UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA DE TECNOLOGÍAS Y SERVICIOS DE TELECOMUNICACIÓN

DISEÑO E IMPLEMENTACIÓN DE UNA APLICACIÓN WEB PARA EL CONTROL DE MOTORES DE CORRIENTE CONTINUA

VERÓNICA GONZÁLEZ PÉREZ

2017

TRABAJO FIN DE GRADO

Título: Diseño e implementación de una aplicación web para el control de motores de corriente continua

Autora: Dña. Verónica González Pérez

Tutor: D. Álvaro Gutiérrez Martín

Departamento: Departamento de Tecnología Fotónica y Bioingeniería (TFB)

TRIBUNAL:

Presidente:

Vocal:

Secretario:

Suplente:

Fecha de lectura:

Calificación:

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA DE TECNOLOGÍAS Y SERVICIOS DE TELECOMUNICACIÓN

DISEÑO E IMPLEMENTACIÓN DE UNA APLICACIÓN WEB PARA EL CONTROL DE MOTORES DE CORRIENTE CONTINUA

VERÓNICA GONZÁLEZ PÉREZ

2017

Resumen

En el presente Trabajo Fin de Grado se ha buscado una alternativa a la actividad presencial desarrollada en el Laboratorio de Robótica y Control del Departamento de Tecnología Fotónica y Bioingeniería promoviendo el acceso remoto a un puesto del mismo a través de una aplicación web.

Con este software conseguimos el intercambio seguro y controlado de información entre usuarios y un puesto físico del laboratorio, además de ofrecer al usuario el acceso desde cualquier dispositivo.

Dicha aplicación web se compone de tres partes: Comunicación, Configuración y Gestión, aportando distinto valor y distintas funcionalidades a la misma. Cada una de las partes se ha desarrollado con tecnologías diferentes teniendo en cuenta las que mejor se adaptan al contexto y a la finalidad y funcionalidades respectivas ya que existen multitud de tecnologías en la actualidad.

Dicha división de la aplicación en tres áreas nos permite poder disponer de dos versiones, una completa (con usuarios registrados y con todas las funcionalidades) y otra parcial (sin registros y sin necesidad de base de datos, pero con funcionalidades limitadas).

Al finalizar el desarrollo de la aplicación, se ha adjuntado un manual para su instalación y para posibles modificaciones que se requiriesen durante su mantenimiento.

Palabras clave

Aplicación web; acceso remoto; gestión; comunicación; PHP; Slim; framework; Websocket; Manual; Laboratorio; Robótica; Motor; Ansi-C;

Summary

In the following Final Degree Project, it has been explored an alternative to the face-to-face activity developed in the Laboratory of Robotics and Control of the Department of Technology Photonics and Bioengineering, promoting remote access to one of its workplaces through a web application.

Thanks to this software we can achieve the secure and controlled exchange of information between users and a physical workplace of the laboratory, besides offering these users the access from any device.

This web application is composed of three parts: Communication, Configuration and Management, providing each of them a different value and different functionalities to the whole application. Each of the parts has been developed with distinct technologies, taking into account those that best fit the context and purpose, as well as their respective functionality, since there are many technologies nowadays.

This division of the application in three areas allows us to have two versions at our disposal, one complete (with registered users and with all the functionalities) and another partial (without registries and without the need of a database, but with limited functionalities).

At the end of the development of the application, a manual has been attached for its installation and for possible modifications that could be required during its maintenance.

Keywords

Web application; Remote access; Management; Communication; PHP; Slim; Framework; Websocket; Manual; Laboratory; Robotics; Engine; Ansi-C.

Índice

1 INTRODUCCIÓN

1.1 MOTIVACIÓN

En la actualidad, debido al gran avance de la tecnología (Internet es actualmente el mayor medio de comunicación y transmisión de información), nos encontramos frente a una gran tendencia a la digitalización de nuestro día a día, desde la visión de negocio de las empresas en las que trabajamos hasta la forma en la que realizamos prácticamente cualquier tarea. Por ejemplo, plataformas como Amazon, Deliveroo, 21Butttons… tienen su gran negocio en que cada día la forma de llevar a cabo cualquier acción depende más y más de la tecnología.

Por éste motivo, el desarrollo de "robots" (como concepto de instrumento electrónico que realiza una función concreta), aplicaciones móviles, redes sociales, juegos interactivos, aplicaciones de escritorio y aplicaciones web es esencial en nuestros días.

Todos los sistemas mencionados previamente son necesarios y útiles para la digitalización y no se puede imaginar la vida actual con la ausencia de uno de ellos, pero en este proyecto, debido a la necesidad que se desea solventar y a las ventajas que el sistema aporta, se ha elegido el desarrollo de una aplicación web.

La tarea a resolver con la aplicación es el acceso y la manipulación de los parámetros de los controladores de un sistema de lazo cerrado para el control de motores de corriente continua.

Aunque se pretende implementar una tarea muy concreta, el TFG se podría generalizar para cualquier aplicación con acceso remoto a un sistema que necesita de unos parámetros para realizar su función que se van a proporcionar desde la aplicación. Es un TFG que podría solventar muchas más tareas manteniendo el diseño, la estructura y los protocolos y modificando exclusivamente los datos de comunicación que se deben proporcionar, a quién se le proporcionan y el formato en que se hace.

Especificando un poco más, la herramienta concreta que se va a desarrollar será destinada a su uso por una organización específica: el Laboratorio de Robótica y Control deldepartamento de Tecnología Fotónica y Bioingeniería (TFB) de la Escuela Técnica Superior de Ingeniería de Telecomunicaciones (ETSIT) de la Universidad Politécnica de Madrid (UPM); en particular por sus investigadores y alumnos y profesores de las asignaturas que el departamento imparte.

El hecho de que sea un acceso remoto es la fundamentación principal por la que la elección realizada ha sido de una aplicación web; para este tipo de accesos, las aplicaciones web aportan ciertas ventajas desde el punto de vista del usuario final frente a otros sistemas, las principales son:

❖ **Disponibilidad** de acceder a través de cualquier dispositivo con un **navegador web** y la consiguiente **independencia** de las características tecnológicas del mismo. Aportando así un servicio **multiplataforma**.

❖ Facilidad para el **teletrabajo** y el **acceso concurrente**.

❖ **Facilidad** en la gestión de **actualizaciones** y **ausencia de** la necesidad de una instalación con unos **requisitos** de librerías y software **concretos**. Además se generan **menor** número de *bugs* debidos a **conflictos** con otros softwares, protocolos o versiones. Y con una **menor** necesidad de **memoria** y, por tanto, **menor carga** en la máquina del usuario.

Por otro lado, la herramienta solventará un problema al que se enfrenta el departamento (igual que la mayoría de las organizaciones de índole similar): la disposición de puestos físicos en los que realizar experimentos es muy costosa además de necesitar disponer de espacio suficiente. Tener un puesto por cada una de las personas que forman permanente o temporalmente parte de la organización es imposible; problema que actualmente se intenta resolver con una organización de horarios (cosa complicada ya que la necesidad de realizar un experimento es solo parte de la labor del personal y surge en momentos impredecibles la mayoría de las veces) y a través de una aplicación de escritorio desarrollada por personal del departamento, pero con las desventajas que ésta conlleva de peso, actualizaciones e incompatibilidades entre otras.

2.1 CONTEXTO: EL LABORATORIO

Como se ha mencionado anteriormente, la aplicación a desarrollar está destinada para el Laboratorio de Robótica y Control del Departamento TFB, por lo que se debe conocer los trabajos que en él se llevan a cabo, los dispositivos electrónicos que se van a utilizar y la aplicación de escritorio disponible, para que con dicha información junto con la de las nuevas funcionalidades que se quieren implementar, se pueda realizar y especificar el análisis de requisitos que permita elegir cómo desarrollar la aplicación web.

1.1.1. El laboratorio físico

Entre otras finalidades, en éste laboratorio se llevan a cabo dos tipos de experimentos con motores de corriente continua: los que corresponden al conocido entre el personal como *"RealLabo"* que deben realizarse personalmente en los 10 puestos que están dispuestos para dicho fin con un alimentador, un conjunto de Motor-Encoder-Reductora, un arduino y una tarjeta de motores; los otros son los que corresponden al conocido como *"TeleLabo"* en el que se pretende que se opere a distancia para realizar un estudio experimental de un controlador y de la señal de saturación. Para este último tipo de experimentos se utilizará la herramienta de este proyecto y físicamente en el laboratorio se dispone de un único puesto de trabajo con dos fuentes de alimentación, una de 5V y otra de 12V con la correspondiente protección; además se dispone de un conjunto Motor-Encoder-Reductora que se puede ver en la Figura 1a extraída de [1] y el hardware apropiado para el control de los motores (una tarjeta con un microcontrolador ARM con capacidad para cuatro motores, otra tarjeta con un procesador ATMEL con Linux y una última tarjeta encargada de la comunicación periférica) que se puede observar en la Figura 1b. Además el puesto está provisto con una cámara web conectada a la tarjeta de control de periféricos para visualizar el movimiento del motor en directo como puede verse en la Figura 1c.

