha";
                    datas={ tabla: 1, Nombre: NombreElegido, filtro: pulsado, FechaIni:$("#FechaIni").val(),
                FechaFin:$("#FechaFin").val()}
                    muestraTabla();
                    $("#Quefiltro p").HTML("Mostrando datos desde " +$("#FechaIni").val() + " hasta "+
```

```
$("#FechaFin").val()+"(<span></span> datos mostrados)");

        });

$(document).on("click","#MuestraTodo",function(e)
        {
            pulsado="todo";
            datas={ tabla: 1, Nombre: NombreElegido, filtro: pulsado};
            muestraTabla();
            $("#Quefiltro p").HTML("Mostrando todos los datos guardados (<span></span> datos
mostrados)");
        });

/*****************************************************************
                *Editamos Valores unitarios de tabla Dato
/*****************************************************************/
        var td,campo,valor,Id,fecha;
        $(document).on("click","td.editable span",function(e)
        {
            e.preventDefault();
            $("td").removeClass("editable");
            td=$(this).closest("td");
            campo=$(this).closest("td").data("campo");
            valor=$(this).text();
            Id=$(this).closest("tr").find("#id").text();
            fecha=$(this).closest("tr").find("#fecha").text();
            td.text("").HTML("<input title='Edita dato' type='text' name='"+campo+"'
value='"+valor+"'><a class='enlace guardar' href='#'>Guardar</a><a class='enlace cancelar'
href='#'>Cancelar</a>");
        });

/*****************************************************************
            *Acciones al pulsar icono cancelar*
*****************************************************************/
        $(document).on("click",".cancelar",function(e)
        {
            e.preventDefault();
            td.HTML("<span>"+valor+"</span>");
            $("td").addClass("editable");
        });


/*****************************************************************
                *Acciones al pulsar icono guardar*
/*****************************************************************/
        $(document).on("click",".guardar",function(e)
        {
            $("#aciertoDatoAjax").HTML("<img src='../img/loading.gif'>");
            e.preventDefault();
            nuevovalor=$(this).closest("td").find("input").val();
            /*********************************************
                    VALIDAMOS DATOS INTRODUCIDOS
            *********************************************/
            //Sacamos la columna en la que estamos para saber que dato validamos
            switch ($(this).closest("td").prevAll().length)
            {
case 1: //dato
  {
```

```
        if (Valdato(nuevovalor,max,min,"#falloDatoAjax")){}
                else
                {
                                setTimeout(function() {$('.ok,.ko').fadeOut('fast');}, 3000);
                                return false ;
                        }
                break;
          }
        }

                    $.Ajax({
                        type: "POST",
                        url: "ModificaDatosAjax.PHP",
                        data: { tabla: 3, campo: campo, valor: nuevovalor, Id: Id, Fecha: fecha }
                    })
                    .done(function( msg ) {
                        $("#aciertoDatoAjax").HTML(msg);
                        td.HTML("<span>"+nuevovalor+"</span>");
                        $("td").addClass("editable");
                        setTimeout(function() {$('.ok,.ko').fadeOut('fast');}, 3000);
                    })
                    .fail(function(jqXHR, textStaus, errorThorown){
                        $("#falloDatoAjax").HTML("<span class='ko'>Fallo al guardar datos: " +textStaus +
        "</span>");
                    });
                });

        /*****************************************************************
                        *Acciones al pulsar el boton eliminar dato*
        *****************************************************************/
                $(document).on("click","a.eliminarDato",function(e)
                {
                    e.preventDefault();
                    id=$(this).closest("tr").find("#id").text();
                    $.Ajax({
                        type: "POST",
                        url: "ModificaDatosAjax.PHP",
                        data: { tabla: 2 , ID: id }
                    })
                    .done(function( msg ) {
                        $("#buscando2").HTML("<spam><img src='../img/Ajax-loader.gif'></spam>");
                        muestraTabla();
                        $("#aciertoDatoAjax").HTML("<span class='ok'>El Dato "+id+" ha sido eliminado
        correctamente "+ msg +"</span>");

                        setTimeout(function() {$('.ok,.ko,#buscando2').fadeOut('fast');}, 3000);
                    })
                    .fail(function(jqXHR, textStaus, errorThorown){
                        $("#falloDatoAjax").HTML("<span class='ko'>Fallo en Eliminacion de dato: " +textStaus +
        "</span>");
                    });
                });



/*****************************************************************
                        *Acciones al pulsar boton Actualizar tabla*
*****************************************************************/
```

```
        $(document).on("click","button.ActualizaDatos",function(e)
                {
                    $('#buscando2').fadeIn();
                    $("#buscando2").HTML("<spam><img src='../img/Ajax-loader.gif'></spam>");
                    muestraTabla();
                    $("#buscando2").HTML("");
                });
});
```

Código 12 - tablas3.js

PHP

Vistas

Index.*PHP*

```
<?PHP
  session_start();

  require_once("./PHP/class/PaqClass.PHP");
  $paquete = new Paq();
?>
<!DOCTYPE HTML>
<HTML>
<head>
 <meta charset="UTF-8">
        <title> Sistema de Sensores </title>
        <link href="CSS/bootstrap.min.CSS" rel="stylesheet" media="screen">
        <link href="CSS/otros.CSS" rel="stylesheet" media="screen">
  <link href="CSS/business-casual.CSS" rel="stylesheet">
 <!-- Fonts -->
 <link
        href="http://fonts.googleapis.com/CSS?family=Open+Sans:300italic,400italic,600italic,700italic,80
        0italic,400,300,600,700,800" rel="stylesheet" type="text/CSS">
 <link
        href="http://fonts.googleapis.com/CSS?family=Josefin+Slab:100,300,400,600,700,100italic,300itali
        c,400italic,600italic,700italic" rel="stylesheet" type="text/CSS">
 <script type="text/JavaScript" src="js/jquery-1.11.1.min.js"> </script>
 <script type="text/JavaScript" src="js/tablas.js"> </script>
        <script type="text/JavaScript" src="js/bootstrap.min.js"> </script>
 <meta name="viewport" content="width=device-width, initial-scale=1">
</head>
<body class="container" onunload=stopWorker();>
<div class="navbar navbar-inverse">
     <a class="navbar-brand" href="#">SCADA-Web</a>
     <ul class="nav navbar-nav">
                <li class="active"><a href="#">Inicio</a></li>
          <li> <a href="PHP/datos.PHP">Sensores</a> </li>
          <li> <a href="PHP/admin.PHP">Administrar</a> </li>
     </ul>
</div>
 <!-- <span class="CirculoVerde"> </span>
 <span class="CirculoRojo"> </span>
 <span class="CirculoAmarillo"> </span>
 <span class="CirculoNaranja"> </span>  -->
<div class="row">
  <div class="col-xs-12 col-sm-12 col-md-4 jumbotron ">
     <h1>Datos Actuales</h1>
     <p>Datos Actuales de sensores activos</p>
     <p>Nos muestra visualmente las alarmas en los datos leidos </p>
     <p><span class="CirculoVerde"> </span> Nivel Normal</p>
     <p><span class="CirculoRojo"> </span> Nivel Anomalo</p>
  </div>
<div id="errorAjaxDatos"></div>
<div class="col-xs-12 col-sm-12 col-md-8 "><!--table-responsive -->
   <div id="errorWorker"></div>
   <div class="table">
```

```
    <table id="editaDatos" class="table table-condensed">
     <caption>Lista de Sensores existentes</caption>
       <thead>
        <tr>

         <th>
           <button type="button"  class="ActualizaDatos btn btn-primary ">
            <span class="glyphicon glyphicon-refresh ">Actualizar</span>
           </button>
          </th>
          <th>
           <button type="button" class="ParaWorker btn btn-primary">
            <span class="glyphicon glyphicon-stop">Parar Autorefresco</span>
           </button>
          </th>
          <th>
           <button type="button" class="EmpiezaWorker btn btn-primary">
            <span class="glyphicon glyphicon-play ">Iniciar Autorefresco</span>
           </button>
          </th>
         </tr>
        <tr>
          <th> Nombre Sensor </th>
          <th> Valor Actual</th>
          <th> Alarmas</th>
        </tr>
        </thead>
        <tfoot>
        </tfoot>
        <tbody>
        </tbody>
    </table>
   </div>
  </div>
 </div>
</body>
</HTML>
```

Código 13 - Index.*PHP*

Datos.*PHP*

```
<?PHP
   session_start();
$menuseleccionado = 2;
?>
<!DOCTYPE HTML>
<HTML>
<head>
   <meta charset="UTF-8">
   <title> Sensores </title>
   <link href="../CSS/bootstrap.min.CSS" rel="stylesheet" media="screen">
   <link href="../CSS/otros.CSS" rel="stylesheet" media="screen">
    <link href="../CSS/business-casual.CSS" rel="stylesheet">
    <link href="../CSS/ui-lightness/jquery-ui-1.10.4.custom.CSS" rel="stylesheet">

   <!-- Fonts -->
   <link
        href="http://fonts.googleapis.com/CSS?family=Open+Sans:300italic,400italic,600italic,700italic,80
        0italic,400,300,600,700,800" rel="stylesheet" type="text/CSS">
   <link
        href="http://fonts.googleapis.com/CSS?family=Josefin+Slab:100,300,400,600,700,100italic,300itali
        c,400italic,600italic,700italic" rel="stylesheet" type="text/CSS">

   <script type="text/JavaScript" src="../js/jquery-1.11.1.min.js"> </script>
   <script type="text/JavaScript" src="../js/jquery-ui-1.10.4.custom.min.js"></script>
   <script type="text/JavaScript" src="../js/highchart/highstock.js"></script>
   <script type="text/JavaScript" src="../js/highchart/exporting.js"></script>
   <script type="text/JavaScript" src="../js/tablas3.js"> </script>
   <script type="text/JavaScript" src="../js/bootstrap.min.js"> </script>
   <meta name="viewport" content="width=device-width, initial-scale=1">
</head>
<body class="container">
   <div class="navbar navbar-inverse">
      <a class="navbar-brand" href="#">SCADA-Web</a>
      <ul class="nav navbar-nav">
           <li><a href="../index.PHP">Inicio</a></li>
           <li class="active"> <a href="datos.PHP">Sensores</a> </li>
           <li > <a href="admin.PHP">Administrar</a> </li>
      </ul>
   </div>

<?PHP
require_once("./class/PaqClass.PHP");
date_default_timezone_set('Europe/Madrid');
//Creamos un objeto de la clase paquete
$paquete = new Paq();
$Elegido=$_GET['eleccion'];

if ($Elegido != "Si" )
   {
?>
      <div class="row">
        <div class="col-xs-12 col-sm-6 col-md-8 jumbotron ">
           <h1>Lista de sensores</h1>
           <p>Aqui podemos ver los sensores dados de alta en nuestro sistema.</p>
           <p>Pulsando sobre ellos veremos el historico de los datos. </p>
```

```
        </div>

        <div class="col-xs-12 col-sm-6 col-md-4 panel panel-default">
          <ul class=" nav nav-list ">

<?PHP
          $ArrNomSen=$paquete->NombreSensor();
          for($i = 0 ;$i<count($ArrNomSen);$i++)
          {

             echo '<li><a
        href="datos.PHP?submit='.$ArrNomSen[$i][0].'&eleccion=Si">'.$ArrNomSen[$i][0].'</li>';

          }
      echo '</ul></div></div>';
    }//Fin if

else
   {
     $Nombre=trim($_GET['submit']);
     echo '<input id="NombreElegido" type="hidden" value="'.$Nombre.'">';
     echo'<div class="col-xs-12 col-sm-12 col-md-12 ">';
     echo '<ul class="nav nav-tabs">';
     $ArrNomSen=$paquete->NombreSensor();
     for($i = 0 ;$i<count($ArrNomSen);$i++)
        {
          if ($ArrNomSen[$i][0]==$Nombre){

             echo '<li class="active"><a
        href="datos.PHP?submit='.$ArrNomSen[$i][0].'&eleccion=Si">'.$ArrNomSen[$i][0].'</a></li>';

          }
          else{
             echo '<li><a
        href="datos.PHP?submit='.$ArrNomSen[$i][0].'&eleccion=Si">'.$ArrNomSen[$i][0].'</a></li>';

          }

        }
     echo '</ul>';
     echo '</div>';
?>

   <div class="row">
        <div class="col-xs-12 col-sm-12 col-md-4 jumbotron">
          <h2>Datos Historicos</h2>
          <p></br></br></p>
          <p>Vemos Datos historicos de sensor <strong><?PHP echo $Nombre ?></strong> </p>
          <p>La grafica muestra los datos elegidos en los filtros de manera visual. </p>
          <p>Se pueden hacer estudios de los datos atraves de lo que se visualiza en esta pagina.</p>
          <p><br></br></p>
        </div>

<?PHP

        if (($_SESSION['per'] == "99") && ($_SESSION['aut']==1))
        {
          echo '<input id="permiso" type="hidden" value="true">';
```

```
        }
        else
        {
          echo '<input id="permiso" type="hidden" value="false">';
        }
?>
    <div class="col-xs-12 col-sm-12 col-md-8">
      <div class="alert alert-info" role="alert">
        <h5>Modificar Datos</h5>
        <p></br></p>
        <p>Para modificar datos del sensor <strong><?PHP echo $Nombre ?></strong> navega por la
      tabla, ponte encima del dato a modificar y haz click con el rat&oacute;n.</p>
        <p>Para eliminar datos, pulsa el boton rojo del dato a eliminar.</p>
        <p>Para ello tienes que autentificarte como administrador en el apartado Administrar</p>
      </div>
      <div class="modal-dialog">
        <div class="modal-content">
          <div class="modal-header">

            <h4 class="modal-title" id="myModalLabel">Filtro de datos</h4>
            <br>
            <div id="buscando" class="col-xs-7 col-xs-offset-5 col-sm-7 col-sm-offset-5 col-md-3 col-md-
      offset-5"></div>
            <br>
          </div>
          <div class="modal-body">
            <p>Pulsa alg&uacute;n filtro para ver los datos</p>
          </div>
          <div class="modal-footer">
            <label for="MuestraTodo">Pulsa para ver todos los datos</label>
            <button id="MuestraTodo" class="btn btn-primary">Mostrar Todo</button>
          </div>
          <div class="modal-footer">
            <label for="FechaIno">Elige rango de fechas</label>
            <input name="FechaIni" id="FechaIni" type="text" readonly placeholder="Fecha Inicio"/>
            <input name="FechaIni" id="FechaFin" type="text" readonly placeholder="Fecha Fin"/>
            <button id="rangoFecha" class="btn btn-primary">Mostrar</button>
          </div>
        </div>
      </div>

      <div id="Quefiltro" class="alert alert-info" role="alert">
        <p></p>
      </div>

      <div class="table">
        <table id="static" class="table">
          <caption>Datos Estad&iacute;sticos</caption>
          <thead>
          </thead>
          <tbody>
          </tbody>
        </table>
      </div>

      <div id="aciertoDatoAjax" class="col-xs-7 col-xs-offset-2 col-sm-7 col-sm-offset-2 col-md-7 col-md-
      offset-3"></div>
      <div id="falloDatoAjax"></div>
```

```
        <div id="contCabe" class="table">

        </div>
        <div  id="contDatos" class="panel panel-default">
          <div  class="panel-body">
            <table class=" table table-hover" >
              <caption>Datos Sensor</caption>
              <thead>
                  <tr>
                    <th>
                      <button type="button" class="ActualizaDatos btn btn-primary ">
                        <span class="glyphicon glyphicon-refresh ">Actualizar</span>
                      </button>
                      <div id="buscando2" class="col-xs-7 col-xs-offset-5 col-sm-7 col-sm-offset-5 col-md-3
       col-md-offset-2"></div>
                    </th>
                     <th> </th>
                    <th id="Unidades"></th>
                  </tr>
                  <tr>
                    <th width='17%'> Valor </th>
                    <th width='10%'> Ud </th>
                    <th width='50%'> Fecha </th>
                    <th width='15%'> Del </th>
                  </tr>
                </thead>
                <tbody >
                </tbody>
              </table>
            <div  class="scroll">
         <!--   <div class="three-columns"> -->
              <table id="editaDatos" class=" table table-hover" >
                <thead>
                </thead>
                <tbody>
                </tbody>
              </table>
            </div>
          </div>
        </div>

        <div id="contGraf">
          <div ><table><caption>Gr&aacute;fica</caption></table></div>
          <div id="contenedor"></div>
        </div>
      </div>
    </div>
<?PHP
  }
?>
</body>
</HTML>
```

Código 14 - datos.*PHP*

Admin.*PHP*

```
<?PHP
  session_start();
  require 'db.ini.PHP';
  require_once("./class/PaqClass.PHP");
  $paquete = new Paq();
  $menuseleccionado = 3;
?>

<!DOCTYPE HTML>
<HTML>
<head>
   <meta charset="UTF-8">
        <title> Administracion de sistema </title>
   <link href="../CSS/bootstrap.min.CSS" rel="stylesheet" media="screen">
   <link href="../CSS/otros.CSS" rel="stylesheet" media="screen">
    <link href="../CSS/business-casual.CSS" rel="stylesheet">
   <!-- Fonts -->
   <link
        href="http://fonts.googleapis.com/CSS?family=Open+Sans:300italic,400italic,600italic,700italic,80
        0italic,400,300,600,700,800" rel="stylesheet" type="text/CSS">
   <link
        href="http://fonts.googleapis.com/CSS?family=Josefin+Slab:100,300,400,600,700,100italic,300itali
        c,400italic,600italic,700italic" rel="stylesheet" type="text/CSS">

   <script type="text/JavaScript" src="../js/jquery-1.11.1.min.js"> </script>
   <script type="text/JavaScript" src="../js/tablas2.js"> </script>
   <script type="text/JavaScript" src="../js/bootstrap.min.js"> </script>
   <meta name="viewport" content="width=device-width, initial-scale=1">
</head>
<body class="container">
  <div class="navbar navbar-inverse">
   <a class="navbar-brand" href="#">SCADA-Web</a>
      <nav>
        <ul class="nav navbar-nav">
          <li <?PHP if ($menuseleccionado == 1){?> class="active"<?PHP }?> > <a href="../index.PHP"
        >Inicio</a></li>
          <li <?PHP if ($menuseleccionado == 2){?> class="active"<?PHP }?> > <a href="./datos.PHP"
        >Sensores</a> </li>
          <li <?PHP if ($menuseleccionado == 3){?> class="active"<?PHP }?> > <a href="./admin.PHP"
        >Administrar</a> </li>
        </ul>
      </nav>
  </div>
<?PHP

$Elegido=$_POST['eleccion'];
$permiso=-1;
if (($Elegido == "Si") || ($_SESSION['aut']==1))
{
   if(isset($_SESSION['user']))
   {
      if ($_SESSION['user']=="")
      {
         $_SESSION['user']=$_POST['user'];
         $_SESSION['pass']=$_POST['pass'];
```

```
    }
    else
    {
      $user=$_SESSION['user'];
      $pass=$_SESSION['pass'];
    }
  }
  else
  {
    $_SESSION['user']=$_POST['user'];
    $_SESSION['pass']=$_POST['pass'];
  }
  $query='SELECT * FROM Sensores.Admin';
  $ArrayDatos= $paquete->getArraySQL($query);
  if($ArrayDatos!=null)
  {
    foreach($ArrayDatos as $row)
    {
      if (($row[0] == $_SESSION['user']) && ($row[1] == $_SESSION['pass']))
      {
        $_SESSION['aut']=1;
        $permiso=$row[2];
        $_SESSION['per']=$permiso;
        break;
      }
      else
      {
        $_SESSION['aut']=0;
      }
    }
  }
  else
   {
    $_SESSION['aut']=0;
   }

  if (($_SESSION['aut']==1) && ($_SESSION['fallo']<3))
  {
    $_SESSION['fallo']=0;
    if ($_SESSION['per'] == "99")
    {
      echo '<input id="permiso" type="hidden" value="true">';
    }
    else
    {
      echo '<input id="permiso" type="hidden" value="false">';
    }
?>
    <div class="row">
      <div class="col-xs-offset-3 col-xs-7 col-sm-offset-5 col-sm-7 col-md-offset-5 col-md-7">
        <form title="Cerrar Sesion" action="GestiomForm.PHP" method="post">
          <input class="btn btn-primary" type="submit" name="submit" value="Cierra Sesion"/>
        </form>
      </div>
    </div>
    <div class="alert alert-info">
     <div class="row">
      <p>&amp;nbsp;&amp;nbsp;Enciende/Apaga Proxy de captaci&amp;oacute;n de datos.</p>
```

```
        <div class="col-xs-6 col-sm-offset-3 col-sm-2  col-md-offset-4 col-md-2" >
          <form title="Enciende" action="GestiomForm.PHP" method="post">
<?PHP
                if ($permiso==99)
                {
?>
                  <input class="btn btn-success" type="submit" name="submit" value="Enciende Proxy"/>
<?PHP
                }else{
?>
                  <input class="btn btn-success" disabled="disabled" type="submit" name="submit"
        value="Enciende Proxy"/>
<?PHP
                }
?>
          </form>
        </div>
        <div class="col-xs-6 col-sm-offset-1 col-sm-3 col-md-2">
          <form title="Apaga" action="GestiomForm.PHP" method="post">
<?PHP
                if ($permiso==99)
                {
?>
                  <input class="btn btn-danger" type="submit" name="submit" value="Apaga Proxy"/>
<?PHP
                }else{
?>
                  <input class="btn btn-danger" disabled="disabled" type="submit" name="submit"
        value="Apaga Proxy"/>
<?PHP
                }
?>
          </form>
        </div>
       </div>
      </div>

   <div class="table">
     <table class="table">
       <caption>Estado de comunica&oacute;n con Proxy</caption>
       <thead>
         <th>Id</th>
         <th>Modifica</th>
         <th>Enciende</th>
       </thead>
       <tbody>
<?
         $query='SELECT * FROM Sensores.JavaPHP';
         $ArrayDatos= $paquete->getArraySQL($query);
         if($ArrayDatos!=null)
         {
           foreach($ArrayDatos as $row)
           {
             echo '<tr>';
               foreach($row as $value)
               {
                 echo '<td>' . $value . '</td>';
               }
```

```
                echo '</tr>';
            }
        }
?>
        </tbody>
    </table>
  </div>
  <div class="alert alert-info">
   <div class="row">
    <p>  Modificar BBDD</p>
    <div class="col-xs-5 col-sm-4  col-md-3">
      <form title="EliminaBBDD" action="GestiomForm.PHP" method="post">
<?PHP
          if ($permiso==99)
          {
?>
            <input class="btn btn-primary" type="submit" name="submit" value="Elimina/Crea BBDD"/>
<?PHP
          }else{
?>
            <input class="btn btn-primary" disabled="disabled" type="submit" name="submit"
        value="Elimina/Crea BBDD"/>
<?PHP
          }
?>
        </form>
    </div>
    <div class="col-xs-5 col-sm-4  col-md-3">
      <form title="CrearBBDD" action="GestiomForm.PHP" method="post">
<?PHP
          if ($permiso==99)
          {
?>
            <input class="btn btn-primary" type="submit" name="submit" value="Crea Sensores"/>
<?PHP
          }else{
?>
            <input class="btn btn-primary" disabled="disabled" type="submit" name="submit" value="Crea
        Sensores"/>
<?PHP
          }
?>
        </form>
    </div>
    <div class="col-xs-5 col-sm-4  col-md-3" hidden="hidden">
      <form title="CreaDatos" action="GestiomForm.PHP" method="post">
<?PHP
          if ($permiso==99)
          {
?>
            <input class="btn btn-primary" type="submit" name="submit" value="Crea Datos"/>
<?PHP
          }else{
?>
            <input class="btn btn-primary" disabled="disabled" type="submit" name="submit" value="Crea
        Datos"/>
<?PHP
          }
```