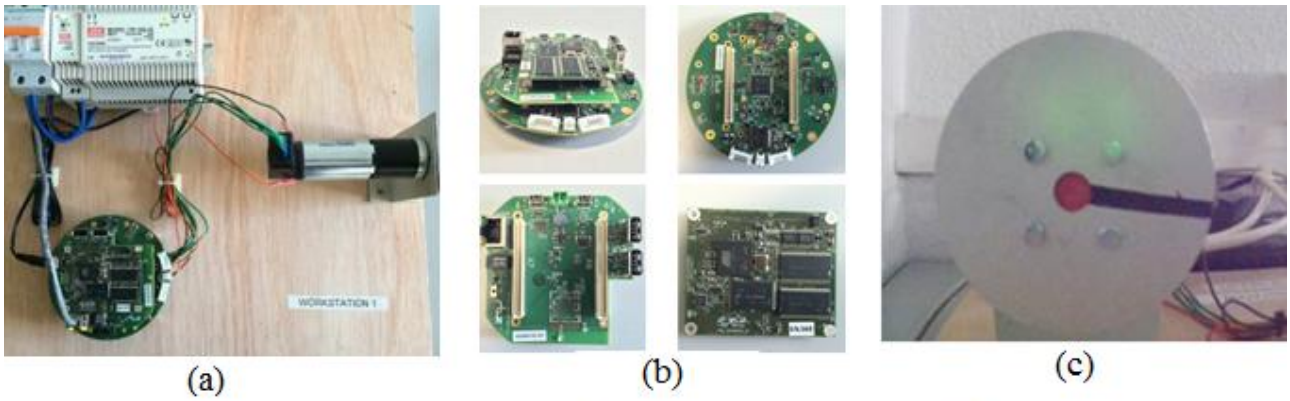

Figura 1. Material electrónico y mecánico disponible en el Laboratorio de Robótica y Control

En cuanto al conjunto Motor-Encoder-Reductora, en concreto se dispone de:

- Motor DC con escobillar **A-max 32 12V**

- Reductora Planetaria **GP 32A 23:1**
- Encoder **HEDS 5540 de 500 pulsos por vuelta**

Sin ahondar mucho en las características de los dispositivos ya que no es la finalidad del TFG, en la Figura 2 extraída de [1] se muestra el esquema del *"Firmware"* de la tarjeta de motores.

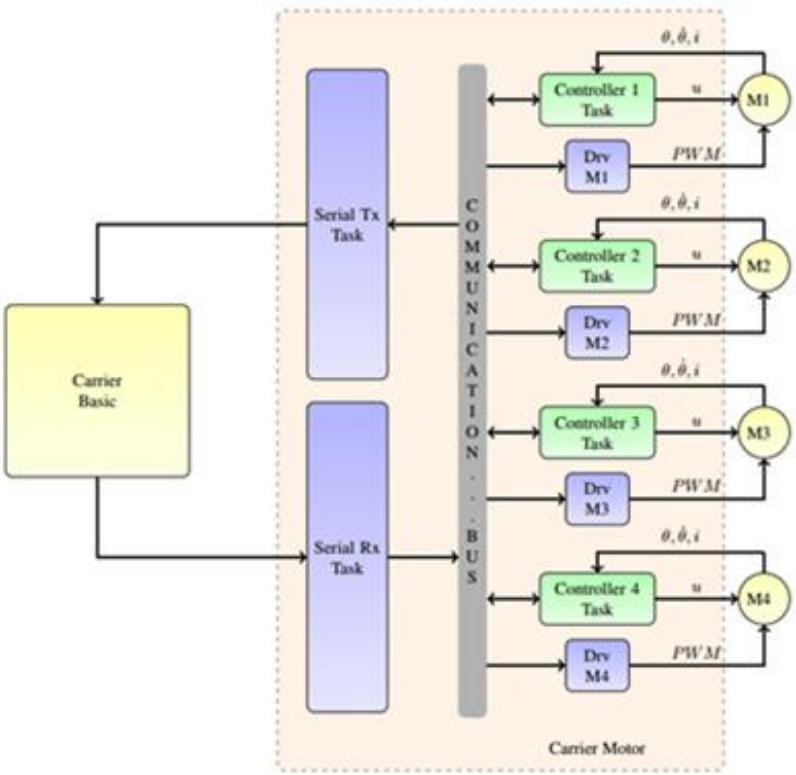

Figura 2 Firmware de la tarjeta de motores

Como puede verse en la Figura 3, la estructura de cada uno de los motores es sencilla, le entran la señal del controlador y de sus drivers que provienen del bus de comunicaciones, su salida realimenta el controlador que la trata y la devuelve al bus.

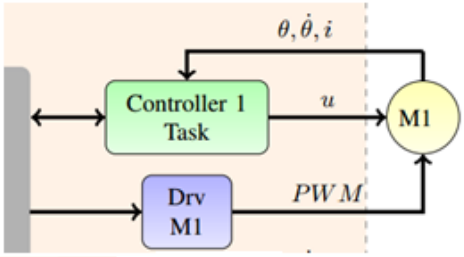

Figura 3 Firmware de un solo motor

A continuación la interconexión de los cuatro motores se hace de manera serializada para la entrada y para la salida, información que pasa por el carrier oportuno.

En cambio, a pesar de la posibilidad de trabajar con los cuatro motores, el planteamiento del *"TeleLabo"* y del puesto físico están hecho para trabajar siempre sobre el motor3, por lo tanto ésta estructura no es relevante de cara a la herramienta ya que para ella solo "existirá" dicho motor.

Para terminar la descripción física del puesto se añade un esquema del *"Firmware"* del mismo en la Figura 4.

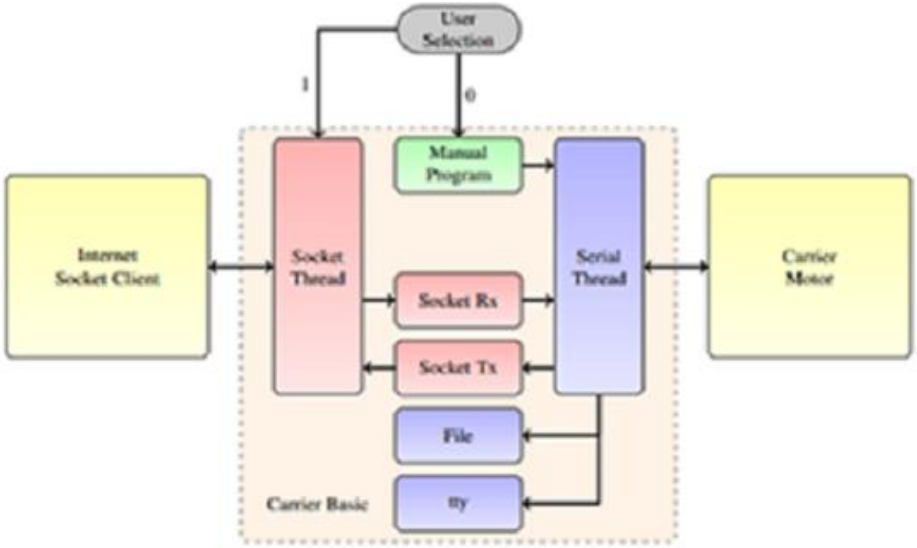

Figura 4 Firmware del puesto del laboratorio

Antes de estudiar la aplicación de escritorio ya existente, se considera relevante conocer algunos conceptos teóricos para entender cada una de las variables que se deben definir para ejecutar un experimento concreto, para ello se define la Tabla 1 y la Figura 5.

PARÁMETROS DE COMUNICACIÓN		
NOMBRE	SEÑAL	UNIDADES
Position	θ(t)	rad
Speed	$\dot{\theta}$ (t)	rad/s
Current	i(t)	A
Reference	r(kT)	rad o rad/s
Error	e(kT) = r(kT) – y(kT)	
Error_ref_fb	$\grave{e}$ (kT) = r(kT) – u_{FB} (kT)	
Control FeedForward	u_{FF} (kT)	V
Control Direct	u_{DI} (kT)	V
Control FeedBack	u_{FB} (kT)	V
Control Parallel	u_{PA} (kT)	V
Control	u(kT)	V
ControlSat	û(kT)	V
Output	y(kT)	rad o rad/s
Tx Period	Período de envío de datos	ms
Time Limit	Límite de ejecución del controlador	ms

Tabla 1

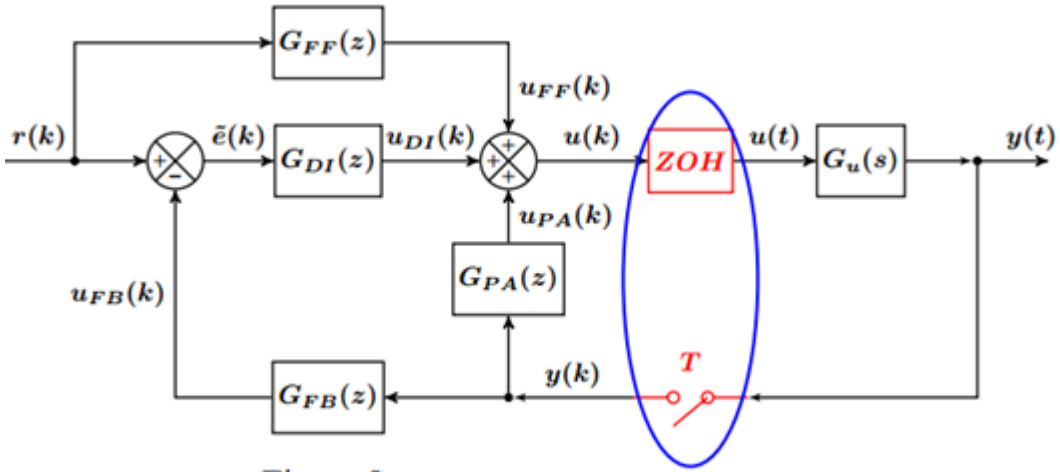

Figura 5 Esquema del circuito de los experimentos

En ellas se puede observar que tanto la velocidad es sólo función de la posición y que ésta junto con la corriente son independientes de la señal de referencia y de la salida, que son las que principalmente están relacionadas.

Afectada por las ganancias del circuito paralelo (G_{PA}) y de realimentación (G_{FB}) que actúan sobre la salida, de la de feedforward (aprendizaje sobre memoria) (G_{FF}) que actúa sobre la señal de referencia y de la directa (G_{DI}) que actúa sobre una diferencia entre la señal de referencia y la señal de salida afectada por G_{FB} y por la propia ganancia de salida (G_u), la salida (y(t)) es una combinación lineal de sí misma y de la señal de referencia establecida.

Comentar además que el error se utiliza según su propia definición como la diferencia entre la señal de referencia y la señal de salida. Nótese además que la señal de referencia es la señal de entrada al sistema como se puede observar en la Figura5.