```
?>
        </form>
      </div>
      <div class="col-xs-5 col-sm-4  col-md-3" hidden="hidden">
        <form title="CreaActuales" action="GestiomForm.PHP" method="post">
<?PHP
          if ($permiso==99)
          {
?>
            <input class="btn btn-primary" type="submit" name="submit" value="Crea Actuales"/>
<?PHP
          }else{
?>
            <input class="btn btn-primary" disabled="disabled" type="submit" name="submit" value="Crea
        Actuales"/>
<?PHP
          }
?>
        </form>
      </div>
    </div>
  </div>
  <div>
<?PHP
    if ($permiso==99)
    {
?>      <div id="errorCero"></div>
        <div id="aciertoCero"></div>
        <div class="alert alert-info">
          <div class="row">
            <p>  Puesta a cero de sensores</p>
<?PHP
          $ArrNomSen=$paquete->NombreSensor();
          for($i = 0 ;$i<count($ArrNomSen);$i++)
          {
            echo '<div class="col-xs-6 col-sm-4  col-md-3">';
            echo '<button class="Pcero btn btn-primary"
        id="'.$ArrNomSen[$i][0].'">'.$ArrNomSen[$i][0].'</button></div>';
          }
?>
          </div>
        </div>
  </div>

        <div class="table">
          <table class="table">
            <caption>Lista BBDD existentes</caption>
            <thead>
              <th>Id</th>
              <th>BBDD</th>
            </thead>
            <tbody>
<?PHP
            //Mostrar BBDD
              $query='SHOW databases';
              $i=1;
              $ArrayDatos= $paquete->getArraySQL($query);
              if($ArrayDatos!=null)
```

```
            {
              foreach($ArrayDatos as $row)
              {
                echo '<tr>';
                foreach($row as $value)
                {
                  echo '<td>' . $i . '</td>';
                  echo '<td>' . $value . '</td>';
                }
                echo '</tr>';
                $i++;
              }
            }
?>
            </tbody>
          </table>
        </div>
      </div>

<?PHP
    }

    if ($permiso==99)
    {
?>
      <div id="errorAjaxUser" ></div>
      <div id="aciertoAjaxUser"></div>
       <div class="table">
         <table class="table" id="editaUser">
            <caption>Usuarios registrados</caption>
            <thead>
              <tr>
                <th>
                  <button type="button" class="ActualizaUser btn btn-primary ">
                    <span class="glyphicon glyphicon-refresh ">Actualizar</span>
                  </button>
                </th>
              </tr>
              <tr>
                <th>Usuarios</th>
                <th>Password</th>
                <th>Privilegios</th>
                <th>Del</th>
              </tr>
            </thead>
            <tfoot>
              <tr>
                <td>
                  <input class="form-control" type="text" id= "NuevoUser" name="NuevoUser"
      placeholder="Usuario"/>
                </td>
                <td>
                  <input class="form-control" type="password" id= "NuevoPass" name="NuevoPass"
      placeholder="Contrase&ntilde;a"/>
                </td>
                <td>
                  <input class="form-control" type="text" id= "NuevoPriv" name="NuevoPriv"
      placeholder="Privilegios de 0 o 99"/>
```

```
                    </td>
                  </tr>
                  <tr>
                    <td>
                      <button title="Crea Usuario" id="btnCreaUser" class="btn btn-primary "> Crea Usuario
        </button>
                    </td>
                  </tr>
                </tfoot>
                <tbody>
                </tbody>
              </table>
            </div>
<?PHP
    }
?>
  <div id="errorAjaxSensor" ></div>
  <div id="aciertoAjaxSensor"></div>
  <div class="table">
   <div class="table-responsive"><!--table-responsive -->
    <table class="table table-condensed" id="editaSensor">
      <caption>Lista de Sensores existentes</caption>
      <thead>
        <tr>
          <th>
            <button type="button" class="ActualizaSensor btn btn-primary form-control">
              <span class="glyphicon glyphicon-refresh ">Actualizar</span>
            </button>
          </th>
        </tr>
        <tr>
          <th> Nombre Sensor </th>
          <th> Direccion </th>
          <th> Intervalo </th>
          <th> Unidades </th>
          <th> Ud. </th>
          <th> Valor Maximo </th>
          <th> Valor Minimo </th>
          <th> Valor Actual</th>
          <th> del</th>
        </tr>
      </thead>
      <tfoot>
        <tr>
          <td>
            <input class="form-control" type="text" name="NombreS" id="NombreS"
        placeholder="Nombre">
          </td>
          <td>
            <input class="form-control" type="text"  name="DirS" id="DirS" placeholder="Direccion ">
          </td>
          <td>
            <input class="form-control" type="text"  name="DelS" id="DelS" placeholder="Intervalo ">
          </td>
          <td>
            <input class="form-control" type="text"  name="UnidadS"id="UnidadS"
        placeholder="Unidades ">
          </td>
```

```
            <td>
              <input class="form-control" type="text"  name="UdS" id="UdS" placeholder="Ud ">
            </td>
            <td>
              <input class="form-control" type="text"  name="MaxS" id="MaxS" placeholder="Maximo">
            </td>
            <td>
              <input class="form-control" type="text"  name="MinS" id="MinS" placeholder="Minimo">
            </td>
            <td >
            </td>
          </tr>
          <tr>
            <td>
<?PHP
              if ($permiso==99)
              {
?>
                <input  class="CreaSensor btn btn-primary form-control" value="Crear Sensor">
<?PHP
              }
              else
              {
?>
                <input  class="CreaSensor btn btn-primary form-control" disabled="disable" value="Crear
        Sensor">
<?PHP
              }
?>
            </td>
          </tr>
        </tfoot>
        <tbody>
        </tbody>
      </table>
    </div>
  </div>
<?PHP
  }
  else
  {
    echo"<script language='JavaScript'>window.location='./admin.PHP';</script>";
  }
}
else if ($_SESSION['fallo']<3)
{
?>
 <div class="modal-dialog">
  <div class="modal-content">
   <div class="modal-header">
    <h4 class="modal-title" id="myModalLabel">Identificate</h4>
<?PHP
    $_SESSION['user']="";
    $_SESSION['pass']="";
    if(!isset($_SESSION['fallo']))
    {
      $_SESSION['fallo']=0;
    }
```

```
    else
    {
      $_SESSION['fallo']+=1;
      echo "<span class='ko'>Fallo n&uacute;mero ". $_SESSION['fallo']." en la
       autentificaci&oacute;n</span>";
    }
    if ($_SESSION['fallo']==3)
    {
      echo "<span class='ko'>N&uacute;mero m&aacute;ximo de fallos superado.<span>";
       echo"<script language='JavaScript'>window.location='./admin.PHP';</script>";
    }
?>
   </div>
   <div class="modal-body">
    <form class="form-horizontal" action="#" method="POST">
      <div class="form-group">
        <label class="col-sm-2 control-label" for="user">Usuario</label>
        <input class="form-control" type="text" name="user">
      </div>
      <div class="form-group">
        <label class="col-sm-2 control-label" for="pass">Contrase&ntilde;a</label>
        <input class="form-control" type="password" name="pass">
        <input type="hidden" name="eleccion" value="Si">
      </div>
      <input class="btn btn-primary" type="submit" value="Entrar">
    </form>
   </div>
   <div class="modal-footer">

   </div>
  </div>
 </div>
<?PHP
}
else
{
  echo "<span class='ko'>Fallo n&uacute;mero ". $_SESSION['fallo'] ." en la
       autentificaci&oacute;n</span>";
  echo "<span class='ko'>N&uacute;mero m&aacute;ximo de fallos superado.<span>";
}
?>
</body>
</HTML>
```

Código 15 - admin.*PHP*

Controladores

GestiomForm.*PHP*

```
<?PHP
require_once("./class/PaqClass.PHP");

//Creamos un objeto de la clase paquete
$paquete = new Paq();
$redirect='';

switch($_POST['submit'])
{

  case 'Crea BBDD':
    require 'CrearDb.PHP';
    $redirect='admin.PHP';
    break;

  case 'Elimina BBDD':
    $redirect='admin.PHP';
    $paquete->eliminaBBDD ();
    require 'CrearDb.PHP';
    break;

  case 'Crea Datos':
    $query='SELECT `Nombre`, `Maximo`, `Minimo` FROM `TipoSensor`;';
    $ArrayDatos= $paquete->getArraySQL($query);
    if($ArrayDatos!=null)
    {
      foreach($ArrayDatos as $row)
      {
        $paquete->insertDatos($row[0],$row[2],$row[1],20);
      }
    }
    $redirect='./admin.PHP';
    break;

  case 'Crea Sensores':
    require 'CreaSensores.PHP';
    $redirect='admin.PHP';
    break;

  case 'Cierra Sesion':
    require 'cerrarSesion.PHP';
    $redirect='./admin.PHP';
    break;

  case 'Crea Actuales':

    $query='SELECT `Nombre`,`Maximo`, `Minimo` FROM `TipoSensor`;';

    $ArrayDatos= $paquete->getArraySQL($query);

    if($ArrayDatos!=null)
    {
      foreach($ArrayDatos as $row)
```

```
        {
          $paquete->datoActual($row[0],((int)$row[2])-20,((int)$row[1])+20,20);
        }
      }
      $redirect='./admin.PHP';
      break;

    case 'Enciende Proxy':
      $Enc="1";
      $IdT=1;
      $query='UPDATE  `JavaPHP`
            SET
               `Enciende`="' . $Enc . '"
               WHERE `Id`=' . $IdT . ';';
      $paquete->ejeConsulta($query);
      $redirect='admin.PHP';
      break;

    case 'Apaga Proxy':
      $Enc="0";
      $IdT=1;
      $query='UPDATE  `JavaPHP`
            SET
               `Enciende`="' . $Enc . '"
               WHERE `Id`=' . $IdT . ';';
      $paquete->ejeConsulta($query);
      $redirect='admin.PHP';
      break;

    default:
      header('Location:admin.PHP');
}
header('Location:' . $redirect);
?>
```

Código 16 - gestiomForm.*PHP*

datos*Ajax*.*PHP*

```
<?PHP
require_once("./class/PaqClass.PHP");

//Creamos un objeto de la clase paquete
$paquete = new Paq();

        $query='SELECT `Nombre`,`DatoActual`,`Maximo`,`Minimo`,`Unidades`,`Ud` FROM TipoSensor;';
                $datos=array();
                $i=1;
                $ArrayDatos= $paquete->getArraySQL($query);

                if($ArrayDatos!=null)
          {
                $datos["vacio"]="no";
              foreach($ArrayDatos as $row)
              {
                                $datos["datos"][]=array(
                                    "Sensor"=>$row[0],
                                    "Dato"=>$row[1],
                                    "Max"=>$row[2],
                                    "Min"=>$row[3],
                                    "uni"=>$row[4],
                                    "ud"=>$row[5]
                                );
                                $i++;
                            }
                $datos["tam"]=$i;
                        }
                        else
                        {
                            $datos["vacio"]="si";
                        }
                header('Content-type: application/JSON; charset=utf-8');
                echo JSON_encode($datos,JSON_FORCE_OBJECT);

                }

        }*/
?>
```

Código 17 - datos*Ajax*.*PHP*

ModificaDatos*Ajax.PHP*

```
<?PHP
require_once("./class/PaqClass.PHP");

//Creamos un objeto de la clase paquete
$paquete = new Paq();

	switch($_POST['tabla'])
	{
	case 1: //JSON tabla
		{
		    $Nombre=$_POST['Nombre'];
		    $datos=array();
		    $i=1;
		    switch($_POST['filtro'])
		    {
		    case 'todo':
			    {

		     $query = 'SELECT Id,Dato,Fecha_hora FROM Sensores.DatoSensor WHERE
	Nombre="'.$Nombre.'"ORDER BY Fecha_hora ASC;';
				    $ArrayDatos= $paquete->getArraySQL($query);

				    if($ArrayDatos!=null)
	{
		    $datos["vacio"]="no";
		    foreach($ArrayDatos as $row)
		    {
			    $parte = explode(" ", $row[2]);
			    $valor= number_format((float)$row[1], 2, '.', '');
				    $datos["datos"][]=array(
				        "Id"=>$row[0],
				        "Valor"=>$valor,
				        "Fecha"=>$row[2],
				        "FechaZulu"=>(""."$parte[0]"."T"."$parte[1]"."Z")
				    );
				    $i++;
			    }
			    $datos["tam"]=$i;

			    $query1 = 'SELECT Maximo, Minimo, Unidades, Ud FROM
	Sensores.TipoSensor WHERE Nombre="'.$Nombre.'";';
			    $ArrayMaxMin= $paquete->getArraySQL($query1);
			    $datos["max"]=$ArrayMaxMin[0][0];
			    $datos["min"]=$ArrayMaxMin[0][1];
			    $datos["uni"]=$ArrayMaxMin[0][2];
			    $datos["ud"]=$ArrayMaxMin[0][3];

			    $query2 = 'SELECT
	AVG(Dato),MAX(Dato),MIN(Dato),STD(Dato),VARIANCE(Dato) FROM (SELECT * FROM
	Sensores.TipoSensor WHERE Nombre="'.$Nombre.'") AS j join Sensores.DatoSensor on
	j.Nombre=DatoSensor.Nombre ORDER BY Fecha_hora ASC;';
			    $ArrayMaxMin= $paquete->getArraySQL($query2);
			    $datos["estatic"][]=array(
			        "AVG"=>number_format((float)$ArrayMaxMin[0][0], 2, '.', ''),
			        "MAX"=>number_format((float)$ArrayMaxMin[0][1], 2, '.', ''),
```

```
                                "MIN"=>number_format((float)$ArrayMaxMin[0][2], 2, '.', ''),
                                "DES"=>number_format((float)$ArrayMaxMin[0][3], 2, '.', ''),
                                "VAR"=>number_format((float)$ArrayMaxMin[0][4], 2, '.', '')
                            );
                        }
                        else
                        {
                            $datos["vacio"]="si";
                        }
                        break;
                    }

                    case 'rangoFecha':
                    {

                        $a=strtotime($_POST['FechaIni']);
                        $fechaIni=new DateTime();
                        $fechaIni->setTimestamp($a);

                        $b=strtotime($_POST['FechaFin']);
                        $fechaFin=new DateTime();
                        $fechaFin->setTimestamp($b);
                        $query = 'SELECT Id,Dato,Fecha_hora FROM Sensores.DatoSensor WHERE
Nombre="'.$Nombre.'" AND Fecha_Hora BETWEEN "'.date_format($fechaIni, 'Y-m-d H:i:s').'" AND
"'.date_format($fechaFin, 'Y-m-d H:i:s').'" ORDER BY Fecha_hora ASC;';

                        $ArrayDatos= $paquete->getArraySQL($query);
                        if($ArrayDatos!=null)
        {
            $datos["vacio"]="no";
            foreach($ArrayDatos as $row)
            {
                    $parte = explode(" ", $row[2]);
                    $valor= number_format((float)$row[1], 2, '.', '');
                                $datos["datos"][]=array(
                                    "Id"=>$row[0],
                                    "Valor"=>$valor,
                                    "Fecha"=>$row[2],
                                    "FechaZulu"=>(""."$parte[0]"."T"."$parte[1]"."Z") //Formato necesario
para ie y firefox
                                );
                                $i++;
                            }
                            $datos["tam"]=$i;

                            $query1 = 'SELECT Maximo, Minimo, Unidades, Ud FROM
Sensores.TipoSensor WHERE Nombre="'.$Nombre.'";';
                            $ArrayMaxMin= $paquete->getArraySQL($query1);
                            $datos["max"]=$ArrayMaxMin[0][0];
                            $datos["min"]=$ArrayMaxMin[0][1];
                            $datos["uni"]=$ArrayMaxMin[0][2];
                            $datos["ud"]=$ArrayMaxMin[0][3];

                            $query2 = 'SELECT
AVG(Dato),MAX(Dato),MIN(Dato),STD(Dato),VARIANCE(Dato) FROM Sensores.DatoSensor WHERE
Nombre="'.$Nombre.'" AND Fecha_Hora BETWEEN "'.date_format($fechaIni, 'Y-m-d H:i:s').'" AND
"'.date_format($fechaFin, 'Y-m-d H:i:s').'";';
                            $ArrayMaxMin= $paquete->getArraySQL($query2);
```

```
                                $datos["estatic"][]=array(
                                "AVG"=>number_format((float)$ArrayMaxMin[0][0], 2, '.', ''),
                                "MAX"=>number_format((float)$ArrayMaxMin[0][1], 2, '.', ''),
                                "MIN"=>number_format((float)$ArrayMaxMin[0][2], 2, '.', ''),
                                "DES"=>number_format((float)$ArrayMaxMin[0][3], 2, '.', ''),
                                "VAR"=>number_format((float)$ArrayMaxMin[0][4], 2, '.', '')
                                );
                            }
                            else
                            {
                                $datos["vacio"]="si";
                            }
                            break;
                        }
                    }
                header('Content-type: application/JSON; charset=utf-8');
                echo JSON_encode($datos,JSON_FORCE_OBJECT);
                break;
                }
                case 2: //elimina
                {
                    $id=$_POST['ID'];
                    $query='DELETE FROM Sensores.DatoSensor WHERE Id="'. $id .'";';
                    if ($paquete->ejeConsulta($query)) echo "<span class='ok'>Valores eliminados
        correctamente.</span>";
                    else echo "<span class='ko'>Error en eliminación</span>";
                    break;
                }
                case 3: //Actualiza Datos
                {
                     $query='UPDATE  `Sensores`.`DatoSensor`
          SET
              `'. $_POST["campo"] . '`="' . $_POST["valor"] . '",
             `Fecha_hora`="' . trim($_POST["Fecha"]) . '"
             WHERE `Id`="' . $_POST["Id"] . '";';
                    if ($paquete->ejeConsulta($query)) echo "<span class='ok'>Valores modificados
        correctamente.</span>";
                    else echo "<span class='ko'>Error en actualización</span>";
                    break;
                }
        }
?>
```

Código 18 - ModificaDatos*Ajax.PHP*

ModificaSensores*Ajax.PHP*

```
<?PHP
require_once("./class/PaqClass.PHP");
//Creamos un objeto de la clase paquete
$paquete = new Paq();

        switch($_POST['tabla'])
        {
        case 1: //JSON tabla
                {
                    $query='SELECT * FROM Sensores.TipoSensor';
                    $datos=array();
                    $i=1;
                    $ArrayDatos= $paquete->getArraySQL($query);

                    if($ArrayDatos!=null)
         {
                $datos["vacio"]="no";
               foreach($ArrayDatos as $row)
               {
                                    $datos["datos"][]=array(
                                        "NombreS"=>$row[0],
                                        "DirS"=>$row[1],
                                        "DelS"=>$row[2],
                                        "UnidadS"=>$row[3],
                                        "UdS"=>$row[4],
                                        "MaxS"=>$row[5],
                                        "MinS"=>$row[6],
                                        "Act"=>$row[7]
                                    );
                                    $i++;
                                }
                    $datos["tam"]=$i;
                            }
                    else
                            {
                                $datos["vacio"]="si";
                            }
                header('Content-type: application/JSON; charset=utf-8');
                echo JSON_encode($datos,JSON_FORCE_OBJECT);
                break;
                }
                case 2: //elimina sensor
                {
                    $id=$_POST['ID'];
                    $query='DELETE FROM Sensores.TipoSensor WHERE Nombre="'. $id .'";';
                    if ($paquete->ejeConsulta($query)) echo "<span class='ok'>Valores eliminados
        correctamente.</span>";
                    else echo "<span class='ko'>Error en eliminación</span>";

                    $query2='DELETE FROM Sensores.DatoSensor WHERE Nombre="'. $id .'";';
                    ($paquete->ejeConsulta($query2));

                    $mod="1";
         $IdT=1;
        $query3='UPDATE  `JavaPHP`
```

```
                SET
                    `Modifica`="' . $mod . '"
                WHERE `Id`=' . $IdT . ';';
        if ($paquete->ejeConsulta($query3)) {}
                    else echo "<span class='ko'>Error en actualización modifica</span>";
                    break;
                }
                case 3: //crea Sensor
                {
                    $nombre=trim($_POST['N']);
                    $dir=trim($_POST['D']);
                    $delay=trim($_POST['Del']);
                    $unidades=trim($_POST['UN']);
                    $ud=trim($_POST['U']);
                    $max=trim($_POST['M']);
                    $min=trim($_POST['MI']);
                    $DatoActualS="0";
                    $query='INSERT IGNORE INTO
        Sensores.TipoSensor(Nombre,Direccion,Delay,Unidades,Ud,Maximo,Minimo,DatoActual)
          VALUES
            ("' . $nombre . '","' . $dir . '","' . $delay . '","' . $unidades. '","' . $ud . '","' . $max . '","' . $min . '","' .
        $DatoActualS . '")';
       if ($paquete->ejeConsulta($query)) echo "<span class='ok'>Valores insertados
        correctamente.</span>";
                    else echo "<span class='ko'>Error en inserción</span>";
                    $mod="1";
         $IdT=1;
        $query2='UPDATE  `JavaPHP` SET `Modifica`="' . $mod . '" WHERE `Id`=' . $IdT . ';';
        if ($paquete->ejeConsulta($query2)) {}
                    else echo "<span class='ko'>Error en actualización modifica</span>";
                    break;
                }
                case 4: //Actualiza Sensor
                {
                    $query=("UPDATE Sensores.TipoSensor set ".$_POST["campo"]."='".$_POST["valor"]."'
        WHERE Nombre='". $_POST["Nombre"] ."';");
                    if ($paquete->ejeConsulta($query))
                    echo "<span class='ok'>Valores modificados correctamente.</span>";
                    else
                            echo "<span class='ko'>Error en actualización</span>";
        $mod="1";
         $IdT=1;
        $query2='UPDATE  `JavaPHP` SET `Modifica`="' . $mod . '" WHERE `Id`=' . $IdT . ';';
        if ($paquete->ejeConsulta($query2)) {}
                    else echo "<span class='ko'>Error en actualización modifica</span>";
                    break;
                }
                case 5: //Puesta a 0 de sensor
                {
                    $nombreSensor=trim($_POST['Nombre']);
                    //llamada a sensor
                    break;
                }
        }
?>
```

Código 19 - modificaSensores*Ajax.PHP*

ModificaUser*Ajax.PHP*

```
<?PHP
require_once("./class/PaqClass.PHP");

//Creamos un objeto de la clase paquete
$paquete = new Paq();

        switch($_POST['tabla'])
        {
        case 1: //JSON tabla
                {
                    $query='SELECT * FROM Sensores.Admin order by Permisos asc';
                    $datos=array();
                    $i=1;
                    $ArrayDatos= $paquete->getArraySQL($query);

                    if($ArrayDatos!=null)
          {
            foreach($ArrayDatos as $row)
            {
                                        $datos["datos"][]=array(
                                            "User"=>$row[0],
                                            "Pass"=>$row[1],
                                            "Priv"=>$row[2],
                                        );
                                        $i++;
                                    }
                    $datos["tam"]=$i;
                            }
                header('Content-type: application/JSON; charset=utf-8');
                echo JSON_encode($datos,JSON_FORCE_OBJECT);
                break;
                }

                case 2: //elimina
                {
                    $id=$_POST['ID'];
                    $query='DELETE FROM Sensores.Admin WHERE User="'. $id .'";';
                    if ($paquete->ejeConsulta($query)) echo "<span class='ok'>Valores eliminados
        correctamente.</span>";
                    else echo "<span class='ko'>Error en eliminación</span>";
                    break;
                }
                case 3: //crea usuario
                {
                    $user=trim($_POST['U']);
                    $pass=trim($_POST['Pa']);
                    $priv=trim($_POST['p']);

                    $query='INSERT IGNORE INTO Sensores.Admin(User,Pass,Permisos)
          VALUES
            ("' . $user . '","' . $pass . '","' . $priv . '")';
                    if ($paquete->ejeConsulta($query)) echo "<span class='ok'>Valores insertados
        correctamente.</span>";
                    else echo "<span class='ko'>Error en actualización</span>";
                    break;
```

```
            }
            case 4: //Actualiza Usuario
            {
                $query=("UPDATE Sensores.Admin set ".$_POST["campo"]."='".$_POST["valor"]."' WHERE
        User='". $_POST["Usuario"] ."';");
                if ($paquete->ejeConsulta($query)) echo "<span class='ok'>Valores modificados
        correctamente.</span>";
                else echo "<span class='ko'>Error en actualización</span>";
                break;
            }

        }
?>
```

Código 20 - ModificaUser*Ajax.PHP*

CierraSesion.*PHP*

```
<?PHP
 session_start();
 // Borra todas las variables de sesión
 $_SESSION = array();
 // Borra la cookie que almacena la sesión
 if(isset($_COOKIE[session_name()]))
 {
        setcookie(session_name(), '', time() - 42000, '/');
 }
 // Finalmente, destruye la sesión
 session_destroy();
?>
```

Código 21 - CierraSesion.*PHP*

creaSensores.*PHP*

```
<?PHP
require_once("./class/PaqClass.PHP");
 $paquete = new Paq();
 $query='INSERT IGNORE INTO
        Sensores.TipoSensor(Nombre,Direccion,Delay,Unidades,Ud,Maximo,Minimo,DatoActual)
          VALUES
            ("Temperatura1","CoAP://CoAP.me:5683/large-create",20,"Grados
        Centigrados","ºC","100","0","0  ")';
$paquete->ejeConsulta($query);

$query='INSERT IGNORE INTO
        Sensores.TipoSensor(Nombre,Direccion,Delay,Unidades,Ud,Maximo,Minimo,DatoActual)
          VALUES
            ("Humedad1","CoAP://CoAP.me:5683/large-create",10,"Porcentaje","%","40","20","0")';
$paquete->ejeConsulta($query);

$query='INSERT IGNORE INTO
        Sensores.TipoSensor(Nombre,Direccion,Delay,Unidades,Ud,Maximo,Minimo,DatoActual)
          VALUES
            ("Lluvia","CoAP://CoAP.me:5683/large-create",30,"Mililitros","mm","800","1","0")';
$paquete->ejeConsulta($query);

$query='INSERT IGNORE INTO
        Sensores.TipoSensor(Nombre,Direccion,Delay,Unidades,Ud,Maximo,Minimo,DatoActual)
          VALUES
            ("Viento1","CoAP://CoAP.me:5683/large-create",1,"Metros Segundo","m/s","80","0","0")';
$paquete->ejeConsulta($query);

$query='INSERT IGNORE INTO
        Sensores.TipoSensor(Nombre,Direccion,Delay,Unidades,Ud,Maximo,Minimo,DatoActual)
          VALUES
            ("Direccion_Viento","CoAP://CoAP.me:5683/large-create",1,"Grados","º","100","10","0")';
$paquete->ejeConsulta($query);

?>
```

Código 22 - creaSensores.*PHP*

Modelo

Db.ini.*PHP*

```
<?PHP
define('MYSQL_HOST','localhost');
define('MYSQL_USER','davsel');
define('MYSQL_PASS','contraseña');
define('MYSQL_USERR','root');
define('MYSQL_PASSR','contraseñaRoot');
define('MYSQL_DB','Sensores');
define('MYSQL_DBM','MySQL');
?>
```

Código 23 - Db.ini.*PHP*

PaqClass.*PHP*

```
<?PHP
require ('db.ini.PHP');
date_default_timezone_set('Europe/Madrid');
class Paq{
  public $IDr = 0 ;
  function compruebaBBDD()
  {
    //Comprobamos que existe la BBDD
    $SQL='SHOW databases';
    $conexion=NULL;  //variable que guarda la conexión de la base de datos
    $conexion = MySQLi_connect(MYSQL_HOST, MYSQL_USERR, MYSQL_PASSR);
    if(!$result = MySQLi_query($conexion, $SQL)) die("<span class='ko'>ooh, Fallo en consulta
       compruebaBBDD</span>");
     //guardamos en un array multidimensional todos los datos de la consulta
    $i=0;
    $ok=False;
    while($row = MySQLi_fetch_assoc($result))
    {
      foreach ($row as $value)
      {

        if ($value==MYSQL_DB)
          $ok=true;
      }
      $i++;
    }

    $this->descnectarBD($conexion);
    if ($ok)
      return true;
    else
      return false;
  }
  //Función que crea y devuelve un objeto de conexión a la base de datos y chequea el estado de la misma.
  function conectarBD($U){
    if($this->compruebaBBDD())
```

```
    {
      $conexion=NULL;
      if($U='u'||$U='U')
      {
        $conexion = MySQLi_connect(MYSQL_HOST, MYSQL_USER, MYSQL_PASS, MYSQL_DB);
      }
      else if($U='r'||$U='R')
      {
        $conexion = MySQLi_connect(MYSQL_HOST, MYSQL_USERR, MYSQL_PASSR, MYSQL_DB);
      }
      //Comprobamos si la conexión ha tenido exito
      if(!$conexion)
      {
        echo "<span class='ko'> Ha sucedido un error inexperado en la conexion de la base de
       datos</span>";
        return -1;
      }
      //devolvemos el objeto de conexión para usarlo en las consultas
      return $conexion;
    }
    else return -1;
}

function conectarCrearBD($SQL){

      $conexion=NULL;
      $conexion = MySQLi_connect(MYSQL_HOST, MYSQL_USERR, MYSQL_PASSR);
      $SQL2="GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, ALTER ON `Sensores`.* TO
       'MYSQL_USER'@'MYSQL_HOST';";
      //Comprobamos si la conexión ha tenido exito
      if(!$conexion)
      {
        echo "<span class='ko'> Ha sucedido un error inexperado en la conexion crear de la base de
       datos</span>";
        return -1;
      }
      else
      {
        MySQLi_query($conexion ,$SQL) or die("<span class='ko'>ooh, Fallo en consulta
       conectarCrearBD</span>");
        MySQLi_query($conexion ,$SQL2) or die("<span class='ko'>ooh, Fallo en consulta
       conectarCrearBD permisos</span>");
        $this->desconectarBD($conexion);
        return ($SQL);
      }
      //devolvemos el objeto de conexión para usarlo en las consultas
      return $conexion;
}
/*Desconectar la conexion a la base de datos*/
function desconectarBD($conexion){
      //Cierra la conexión y guarda el estado de la operación en una variable
      $close = MySQLi_close($conexion);
      //Comprobamos si se ha cerrado la conexión correctamente
      if(!$close){
        echo "<span class='ko'>Ha sucedido un error inesperado en la desconexion de la base de
       datos</span>";
      }
      //devuelve el estado del cierre de conexión
```

```
        return $close;
    }

    function eliminaBBDD (){
        $conR = MySQLi_connect(MYSQL_HOST, MYSQL_USERR, MYSQL_PASSR);
        $SQL='DROP DATABASE IF EXISTS Sensores';
        if($conR)
        {
            MySQLi_query($conR,$SQL) or die("<span class='ko'>ooh, Fallo en consulta  eliminaBBDD</span>");
            $this->desconectarBD($conR);
            return ($SQL);
         }
        else{
          echo("<span class='ko'>No existe BBDD</span>");
        }
    }

    function ExisteBBDD($NombreBBDD)
    //Nos dice si existe la base de datos.
    {
        //Creamos la conexión
        $conexion = $this->conectarBD('u');
        if ($conexion!=-1)
        {
            $SQL='SHOW databases WHERE `database`="' . $NombreBBDD .'";';
            //generamos la consulta
            if(!$result = MySQLi_query($conexion, $SQL)) die("<span class='ko'>ooh, Fallo en consulta
             ExisteBBDD</span>");

            //guardamos en un array multidimensional todos los datos de la consulta
            $i=0;
            while($row = MySQLi_fetch_assoc($result))
            {
                //Contamos columnas

                $i++;
            }
            //Cerramos la base de datos
            $this->desconectarBD($conexion);
            //devolvemos true si existen columnas si no false
            if ($i==1)
            {
              //print_r($i);
                return true;
            }
            else
                return false;
    }
    else
        return false;
}
    //Devuelve un array multidimensional con el resultado de la consulta
    function getArraySQL($SQL){
        //Creamos la conexión
        $conexion = $this->conectarBD('u');
        //generamos la consulta

        if ($conexion!=-1)
```

```
    {
      if(!$result = MySQLi_query($conexion, $SQL)) die("<span class='ko'>ooh, Fallo en consulta
      getArraySQL </span>");
      $rawdata = [];
      //guardamos en un array multidimensional todos los datos de la consulta
      $i=0;
      while($row = MySQLi_fetch_row($result))
      {
        //guardamos en rawdata todos los vectores/filas que nos devuelve la consulta
        $rawdata[$i] = $row;

        $i++;
      }
      //Cerramos la base de datos
      $this->desconectarBD($conexion);
      //devolvemos rawdata
      return $rawdata;
    }
    else
      echo("<span class='ko'>No existe BBDD</span>");
}

//ejecutamos consultas como root
function ejeConsultaR ($SQL){
    $conR = $this->conectarBD('r');

    print_r($conR );
    if ($conexion!=-1)
    {
      MySQLi_query($conR,$SQL) or die("<span class='ko'>ooh, Fallo en consulta ejeConsultaR</span>");
      $this->desconectarBD($conR);
      return ($SQL);
     }
    else{
      echo("<span class='ko'>No existe BBDD</span>");
    }
}

//ejecutamos cualquier consulta como user
function ejeConsulta ($SQL){
    $con = $this->conectarBD('u');

    if ($conexion!=-1)
    {
      MySQLi_query($con,$SQL) or die("<span class='ko'>ooh, Fallo en consulta ejeConsulta</span>");
      $this->desconectarBD($con);
      return ($SQL);
     }
    else{
      echo("<span class='ko'>No existe BBDD</span>");
    }
}
//inserta en la base de datos datos eleatorios
function insertDatos($Nom,$min,$max,$rep){

    //creamos la conexión
      $conexion = $this->conectarBD('u');
      //Escribimos la sentencia SQL necesaria respetando los tipos de datos
```

```
    for ($i = 1; $i <= $rep; $i++)
    {
      //Generamos un número entero aleatorio entre min y max
      $ran = rand($min, $max);

      $annio = rand(2011, 2014);
      $mes = rand(1, 12);
      $dia = rand(1, 30);
      $Hora = rand(1, 24);
      $Min = rand(1, 59);
      $Seg = rand(1, 59);

      $cadena=sprintf('%d-%d-%d %d:%d:%d',$annio,$mes,$dia,$Hora,$Min,$Seg);
      $SQL = 'insert into DatoSensor (Id, Nombre, Dato,Fecha_hora)
      values (NULL,"'.$Nom.'","'.$ran.'","'.$cadena.'")';
      //hacemos la consulta y la comprobamos
      $consulta = MySQLi_query($conexion,$SQL);
      if(!$consulta)
        {
          echo "<span class='ko'> No se ha podido insertar en la base de
       datos</span>".MySQLi_error($conexion);
        }

    }
    //Desconectamos la base de datos
    $this->desconectarBD($conexion);
    //devolvemos el resultado de la consulta (true o false)
    return $consulta;
  }
  //Genera un dato aleatorio para datoActual
  function datoActual($Nom,$min,$max){
    //creamos la conexión
      $conexion = $this->conectarBD('u');
      //Escribimos la sentencia SQL necesaria respetando los tipos de datos
      //Generamos un número entero aleatorio entre 0 y 100
      $dato = rand($min, $max);

      $SQL = 'UPDATE `TipoSensor` SET `DatoActual`= "'. $dato .'" WHERE `Nombre` ="' . $Nom .'";';
      $consulta = MySQLi_query($conexion,$SQL);
      if(!$consulta)
        {
          echo "<span class='ko'> No se ha podido actualizar en la base de
       datos</span>".MySQLi_error($conexion);
        }

      $SQL = 'insert into DatoSensor (Id, Nombre, Dato,Fecha_hora)
      values (NULL,"'.$Nom.'","'.$dato.'",NULL)';
      //hacemos la consulta y la comprobamos
      $consulta = MySQLi_query($conexion,$SQL);
      if(!$consulta)
        {
          echo "<span class='ko'>No se ha podido insertar en la base de
       datos</span>".MySQLi_error($conexion);
        }
    //Desconectamos la base de datos
    $this->desconectarBD($conexion);
    //devolvemos el resultado de la consulta (true o false)
    return $consulta;
```

```
    }

    function getAllInfo($SQL){
        //obtenemos el array con toda la información
        return $this->getArraySQL($SQL);
    }

    function NombreSensor()
    {
        $SQL = 'SELECT Nombre FROM Sensores.TipoSensor';
        return $this->getArraySQL($SQL);
    }
}
?>
```

Código 24 - PaqClass.*PHP*

Aplicación *Proxy*

Este apartado muestra el código creado para la aplicación java que hace la función de *Proxy*. A continuación, mostramos el sistema de archivos utilizado para la creación. Como vemos, utilizamos el sistema de archivo de la librería californium-core. Aquí es donde introducimos los archivos creados para realizar el *Proxy*.

Ha sido necesario crear tres clases, una es el programa principal, se llama *ProxyCoAP*.java, la clase que hace las funciones de conexión a la base de datos se llama basecon.java. La clase que estructura la función de los hilos se llama Ejc*CoAP*_h.java. Vemos la estructura de carpetas del proxy en la figura 75.

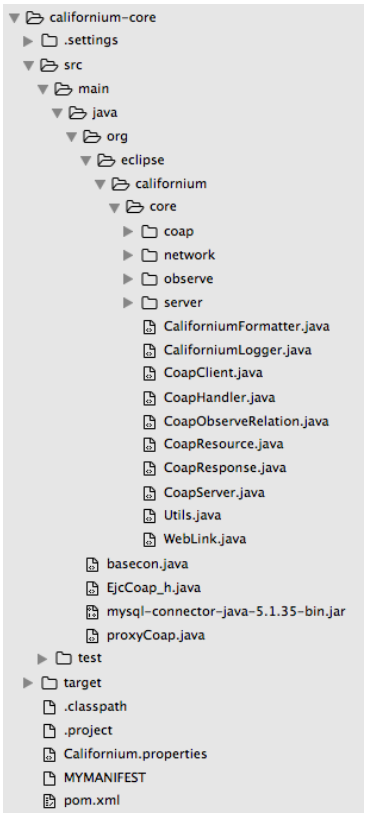


Figura 75 - Estructura de archivos de aplicación *Proxy*

Basecon.java

```
import java.SQL.*;
public class basecon
{
	private static final String userR="root";
	private static final String userU="davsel";
	private static final String passR="ddd";
	private static final String passU="uah";
	private static final String BBDD="Sensores";

	Connection conexion;
	Statement s;
	ResultSet rs;
	int actualizado=0;

	public void ConectaBBDD(String User)
	{
		try
		{
			Class.forName("com.MySQL.jdbc.Driver");

			if (User=="U" || User=="u")
				{
					conexion = DriverManager.getConnection ("jdbc:MySQL://localhost/" + BBDD
	,userU ,passU);
				}
			else if (User=="R" || User=="r")
				{
					conexion = DriverManager.getConnection ("jdbc:MySQL://localhost/"+
	BBDD,userR, passR);
				}


		} catch (Exception e)
		{
			System.out.println("Error de conexion en BBDD" + e);
			e.printStackTrace();
		}


	}

	public ResultSet ConsultaBBDD(String SQL)
	{
		try {

			s = conexion.createStatement();

			rs = s.executeQuery (SQL);

		} catch (SQLException e) {
			System.out.println("Error en consulta en BBDD" + e);
			e.printStackTrace();
		}
		return rs;
```

```
        }

        public int ActualizaBBDD(String SQL)
        {
                try {

                    s = conexion.createStatement();

                    actualizado = s.executeUpdate (SQL);

                } catch (SQLException e) {
                    System.out.println("Error en consulta en BBDD" + e);
                    e.printStackTrace();
                }
                return actualizado;
        }

        public void CerrarBBDD()
        {
                try {
                    conexion.close();
                } catch (SQLException e) {
                    System.out.println("Error de cerrando conexion en BBDD" + e);
                    e.printStackTrace();
                }
        }
}
```

Código 25 - basecon.java

Ejc*CoAP*_h.java

```
import org.eclipse.californium.core.CoAPClient;
import org.eclipse.californium.core.CoAPResponse;


public class EjcCoAP_h implements Runnable
{
		private Thread internalThread;
		private String nombreTabla, nombreSensor; // Nombre de tabla donde escribir datos.
		private int delay; // Tiempo entre escrituras-lecturas
		private String direccion; //Direccion CoAP para lectura se server
		private String dato,dato2; //En realidad será Float.
		private float Maximo, Minimo;
		private int i;
		private boolean bucle;
		private double valorAleatorio=0;
		basecon BBDD;

	public EjcCoAP_h(String NSensor, String nombreT, int tiempo, String Dire, float max, float min)
	{
		nombreSensor = NSensor;
		nombreTabla = nombreT;
		delay = tiempo;
		direccion = Dire;
		Maximo=max;
		Minimo=min;
		dato="";
		dato2="";
		i=0;
		BBDD = new basecon();
		System.out.println(NSensor + " Comienza ejecución de hilo "+ nombreSensor);
		bucle=true;
		//Iniciamos hilo, se ejecuta run()
		internalThread=new Thread(this);
		internalThread.start();

	};

	private boolean bbddInserta( double dato)
	{

		String consulta="INSERT INTO Sensores.DatoSensor (Id,Nombre,Dato,Fecha_Hora) VALUES
	(NULL,'"+ nombreSensor +"',"+ dato +",NULL)";


		//ResultSet rsConsulta;
		try {

		    BBDD.ConectaBBDD("U");
		    BBDD.ActualizaBBDD(consulta);

		} catch (Exception e) {
		    System.out.println("Error en consulta Inserta " + e);
		    e.printStackTrace();
		    return false;
```

```
        }
        BBDD.CerrarBBDD();
        return true;
        }

private boolean bbddModifica( double val)
{


        String consulta="UPDATE Sensores.TipoSensor SET DatoActual = "+ val +" WHERE
Nombre='"+ nombreSensor +"' ";
        //ResultSet rsConsulta;
        try {

            BBDD.ConectaBBDD("U");
            BBDD.ActualizaBBDD(consulta);

        } catch (Exception e1) {
            System.out.println("Error en consulta Modifica " + e1);
            e1.printStackTrace();
            return false;
        }
        BBDD.CerrarBBDD();
        return true;
        }

public void run()
{
        CoAPClient client = new CoAPClient(direccion);
        CoAPResponse response;


        while(bucle)
        {
            i++;
//Cuando funcionen los servidores CoAP se eliminara lo que se indica abajo y quedará solo esto:
            /*
                    response = client.post("",1);
                    dato=response.getResponseText();

             */

        valorAleatorio = Math.random()*((Maximo+20)-Minimo)+(Minimo-20);  //se elimina
        response = client.post(String.valueOf(valorAleatorio),1);          //se elimina
        response = client.get();                                  //se elimina
            dato=response.getResponseText();                          //se elimina

                    bbddInserta(Double.parseDouble(dato));
                    bbddModifica(Double.parseDouble(dato));
                    System.out.println(  nombreSensor+ " " + dato + "\n");

            try{
                    Thread.sleep(delay);
            }catch(InterruptedException e)
                    {
                        System.out.println( nombreSensor + "(): Paramos Sleep \n" + e.toString());
                        Thread.currentThread().interrupt();
                    }
```

```
            }

        }
        public void RompeHilo()
        {
                bucle=false;
                try{
                    Thread.sleep(100);
                }catch(InterruptedException e)
                    {
                            System.out.println( nombreSensor + "(): Paramos Sleep de PararHilo\n" + e);
                    }
                internalThread.interrupt();
                System.out.println( nombreSensor + " interrumpido voluntariamente");

        }

        public String dameDato()
        {
                return dato;
        }

        public String dameNombre()
        {
                return nombreSensor;
        }

}
```

Código 26 - Ejc*CoAP*_h.java

ProxyCoAP.java

```
import java.SQL.ResultSet;
import java.SQL.SQLException;
import java.util.Iterator;
import java.util.ArrayList;

public class ProxyCoAP extends Thread
{


	String nombre,NomSensor;
	int cadaCuantosMs,delay;

	ArrayList<EjcCoAP_h> Hilos; //lista de hilos
	basecon BBDD;
	boolean enciende, modifica;

	private ProxyCoAP()
	{
		enciende=true;
		modifica=true;
		BBDD = new basecon();
		Hilos = new ArrayList<EjcCoAP_h>();
	}

	private boolean getEnciende(){
		return enciende;
	}
	private boolean getModifica(){
		return modifica;
	}
	private basecon getBBDD(){
		return BBDD;
	}
	private ArrayList<EjcCoAP_h> getHilos(){
		return Hilos;
	}

	private void setEnciende(boolean e){
		enciende=e;
	}
	private void setModifica(boolean m){
		modifica=m;
	}
	private void setBBDD(basecon b){
		BBDD=b;
	}
	private void setHilos(ArrayList<EjcCoAP_h> h){
		Hilos=h;
	}

	private static void LimpiaHilos(ArrayList<EjcCoAP_h> Hilos)
	{
```

```
        //**Mejora** compara Hilos existentes con BBDD
        //  Si existe en BBDD y lista   -->Siguiente
        //  elseif existe en lista y no en BBDD      -->Elimina de lista y destruye hilo
        //  elseif si existe en BBDD y no en lista  -->Introducimos en lista y creamos hilo

        //Elimina hilos anteriores
        if(!Hilos.isEmpty())
        {
            Iterator<EjcCoAP_h> RecorreHilos = Hilos.iterator(); //Iterador de lista
            while(RecorreHilos.hasNext())
            {
                    RecorreHilos.next().RompeHilo();
            }
        }

        Hilos.clear();
}

private static boolean creaHilos(ArrayList<EjcCoAP_h> Hilos,basecon BBDD)
{
        //for con consulta SQL sensores-->Nombre, intervalo
        //crea hilos, tantos como sensores