Como se ha podido concluir del análisis de las variables, la señal de referencia es de gran importancia, por ello se resume en la Tabla 2 los tipos de señales que pueden usarse como referencia del controlador.

SEÑAL DE REFERENCIA DEL CONTROLADOR Y SUS VARIABLES					
SEÑAL	ECUACIÓN	VAR1	VAR2	VAR3	VAR4
Delta	r(kT)=A; k=0; r(kT)=0, ∀k=0	A			
Escalón	r(kT)=A	A			
Rampa	r(kT)=A · kT	A			
Parábola	$r(kT)=\frac{1}{2}\cdot A\cdot (kT)^2$	A			
Seno	r(kT)= A · sin(ω · kT)	A	ω		
Coseno	r(kT)= A · cos(ω · kT)	A	ω		
Trapezoidal	Imagen 8	A	t_1	t_2	t_3

Tabla 2

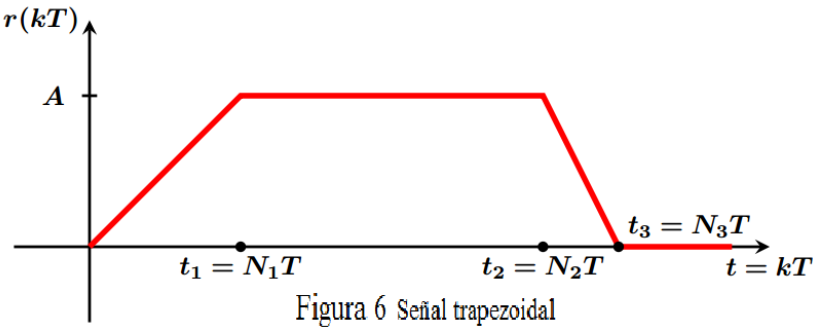

Figura 6 Señal trapezoidal

La señal Delta equivale a una constante (un valor) para t=0 y 0 para el resto de los valores; por su parte, la señal Escalón es una constante para todos los valores; la Rampa es una constante que afectada por la variable hace que el valor aumente al aumentar la variable y disminuya al hacerlo ésta y la parábola es una función curva que afecta a la constante con el cuadrado de la variable, por lo que estas cuatro funciones quedan definidas con indicar el valor de la constante.

En cambio Seno y Coseno dependen de una frecuencia de oscilación además de la constante que corresponde a la amplitud, por lo que se definen con dos variables cada una.

Por último, Trapezoidal como muestra la Figura 6 extraída de [1] necesita de una constante que define el valor máximo, el tiempo en que alcanza dicho valor, el tiempo que lo mantiene y el tiempo en que vuelve a valer cero, por lo que hacen falta cuatro variables para que quede bien definida.

Aunque se han descrito para la señal de referencia del controlador, dichos tipos de señales son válidos para la señal de perturbación que se puede sumar al experimento para hacerlo más realista.

Comentar también que "Sampling Period" corresponde con el periodo de muestreo en ms, que el tipo de controlado es PID único y por defecto, que las dos opciones de la variable de control son Posición o Velocidad y que la frecuencia PWM está fija a 20KHz.

Además las variables, conrrespondientes a saturación del windup, tanto de Direct como de Parallel, FeedBack y FeedForward son fijas con un valor de 12,00.

1.1.2. La aplicación de escritorio

Teniendo en cuenta que el experimento requiere de la configuración de todas las variables ya mencionadas, es fácil entender que la aplicación de escritorio posee una interfaz gráfica en la que se puedan configurar todos ellos, para ello se dispone de un panel de configuración compuesto por tres pestañas activas actualmente: Controller, Perturbation y Communication.

Además dicha interfaz dispone de un espacio donde se va a dibujar la gráfica con los datos obtenidos y de un área donde se va a ver a través de la web-cam el movimiento del motor al ejecutar el experimento que se le ha enviado. También, como puede observarse en Figura 7 que contiene capturas de pantalla de la aplicación, las tres comparten un área común en el que se puede visualizar el botón para el envío, el orden de la cola y el estado en el que se encuentra nuestro experimento.

En la interfaz se puede ver que hay otros servicios implementados como la autenticación pero que están deshabilitados en la actualidad. Lo que se encuentra activo son las funcionalidades mínimas que deben implementarse en la nueva aplicación, además de aquellas otras que se decidan ampliar considerando la finalidad de la aplicación y el entorno que se ha comentado en el que se utilizará.

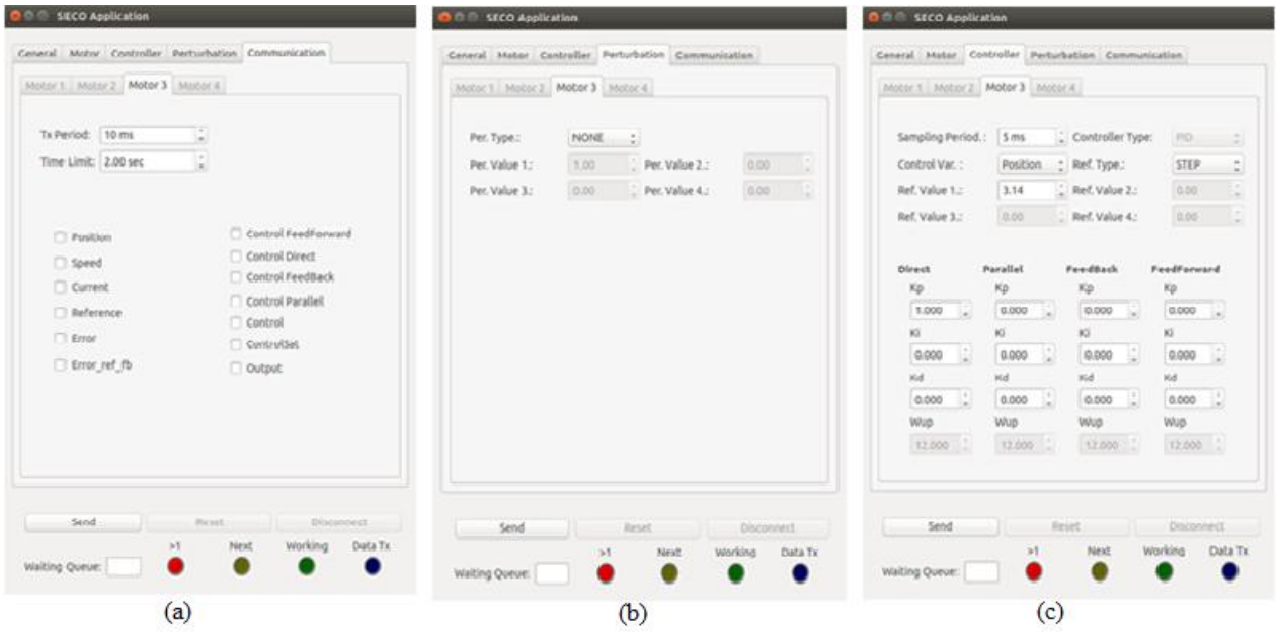

Figura 7 Aplicación de Escritorio

Esta aplicación al ser de escritorio está limitada en aspectos como que se debe ejecutar sobre Linux, ocupa un espacio considerable de la memoria del equipo en el que se almacena y posee dependencias de librerías como *"make"* o *"libqt4-dev"*.

3.1 ANÁLISIS DE FUNCIONALIDADES Y REQUISITOS

La aplicación va a ser utilizada por dos tipos de cargos dentro del departamento, los profesores (u otro tipo de personal del departamento que hará las funciones de gestión del "TeleLabo") y los alumnos. Es por ello que la aplicación debe proporcionar funcionalidades diferentes según dos roles que coinciden con los cargos mencionados: administrador y usuario; aunque hay un servicio común: autenticación correspondiente al caso de uso mostrado en la Figura 8. (1F)

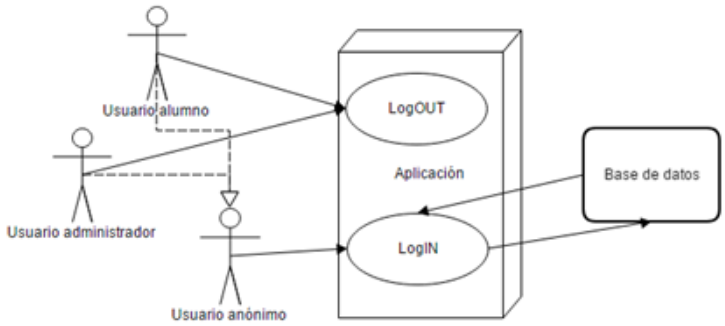

Figura 8 Caso de uso de autenticación

Analizando las acciones que podría desear realizar un usuario del sistema con rol de administrador (un profesor o similar) una vez se haya logueado, se definen los servicios listados a continuación, que se muestran como casos de uso en la Figura 9:

- CRUD (Create, Read, Update, Delete – Crear, Leer, Actualizar, Borrar) de un usuario. (La lectura es de todos los usuarios existentes). (2F)
- Asignar un experimento a un usuario concreto. (3F)
- Editar un experimento asignado a un usuario. (4F)
- Borrar todos los usuarios con rol de usuario al finalizar una asignatura o un proyecto sin necesidad de irlo haciendo uno por uno. (5F)
- Agregar un grupo de usuarios con rol de usuario desde un fichero de texto al iniciar una asignatura o un proyecto sin necesidad de irlo haciendo uno por uno. (6F)

Valorando las acciones que podría desear realizar un usuario del sistema con rol de usuario (un alumno) una vez se haya logueado, se definen los servicios listados a continuación, que se muestran como casos de uso en las Figura 10a y 10b:

- CRUD (Create, Read, Update, Delete – Crear, Leer, Actualizar, Borrar) de un experimento. (La lectura es de todos sus experimentos existentes). (7F)
- Cargar un experimento almacenado en la base de datos en la configuración para ejecutarlo. (8F)
- Cargar un experimento almacenado en un fichero en la configuración para ejecutarlo. (9F)
- Descargar a un fichero la configuración de un experimento. (10F)
- Configurar manualmente un experimento para ejecutarlo. (11F)
- Ejecutar (o enviar para que se ejecute) un experimento. (12F)
- Descargar a un fichero la gráfica con los datos de la ejecución. (13F)
- Cargar de un fichero los datos y dibujarlos en la gráfica. (14F)
- Visualizar el movimiento del motor durante la ejecución de un experimento. (15F)
- Visualizar la posición que se ocupa en la cola y el estado de la comunicación. (16F)

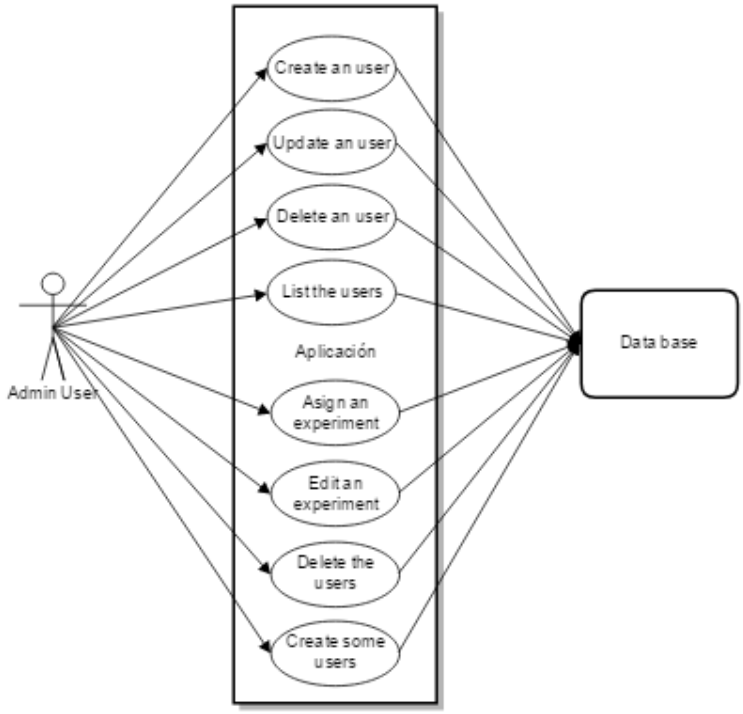

Figura 9 Casos de uso de administrador

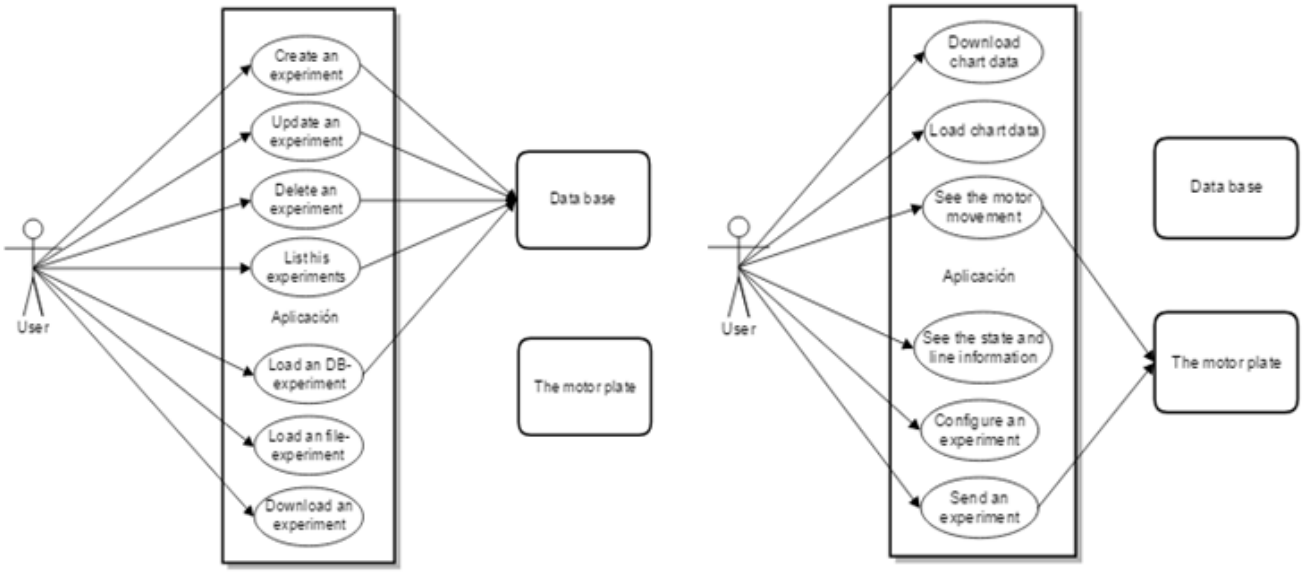

Figura 10 Casos de uso de usuario

Hasta ahora se han descrito los casos de uso de cada posible actor del sistema, pero toda aplicación tiene además unos requisitos técnicos que provienen de las características de la misma. En este caso, por ejemplo, es importante que la aplicación permita la solicitud concurrente de peticiones de ejecución sin que esto produzca errores en la ejecución.

También es relevante que aunque no nos encontramos ante un sistema que requiera de una comunicación en tiempo real, no se trata de un sistema de tráfico habitual como puede ser el de la difusión de noticias, ya que la transmisión imprescindible de una cámara web es tráfico multimedia con cierta sensibilidad al retardo. Por lo tanto, la aplicación no puede ser muy pesada para que no se convierta en una herramienta lenta.

Otro aspecto importante a tener en cuenta es que el servidor que aloje la aplicación es un servidor de un departamento con otros servicios, por lo que deberá ser un sistema que pueda convivir en el actual entorno de la organización y que no cargue en exceso el mismo.

Por otro lado, puesto que el sistema debe ser exportable y reutilizable para cualquier sistema de acceso remoto, el entorno deberá ser intuitivo y sencillo en su uso para que pueda utilizarlo personal sin conocimientos técnicos.

De los últimos comentarios se puede extraer que como requisitos no funcionales del sistema tenemos:

- Fácil mantenimiento. (1NF)
- Tablas de tamaño reducido para una correcta convivencia con el resto de herramientas existentes en el servidor donde se alojará. (2NF)
- Gestión de los datos en los accesos concurrentes. (3NF
- Uso intuitivo y sencillo. (4NF)
- Confiable y robusto. (5NF)
- Adaptable a cambios de hardware. (6NF)
- No necesita ser una aplicación en tiempo real pero para la correcta visualización de la imagen no debe haber un gran retardo. (7NF)
- A nivel legal se deberán utilizar licencias libres. (8NF)

Las funcionalidades y los requisitos expuestos están identificados con un código entre paréntesis en su enumeración para poder referirse a ellos fácilmente en la siguiente lista en la que se ordenan por prioridad y distinguiendo entre imprescindibles y deseables.

- Requisitos no funcionales:
  - Imprescindibles y de prioridad decreciente:
    - 8NF
    - 1NF
    - 5NF
    - 2NF
    - 3NF
  - Deseables y de prioridad decreciente:
    - 4NF
    - 7NF
    - 6NF
- Requisitos funcionales:
  - Imprescindibles y de prioridad decreciente:
    - 11F
    - 12F
    - 16F
    - 15F
  - Deseables y de prioridad decreciente:
    - 7F
    - 8F
    - 13F
    - 14F
    - 10F
    - 9F
    - 1F
    - 2F
    - 3F
    - 4F
    - 5F
    - 6F

Por último, explicar que existe un proyecto que se inició en un pasado y está parado actualmente en el que se empezó a desarrollar una aplicación que simulase las funcionalidades básicas como las que provee la aplicación de escritorio explicada anteriormente; dicho código se reutilizará en la medida que sea posible para que el tiempo que se invirtió no sea perdido y agilizar el desarrollo actual.

2 METODOLOGÍA Y PLANIFICACIÓN DEL DESARROLLO

En este capítulo se fija la metodología de trabajo que se va a seguir durante el proyecto; se analizarán los puntos de "ataque" a la aplicación para dividirlos en diferentes tareas y se estimará el tiempo que se va a invertir en cada una de una manera justificada.

4.1 METODOLOGÍA GENERAL

Adaptando a la ejecución individual del proyecto la metodología ágil de trabajo SCRUM, se va a trabajar dividiendo el proyecto en iteraciones de tiempo relativamente estables y breves en las que una tarea sea desarrollada por completo e intentando que cada una de ellas proporcione valor al proyecto. Por ello, en cada iteración se realizarán las pruebas individuales y de integración necesarias para poder considerar una entrega estable.

Por otro lado, para poder llevar a cabo las conclusiones obtenidas del proyecto, se considerarán parte importante a valorar los problemas encontrados durante el desarrollo de cada parte del proyecto y cómo se han solventado o a qué decisiones han llevado. Lo que hace importante que se vaya documentando cada paso que se realice para que todo detalle sea reflejado oportunamente.

2.1 ELECCIÓN DE ÁREAS DE TRABAJO Y JUSTIFICACIÓN

En la Figura 11 se muestra un esquema de la situación a abordar y como puede observarse en ella, la propia situación suscita la primera división de zonas de ataque:

- Cliente o parte "front" de la aplicación que es la parte que se ejecuta en el navegador del usuario.
- Servidor o parte de "back" de la aplicación que se centra en la gestión.
- Servidor o parte de "back" de la aplicación que se comunica con la placa física del puesto de laboratorio (serverSECO).
- Base de datos o gestión del almacenaje de los datos.