            String consulta="SELECT * FROM Sensores.TipoSensor";
            ResultSet rsConsulta;
            BBDD.ConectaBBDD("U");
            rsConsulta = BBDD.ConsultaBBDD(consulta);
            String BBDD="Sensores";
            String Nombre, Dire, unidades, ud;
            float max,min,actual;
            int delay;

            try {

                    while (rsConsulta.next())
                    {
                        Nombre=rsConsulta.getString (1);
                        Dire=rsConsulta.getString (2);
                        delay =rsConsulta.getInt(3)*1000;
                        unidades=rsConsulta.getString (4);
                        ud=rsConsulta.getString(5);
                        max=rsConsulta.getFloat (6);
                        min=rsConsulta.getFloat (7);
                        actual=rsConsulta.getFloat (8);

                        Hilos.add(new EjcCoAP_h(Nombre,BBDD,delay,Dire,max,min));
                    }
                    } catch (SQLException e1) {
                        System.out.println("Error recorrido de rsConsulta " + e1);
                        e1.printStackTrace();
                        return false;
                    }
            BBDD.CerrarBBDD();
            return true;
        }
```

```
private boolean bbddModifica( boolean b)
{
        int val=-1;
        if (b)
            val=1;
        else
            val=0;

        String consulta="UPDATE Sensores.JavaPHP SET Modifica = "+ val +" WHERE Id=1";
        //ResultSet rsConsulta;
        try {

            BBDD.ConectaBBDD("U");
            BBDD.ActualizaBBDD(consulta);

        } catch (Exception e1) {
            System.out.println("Error en consulta Modifica " + e1);
            e1.printStackTrace();
            return false;
        }
        BBDD.CerrarBBDD();
        return true;
        }
private boolean consultaOnModifica()
{
        String consulta="SELECT * FROM Sensores.JavaPHP";
        ResultSet rsConsulta;
        BBDD.ConectaBBDD("U");
        rsConsulta = BBDD.ConsultaBBDD(consulta);

        try {

            while (rsConsulta.next())
            {
                    if (rsConsulta.getInt(2)==1)
                    {
                            modifica=true;
                    }
                    else
                    {
                            modifica=false;
                    }

                    if (rsConsulta.getInt(3)==1)
                    {
                            enciende=true;
                    }
                    else
                    {
                            enciende=false;
                    }
              }
            } catch (SQLException e1) {
                    System.out.println("Error recorrido de rsConsulta " + e1);
                    e1.printStackTrace();
                    return false;
```

```
            }
        BBDD.CerrarBBDD();
        return true;
    }

    public static void main(String[] args)
    {
        ProxyCoAP llamada= new ProxyCoAP();
        boolean arranque=true;
        int i = 0;
        llamada.consultaOnModifica();
        while (true)
        {
                //Comprobamos si enciende de nuevo
                llamada.consultaOnModifica();
            while (llamada.getEnciende())
            {
                //Comprobamos si apaga
                llamada.consultaOnModifica();

                //consulta modifica
                if (llamada.getModifica())
                {
                    LimpiaHilos(llamada.getHilos());
                    llamada.bbddModifica( false);//escribir en BBDD
                    arranque =true;
                    //System.out.println ("entra en modifica");
                }
                if (arranque)
                {
                    creaHilos(llamada.getHilos(), llamada.getBBDD());
                    arranque =false;
                    //System.out.println ("entra en arranque");
                }

                i++;
                //System.out.println("main(): I=" + i);
                llamada.setModifica(false);

                try{
                    //Esperamos para ver info de los hilos
                    Thread.sleep(20);
                }catch(InterruptedException e){e.printStackTrace();}
            }
            LimpiaHilos(llamada.getHilos());
            llamada.bbddModifica( false);//escribir en BBDD
            arranque =true;
            i=0;
        }
    }
}
```

Código 27 - *ProxyCoAP*.java

Aplicación *Android*

Mostramos el código necesario para crear la apk de *Android*. Por un lado, tenemos la parte servidor que devuelve los datos de la base de datos a la apk y por otro la apk propiamente dicha.

Parte Servidor

A continuación, se muestra el código necesario para el funcionamiento de la apk *Android* en la parte servidor. Primeramente mostramos el sistema de archivos de la parte servidora en la figura 76.

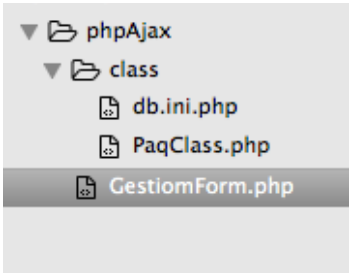


Figura 76 - Estructura de archivos parte servidor apk *Android*

Controlador

GestiomForm.*PHP*

```
<?PHP
require './class/db.ini.PHP';
require_once("./class/PaqClass.PHP");

//Creamos un objeto de la clase paquete
$paquete = new Paq();


switch($_GET['submit'])
{
  case 'listaSensores':

    $query='SELECT `Nombre` FROM Sensores.TipoSensor';
    $ArrayDatos= $paquete->getArraySQL($query);

    $respuestaJSON=array();

    if($ArrayDatos!=null)
    {
      $respuestaJSON["success"]=true;


      $i=0;

      foreach($ArrayDatos as $row)
        {
          $respuestaJSON["data"]["sensores"][]=$row[0];
          $i++;
        }
      $respuestaJSON["data"]["message"]= sprintf("Numero de sensores localizados %d", $i);
    }
    else{
      $respuestaJSON["success"]=false;
      $respuestaJSON["data"]["message"]= sprintf("No tenemos sensores en la base de datos");

    }
    header('Content-type: application/JSON; charset=utf-8');
    echo JSON_encode($respuestaJSON,JSON_FORCE_OBJECT);
    break;

  case 'datosActuales':
    $query='SELECT `Nombre`,`DatoActual`,`Minimo`,`Maximo` FROM Sensores.TipoSensor';
    $ArrayDatos= $paquete->getArraySQL($query);

    $respuestaJSON=array();

    if($ArrayDatos!=null)
    {
      $respuestaJSON["success"]=true;


      $i=0;
```

```
        foreach($ArrayDatos as $row)
          {
            $respuestaJSON["data"]["sensores"][]=$row;
             $i++;
          }
        $respuestaJSON["data"]["message"]= sprintf("Numero de datos actuales localizados %d", $i);
      }
      else{
        $respuestaJSON["success"]=false;
        $respuestaJSON["data"]["message"]= sprintf("No tenemos sensores en la base de datos");

      }
      header('Content-type: application/JSON; charset=utf-8');
      echo JSON_encode($respuestaJSON,JSON_FORCE_OBJECT);
      break;

    case 'listaDatos':

      $Nombre=$_GET['nombreS'];


      $query='SELECT Dato,Fecha_hora FROM (SELECT * FROM Sensores.TipoSensor WHERE
         Nombre="'.$Nombre.'") AS j join Sensores.DatoSensor on j.Nombre=DatoSensor.Nombre ORDER BY
         Fecha_hora ASC;';
      $ArrayDatos= $paquete->getArraySQL($query);

      $respuestaJSON=array();

      if($ArrayDatos!=null)
      {
        $respuestaJSON["success"]=true;


        $i=0;
        foreach($ArrayDatos as $row)
          {


            $respuestaJSON["data"]["datos"][]=array(
             0=>$row[0],
             1=>$row[1],

             );
             $i++;
          }

        $query1 = 'SELECT Maximo, Minimo, Unidades, Ud FROM Sensores.TipoSensor WHERE
         Nombre="'.$Nombre.'";';
        $ArrayMaxMin= $paquete->getArraySQL($query1);
        $respuestaJSON["max"]=$ArrayMaxMin[0][0];
        $respuestaJSON["min"]=$ArrayMaxMin[0][1];
        $respuestaJSON["uni"]=$ArrayMaxMin[0][2];
        $respuestaJSON["ud"]=$ArrayMaxMin[0][3];

        $query2 = 'SELECT AVG(Dato),MAX(Dato),MIN(Dato),STD(Dato),VARIANCE(Dato) FROM (SELECT *
         FROM Sensores.TipoSensor WHERE Nombre="'.$Nombre.'") AS j join Sensores.DatoSensor on
         j.Nombre=DatoSensor.Nombre ORDER BY Fecha_hora ASC;';
        $ArrayMaxMin= $paquete->getArraySQL($query2);
```

```
        $respuestaJSON["estatic"][]=array(
           "AVG"=>$ArrayMaxMin[0][0],
           "MAX"=>$ArrayMaxMin[0][1],
           "MIN"=>$ArrayMaxMin[0][2],
           "DES"=>$ArrayMaxMin[0][3],
           "VAR"=>$ArrayMaxMin[0][4],
        );
        $respuestaJSON["data"]["message"]= sprintf("Numero de datos localizados %d", $i);
        $respuestaJSON["data"]["NumDatos"]=$i;
     }
     else{
        $respuestaJSON["success"]=false;
        $respuestaJSON["data"]["message"]= sprintf("No tenemos datos del sensor %s en la base de datos",
         $Nombre);

     }
     header('Content-type: application/JSON; charset=utf-8');
     echo JSON_encode($respuestaJSON,JSON_FORCE_OBJECT);
   break;

   default:

}

exit();

?>
```

Figura 77 - GestiomForm.*PHP*

Modelo

Db.ini.*PHP*

```
<?PHP
define('MYSQL_HOST','localhost');
define('MYSQL_USER','davsel');
define('MYSQL_PASS','uah');
define('MYSQL_USERR','root');
define('MYSQL_PASSR','ddd');
define('MYSQL_DB','Sensores');
define('MYSQL_DBM','MySQL');
?>
```

Código 28 - db.ini.*PHP*

PaqClass.*PHP*

```
<?PHP
require ('db.ini.PHP');
date_default_timezone_set('Europe/Madrid');

class Paq{

  public $IDr = 0 ;


  function compruebaBBDD()
  {
    //Comprobamos que existe la BBDD
    $SQL='SHOW databases';
    $conexion=NULL;  //variable que guarda la conexión de la base de datos
    $conexion = MySQLi_connect(MYSQL_HOST, MYSQL_USERR, MYSQL_PASSR);
    if(!$result = MySQLi_query($conexion, $SQL)) die('ooh, Fallo en consulta compruebaBBDD');
     //guardamos en un array multidimensional todos los datos de la consulta
    $i=0;
    $ok=False;
    while($row = MySQLi_fetch_assoc($result))
    {
      foreach ($row as $value)
      {

        if ($value==MYSQL_DB)
          $ok=true;

      }

      $i++;
    }
    //print_r($i);
    $this->desconectarBD($conexion);
    if ($ok)
      return true;
    else
      return false;
  }
```

```
//Función que crea y devuelve un objeto de conexión a la base de datos y chequea el estado de la misma.
function conectarBD($U){

   if($this->compruebaBBDD())
   {
      $conexion=NULL;
      if($U='u'||$U='U')
      {
         $conexion = MySQLi_connect(MYSQL_HOST, MYSQL_USER, MYSQL_PASS, MYSQL_DB);
      }
      else if($U='r'||$U='R')
      {
         $conexion = MySQLi_connect(MYSQL_HOST, MYSQL_USERR, MYSQL_PASSR, MYSQL_DB);
      }
      //Comprobamos si la conexión ha tenido exito

      if(!$conexion)
      {
         echo 'Ha sucedido un error inexperado en la conexion de la base de datos<br>';
         return -1;
      }
      //devolvemos el objeto de conexión para usarlo en las consultas

      return $conexion;
   }
   else return -1;

}

/*Desconectar la conexion a la base de datos*/
function desconectarBD($conexion){
      //Cierra la conexión y guarda el estado de la operación en una variable
      $close = MySQLi_close($conexion);
      //Comprobamos si se ha cerrado la conexión correctamente
      if(!$close){
        echo 'Ha sucedido un error inexperado en la desconexion de la base de datos<br>';
      }
      //devuelve el estado del cierre de conexión
      return $close;
}

function ExisteBBDD($NombreBBDD)
//Nos dice si existe la base de datos.
{

   //Creamos la conexión

   $conexion = $this->conectarBD('u');
   if ($conexion!=-1)
   {
      $SQL='SHOW databases WHERE `database`="' . $NombreBBDD .'";';
      //generamos la consulta
      if(!$result = MySQLi_query($conexion, $SQL)) die('ooh, Fallo en consulta ExisteBBDD');

      //guardamos en un array multidimensional todos los datos de la consulta
      $i=0;
```

```
        while($row = MySQLi_fetch_assoc($result))
        {
          //Contamos columnas
          //print_r($row);
          $i++;
        }
        //Cerramos la base de datos
        $this->desconectarBD($conexion);
        //devolvemos true so existen columnas sino false

        if ($i==1)
        {
         //print_r($i);
          return true;
        }
        else
          return false;
    }
    else
      return false;
}

    //Devuelve un array multidimensional con el resultado de la consulta
    function getArraySQL($SQL){
      //Creamos la conexión

      $conexion = $this->conectarBD('u');
      //generamos la consulta

      if ($conexion!=-1)
      {
        if(!$result = MySQLi_query($conexion, $SQL)) die('ooh, Fallo en consulta getArraySQL');

        $rawdata = [];
        //guardamos en un array multidimensional todos los datos de la consulta
        $i=0;
        while($row = MySQLi_fetch_row($result))
        {
          //guardamos en rawdata todos los vectores/filas que nos devuelve la consulta
          $rawdata[$i] = $row;

           $i++;
        }
        //Cerramos la base de datos
        $this->desconectarBD($conexion);
        //devolvemos rawdata

        return $rawdata;
      }
      else
        print_r("No existe BBDD");
    }

    //inserta en la base de datos un nuevo registro en la tabla usuarios
    function insertDatos($Nom,$min,$max,$rep){

      //creamos la conexión
        $conexion = $this->conectarBD('u');
```

```
        //Escribimos la sentencia SQL necesaria respetando los tipos de datos
    for ($i = 1; $i <= $rep; $i++)
    {

        //Generamos un número entero aleatorio entre 0 y 100
        $ran = rand($min, $max);

        $annio = rand(2011, 2014);
        $mes = rand(10, 12);
        $dia = rand(10, 30);
        $Hora = rand(10, 24);
        $Min = rand(10, 59);
        $Seg = rand(10, 59);

        $cadena=sprintf('%d-%d-%d %d:%d:%d',$annio,$mes,$dia,$Hora,$Min,$Seg);


        $SQL = 'insert into DatoSensor (Id, Nombre, Dato,Fecha_hora)
        values (NULL,"'.$Nom.'","'.$ran.'","'.$cadena.'")';
        //hacemos la consulta y la comprobamos
        $consulta = MySQLi_query($conexion,$SQL);
        if(!$consulta)
          {
            echo "No se ha podido insertar en la base de datos<br><br>".MySQLi_error($conexion);
          }

    }
    //Desconectamos la base de datos
    $this->desconectarBD($conexion);
    //devolvemos el resultado de la consulta (true o false)
    return $consulta;
  }

  function getAllInfo($SQL){
    //obtenemos el array con toda la información
    return $this->getArraySQL($SQL);
  }

  function NombreSensor()
  {
    $SQL = 'SELECT Nombre FROM Sensores.TipoSensor';
    return $this->getArraySQL($SQL);
  }

}

?>
```

Código 29 - PaqClass.*PHP*

Parte *Android*

La aplicación *Android* se convierte a partir de la aplicación web cuyo código mostramos a continuación. La estructura de archivos se muestra en la siguiente figura:

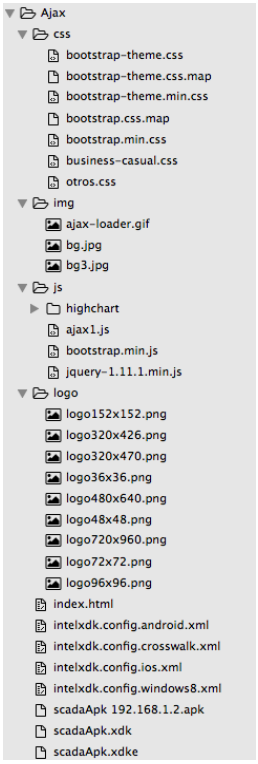


Figura 78 - Estructura de archivo apk *Android*

CSS

Los archivos *CSS* son exactamente los mismos que los utilizados en la aplicación web general, no se muestra el código en este apartado por estar en el subapartado *CSS* de Aplicación web.

JavaScript

*Ajax*1.js

```
var SensorElegido="";
var ip="192.168.1.2";
var direccion="http://" + ip + "/~davsel/PHPAjax/GestiomForm.PHP";

function irASensor(nombreSens) {

  SensorElegido=nombreSens;

  yaPuedes=true;
      $("#CabeceraDatos").HTML("</br><img src=\"./img/Ajax-loader.gif\" alt=\"cargando\"
      height=\"42\" width=\"42\"><p>...Buscando...</p> ");

      var llamadaAjax = function(dato,nombreS){
      $("#CabeceraDatos").HTML("</br> <img src=\"./img/Ajax-loader.gif\" alt=\"cargando\"
      height=\"42\" width=\"42\"><p>...Buscando datos...</p> ");
      return $.getJSON(direccion,{"submit" : dato , "nombreS" : nombreS});
      }

  $("#CabeceraDatos").HTML("<img src=\"./img/Ajax-loader.gif\" alt=\"cargando\" height=\"42\"
      width=\"42\"><p>...Buscando valor de sensores...</p> ");

          llamadaAjax('listaDatos',nombreSens)
            .done(function(response){
              if(response.success){
      var salida="<div class='jumbotron'>"
      salida+="<h3> Datos de sensor " + SensorElegido +"</h3>"
                  salida+="<h4>" + response.data.message + "</h4>";

          var salida2="";
          var salida3="";
          var graficar=[];
          var date;
          var anioMin=9000;
          var anioMax=0;

                  salida+="</div>";
      salida3+="<table class=\"table table-hover\">";
          salida3+= " <caption>Datos Historicos </caption>";
          salida3+= "<thead>";
          salida3+= " <tr><th></th><th>Datos Mostrados en "+ response.uni + "</th></tr>";
          salida3+= " <tr>";
          salida3+= "   <th width='15%'> Id </th>";
          salida3+= "    <th width='52%'> Valor </th>";
          salida3+= "    <th width='40%'> Fecha </th>";
          salida3+= " </tr>";
          salida3+= "</thead>"
          salida3+="<tbody></tbody></table>";

      salida2+="<table class=\"table table-hover\">";

          salida2+= "<thead>";
          salida2+= "</thead>";
          salida2+="<tbody>";
```

```
    for (i = 0;  i< response.data.NumDatos; i++)

    {
       salida2+="<tr><td>" +(i+1)+ "</td><td>"  + response.data.datos[i][0] +"  "+
response.ud + "</td><td>" + response.data.datos[i][1]+ "</td></tr>"
       date= new Date(response.data.datos[i][1]);
       var anio=parseInt(date.getFullYear())

if(anioMin>anio){
          anioMin=anio;
       }
       if(anioMax<anio){
          anioMax=anio;
       }

graficar.push([(Date.UTC(date.getUTCFullYear(),date.getUTCMonth(),date.getUTCDate(),date.getU
TCHours(),date.getUTCMinutes(),date.getUTCSeconds())),parseInt(response.data.datos[i][0])]);

    }
    salida2+="</tbody>";
    salida2+="</table>";
    $("#CabeceraDatos").HTML(salida);
    $("#datos3").HTML(salida3);
   $("#datos").HTML(salida2);


   var rangoAnio=new Array();
   rangoAnio[0]=anioMin;

   for (j=1;j<(anioMax-anioMin)+1;j++){
      rangoAnio[j]=rangoAnio[j-1]+1;
   }


  chartCPU = new Highcharts.StockChart({
       chart: {
          renderTo: 'grafico'
          //defaultSeriesType: 'spline'
          //Se puede usar tipo de grafico activo con llamada Ajax en el interior Spline updating each
second
       },
     rangeSelector : {
       enabled: false
     },
     title: {
       text: 'Grafica'
     },
     xAxis: {
       type: 'datetime',
       labels: {
          formatter: function () {
             return Highcharts.dateFormat('%d %m %Y', this.value);
          },
          dateTimeLabelFormats: {
             minute: '%H:%M',
             hour: '%H:%M',
             day: '%e. %b',
```

```
                    week: '%e. %b',
                    month: '%b \'%y',
                    year: '%Y'
                }
            }
        },
        yAxis: {
            minPadding: 0.2,
            maxPadding: 0.2,
            title: {
                text: 'Valores',
                margin: 10
            }
        },
        series: [{
            name: 'valor',
            data: (function() {
                var data = graficar;
                return data;
            })()
        }],
        tooltip: {
            formatter: function () {
                return '<b>Dato: </b>'+Highcharts.numberFormat(this.y, 2)+'<br/>' +
                '<b>Fecha: </b>'+ Highcharts.dateFormat('%d/%m/%Y %H:%M:%S', this.x) + '<br/>'
                ;
            }
        },
      credits: {
        enabled: false
      }
      });

          } else{
              $("#CabeceraDatos").HTML('No ha funcionado conexion Ajax de datos'
    +response.data.message);
          }
        })
        .fail(function(jqXHR, textStaus, errorThorown){
          $("#CabeceraDatos").HTML('Fallo en conexion de datos: ' +textStaus);
        });

  $("#refreSCADAtos").show();
  $("#datos1").show();
  $("#datos2").show();


}

function listaSensores() {
      var llamadaAjax = function(dato){
      $("#inserta").HTML("</br><img src=\"./img/Ajax-loader.gif\" alt=\"cargando\" height=\"42\"
      width=\"42\"><p>...Buscando...</p> ");
      return $.getJSON(direccion,{"submit" : dato});
      }

          llamadaAjax('listaSensores')
            .done(function(response){
```

```
                if(response.success){
                  var salida="<div class='jumbotron' >"
                  salida+="<h3>" + response.data.message + "</h3>";
                  $.each(response.data.sensores, function(key,valor){
                    salida+= "<button class='btn btn-success' id=" + key + " onclick=\"irASensor('" + valor +
        "')\"> " + valor +" </button> "

                  });
        salida+="</div>"
                  $("#inserta").HTML(salida);
                } else{
                  $("#inserta").HTML('No ha funcionado conexion Ajax' +response.data.message);
                }
              })
              .fail(function(jqXHR, textStaus, errorThorown){
                  $("#inserta").HTML('Fallo en conexion: ' +textStaus);
              });


}

function DatosActuales() {
var llamadaAjax = function(dato){
    $("#actuales").HTML("</br> <img src=\"./img/Ajax-loader.gif\" alt=\"cargando\" height=\"42\"
       width=\"42\"><p>...Buscando datos actuales...</p> ");
    return $.getJSON(direccion,{"submit" : dato});
    }

   llamadaAjax('datosActuales')
      .done(function(response){
        if(response.success){
          var min;
          var max;
          var valor;
          var nombre;
          var salida2="<div class='jumbotron' >"
          salida2+="<h3>Datos Actuales</h3></div>";
          salida2+="<table class=\"table table-condensed\">";
          salida2+= " <caption>Datos Actuales</caption>";
          salida2+= "<thead>";
          salida2+= " <tr>";
          salida2+= "   <th> Nombre </th>";
          salida2+= "    <th> Valor </th>";
          salida2+= "    <th> indicador </th>";
          salida2+= "    </tr>";
          salida2+= "</thead>"
          salida2+="<tbody>";

          $.each(response.data.sensores, function(key,valor){
            nombre=valor[0];
             dato=parseInt(valor[1]);
             min=parseInt(valor[2]);
             max=parseInt(valor[3]);


             salida2+="<tr><td>"+nombre+"</td>"
             salida2+="<td>" + dato +"</td>"
```

```
                if((dato<max) && (dato>min))
                  salida2+= "<td> <span class=\"CirculoVerde\"> </span>  </td>";
                else
                  salida2+= "<td> <span class=\"CirculoRojo\"> </span>  </td>";

                salida2+="</tr>";
             });

              salida2+="</tbody></table>"

              $("#actuales").HTML(salida2);
            } else{
              $("#actuales").HTML('No ha funcionado conexion Ajax' +response.data.message);
            }
          })
          .fail(function(jqXHR, textStaus, errorThorown){
            $("#actuales").HTML('Fallo en conexion: ' +textStaus);
          });


}


$(document).ready(function(){
   $("#refreSCADAtos").hide();
   $("#datos1").hide();
   $("#datos2").hide();
         //Inicia lista sensores
   listaSensores();
   DatosActuales() ;

});
```

Código 30 - *Ajax*1.js

HTML

```
<!DOCTYPE HTML>
<HTML>
<head>
        <meta charset="UTF-8">
        <title>Prueba Ajax</title>
        <script type="text/JavaScript" src="./js/jquery-1.11.1.min.js"> </script>
        <link href="./CSS/bootstrap.min.CSS" rel="stylesheet" media="screen">
        <link href="./CSS/otros.CSS" rel="stylesheet" media="screen">
        <link href="./CSS/business-casual.CSS" rel="stylesheet">
  <script type="text/JavaScript" src="./js/bootstrap.min.js"> </script>
  <script type="text/JavaScript" src="./js/highchart/highstock.js"> </script>
        <script type="text/JavaScript" src="./js/highchart/exporting.js"> </script>
  <script type="text/JavaScript" src="./js/Ajax1.js"> </script>
  <meta name="viewport" content="width=device-width, initial-scale=1">
</head>
<body class="container">
        <div class="navbar navbar-inverse">
    <p class="navbar-brand">Android SCADA Web</p>
  </div>
        <div>
            <div id="actuales" class="table">
                <img src="./img/Ajax-loader.gif" alt="cargando" height="42" width="42">
            </div>
            <button id="refrescaSen" class="btn btn-primary" onclick="DatosActuales()">Refresca Datos
        Actuales</button>
            <br><br>
        </div>

        <div>
            <div id="inserta" class="center-block"></div>
            <button id="refrescaSen" class="btn btn-primary" onclick="listaSensores()">Refresca
        Sensores</button>
              <br><br>
        </div>

        <div>
            <div id="CabeceraDatos"></div>
            <div id="datos1" class="panel panel-default">
              <div id="datos2" class="panel-body">
                <div id="datos3" class="text-right"></div>
                <div id="datos" class="scroll"></div>
              </div>
            </div>

        <div id="grafico"></div>
            <button id="refreSCADAtos" class="btn btn-primary" onclick="irASensor(SensorElegido)"
        >Refresca datos</button>
            <br><br><br><br><br><br><br><br><br><br><br><br>
        </div>