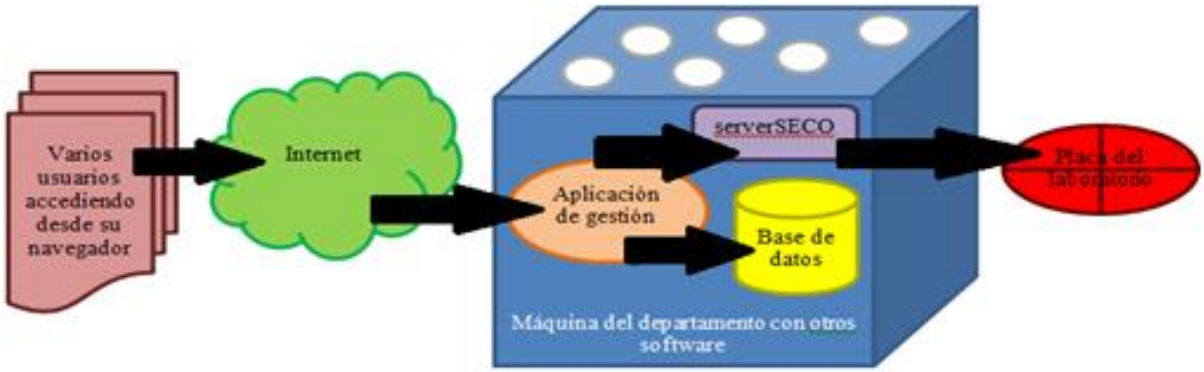

Figura 11 Esquema de la funcionamiento

Una vez encontrados los puntos de ataque al sistema a desarrollar, cada uno de ellos se debe subdividir en tareas a desarrollar:

- Implementación de una capa de "front" que consistirá en:
  - Instalación del entorno suministrado del proyecto previo.
  - Análisis del código proporcionado y de las herramientas que en él se utilicen.

- ➢ Implementación de la capa de cliente que permita la interacción con el servidor para la configuración de los parámetros de los experimentos.
- ➢ Análisis de las herramientas y tecnologías necesarias e implementación de la funcionalidad de la descarga de los resultados de la gráfica en un fichero para su posterior carga y visualización.
- ➢ Análisis de las herramientas y tecnologías necesarias e implementación de la funcionalidad de la descarga de los valores de la configuración en ficheros para su posterior carga y repetición.
- ➢ Pruebas de un correcto funcionamiento de las funcionalidades y de integración.

❖ Implementación de una capa de "back" correspondiente a la aplicación de gestión y la implementación del sistema de gestión de los mismos, parte que consiste en:

- ➢ Análisis y comparativa de las tecnologías para seleccionar aquellas a utilizar.
- ➢ Análisis y comparativa de las arquitecturas para seleccionar la más recomendable.
- ➢ Conclusiones, diseño final y su justificación.
- ➢ Análisis, diseño e implementación del sistema de datos o base de datos.
- ➢ Implementación de la solución elegida.
- ➢ Pruebas de funcionalidad individual y de integración.

❖ Implementación de una capa de "back" correspondiente a serverSECO que se comunique con la placa del laboratorio; como ésta es la parte más trabajada en el proyecto que se suministra, realmente se implementarán modificaciones pertinentes para que funcione con los nuevos servicios desarrollados y para tratar el acceso concurrente o gestión de colas. Para ello se deberán realizar las siguientes tareas:

- ➢ Adaptación de la capa de cliente para la recepción de los datos de la aplicación de gestión.
- ➢ Análisis, diseño e implementación de una solución para la gestión de las colas en serverSECO.
- ➢ Pruebas de funcionamiento individual y de integración del sistema.

Por último, incidir en que la división que se ha hecho ha buscado ser lo más disjunta posible, separando en "piezas" para facilitar el mantenimiento del código y la posterior reutilización de dichas "piezas" si se desease. En la Figura 12 se muestra más esquemáticamente dichas "piezas".

Figura 12 Esquema de interacción entre áreas del TFG

3. DESARROLLO, IMPLEMENTACIÓN Y PRUEBAS

En este capítulo se desarrolla el análisis, diseño y posterior implementación y prueba de la aplicación. Se incluirán los aspectos más relevantes y pequeños fragmentos de código concretos para su explicación.

3.1. IMPLEMENTACIÓN DE LA CAPA DE CLIENTE

El primer área que se abordará es la capa de cliente. En este apartado se instalará y pondrá en funcionamiento el servidor que proporciona la comunicación con la placa del sistema para comprender su funcionamiento; a continuación se realizará la investigación y documentación necesarias sobre las bibliotecas utilizadas y sus diferentes módulos y se implementarán las nuevas funcionalidades correspondientes a esta pieza.

3.1.1. Instalación del entorno existente

Tras una primera lectura del código que conforma el servidor, se podría resumir su estructura como se muestra en la Figura 13 en la que se aprecian los puntos más importantes: hay un fichero web para comunicarse con la placa que contiene el motor utiliza una función de JavaScript con la que conectarse a través de websocket al socket en la placa. La función JS por su parte utiliza la parte de "server" para realizar dicho puente y tratar la información oportunamente.

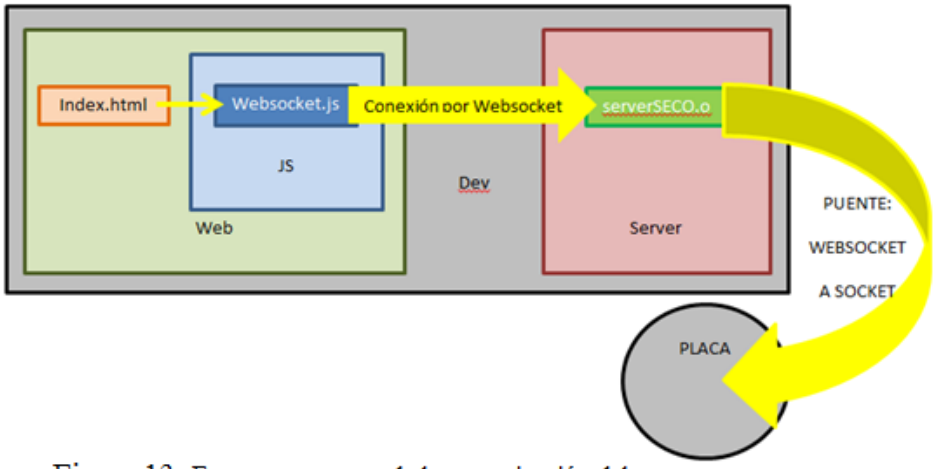

Figura 13 Esquema resumen de la comunicación del acceso remoto

Por otro lado y debido a que serverSECO está programado en Ansi-C y coincidiendo con el sistema operativo que tiene la máquina donde se alojará la aplicación, el sistema deberá ejecutarse en una distribución Unix y usando las librerías necesarias para poder realizar las conexiones de websocket. Por ello, lo primero para desplegar el entorno es instalar las dependencias que se deben satisfacer que son: *gcc, cmake, zlib1g-dev* y *libssl-dev*. En cuanto a las librerías necesarias para trabajar con websockets, se adjunta en un fichero comprimido la librería para Linux: libwebsockets; se deberá descomprimir *(tar –xvzf libwebsockets-1.4.-chrome43-firefox-36.tar.gz)*, instalar con *make install* y establecer dos vínculos simbólicos con *ln –s* entre los dicheros libwebsockets.so y libwebsockets.so.5 de la librería y de la ruta local.

Una vez tenemos el entorno preparado, se debe instalar un servidor Apache desde donde se proveerán los ficheros estáticos de cliente (parte Web de la Figura 13). Podría trabajarse con otro servidor pero se ha elegido Apache ya que es con diferencia el servidor web más usado en la actualidad, posiblemente porque es un servidor muy robusto y eficiente además de ser gratuito.

Una vez instalado y configurado Apache, para trabajar con el código en local, es importante que todas las variables que entran en juego estén correctamente definidas. Todo este tema se trata en profundidad en el manual explicado en el Anexo C. Tras tener todo configurado e instalado, el entorno estará levantado adecuadamente y se realiza apropiadamente la comunicación deseada.

Análisis y explicación del código y su estructura

Investigando más profundamente los ficheros de que se dispone podemos ver que se tiene una página HTML en la que se desarrolla todo, tanto la configuración de los parámetros, como el envío de los mismos y la recepción de los datos para su posterior muestra gráfica y la visualización de la imagen de video.

Por su parte, en el documento HTML podemos diferenciar varias partes:

- Un *div* que contiene el canvas que conectará con la web-cam.
- Un *div* en el que se dibujará la gráfica con los valores obtenidos.
- Un *div* que alberga un selector de tipo de perturbación y cuatro inputs de tipo numéricos para configurar los parámetros de la misma.
- Un *div* con dos inputs para indicar el periodo de ejemplo y el tiempo límite de la comunicación y doce checkbox para seleccionar las variables de las que queremos recibir el valor tras la ejecución del experimento y que serán las variables representadas en la gráfica.
- Un *div* que contiene una tabla con varios inputs y selectores en los que podemos configurar las variables de Direct, Paralled, Feedback, Feedforward, el tipo de controlador, el periodo de ejemplo del controlador, la variable de control y el tipo de referencia y sus parámetros.
- Una sección con el botón para el envío, el valor de la cola y los leds que indicarán el estado de ejecución del experimento.
- Una cabecera con un título y un hipervínculo a la página del departamento.

Además, dicha página dispone de un estilo y usa varios scripts de JS como diversas librerías de JQuery: la básica en su versión 1.5.2; tablesorter, equalHeight y hideshow; que son funcionalidades públicas que se utilizan en el código. Concretando lo que cada una de ellas realiza:

- **Hideshow.js** es una función que corre una vez el DOM está cargado (Document Object Model, es la estructura de objetos que se genera en el navegador. Es una estructura jerárquica de varios objetos donde unos dependen de otros. Dichos objetos modelan tanto la ventana del navegador como el historial, la página web en sí y todos los elementos que pueda tener dentro. Además todos ellos son accesibles, modificables e incluso se pueden eliminar a través de JavaScript y CSS -Cascading Stylesheets-). Utiliza la clase *toogle* para alternar la visibilidad del elemento además de cambiar el texto del link que alterna la visibilidad del elemento en función del estado de visibilidad que posea.

- **EqualHeight.js** se añade a JQuery que encuentra la mayor altura en una colección.