 </body>
</HTML>
```

Código 31 - Index.*HTML*

Capítulo 6

Manual de usuario

El capítulo que nos ocupa pretende ser una guía de funcionamiento del sistema. En él se muestra cómo funciona tanto la aplicación web como la app *Android* detectando qué cambios realiza cada acción en el sistema creado.

El servidor tiene que estar funcionando con el sistema de gestión de base de datos, el servidor *Apache* y el *Proxy Proxy*Final.jar funcionando. Una vez está listo todo esto, los usuarios podrán acceder a los datos y administrar el servidor web.

Aplicación Web

Index.*PHP*

La página principal de la aplicación muestra los últimos datos recogidos por la aplicación *Proxy*. Estos datos se muestran sobre una tabla junto con su nivel de alarma. Las alarmas se muestran en verde cuando el dato leído está entre los límites máximo y mínimo configurados en la pestaña administrar. Cuando el valor leído se sale de esos límites establecidos en la configuración del sensor, la alarma se muestra en rojo.

Los datos se refrescan automáticamente cada segundo sin realizar ninguna acción, es un sistema de autorefresco.

El autorefresco funciona si el navegador que se utilice soporta worker, si no soporta esta faceta de *HTML*5, nos saldrá un mensaje como el de la figura

79. Si no soporta worker tenemos disponible el botón Actualizar que al pulsarlo recoge los datos actuales.

Figura 79 - Navegador sin soporte de worker

Los navegadores que soporten workers, actualizan los datos de manera automática. No obstante, existe la posibilidad de parar esa actualización mediante el botón Parar Autorefresco, para volver a iniciar se debe pulsar Iniciar Autorefresco. Estos botones se muestran en la figura 80:

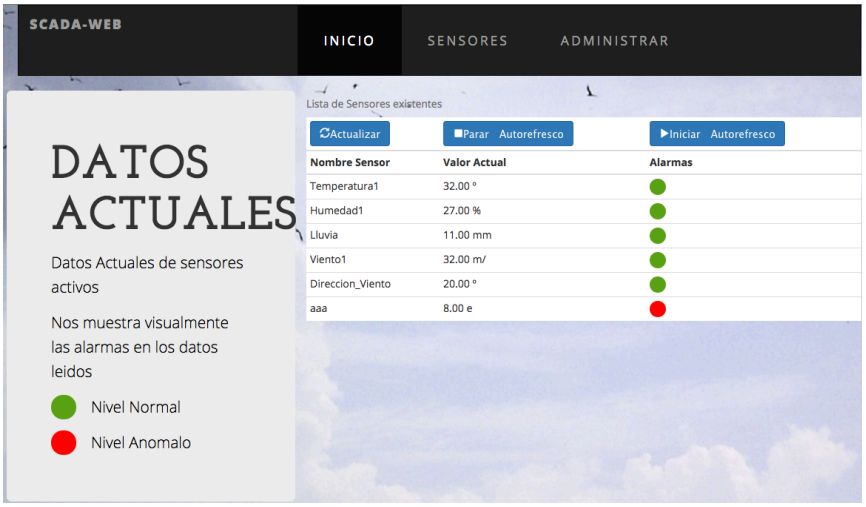


Figura 80 - Vista de Index.*PHP*

Datos.*PHP*

Al pulsar la pestaña sensores en la barra de navegación, accedemos a la página datos.*PHP*. En esta página se visualizan los datos históricos guardados hasta el momento.

Una vez accedemos, se nos muestra una barra de navegación vertical con todos los sensores dados de alta en el sistema. Pulsando sobre cualquiera de ellos accederemos a los datos de ese sensor, se muestra en la figura 81.

Figura 81 - Vista datos.*php* sin elegir

Cuando elegimos el sensor deseado se nos muestra un cuadro de filtrado de datos a mostrar como el que vemos en la figura 82. Podemos ver todos los datos pulsando el botón mostrar todo o ver solo los datos de un determinado rango de fechas.

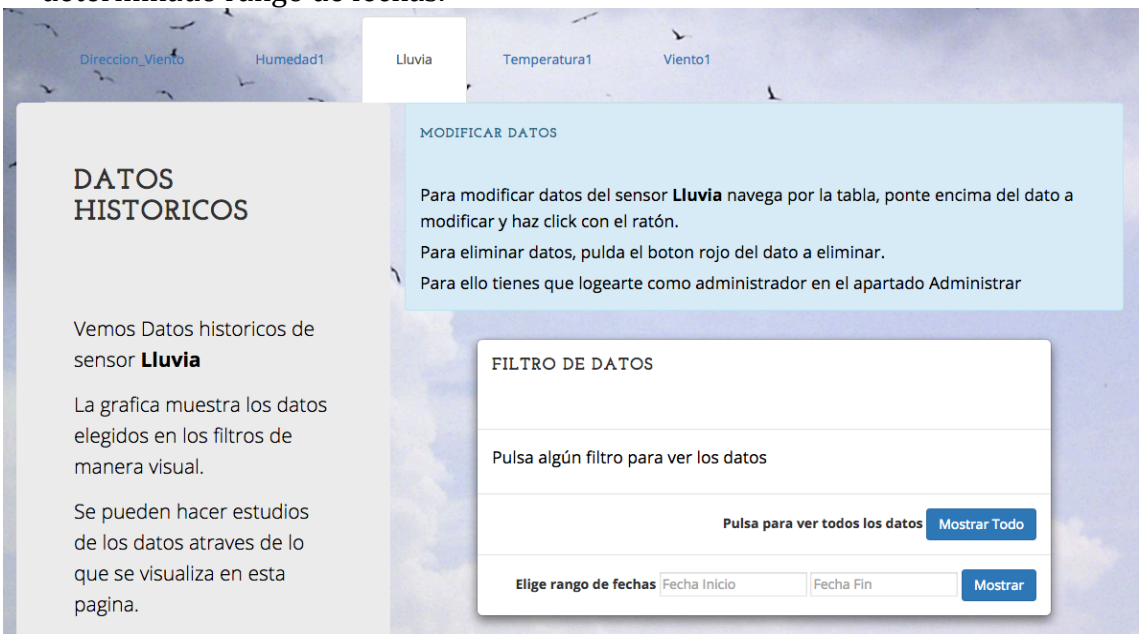


Figura 82 - Vista datos.*PHP* sin filtro

El filtro por fecha permite elegir gráficamente la fecha mediante un calendario desplegable. Si elegimos la fecha de inicio, al elegir la fecha de fin del rango vemos en la figura 83 como sombrea los días anteriores a la fecha elegida en fecha de inicio. De esta forma evitamos consultas incoherentes a la base de datos, hacemos una validación de los datos introducidos.

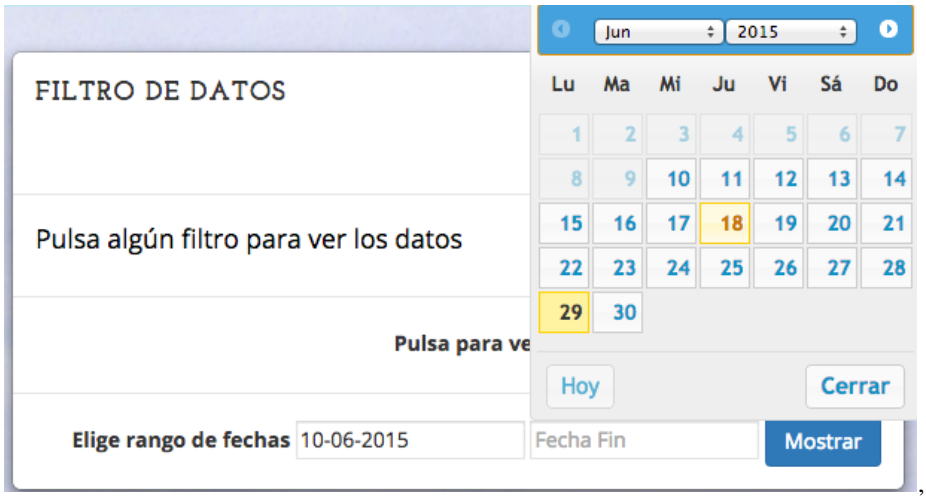


Figura 83 - Detalle de filtro por fecha de datos.*PHP*

Si volvemos a pinchar sobre la fecha inicial del rango, vemos que nos sombrea los días posteriores a la fecha elegida en fecha fin. Igual que se comentaba anteriormente, para validar un rango correcto de fechas.

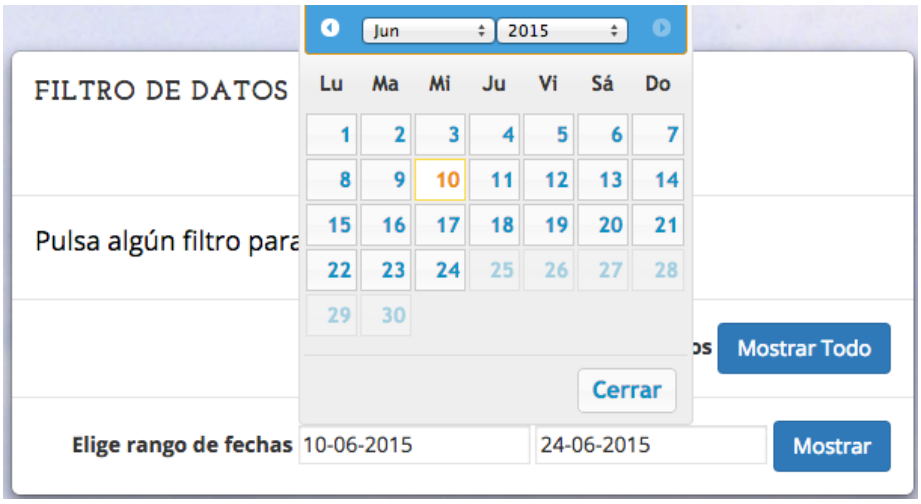


Figura 84 - Detalle de bloqueo de fecha en filtro de datos.*PHP*

Una vez elegido el filtro y pulsado su correspondiente botón para mostrar los datos, se generan tres contenedores: uno con los datos estadísticos de los datos seleccionados en el filtro, otro con todos los datos elegidos en el filtro y un tercero con el gráfico de los datos. Además, se nos muestra qué filtro hemos elegido y el número de datos mostrados, com vemos en figura 85.

En los datos estadísticos se observan la media, el valor máximo y mínimo, la desviación típica y la varianza de los datos elegidos en el filtro.

Figura 85 - Detalle datos.*PHP*, numero de datos mostrados y datos estadísticos

Figura 86 - Detalle de datos.*PHP*, tabla de datos y gráfico

La tabla de datos introduce los valores dentro de un scroll para mostrar los datos de manera comprimida en la pantalla. Se observa en la figura 87.

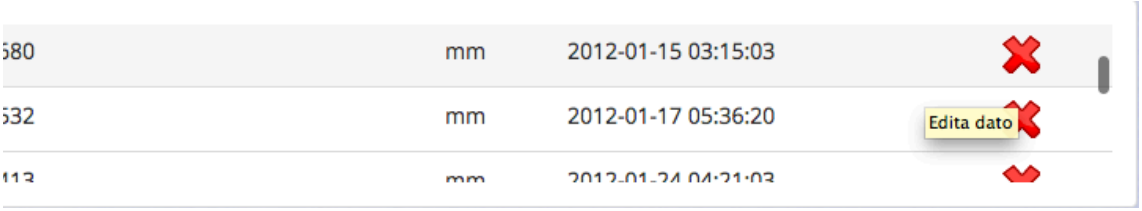


Figura 87 - Detalle de Scroll en tabla de datos

Los datos mostrados se pueden editar. Para poder editar tenemos que autentificarnos como administradores en la pestaña de navegación Administrar. Una vez estemos autentificados ya podemos modificar datos. Para saber que podemos editar los datos, al mostrar la tabla se ven unas "X" rojas que dan la opción de eliminar el dato. Si estas "X" no se muestran, no estamos autentificados como administradores. En datos.*PHP* solo podemos modificar el campo valor de la tabla, los demás valores no son editables.

Para poder editar un valor pasamos por encima de él, vemos cómo sale un lapicero (figura 88) , pulsamos sobre el valor y nos aparece el valor con la posibilidad de editarlo (figura 89). Además aparecen dos botones, uno para guardar y otro para descartar cambio. El botón de guardar se representa con un disquete y el de descartar con una x. Los valores introducidos en la edición de datos, se validan para que tengan unos valores concretos y la base de datos no se llene de valores erróneos.

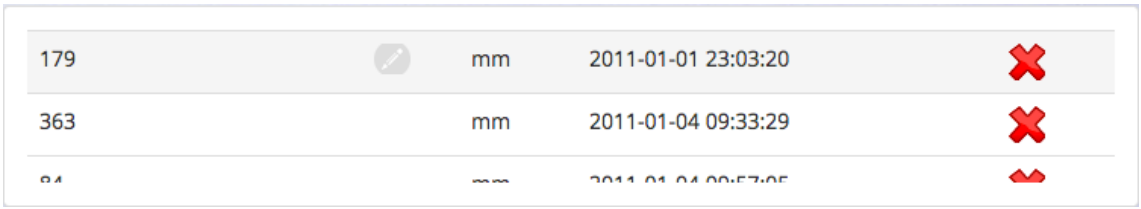


Figura 88 - Detalle de edición de datos tabla de valores de datos.*PHP*

Figura 89 - Detalle de edición de datos

Cuando modificamos el valor y damos a guardar se sustituye en la base de datos el valor antiguo por el que acabamos de introducir. Cuando esto ocurre se muestra un mensaje como el de la figura 90.

Figura 90 - Detalle de edición de dato realizada correctamente

Al pulsar las "X" rojas que aparecen a la derecha de cada dato, eliminamos el dato por completo sin posibilidad de recuperación. Cuando se elimina correctamente se muestra el mensaje de la figura 91.

Figura 91 - Eliminación de datos correcta.

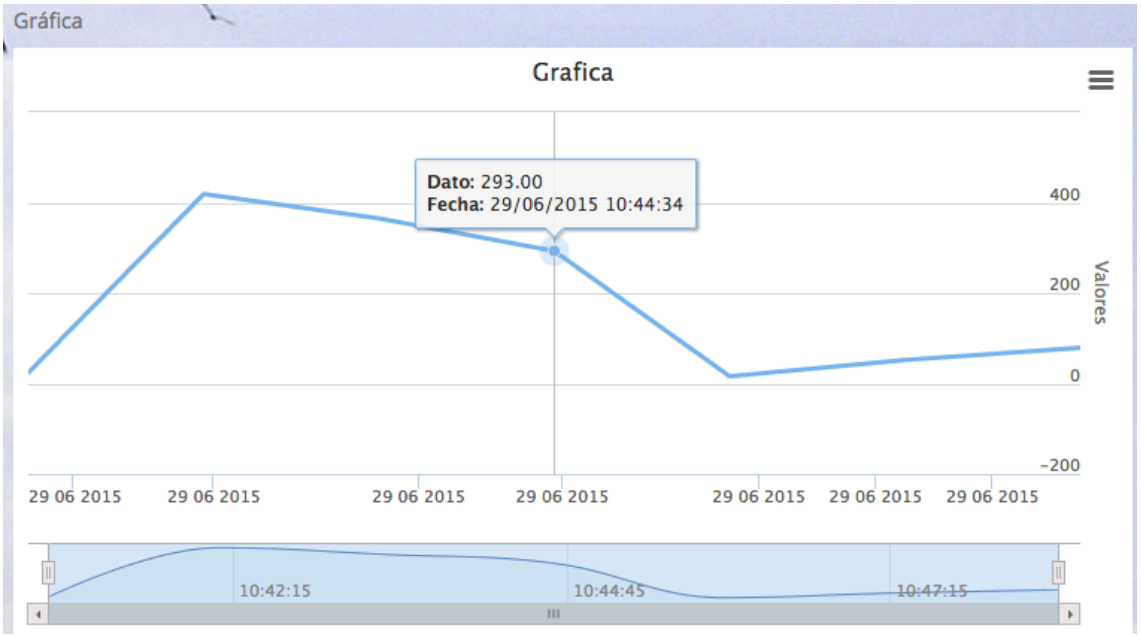


Figura 92 - Detalle de gráfica, muestra de datos al pasar sobre la línea.

La gráfica muestra la representación de los datos seleccionados mediante el filtro anteriormente elegido. La gráfica se implementa con una librería de *Javascript* llamada *Highchart*, es interactiva, pasando el cursor sobre la línea de representación de datos muestra el valor en ese punto junto con la fecha y hora en que se tomó el valor.

También tenemos opciones de zoom, en la barra inferior se puede acortar la parte sombreada en azul para mostrar en la gráfica los datos de manera más exacta a la fecha que nos interesa. En la figura 94 se ve en detalle.

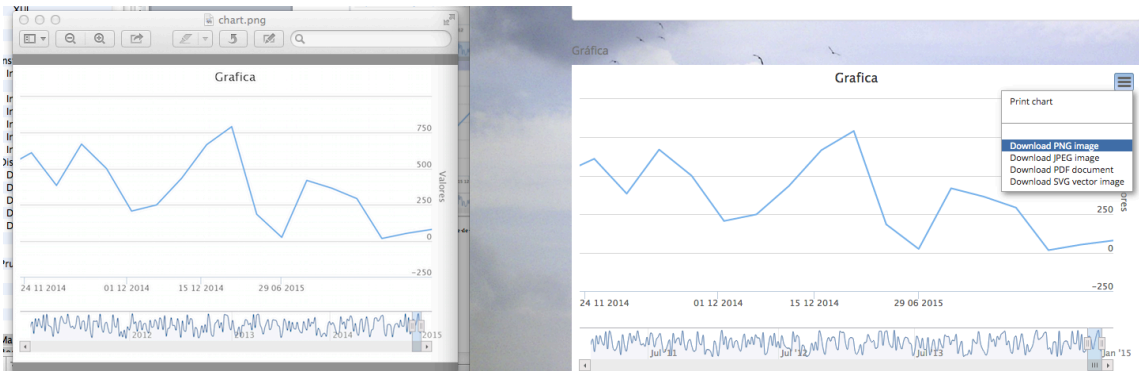


Figura 93 - Detalle de exportación a imagen de gráfico

Los gráficos se pueden exportar en multitud de formatos, pulsando sobre las tres líneas horizontales paralelas de la parte superior derecha nos despliega un listado con el formato deseado de descarga.

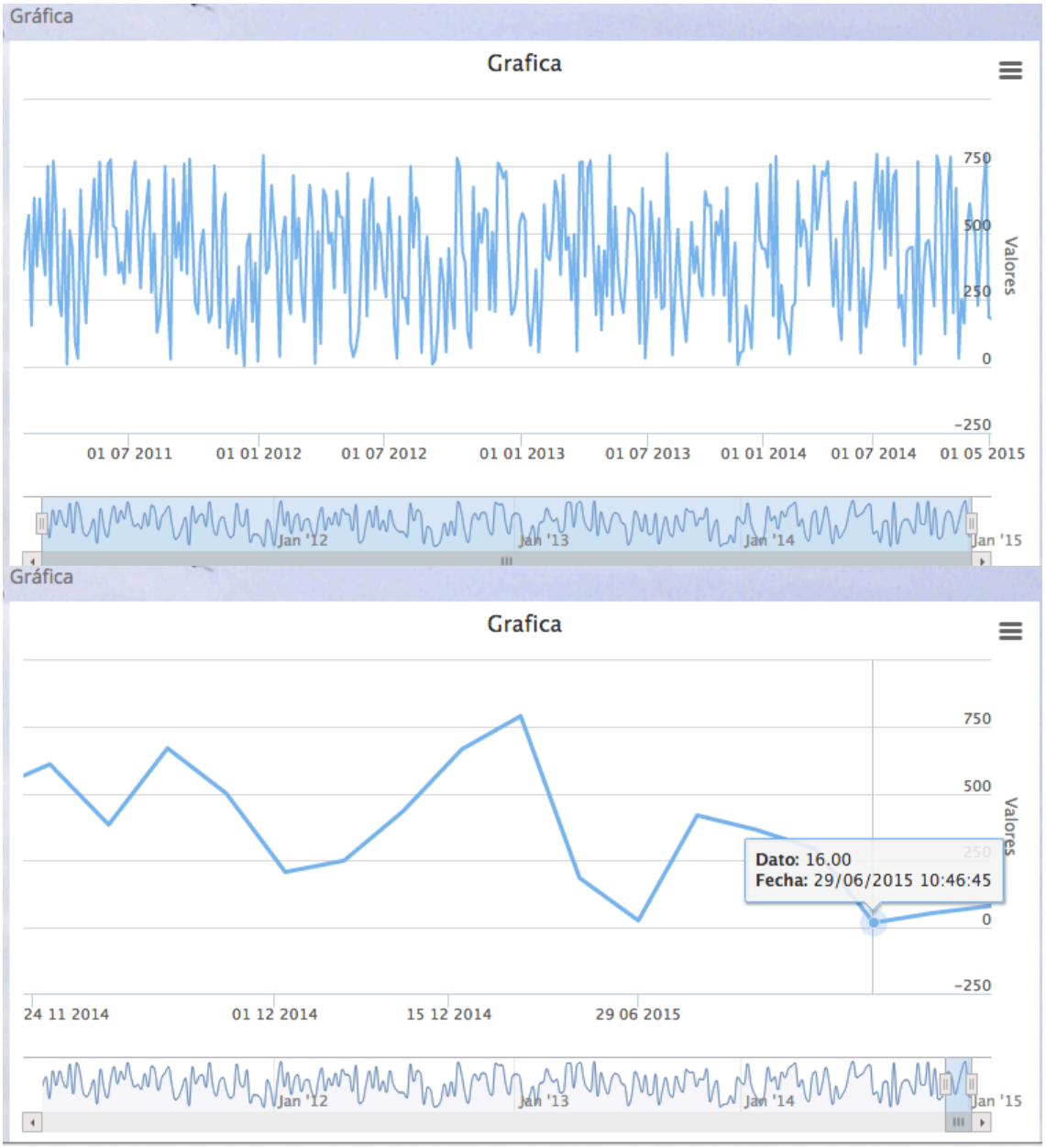
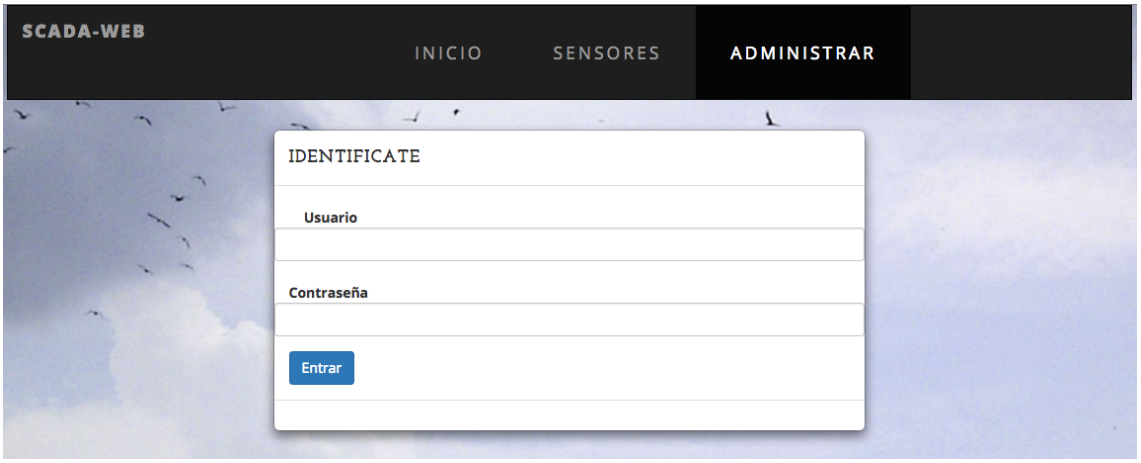


Figura 94 - Detalle de uso del zoom en gráfico de datos.*PHP*

Admin.*PHP*

Admin.*PHP* es la manera de configurar el sistema. Desde esta página se configuran desde la base de datos hasta el *Proxy*, pasando por los usuarios.

Debido a la importancia que tienen estas configuraciones se bloquea el acceso mediante usuario y contraseña. Para evitar ataques por fuerza bruta para la detección de contraseñas, se protege la autentificación mediante un límite de intentos. En concreto 3 intentos, si en tres intentos no se ha autentificado el usuario, se bloquea la página administrador. Solo pudiendo desbloquearse si se borran las cookies creadas por la sesión de usuario. En la figura 90 vemos el cuadro de diálogo para introducir usuario y contraseña, también se observa el mensaje mostrado tras un fallo en la autentificación.



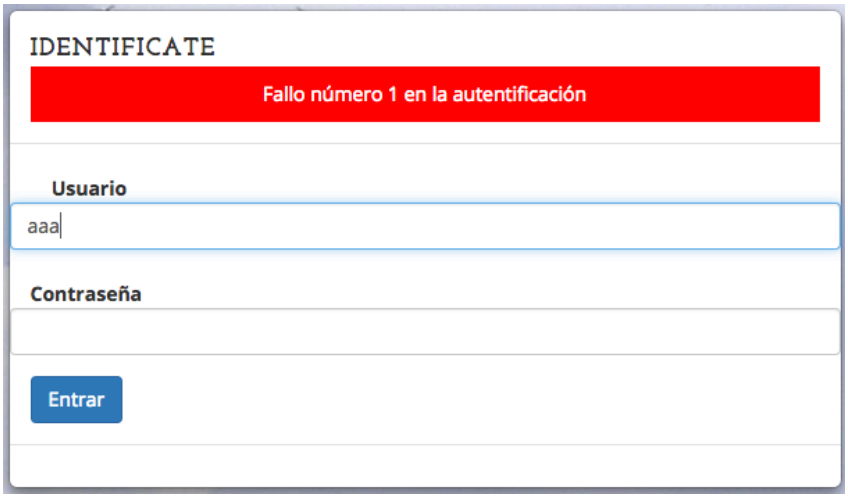


Figura 95 - Fallo de autentificación en admin.*PHP*

Como decíamos, si fallamos tres veces en la autentificación, se bloquea la página de administración, no dando la opción de introducir nuevos datos. Para evitar el bloqueo tenemos que entrar en preferencias de nuestro navegador y borrar datos de navegación. En la figura 97 se muestra el ejemplo con el navegador chrome.

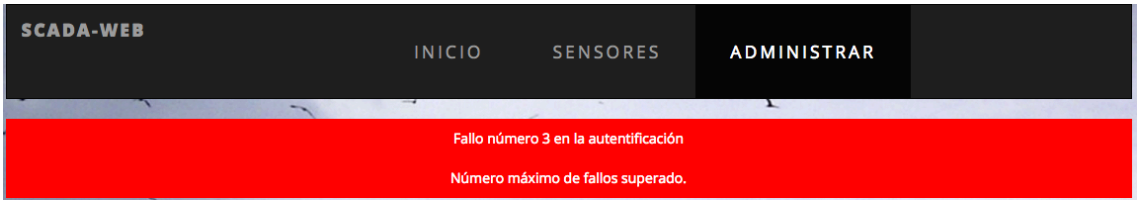


Figura 96 - Tercer fallo de autentificación, bloqueo de admin.*PHP*

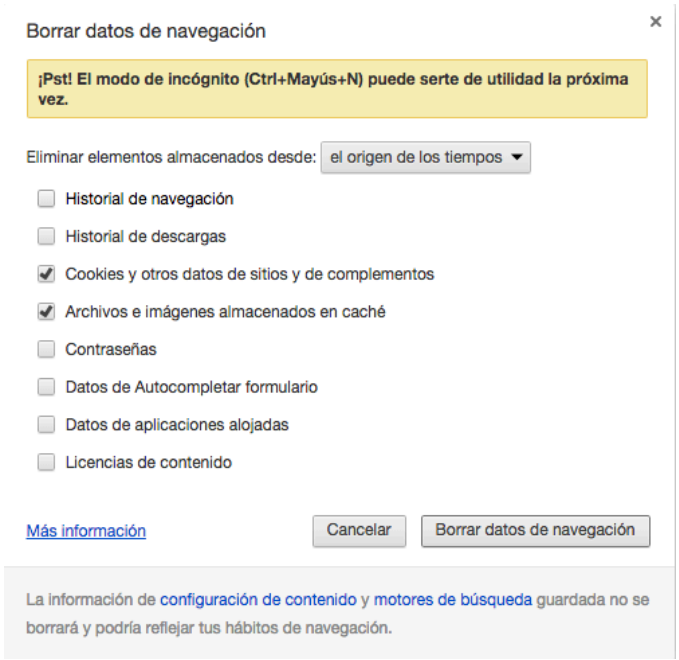


Figura 97 - Resolver bloqueo de autentificación.

Las siguientes tres imágenes muestran la visión de la página de administración una vez nos hemos autentificado como administradores.

La figura 98 muestra primeramente el botón cerrar sesión, este destruye la sesión actual pudiendo autentificarnos con otro usuario posteriormente.

El siguiente cajón engloba los botones de parada y arranque del *Proxy* y justo abajo muestra el estado de los campos *Enciende* y *Modifica*. Si *Enciende* está a cero el *Proxy* no está capturando datos, si está a uno está capturando datos. El campo *Modifica* solo se cambia de estado cuando modificamos un valor en la tabla de sensores que veremos en breve.

El cajón modifica BBDD muestra dos botones, uno para eliminar la base de datos y crearla sin datos y el otro para crear los sensores iniciales que se configuran al diseñar la aplicación. Es la gestión de la BBDD.

El siguiente cajón muestra tantos botones como sensores tenemos creados, tienen como funcionalidad ajustar a cero el hardware del sensor pulsado.

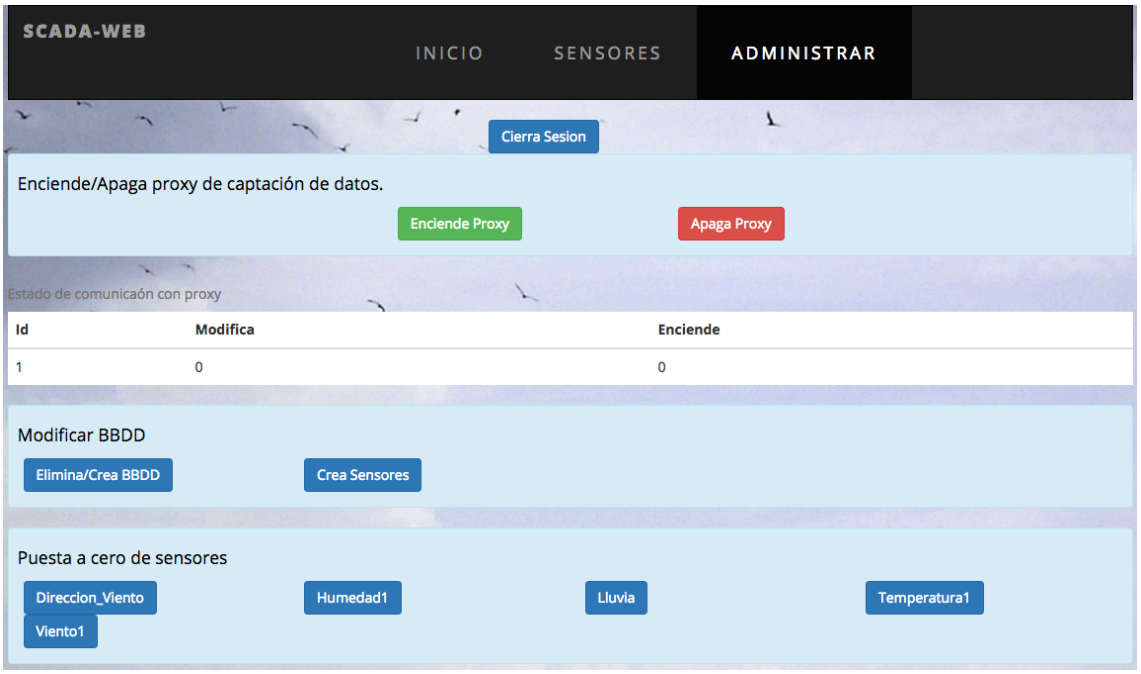


Figura 98 - Detalle de admin.*PHP*

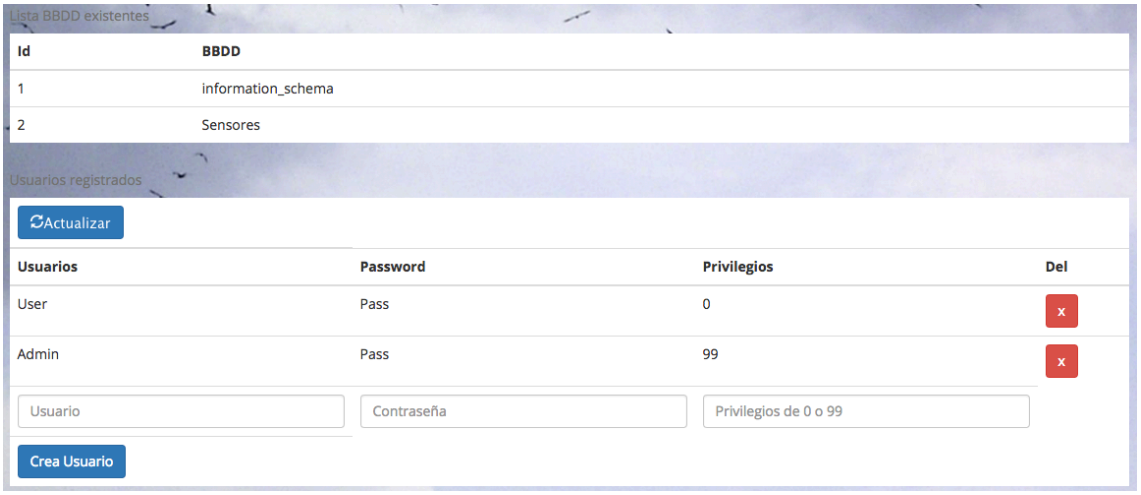


Figura 99 - Detalle 2 de Admin.*PHP*

En la figura 99 primeramente se observan las tablas de la BBDD que ve el usuario que gestiona la BBDD. Information_schema es una tabla de *MySQL* de gestión de datos y la tabla sensores es la tabla con la que trabaja nuestra aplicación web. También vemos la tabla de gestión de usuarios desde la que podemos crear, eliminar o modificar cualquier dato de usuario. Todas las modificaciones que se producen en la tabla usuarios se muestran en directo sin cargar la página entera, solo cargando la tabla usuarios.

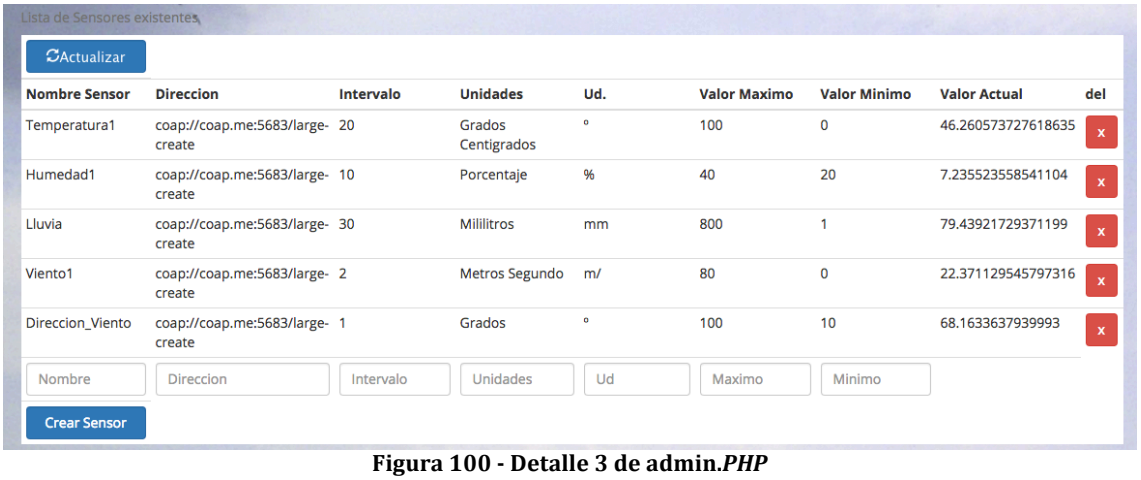


Figura 100 - Detalle 3 de admin.*PHP*

La figura 100 visualiza la tabla de gestión de sensores con la que el *Proxy* se autoconfigura. En ella se ven los sensores creados, se pueden editar, eliminar y crear nuevos sensores. Todo ello sin recargar la página completa, solo cargando la tabla en cuestión.

La figura 101 muestra la página admin.*PHP* cuando nos autentificamos con permisos de usuario. Como vemos, desactiva todos los botones menos cerrar sesión, no muestra tabla de gestión de usuarios y la tabla sensores no es editable, sólo se muestra. Además, tampoco da opción de crear nuevos sensores.

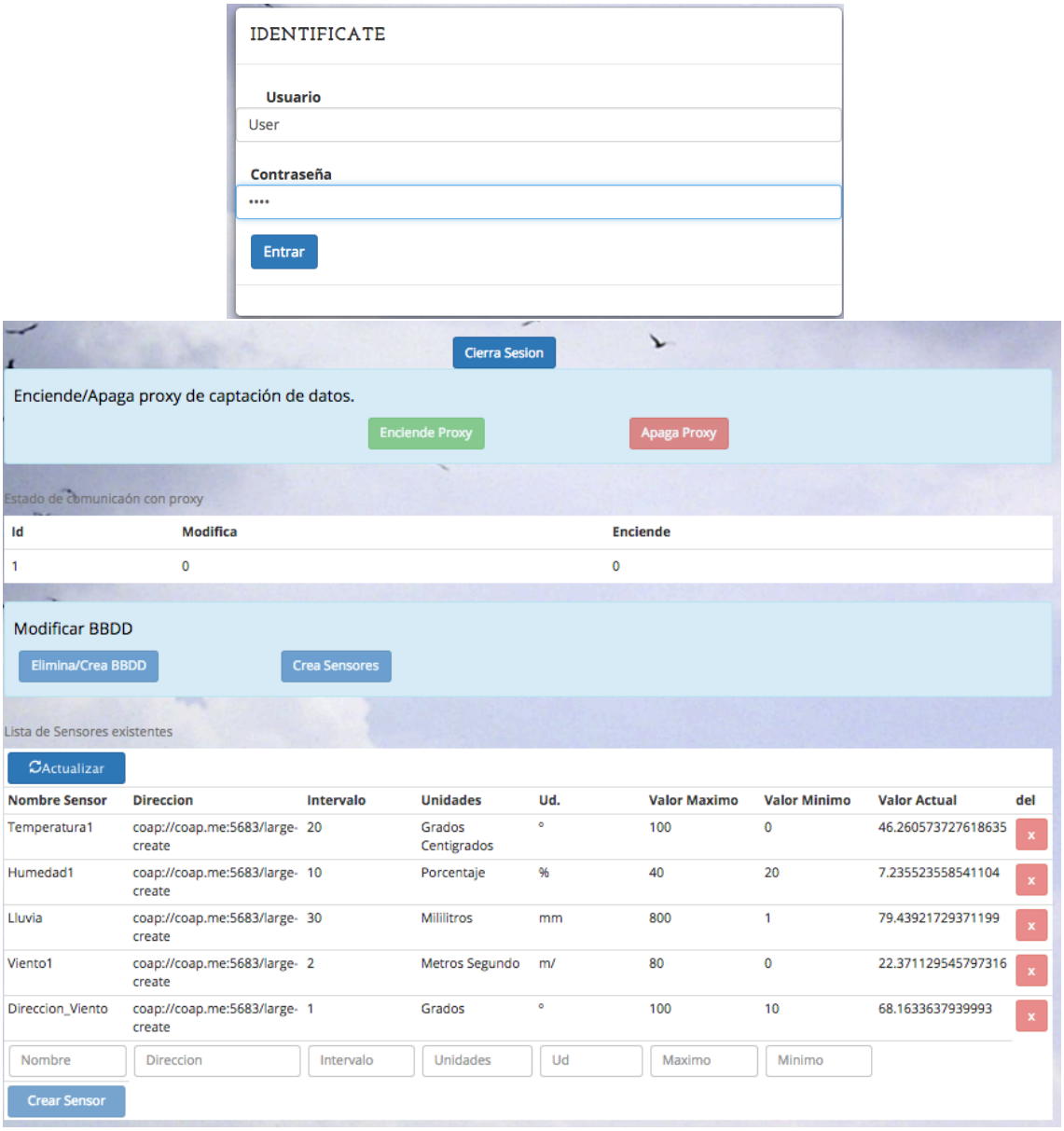


Figura 101 - Admin.*PHP* autentificado con privilegios de user (privilegio 0)

Si autentificados como usuarios vamos a la página datos.*PHP*, los valores de los sensores tampoco serán editables ni se podrán eliminar.

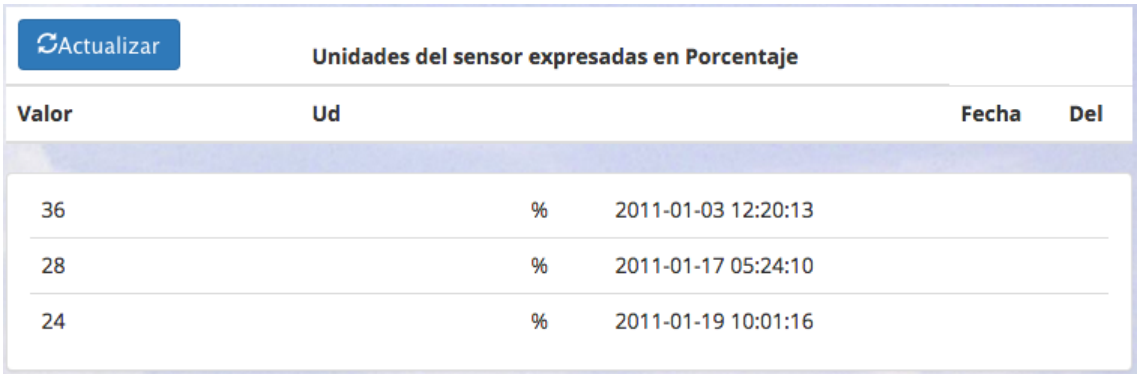


Figura 102 - datos.*PHP* autentificados con privilegios de user. No deja eliminar ni editar datos.

A continuación mostramos el funcionamiento de los botones de activar el *Proxy* viendo cómo para y arranca la aplicación *Proxy*. Primero se tiene que ejecutar el *Proxy* de la siguiente manera:

Mac: User$ java -jar *Proxy*Final.jar

Pulsando el botón Enciende *Proxy* vemos en la figura 103 como en la consola empiezan a verse las lecturas de los sensores. Esto conlleva que esos datos se están escribiendo en la base de datos.

Si pulsamos el botón Apaga *Proxy* se observa en la figura 104 cómo se paran todos los hilos de ejecución de sensores, anulando las lecturas.

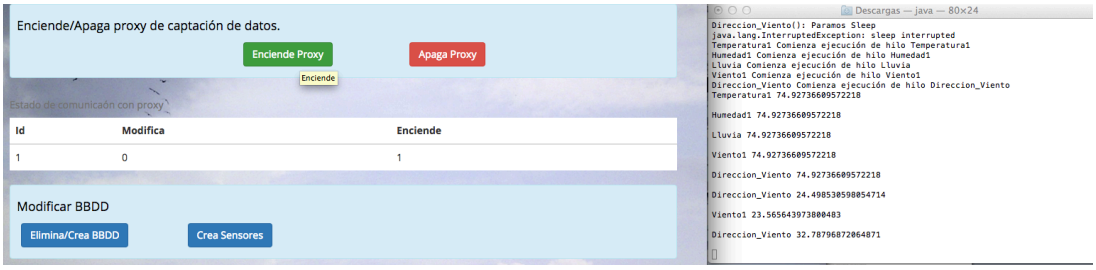


Figura 103 - Uso del botón Enciende *Proxy* en admin.*PHP*

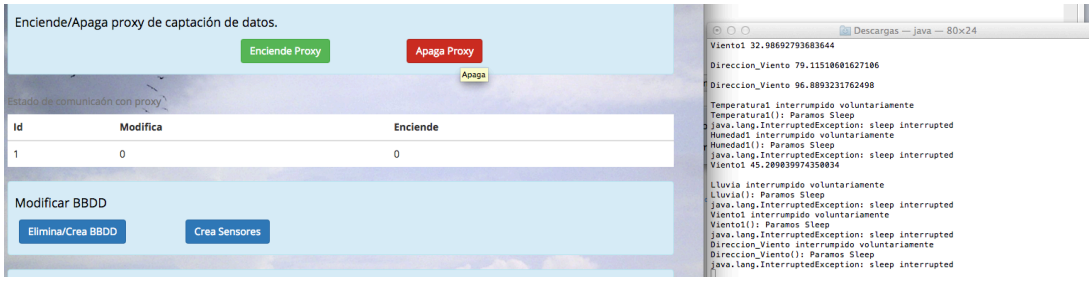


Figura 104 - Uso del botón Apaga *Proxy* en admin.*PHP*

En imágenes sucesivas se observan las distintas validaciones, errores y mensajes de tarea realizada que se muestran en las diferentes casuísticas de uso de la tabla de gestión de usuarios.

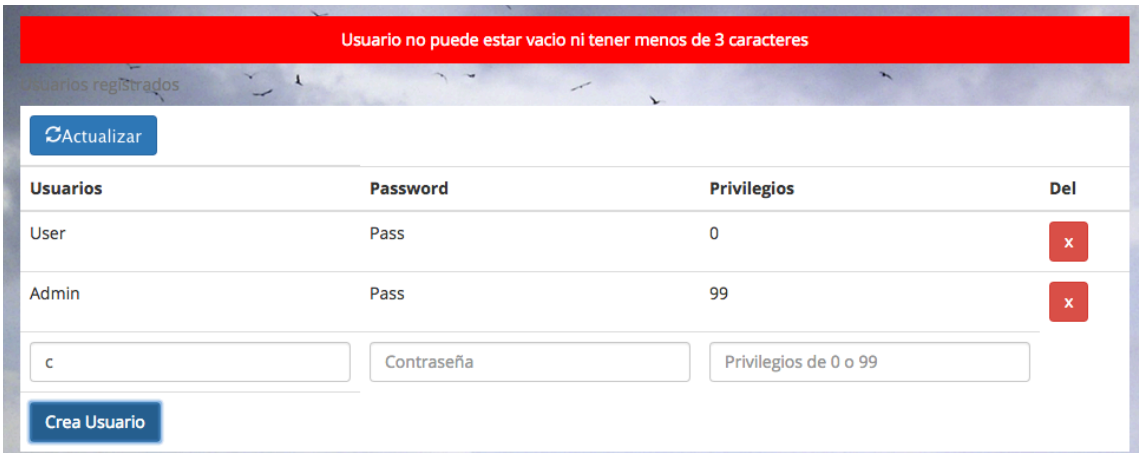


Figura 105 - Validación de datos en tabla Usuarios de Admin.*PHP*

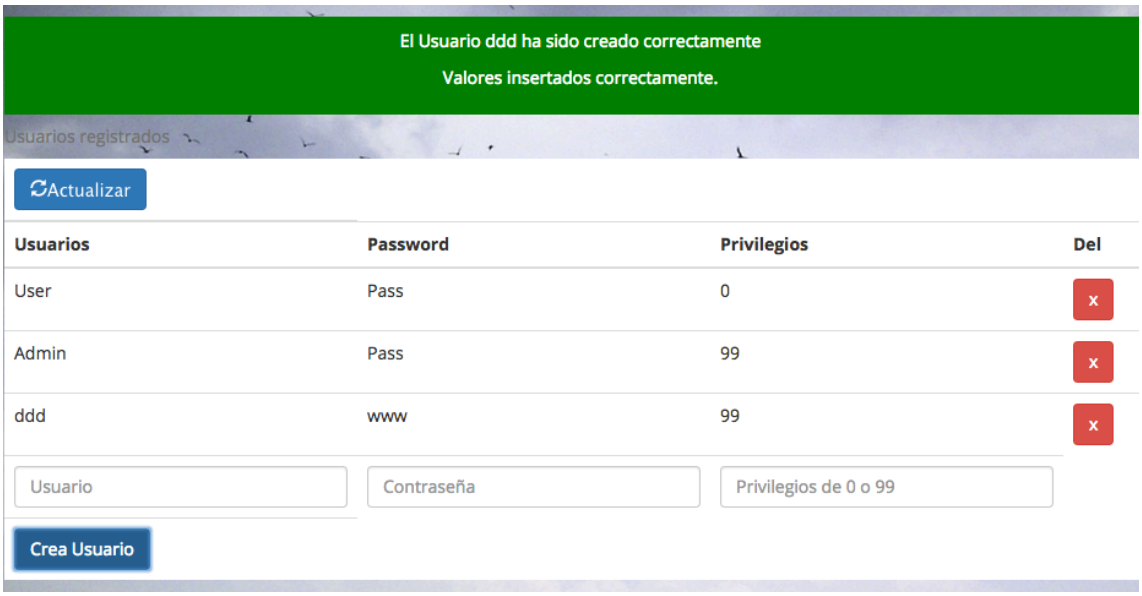


Figura 106 - Detalle de creación de usuario en admin.*PHP*