- **Tablesorter.js** permite crear tablas ordenables haciendo click sobre las cabeceras de las columnas a partir de una tabla HTML estándar. Para convertir una tabla sólo debe invocarse el constructor de la librería y esto se hace a través de la siguiente sentencia de JavaScript: *$("#idDeLaTabla").tableSorter().*

Además utiliza otros ficheros JavaScript desarrollados para este proyecto que son camera.js, leds.js, dynamics.js, plotData.js y websocket.js. Profundizando un poco más en lo que cada uno de ellos realiza:

- **Camera.js** es una función que coge el elemento canvas del DOM, carga una imagen en el área centrada en (0,0) y la refresca cada 10 milisegundos. Si no se modifica externamente, se trata de una imagen negra.

- **Leds.js** es una función que carga primero las constantes de los colores de encendido y apagado de unos "leds" simulados con elementos canvas que se utilizan para indicar el estado en el que se encuentra la petición, coge cada elemento y le carga un contexto de dos dimensiones, posteriormente dibuja círculos en los canvas con los colores de apagado inicialmente pero cuando se llame del exterior a uno de ellos con el color de encendido, se repintará el correspondiente elemento.

- **PlotData.js** es una función que añade o no a dos arrays: en uno el path de un fichero json y en el otro el nombre de la variable correspondiente; en función de si los checkbox de los elementos del apartado de comunicación están seleccionados o no. Posteriormente llama a la siguiente función almacenada en el fichero que es la que dibuja la gráfica en el área dispuesta pasándole como parámetros los dos arrays. En esta función, se configura el gráfico de HighCharts recorriendo ambos arrays. El funcionamiento de HighCharts se investiga y documenta más en detalle en la siguiente tarea.

- **Dynamics.js** contiene dos funciones que con respecto al valor de un selector del DOM, activa o desactiva otros elementos del mismo, para que en función del tipo de señal de referencia del controlador y del tipo de señal de perturbación, se puedan o no modificar los distintos inputs de los valores de las variables relacionadas. Se podría decir que proporciona una funcionalidad de dinamismo en la página HTML como se muestra es la Figura 14.
  Como se explicaba al comentar la Tabla 2 y la Figura 6 en el capítulo anterior, si la entrada (sea de reference o de perturbation) es nula se deben modificar las variables con (0,0,0,0) correspondiendo a (X1, X2, X3, X4), en cambio si es Delta, Escalón, Rampa o Parábola hay que modificarlas con (1,0,0,0), con Seno y Coseno es como activarlas según (1,1,0,0) y si la entrada es Trapezoidal hay que activarlas todas con (1,1,1,1).

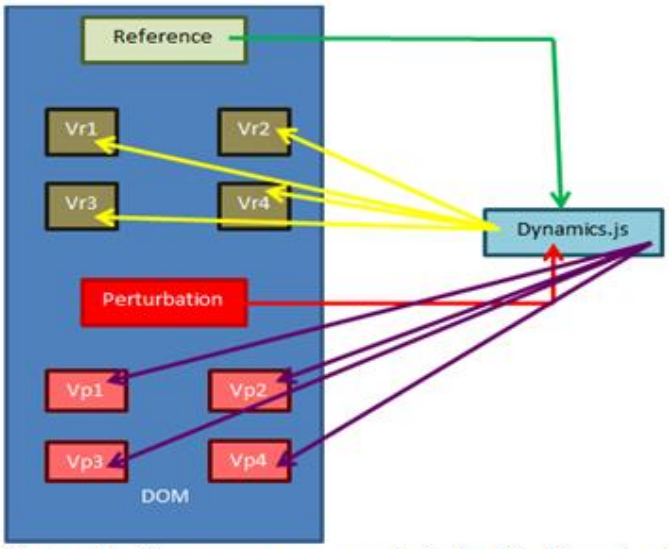

Figura 14 Esquema que representa la función Dynamics.js

- **Websocket.js** es un fichero que alberga dos funciones: socketConnect y socketSendData:

  - **socketSendData** es la función que envía los datos en caso de que el websocket esté en el estado "START". Para ello, por cada parámetro (elemento de configuración del DOM), se toma el valor y se hace un send de websocket con el par (nombre de la variable:valor de la variable). Para aligerar la carga de envíos, si el valor es 0.00 en las variables Direct, Parallel, FeedBack y FeedForward, éstas no se envían; además en las variables de perturbación sólo se envían si el tipo es distinto del equivalente al valor 0 que es None y las variables de comunicación sólo se envían si han sido seleccionadas. Por último, para indicar el fín de la trasmisión se envía "START:".
  - **socketConnect** es una función lanzada al pulsar el elemento botón enviar. Lo primero que hace es establecer el websocket para la comunicación con la dirección IP que se desea. Posteriormente se definen diferentes acciones en función del estado de dicho websocket, esta situación se presenta en la Figura 15.

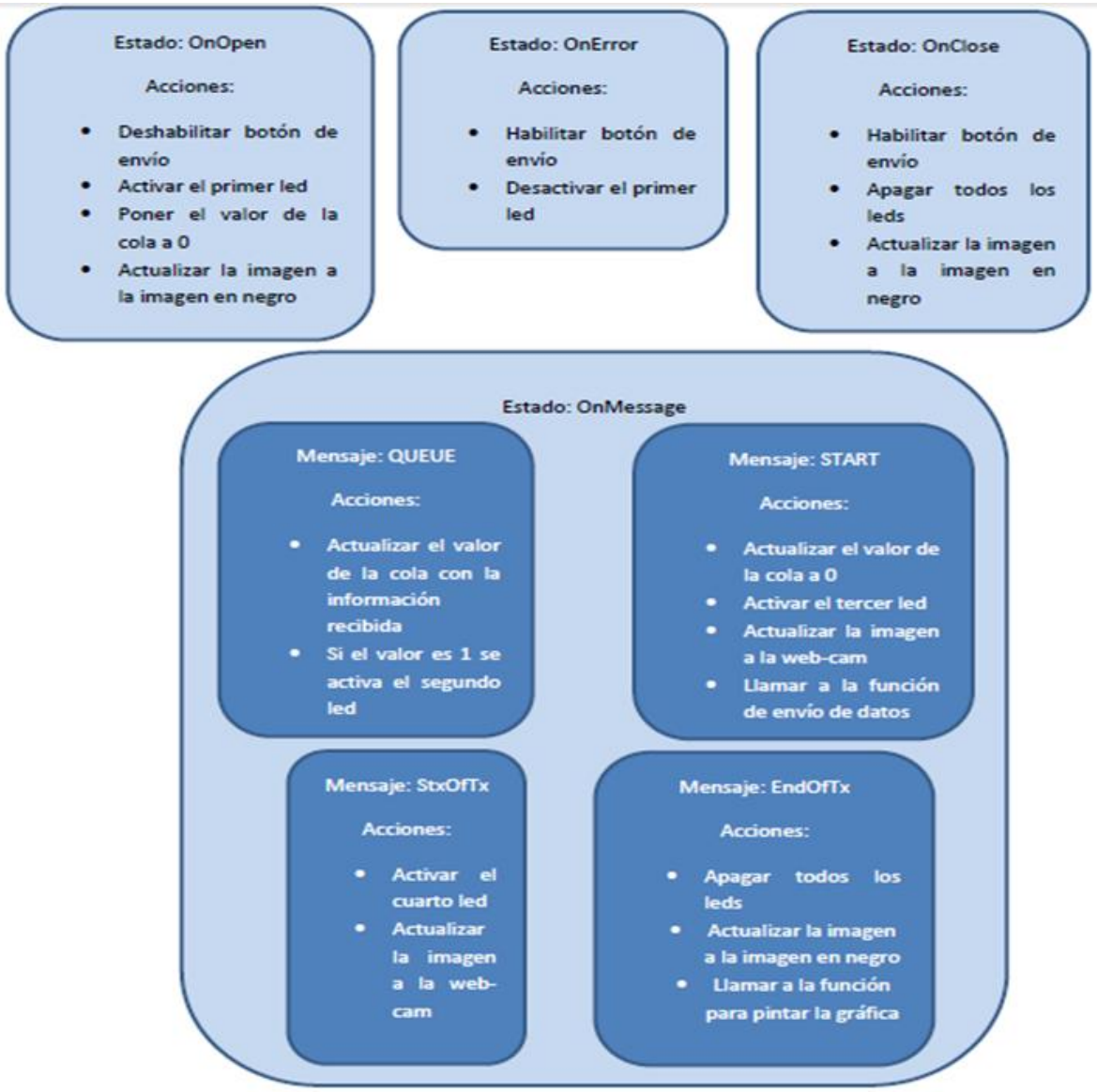

Figura 15 Esquema de las acciones según los estados del websocket

Con ésto y el fichero de estilo se concluye el código de la aplicación de cliente. Adicionalmente, en ServerSECO además de los ficheros .o, .h.gch y binario generados tras la compilación, nos encontramos con varios ficheros:

- **Common.h** es un archivo en el que se definen los valores de las constantes necesarias para el funcionamiento de ServerSECO (BOARD_SOCKET_ADDR, BOARD_SOCKET_PORT, WEBSOCKET_PORT, NUM_CONTROLLERS, MOTOR1, …); además, se definen estructuras (similares a los objetos de la programación orientada a objetos) con sus atributos: la de parámetros de controlador, de señal, de controlador general y de comunicación general. Si se permitiese otro tipo de controlador (de test por ejemplo), habría que desarrollar las estructuras que necesitase.
- **Queue.c** es el fichero que implementa la cola, en él se implementan las diferentes funciones para inicializar, comprobar el estado, introducir y sacar elementos, conocer la posición de cada elemento y limpiar la cola.
- **Queue.h** es el fichero en el que se definen las constantes de las colas (QUEUESIZESOCKETTX, QUEUESIZESOCKETRX, QUEUESIZESINGLE), las estructuras que se necesitan: cola de transmisión, de recepción y una cola sin definir finalidad con sus punteros y sus contadores y se definen además los métodos que se deben implementar en el fichero .c anteriormente explicado.

- **Makefile** es el fichero de compilación del servidor, se indican las dependencias de compilación y los comandos de borrado necesarios.
- **ServerSECO.c** es el fichero de implementación de ServerSECO.

Lo primero que se hace es definir las variables que van a ser utilizadas en el mismo: cola, estructura de parámetros de controlador general y de comunicación general, arrays de nombres de variables a imprimir y de los jsons y enteros que indican el estado del socket de la placa y de FD.

Posteriormente se definen las constantes que indicarán el estado del websocket y se inicializa la variable a 0 (esperando entrada de websocket).

A continuación se especifican algunos métodos que serán implementados más adelante para que se puedan usar antes de llegar a su implementación. Se pasa a implementar los métodos necesarios: los dos primeros son los callback de http y del protocolo de comunicación a websocket (dumb increment) en el que a través de un switch se realizan determinadas acciones en función de la razón por la que se debe reaccionar. Además de explicarse a continuación, se resume gráficamente en la Figura 16 en la que cabe mencionar que las flechas rojas con la bola negra indican que a ése estado se puede llegar desde cualquier otro siempre que el evento sucedido sea el descrito al lado y que las flechas amarillas son transiciones entre estados que se realizan externamente, en cambio las rojas indican transiciones que se realizan concretamente en el fichero que se está explicando.

- ➢ Si existe una conexión previa: se introduce en la cola.
- ➢ Si se está desconectando: se borra su posición de la cola.
- ➢ Si algo se está recibiendo del socket: se llama a la función que verifica los datos.
- ➢ Si el servidor está esperando: comprueba la posición de la cola y si estamos los primeros pueden darse varios casos:
  - El estado señalado es "esperando entrada del websocket": pasamos al estado siguiente, y mandamos un START.
  - El estado señalado es "iniciar la trasmisión de datos de la placa": pasamos al estado siguiente y mandamos un StxOfTx.
  - El estado señalado es "acabar trasmisión de datos de la placa": mandamos EndOfTx y pasamos al primer estado de nuevo y devolvemos un -1.
- ➢ Si el servidor está esperando pero no somos los primeros en la cola, se manda QUEUE y la posición de la cola que tenemos.

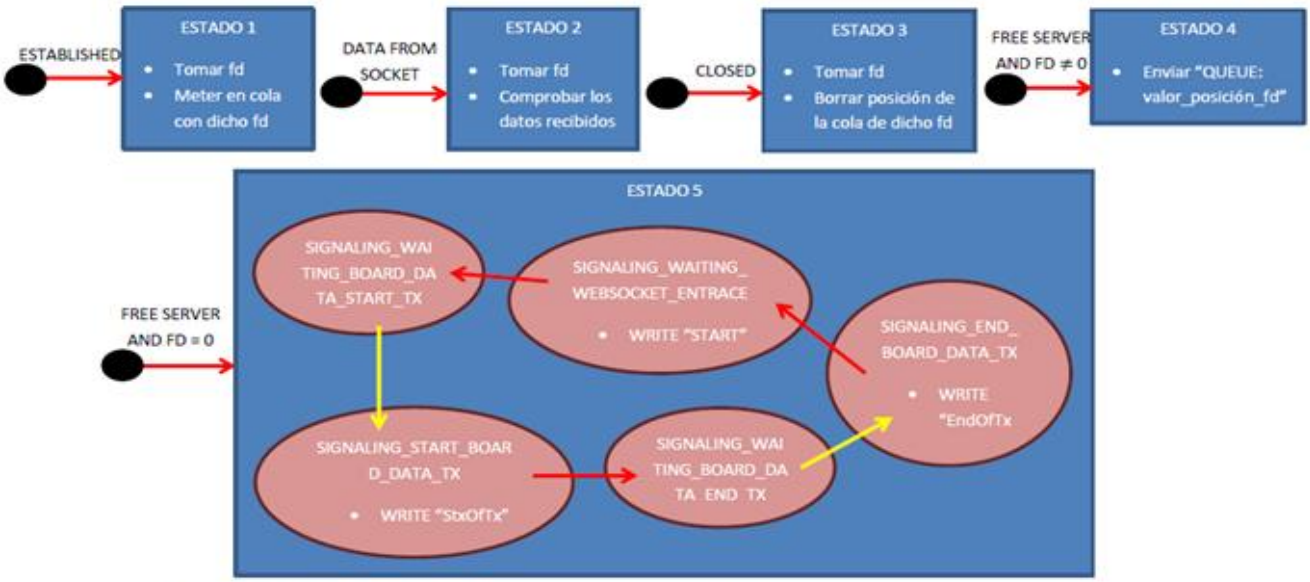

Figura 16 Esquema de reacciones en función del mensaje recibido por el websocket

También se definen varias estructuras relacionadas con indicar qué métodos son los protocolos para el websocket. A continuación se implementa el método main en el que se inicializa la cola, los parámetros de controlador y de comunicación, se pone el flag del socket a false, se llama al método de crear socket, se crean websockets y sus propiedades. Posteriormente entra en un bucle infinito que se describe como una máquina de estados (que comprueba el flag cada 50 ms) en la Figura 17.

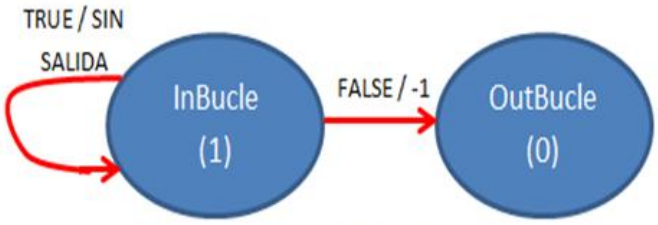

Figura 17 Bucle del método main

Para continuar se implementan otra serie de métodos como:

➢ **checkReceivedData(int fd, char* in)** para comprobar los datos recibidos. Si coincide con START se ejecuta el borrado de todo fichero que contenga MOTOR de la ruta definida previamente y llama al método que manda la información al socket. Si coincide con el nombre de una variable o parámetro, coge el valor, lo convierte en un número con coma flotante y lo asigna a la estructura correspondiente (se cogen los valores de la comunicación y se almacenan apropiadamente).

➢ **SendDataToSocket()** en el que se resetea la placa llamando a sendData con los valores de reset, se duerme mientras se reinicia el sistema, manda PWM con sendData y tras comprobar lo que debe mandar, utiliza los valores oportunos para enviarlo a través de sendData.

➢ **SendData(unsigned char frame_type, unsigned char funct, unsigned char u_motor, float data1, float data2, float data 3)** en el que tras calcular el timestamp se crea el frame para la comunicación con el timestamp, los parámetros y los finales de línea y lo manda al socket con send. Si devuelve un error lo capta y pone el flag a false para abortarlo todo, si no hay error espera "durmiendo" un tiempo a que todo haya ido bien.

➢ **TxtToJson()** en el que los datos se convierten a JSON. Para cada variable seleccionada de comunicación se crea el fichero donde se va a generar el JSON y se tienen abiertos el de texto y el de json y mientras el de texto tenga nueva línea se cogen los valores que son los dígitos que están después del espacio, se coge el timestamps que son los digitos que están desde el principio hasta el espacio y se crea la entrada json como [timestamp, value]. A continuación, coloca en la ruta configurada dichos ficheros json tras borrar los antiguos.

➢ ***ClientRx(void *threadid)** método que capta todas las recepciones a través del socket. Para ello con un bucle infinito se comprueba si algo ha sido recibido en la placa, si es así pueden darse varias circunstancias que además de explicarse a continuación se resumen en la Figura 18:

- Si se recibe MOTOR, éste es el nombre del fichero, y el estado de la señalización es esperando el inicio de la trasmisión de datos de la placa, se actualiza a iniciada la trasmisión y se introduce la información en un fichero.
- Si se recibe EndOfTx, la trasmisión ha acabado, se actualiza el estado de la señalización a acabada y se llama para resetear con sendData, se espera a que se reinicie el sistema y reinicia el PWM. Convierte los datos recibidos a un JSON con TxtToJson y se inicializan los parámetros del controlador y de la comunicación.
- Si se recibe Queue no se hace nada porque está aún en cola.
- Si no es ninguno de los anteriores, es porque se están recibiendo datos de un fichero, por lo que se introducen en el fichero.

Además de métodos auxiliares como commVarToJsonFile, CommInitCommParams, commVarToTxtFile, ControllerInitControllerParams, CreateSocket, replace_str y AliveTx.

Con todo esto se ha explicado dónde y cómo se realiza la tarea de comunicación con la placa a través de ServerSECO y las distintas funciones de Javascript. El entorno ha quedado configurado y funcionando y se conoce dónde se debe modificar para cambiar o añadir alguna funcionalidad.

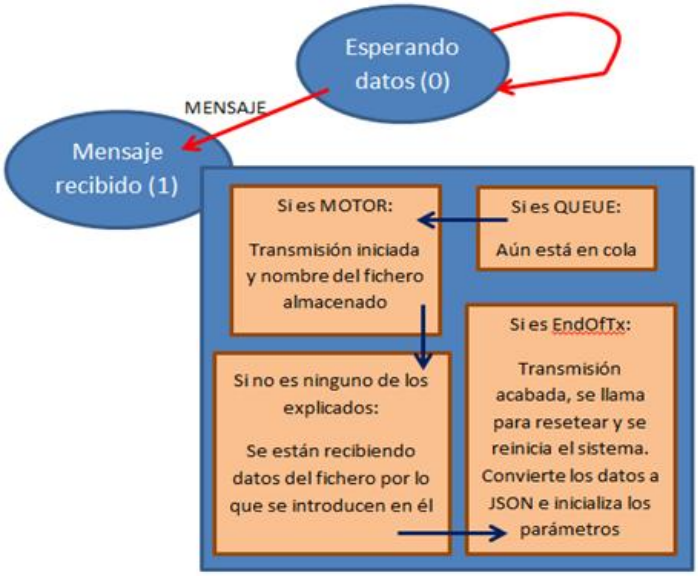

Figura 18 Bucle infinito del método ClientRx

3.1.2. Documentación sobre HighCharts y sus módulos

En esta parte se pretende explicar ligeramente la librería HighCharts escrita en JavaScript que se ha utilizado para implementar la gráfica dónde se volcarán los resultados del experimento para que sean representados.