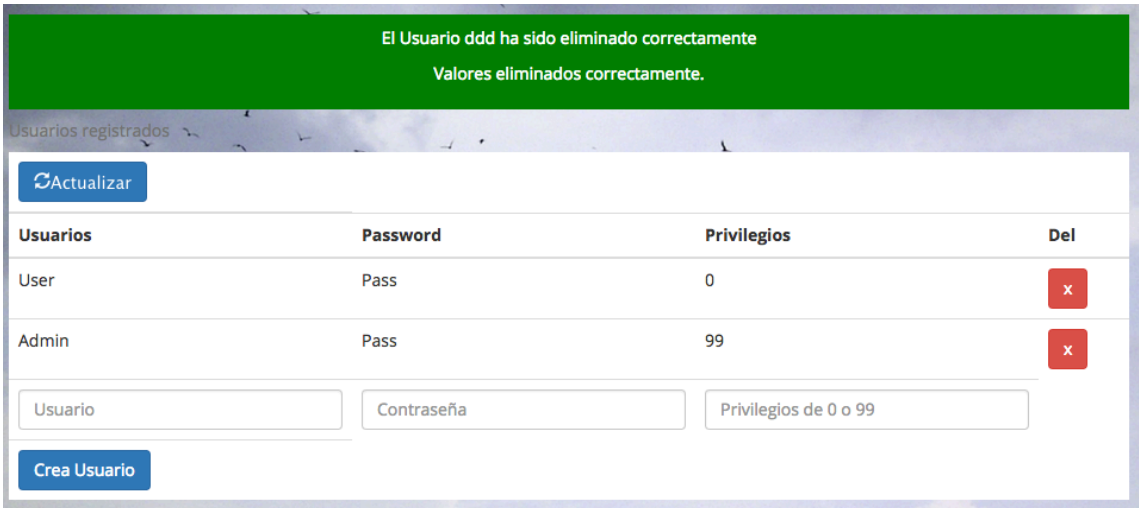


Figura 107 - Detalle de eliminación de usuario en admin.*PHP*

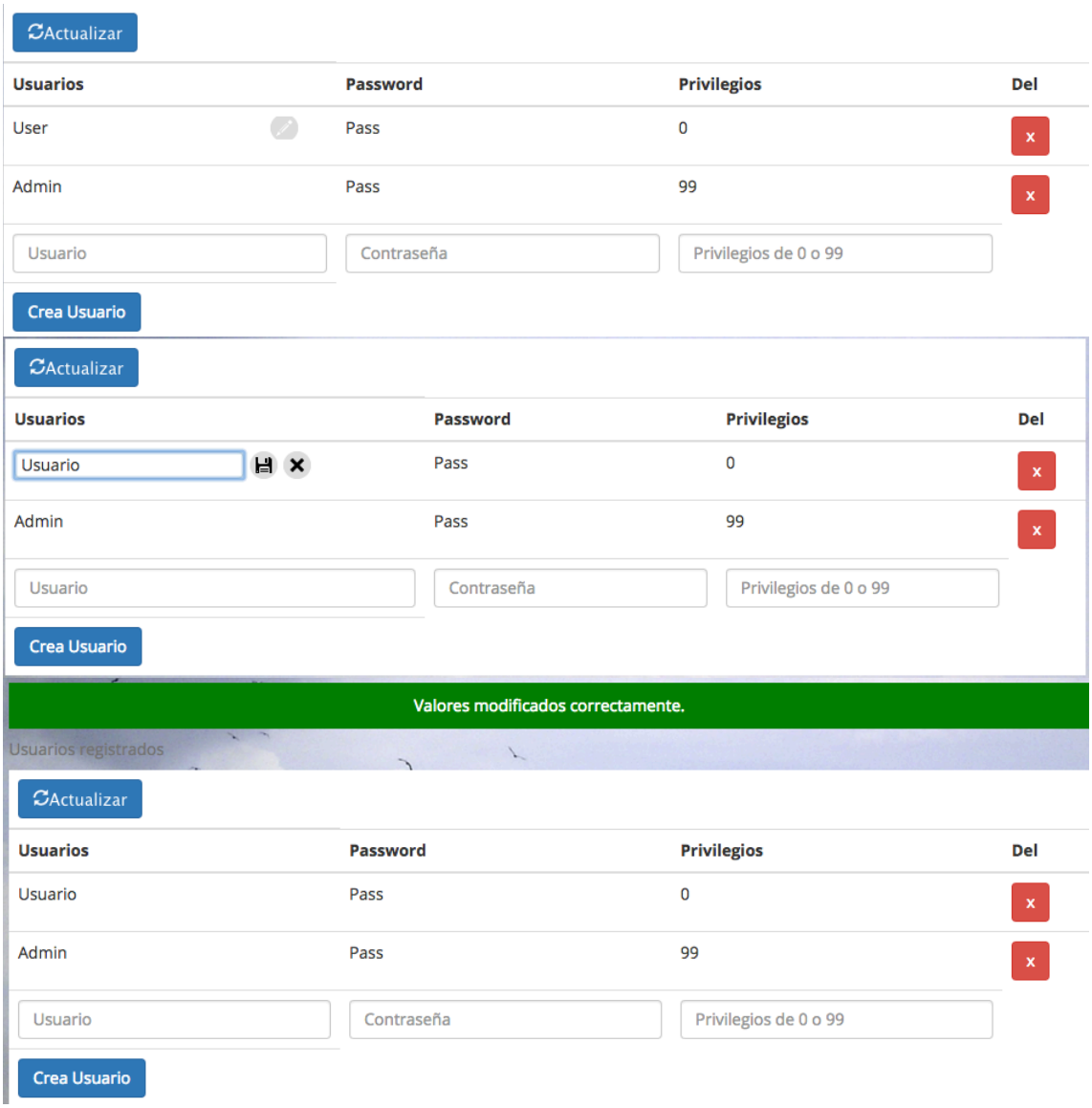


Figura 108 - Edición de usuarios y sus atributos en admin.*PHP*

Para ver la figura 109 se desactivó el servidor de base datos y se realizo una consulta. Así se ve el error que mostraría la aplicación si existe un error con la base de datos.

Figura 109 - Detalle de error en eliminación de usuarios en admin.*PHP*

Vemos en la sucesivas figuras mostradas a continuación las distintas interactuaciones posibles con la tabla de gestión de sensores. Primero observamos cómo se edita un dato. Más tarde cómo se crea un sensor y seguidamente cómo se elimina.

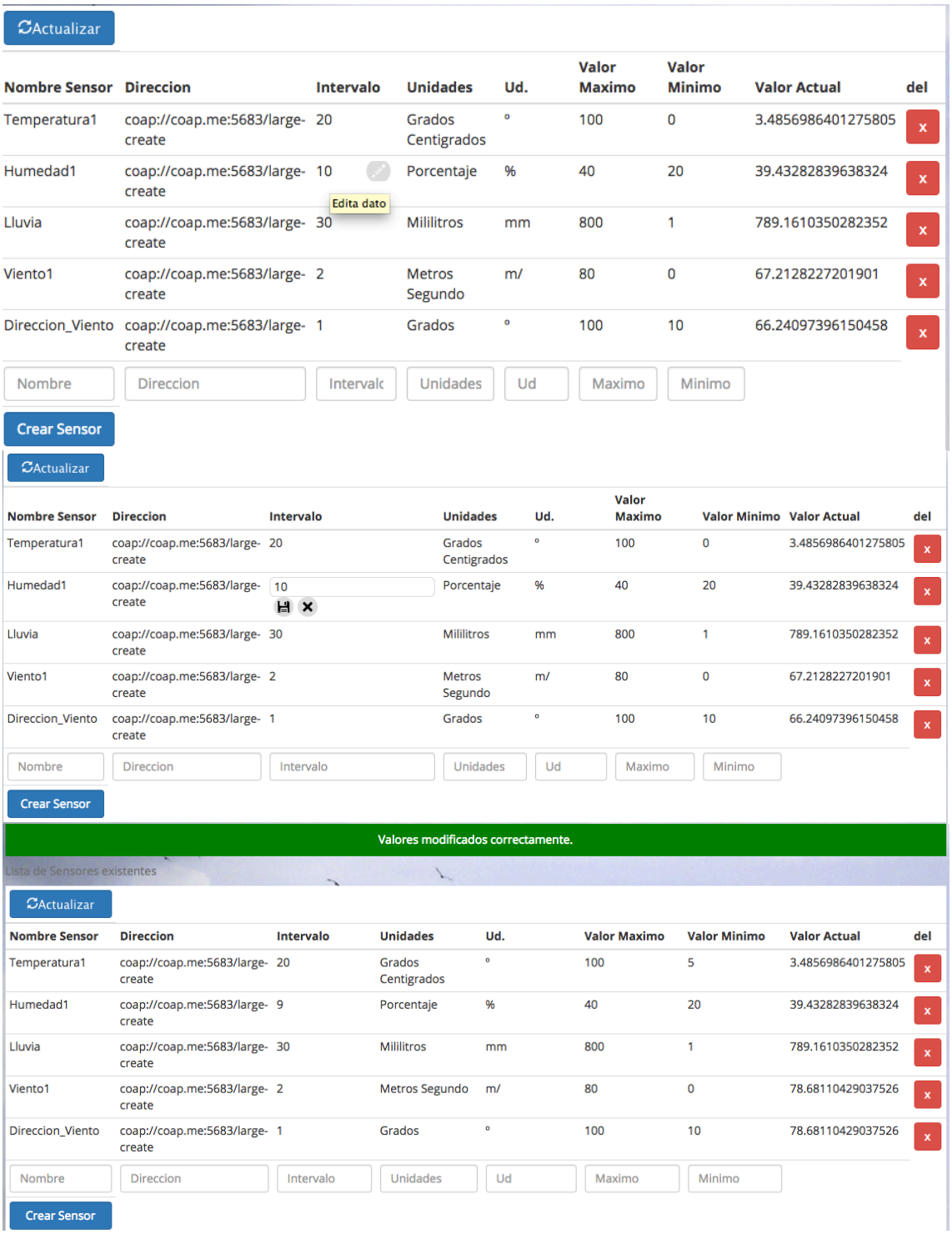


Figura 110 - Edición de sensores y sus atributos en admin.*PHP*

Todos los campos de creación de sensores como de usuarios se validan para que no se introduzcan datos erróneos en la base de datos, vemos un ejemplo en la Figura 111.

Figura 111 - Validación de datos en tabla Sensores de admin.*PHP*

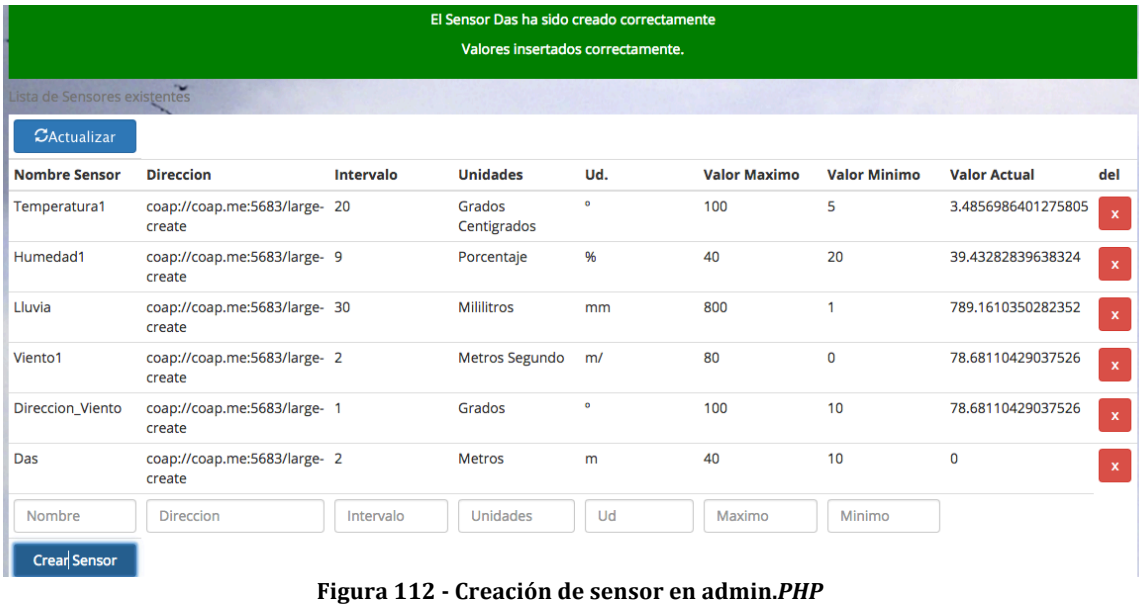


Figura 112 - Creación de sensor en admin.*PHP*

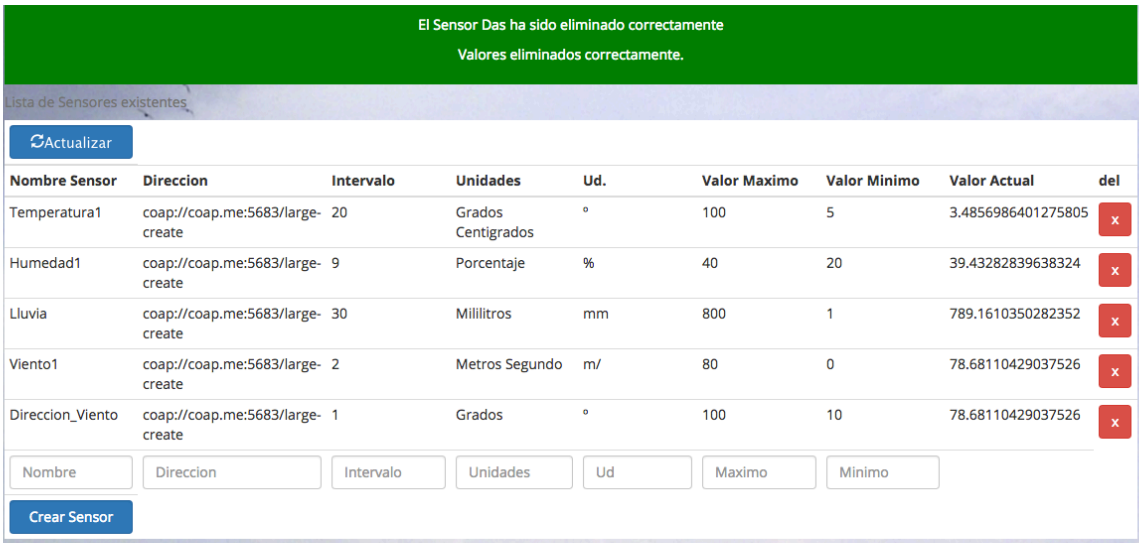


Figura 113 - Eliminación de sensor en admin.*PHP*

La figura 114 muestra el mensaje mostrado tras poner a cero con éxito el hardware de un sensor.

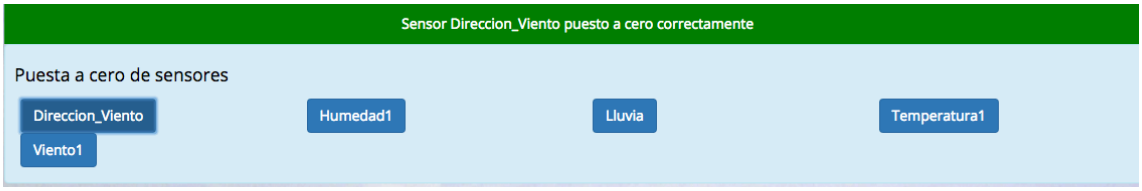


Figura 114 - Puesta a cero de sensores en admin.*PHP*

Aplicación *Android*

La aplicación *Android* tiene un modo de funcionamiento sencillo e intuitivo. Muestra todos los datos en el mismo bloque de visión. Primero muestra los datos actuales leídos en el momento de la carga, estos no se actualizan de manera automática sino mediante el botón *refresca datos actuales.*

Figura 115 - App *Android*

Seguidamente muestra unos botones, habrá tantos botones como sensores hay dados de alta en el sistema. Si queremos ver si se ha introducido algún sensor nuevo desde que se cargó la aplicación tenemos que pulsar el botón *Refresca sensores*.

Cuando pulsemos sobre cualquier botón de la lista de sensores, la app conectará con el servidor para pedir todos los valores que existen de ese sensor. Una vez la app recupera los datos, los muestra en una tabla con scroll y genera un gráfico con los valores recogidos. Si queremos ver los últimos datos actualizados de ese sensor solo tenemos que pulsar refresca datos de sensor.

Todos los botones de refresco, cargan los datos en cuestión sin tener que cargar los datos de toda la app.

Capítulo 7

Pliego de condiciones

Los medios necesarios para la realización del proyecto han de ser variados debido a la complejidad del sistema a realizar. El proyecto interactua con servidor web, navegador web, aplicación android, servidor de sensores mediante CoAP. Por lo que necesita una cantidad importante de hardware y software.

Hardware.

- Ordenador personal para el diseño de aplicaciones. Marca Apple, modelo IMAC 21", 8Gb de memoria RAM, 1Tb de HDD, procesador Intel i5 2,5Ghz y sistema operativo MAC OSX 10.9.5.
- Máquina servidora de gran potencia para albergar los distintos servicios necesarios para el funcionamiento del sistema global. La maquina mencionada como ordenador personal, sirve de maquina servidora.
- Infraestructura de red para conseguir seguridad en el sistema, al ser un servicio web, necesitamos instalar una DMZ para no tener vulneraciones de nuestros servicios. La DMZ consta del router Cisco ASA 5510 Firewall para interconectar la parte externa de la red (internet) y el router Cisco 2911 para hacer de puerta de enlace en la red interna del servidor.

- Dispositivo móvil con sistema operativo Android para realización de pruebas con al menos 500Mb de RAM, procesador doble nucleo de 1GHz y S.O. Android V4.3.

Software

- IDE (Entorno de desarrollo integrado) *Sublime* Text V3062 para diseño con el lenguaje de programación web dinámico *PHP*.
- IDE (Entorno de desarrollo integrado) *Eclipse Kepler* para el desarrollo del proxy bajo el lenguaje de programación java *java*.
- Librería *CoAP Californium V1.0*.
- Librería de *javascript JQUERY V1.11.1.*
- Librería de *javascript HighCharts V4.0.4.*
- *Librería de CSS bootstrap V3.3.4.*
- Servidor web *Apache V2.2.29*.
- Servidor de BBDD *MySql V5.6.25* con gestor *MyPhpAdmin V4.5.0*.
- Conversor de *HTML5 a Android Intel XDK V2170* .
- Software ofimático *Microsoft Office 2011 para MACOSX*.

Capítulo 8

Presupuesto

En capítulo siguiente muestra la inversión que se tiene que realizar para poner en funcionamiento el proyecto realizado en este trabajo fin de grado. El presupuesto se divide en varias partidas. Las partidas son:

- Estudio del sistema
- Instalación de las herramientas para el desarrollo del sistema.
- Diseño del sistema.
- Creación y compilación del sistema.
- Otros.
- Total.

En los siguientes epígrafes se desglosan las subpartidas contenidas en cada partida así como su precio y descripción.

Estudio del sistema

Se muestra a continuación el presupuesto para iniciar el proyecto. Es la fase en la que se estudian todos los lenguajes de programación, librerías y software necesario para la realización del proyecto.

Descripción	Ud	Precio Unitario	Precio Total
Horas de estudio del protocolo CoAP para comprender su funcionamiento. Busqueda de información de distintos modos de comunicación de *CoAP* para elegir el más adaptatado a nuestro proyecto.	25	40€	1000€
Horas destinadas al estudio de lenguajes de programación para la realización del *Proxy*. Realización de pequeñas pruebas con distintos lenguajes de programación que implementan librerías *CoAP* con el fin de elegir el lenguaje más adaptado a nuestras especificaciones	30	40€	1200€
Horas de estudio de lenguajes de programación dinámicos del lado servidor. Instalación y pruebas con los dos principales lenguajes elegidos, PHP y Ruby on Rails. Realización de pruebas para la elección del lenguaje más adaptado a nuestro proyecto.	60	40€	2400€
Horas dedicadas al estudio y diseño de la aplicación web a realizar para la supervisión y administración del sistema de sensores	15	40€	600€
Horas de estudio de librerías *javascript* para realización de gráficos. Realización de pequeñas pruebas para elegir la librería más útil en nuestro proyecto.	10	40€	400€
Horas dedicadas al estudio de librerías *javascript* para facilitar la ejecución de código en el lado cliente de la aplicación web.	5	40€	200€
Horas de estudio para comprender el patrón	22	40€	880€

Modelo-Vista-Controlador y adaptarlo a nuestro diseño de aplicación web.			
Horas de estudio para elección de software que realice la función de servidor web en nuestro sistema.	5	40€	200€
Horas de estudio para la elección del servidor de base de datos a utilizar en nuestro proyecto. Busqueda de especificaciones en distintos fabricantes de software y realización de distintas pruebas para realizar la elección.	36	40€	1440€
Horas de estudio de Sistema operativo Android para conocer especificaciones y formas de programar sobre éste sistema operativo.	30	40€	1200€
Horas de estudio de distintos frameworks de conversión entre HTM5 y aplicaciones Android. Realización de pruebas con distintos fabricantes para ver su versatilidad y funcionamiento.	55	40€	2200€
SUBTOTAL			**11720€**

Instalación de las herramientas para el desarrollo del sistema

Se muestra a continuación el presupuesto de las aplicaciones necesarias, así como su instalación. Estas aplicaciones son necesarias para diseñar y hacer operativo el proyecto.

Descripción	Ud	Precio Unitario	Precio Total
Horas de instalación y configuración de lenguaje de programación PHP.	5	40€	200€
Software PHP	1	0€	0€
Software Sublime text para el desarrollo de aplicación web.	1	30€	30€
Horas de instalación y configuración de servidor de base de datos MySql.	10	40€	400€
Software MySql	1	0€	0€
Horas de instalación y configuración de gestor de base de datos PHPMyAdmin.	5	40€	200€
Software PHPMyAdmin	1	0€	0€
Horas de instalación y configuración de Servidor web Apache.	5	40€	200€
Software Apache	1	0€	0€
Horas de instalación y configuración de IDE de diseño para Proxy Eclipse.	10	40€	400€
Software Eclipse	1	0€	0€
Horas de instalación y configuración de IDE de diseño para para Android XDK.	5	40€	200€
Software Eclipse	1	0€	0€
Librerías *javascript Jquery* y *HighCharts*	1	0€	0€
Librería *CoAP* para *java Californium*	1	0€	0€
SUBTOTAL			**1830€**

Diseño del sistema

A continuación se muestra el presupuesto del diseño de las distintas partes del projecto.

Descripción	Ud	Precio Unitario	Precio Total
Horas de diseño de base de datos.	5	40€	200€
Horas de diseño según patrón MVC de la aplicación web.	30	40€	1200€
Horas de diseño de servidor de base de datos.	5	40€	200€
Horas de diseño del proxy.	25	40€	1000€
Horas de diseño de aplicación *Android*.	20	40€	800€
Horas de diseño de parte servidora para aplicación *Android.*	10	40€	400€
SUBTOTAL			**3800€**

Creación y compilación del sistema

Se muestra a continuación el presupuesto de generación de código necesario para hacer funcionar la proyecto diseñado. En esta partida se incluye la creación del código, la generación y las pruebas de funcionamiento del programa creado.

Descripción	Ud	Precio Unitario	Precio Total
Horas de creación de base de datos. Incluye la creación de las tablas y atributos así como la conexión a los distintos sistemas empleados en el proyecto (*java y PHP*).	20	40€	800€
Horas de creación según patrón MVC de la aplicación web. Creación y pruebas de las distintas páginas web necesarias para las vistas de la aplicación, creación y pruebas del código con función controlador necesarias para comunicar las vistas con el modelo. Creación y pruebas del código necesario para hacer funciones de modelo. Incluye todo el díseño de la aplicación web, desde código ejecutable en la parte cliente (*javascript y HTML*) como en la parte servidora (*PHP*) así como creación de vistas mediante *CSS*.	120	40€	4800€
Horas de creación del proxy. Creación de código multihilo en *java* para la ejecución de lecturas *CoAP* al servidor de sensores. Así como funciones de lectura/escritura en BBDD. Creación de clases necesarias para realizar las labores diseñadas. Así como realización de pruebas para garantizar la estabilidad del proxy.	80	40€	3200€
Horas de creación de aplicación *Android*. Creación de vista en HTML5 para su conversión mediante XDK a aplicación *Android*. Pruebas de comunicación entre vista y controlador-modelo.	60	40€	2400€
Horas de creación de parte servidora para	60	40€	2400€

aplicación *Android.* Creación de controlador y modelo para comunicar con vista ejecutada en dispositivo Android. Pruebas en distintos telefónos.			
SUBTOTAL			**13600€**

Otros

Se muestra los dispositivos y programas necesarios para el funcionamiento del proyecto totalmente terminado.

Descripción	Ud	Precio Unitario	Precio Total
Ordenador Apple IMAC 21", 8Gb de memoria RAM, 1Tb de HDD, procesador Intel i5 2,5Ghz y sistema operativo MAC OSX 10.9.5. Incluida configuración inicial. Sirve como servidor y como maquina para desarrollo.	1	2000€	2000€
Dispositivo móvil Android de HTC Desire X con 768Mb de RAM y procesador doble nucleo de 1GHz y S.O. Android V4.3. Incluida configuración inicial del dispositivo e instalación de aplicación desarrollada.	1	300€	300€
Software Microsoft Office.	1	400€	400€
Software Firewall profesional.	1	200€	200€
Software y hardware para copia de seguridad. Software Symantec Backup Exec y HDD 4TB.	1	800€	800€
Horas de experto en seguridad para configuración de red y ordenador. Creación de DMZ para asegurar el servidor. La DMZ consta del router Cisco ASA 5510 Firewall para la parte externa de la red (internet) y el router Cisco 2911 para hacer de puerta de enlace en la red interna. Incluye hardware y software necesario.	1	12000€	12000€
Horas de escritura de la memoria	60	40€	2400
SUBTOTAL			**18100€**

Gastos Anuales

A continuación se muestra el presupuesto de los gastos fijos anuales que son necesarios para el funcionamiento del servidor web.

Descripción	Ud	Precio Unitario	Precio Total
Alquiler de dominio.	1	60€	60€
Actualización Anual del Antivirus.	1	40€	1200€
Mantenimiento Anual del servidor.	1	900€	900€
Manteniemiento Anual de copia de seguridad.	1	400€	400€
Gastos de luz.	1	1900€	1900€
SUBTOTAL			**4460€**

Total

Partida	Subtotal
Estudio del sistema.	11720€
Instalación de las herramientas para el desarrollo del sistema.	1830€
Diseño del sistema.	3800€
Creación y compilación del sistema.	13600€
Otros.	18100€
Gastos del primer año.	4460€
Total	**53510€**

Capítulo 9

Conclusiones

El trabajo fin de grado ha sido creado para que sea un proyecto útil en futuras implementaciones con redes de sensores. Se ha pretendido realizar un sistema versátil, configurable y escalable, de modo que pueda ser utilizado con cualquier sensor que funcione con el protocolo *CoAP*.

Líneas Futuras

Todo proyecto deja vías abiertas para continuar mejorando. Este no es distinto. La líneas futuras de estudio son variadas.

Se puede mejorar la interacción entre el servidor *CoAP* y el *Proxy* mediante la opción *observe* del protocolo *CoAP*. Esta opción tiene que ser soportada tanto por el cliente *CoAP*(*Proxy*) como por el servidor de sensores *CoAP*. Con esta opción se mejoraría el rendimiento de las comunicaciones entre sistemas, debido a que el proxy no tendría que estar continuamente llamando al servidor para conseguir datos actuales. Aunque el servidor no tenga un nuevo dato, el *proxy* cada tiempo programado tiene que recuperar el dato del servidor. Con la función *observe,* es el servidor el que manda el dato, pero solo cuando este dato se ha modificado por el sensor encargado de ello. Como percibimos, esto evita una ingente cantidad de transito de datos entre el *proxy* y el servidor *CoAP*.

La aplicación *Android* es mejorable incorporando workers en *JavaScript* para el procesamiento de grandes cantidades de datos. Al trabajar con el muestreo de todos los datos, la recepción de datos y conversión de *JSON* es pesada y deja la aplicación en un estado latente hasta que trata todos los datos. Ese proceso lo debe de hacer un hilo de ejecución. Otra solución es mostrar solo los últimos 100 datos.

La aplicación *Android* mejoraría también creando un worker que autoreferesque los datos actuales, como hace la aplicación web general.

La aplicación web se puede mejorar afinando las validaciones de datos. Por ejemplo, al eliminar usuarios, ahora mismo deja eliminar todos los existentes. Se tiene que tener la precaución de no eliminar el último administrador que quede. Si esto ocurre, no se puede seguir administrando la web. Validar las direcciones *CoAP* introducidas de manera precisa es otra línea futura de estudio.

El *Proxy* genera muchos datos, esto al cabo del tiempo, hace imposible el tratamiento de todos los datos mediante *JavaScript*. La solución propuesta como línea futura, es crear una tabla intermedia con los datos de un solo día. Al pasar el día, se hace la media de ese día y se guarda en la tabla de datos históricos. Cuando empiece un nuevo día, los datos de la tabla día se borrarán, dejando solo los datos estadísticos del día. De esta forma, si tomamos datos cada segundo, no nos genera 1440 datos al día cada sensor. El dato actual está actualizado al segundo; pero el histórico guarda solo la media.

La base de datos se puede optimizar aun más, es otra línea de estudio interesante. Además de incrementar la seguridad, como por ejemplo cifrando las contraseñas de los usuarios.

La seguridad en los sistemas web es fundamental, es necesario un estudio a fondo de este aspecto tanto en la aplicación web como en la app *Android*. Además de la creación de una red DMZ que asegure la confidencialidad de la red donde se instale el sistema.

El *Proxy* puede mejorar su eficiencia de dos maneras: una la hemos explicado en los párrafos anteriores, (el uso de la opción observe). Otra mejora de la eficiencia se consigue parando solo el hilo afectado de una modificación. Sabemos que cuando desde la aplicación web se modifica un valor, se cambia el campo de la BBDD M*odifica* para indicar al *Proxy* la recarga de los hilos. Una manera de mejorar esto es parar solo el hilo que se ha modificado y dejar los demás funcionando. Si creamos un sensor nuevo: lo mismo, es decir, incluir el sensor a la lista de hilos y dejar los demás hilos como están funcionando.

Capítulo 10

Bibliografía

BBDD

Resumen base de datos

[1]https://dinahosting.com/hosting/bases-de-datos

Administración de Base de datos *MySQL*

[2]http://manuales.guebs.com/MySQL-5.0/MySQL-database-administration.HTML#MySQLd-safe

Instalar configurar *MySQL* y *PHP*MyAdmin en Mac OSX

[3]http://www.elguisante.com/2011/12/como-instalar-y-desinstalar-MySQL-y-PHPmyadmin-en-mac-osx-lion/

[4]http://www.ooscarr.com/nerd/elblog/2009/11/instale-MySQL-en-mac-os-x.PHP

[5]http://www.vichaunter.com/como-se-hace/resetear-contrasena-de-root-en-MySQL-de-mac
[6]http://tuxjm.net/docs/MySQL-basic.txt

Modificar contraseña Root en *MySQL*

[7]http://configuracionpordefecto.com/como-cambiar-la-contrasena-root-de-MySQL/

[8]http://www.cyberciti.biz/faq/MySQL-change-root-password/

Almacenar contraseñas en *MySQL*

[9]http://www.solingest.com/blog/almacenar-contrasenas-en-MySQL

Insertar automáticamente la fecha y hora en *MySQL*

[10]http://cambrico.net/MySQL/como-insertar-automaticamente-la-fecha-y-hora-en-MySQL

SQL - JOIN básico

[11]http://ariel.esdebian.org/27200/SQL-join-basico

Driver Java de *MySQL*

[12]http://dev.MySQL.com/downloads/file.PHP?id=456316

Curso *MySQL*

[13]http://MySQL.conclase.net/

ServidorWeb

Apache

[14]http://techtastico.com/post/instalar-Apache-PHP-MySQL-mavericks/

Lenguaje web dinámico

Ruby on Rails

[15]http://www.rubyonrails.org.es/

[16]https://carlossanchezperez.wordpress.com/2013/05/19/mi-guia-de-ruby-las-clases-y-curiosidades-oop/

[17]https://rvm.io/rvm/install

[18]https://www.macupdate.com/app/mac/11136/macgpg2/download

[19]https://github.com/twbs/bootstrap-sass

[20]http://rbblog.foxandxss.net/desplegando-rails-parte-3-passenger

[21]https://www.railstutorial.org/book/filling_in_the_layout

[22]http://www.tutorialspoint.com/ruby/ruby_tutorial.pdf

[23]Ruby on rails Desarrollo práctico de aplicaciones web
editorial: Rc libros

Autor: Santiago Ponce Moreno.

PHP

[24]http://PHP.net/manual/es/

[25]http://programacion.net/

[26]Desarrollo Web con *PHP* 6, *Apache* y *MySQL*
Editorial Anaya.

Autores: Timothy Boronczyk, Elizabeth Naramore, Jason Gerner, Yann Le Scouarnec, Jeremy Stolz, Michael K. Glass.

Lenguajes *Proxy*

Ruby

*[27]**Ruby on rails Desarrollo práctico de aplicaciones web***

editorial: Rc libros

Autor: Santiago Ponce Moreno.

[28]http://www.tutorialspoint.com/ruby/ruby_tutorial.pdf

Java

*[29]**Java,Cómo programar Deitel***

Editorial Pearson.

Autores: Paul Deitel y Harvey Deitel.

[30]http://www.javaya.com.ar/
[31]https://docs.oracle.com/javase/tutorial/
[32]http://www.ctr.unican.es/asignaturas/procodis_3_II/Doc/Procodis_3_01.pdf
[33]http://www.elclubdelprogramador.com/2012/01/18/ide-importar-proyectos-al-Eclipse/
[34]http://www.chuidiang.com/java/MySQL/EjemploJava.PHP
[35]http://jarroba.com/arraylist-en-java-ejemplos/
[36]http://codejavu.blogspot.com.es/2013/06/ejemplo-conectando-java-con-MySQL.HTML

Ejecutar Applet Java en *PHP*

[37]http://PHP-java-bridge.sourceforge.net/doc/installation.PHP
[38]http://www.adictosaltrabajo.com/tutoriales/puente-PHP-java/

Python

[39]https://www.python.org/

HTTP-*CoAP Proxy* using WebSocket and FastCGI.

[40]https://github.com/AleLudovici/CoAPXY/blob/master/examples/CoAP_client.c

CSS

[41]http://www.desarrolloweb.com/manuales/CSS3.HTML

[42]http://librosweb.es/CSS/capitulo_7.HTML

[43]http://www.Ajaxload.info/

bootstrab

[44]http://getbootstrap.com/

[45]https://librosweb.es/libro/bootstrap_3/

[46]http://www.mediavida.com/foro/dev/tutorial-bootstrap-para-principantes-487865

[47]http://bootstrapbay.com/blog/working-bootstrap-contact-form/

[48]http://www.anidocs.es/bootstrap/docs/scaffolding.PHP

[49]http://ksesoCSS.blogspot.com/2012/05/centrando-al-centro-con-CSS-16-maneras.HTML

Android

Android, Guia para desarrolladores

Editorial Anaya.

Autores: W.Frank Ableson, Robi Sen y Chis King.

Framework *HTML*5+*CSS*+*JavaScript*→*Android*

[50]http://ivanprego.com/diseno-web/CSS/desarrollar-aplicaciones-para-ios-y-Android-con-HTML5-CSS-y-JavaScript/

[51]http://diseñorapido.es/frameworks-para-crear-aplicaciones-Android-en-HTML/

[52]http://www.lewebmonster.com/como-convertir-una-aplicacion-web-HTML5-a-Android-y-ios-facilmente/

[53]http://programacion.net/articulo/10_framework_para_desarrollar_aplicaciones_para_dispositivos_moviles_basadas_en_HTML_498

[54]http://www.campusmvp.es/recursos/post/PhoneGap-o-Apache-Cordova-que-diferencia-hay.aspx

Intel XDK

[55]https://software.intel.com/es-es/HTML5/tools

Android Nativo

[56]http://programadorygeek.blogspot.com.es/2011/11/HTML5-crear-una-aplicacion-nativa-de.HTML

BBDD y *Android*

[57]http://moztrodev.blogspot.com.es/2013/10/hazlo-facil-conectar-aplicacion-Android.HTML

[58]http://picarcodigo.blogspot.com.es/2014/05/webservice-conexiones-base-de-datos.HTML

JavaScript

[59]http://cybmeta.com/Ajax-con-JSON-y-PHP-ejemplo-paso-a-paso/

[60]http://www.martiniglesias.eu/blog/editar-campos-de-formularios-en-el-lugar-con-jquery-Ajax-y-PHP/225

[61]http://egatuts.com/2013/como-crear-tablas-editables-con-jquery/

[62]http://www.lewebmonster.com/manipulacion-de-tablas-con-jquery/

JQuery

[63]http://api.jquery.com/jquery.getJSON/

Ajax

[64]http://librosweb.es/libro/Ajax/

JSON

[65]http://api.jquery.com/jquery.getJSON/

[66]http://www.lawebdelprogramador.com/foros/JavaScript/1431086-consulta-SQL-con-Ajax-pero-usando-mas-de-una-variable-como-condicion.HTML

[67]http://hiddentao.github.io/squel/index.HTML

WebWorkers

[68]http://www.HTML5rocks.com/es/tutorials/workers/basics/

Gráficos

[69]http://programacion.net/articulo/clases_en_PHP_para_pintar_graficos_384

[70]http://solutoire.com/plotr/
[71]http://omnipotent.net/jquery.sparkline/#s-docs
[72]http://jsfiddle.net/gh/get/jquery/1.9.1/highslide-
[73]software/highcharts.com/tree/master/samples/highcharts/demo/line-basic/
[74]http://jpgraph.net/
[75]http://blog.vilourenco.com.br/como-montar-graficos-com-dados-dinamicos-em-HTML5-com-PHP-e-MySQL/
[76]http://www.highcharts.com/demo
[77]http://geekytheory.com/PHP-MySQL-highcharts-mostrar-grafica-dinamica-en-funcion-del-tiempo/

CoAP

Pagina oficial de *CoAP* tecnology

[78]http://CoAP.technology/

Ietf-core-*CoAP*-14 - The Constrained Application Protocol (*CoAP*)

[79]http://tools.ietf.org/HTML/draft-ietf-core-CoAP-14-section-1

Libreria *CoAP* para java: j*CoAP*

[80]https://code.google.com/p/jCoAP/

Libreria *CoAP* para *JavaScript*: node-*CoAP*

[81]https://github.com/mcollina/node-CoAP#basic

Libreria *CoAP* para Ruby: SmallLars/*CoAP*

[82]https://github.com/SmallLars/CoAP

Libreria *CoAP* para Twisted framework phyton: siskin/txThings

[83]https://github.com/siskin/txThings

Libreria *CoAP* para Python: aio*CoAP* 0.1

[84]https://pypi.python.org/pypi/aioCoAP/0.1

Libreria *CoAP* para Java : *Californium*

[85]https://github.com/mkovatsc/Californium
[86]http://people.inf.ethz.ch/mkovatsc/Californium.PHP

[87]https://unpocodejava.wordpress.com/2014/10/16/Californium-libreria-java-CoAP/

[88]http://stackoverflow.com/questions/28898501/CoAP-example-for-observe-is-not-working-in-Eclipse

[89]http://grepcode.com/file/repo1.maven.org/maven2/ch.ethz.inf.vs/Californium/0.18.7-final/ch/ethz/inf/vs/Californium/CoAPClient.java?av=f

Presentación: Hands-on with *CoAP*

[90]https://docs.google.com/presentation/d/1dDZ7VTdjBZxnqcIt6qoX742d6dHbzap-D_H8Frf3LRE/edit?pli=1 - slide=id.g38a4dd9e2_0172
[91]https://oracleus.activeevents.com/2014/connect/fileDownload/session/4DC51536A7239D69A8B34DBBA659696C/CON4803_Cabé-JavaOne%202014%20-%20IoT%20protocols%20jungle.pdf

[92]https://docs.google.com/presentation/d/1dDZ7VTdjBZxnqcIt6qoX742d6dHbzap-D_H8Frf3LRE/mobilepresent?pli=1&slide=id.g38a4dd9e2_014

Libreria *CoAP* para Java Eclipse Foundation: *Californium*

[93]https://github.com/Eclipse/?query=Californium

Copper *CoAP* app para Firefox
[94]https://addons.mozilla.org/es/firefox/addon/copper-270430/

Proyectos fin de carrera

*[95]***Uso de protocolo *CoAP* para implementación de una aplicación domótica con redes inalámbricas**
Jorge Castro Heredia
Universidad politécnica de Cartagena.

*[96]***Desarrollo y estudio de protocolo Observe para *CoAP***
Xavi Gimeno Gimenez
Universitat politécnica de Catalunya.

Universidad de Alcalá
Escuela Politécnica Superior

Escuela Politécnica Superior

Universidad de Alcalá