La motivación para desarrollar la librería fue ofrecer una manera sencilla de implementar gráficas interactivas completamente desarrollada en un lenguaje que es ampliamente aceptado. Ofrece una gran variedad de tipos de gráficas (desde gráficas de línea con múltiples opciones a las distintas gráficas de área pasando por opciones como columnas, barras, quesitos, burbujas, esquemas de dispersión o combinaciones de todos ellos) y opciones de incluir acciones dinámicas en la misma o visualizaciones en 3D o en imágenes de distintos medidores entre otras muchas opciones para encontrar exactamente lo que se adapte perfectamente al concepto y diseño del entorno. Además, otras de las ventajas de usar este framework son:

- Al estar escrito en JavaScript aporta **compatibilidad** con los navegadores actuales.
- Es **gratuito** para el uso no comercial.
- Es **código abierto** tanto para las licencias gratuitas como para las de pago, lo que **permite modificar el código** a las necesidades específicas del uso concreto.
- Posee una **sencilla configuración**; en la documentación oficial se ofrecen multitud de ejemplos y demostraciones de los distintos usos y en ellos se muestra que la configuración es una estructura de notación de objetos JavaScript (conjunto de clave-valor) conectados por puntos, separados por comas y agrupados por llaves como puede verse en la Figura 19.

```
var chart ={
        chart: {
          type: 'line',
          zoomType: 'xy'
        },
        title: {
          text: 'Measurements',
        },
```

Figura 19
Parte de código para configurar el gráfico

- Además contiene diversos **módulos** para la exportación e impresión, la funcionalidad de hacer zoom en un eje, en el otro o en ambos, para cargar datos de una fuente externa o para la modificación estética del gráfico como puede ser revertir el eje teniendo los valores más altos en la zona más cercana al origen entre otras.

Por otro lado, los programadores de HighCharts se preocupan por que sus gráficas puedan ser fácilmente manejadas por personas sin formación en el ámbito de la programación y para ello ponen a disposición del público un editor de gráficas descargable en GitHub: *"https://github.com/highcharts/highcharts-editor/releases"* en el que se configura la gráfica como puede verse en el anexo D. Permite desde habilitar funciones como la exportación a elegir la configuración de los ejes, de los nombres de variables, de la zona del gráfico donde hace efecto el zoom, el tamaño del gráfico, los colores o formatos de las fechas entre otros y poder posteriormente descargar el código en HTML u otros formatos para utilizarlo en su proyecto.

3.1.3. Documentación sobre Blob y FileReader de JavaScript

El almacenaje de los datos de un experimento se puede plantear en múltiples formatos, pero con la finalidad de crear una aplicación con una estructura y unos conceptos que se puedan adaptar fácilmente a cualquier "experimento" con mayor o menor número de parámetros de configuración, se ha considerado el uso de Blobs. Un Blob (Binary Large Objects, *objetos binarios grandes*) representa un objeto tipo fichero de datos planos inmutables. Los Blobs representan datos que no necesariamente se encuentran en un formato nativo de JavaScript.

La interfaz File se encuentra basada en un Blob, heredando y extendiendo su funcionalidad para soportar archivos en el sistema del usuario. Como cualquier objeto tiene un constructor (*Blob()*), atributos *(Blob.size* y *Blob.type [tipo MIME]*) y métodos (*Blob.slice()* que devuelve un rango específico de bytes). Para extraer los datos de un Blob, se necesita un lector que los leerá con alguno de sus métodos como *readAsArrayBuffer(blob)*, por ello se investiga también acerca de ellos. Un FileReader, u objeto lector, se encarga de extraer los datos de ficheros (o información en buffer) almacenador en el cliente de forma asíncrona a través de objetos Blob o File. Los ficheros se pueden obtener de diversas maneras como pueden ser desde un input o como un dataTransfer tras un arrastre entre otras opciones. Como objeto que es, un lector tiene constructor (*FileReader()*), atributos (*FileReader.error, FileReader.readyState, FileReader.result, FileReader.onabort, FileReader.onerror, FileReader.onload, FileReader.onloadstart, FileReader.onloadend y FileReader.onprogress*) y métodos (*FileReader.abort(), FileReader.readAsArrayBuffer(), FileReader.readAsDataURL() y FileReader.readAsText()*).

Combinando estos objetos con los escuchadores, la descarga y la carga de ficheros, su lectura y tratamiento y trabajar con la información que aporten resulta bastante sencillo.

3.1.4. Implementación de la descarga de datos en un fichero CSV

Gracias a HighCharts y todos los módulos que ofrece, esta tarea es sencilla de realizar ahora. Para ello se necesita la incorporación al código HTML de la fuente la siguiente sentencia: *<script src="http://code.highcharts.com/stock/modules/exporting.js"></script>*. Con esto, en el *div* donde se dibuja el chart aparecerán las diferentes opciones en un desplegable desde el icono con las tres rayas horizontales donde se podrá elegir la descarga de los datos o de la gráfica en CSV, XLS, SVG, PDF, JPEG o PNG, además de ver los datos online e imprimir el gráfico.

Con miras a la futura implementación de la carga de datos desde el fichero descargado, se procede a la inspección del fichero que se ha descargado como CSV y se observa que tanto la separación de los valores como el separador decimal son comas, lo cual hará difícil diseñar una función que pueda discernir si una coma en concreto es un separador decimal o un signo de separación entre valores.

Para solventar el problema se intentan desarrollar algoritmos que puedan discernirlo, pero al ver la complicación que esto supone, se opta por descargar y modificar la librería *exporting.js*: se accede a ella en GitHub, se descarga y se busca el fragmento de código donde se estipula cuál será el signo de puntuación utilizado como separador decimal y se cambia a un punto como puede observarse en la Figura 20.

Figura 20 Estracto de código para la modificación de la separación decimal

Además se deberá incluir dicho js además del anterior ya que el anterior generará las opciones y éste modificará la forma en la que se va a generar el fichero en CSV en concreto. Para ello basta con incluir la siguiente línea de código: *<script src="js/export-csv.js"></script>*. De esta manera el fichero que se generará será similar al de la Figura 21.

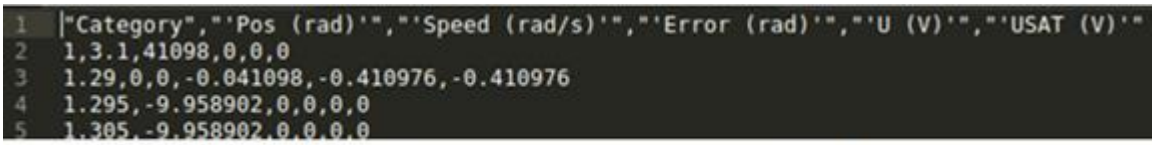

Figura 21 Ejemplo de fichero con datos descargado del gráfico

3.1.5. Implementación de la carga de datos para su representación gráfica desde un fichero CSV

Entre los módulos que HighCharts ofrece está *data.js* que es una herramienta para facilitar el análisis de fuentes de entrada como ficheros CSV o tablas HTML.

Aunque el módulo ofrece múltiples opciones, la implementación que se ha elegido es a través de la opción "rows" en la que se deberá pasar un array de arrays. El primer array contendrá a partir de su posición 1 el nombre de cada variable a representar. El resto de arrays serán interpretados como los valores de dichas variables desde la posición 1. La posición 0 de todos ellos se interpretará como el valor del eje x correspondiente a los valores siguientes de las variables.

El código del constructor del gráfico correspondiente a la entrada desde fichero CSV se puede visualizar en la Figura 40 del anexo D.

La función será llamada desde otra que tratará el evento de cambio en el input del fichero. Para ello una función está escuchando los cambios a través de un *"escuchador de eventos"*. Al producirse, se abre el fichero y se comprueba que sea de la extensión adecuada y se pasa a un lector que lo leerá a través de su método *"readAsText"*. A través de otro escuchador de eventos, cuando el lector termine se disparará otra función que preparará el documento como array de arrays con la información en el formato expuesto anteriormente y llamará al constructor del gráfico.

Todas las funciones se encuentran desarrolladas en el fichero *"tratamientoCSV.js"* que debe ser incluido en el DOM junto con el módulo de datos de HighCharts con las siguientes líneas de código:
<script src="https://code.highcharts.com/modules/data.js"></script>
<script src="js/tratamientoCSV.js"> </script>.

3.1.6. Implementación de la descarga de datos de configuración a un fichero

Para incluir la funcionalidad de descarga de datos en un fichero, lo primero que se necesita es un botón para hacerlo en el DOM que esté vinculado a una función de JavaScript en la que tras obtener el valor de cada variable del DOM con *document.getElementById("id")*.value o *.checked* incluya en un array *"\n" + "nombreDeLaVariable:"+ "valorDeLaVariable"* que agrupe las variables por el tipo de variable que es y posteriormente un array de arrays en los que se inserta cada array tras acabar de añadirle todas sus variables.

A continuación se crea un blob con el array de arrays y de tipo texto que se pasará como parámetros en la llamada a la función que va a realizar la descarga. En dicha función, cuando un fileReader termine de leer el blob, se crea un elemento a en el DOM en el que se configura el href, el target y el download (nombre que tendrá el fichero al descargar: "configuracion.seco"; aquí se le asigna una extensión .seco para que se identifique, pero no es necesario); posteriormente se simula un click de ratón del cliente (con *.dispatchEvent(new MouseEvent('click', {'view' : window, 'bubbles' : true, 'cancelable' : true});)*) en el link creado para lanzar la descarga.

Posteriormente, se crea el fichero y se lanza la descarga del mismo, el fichero descargado tendrá un aspecto similar al de la Figura 22.

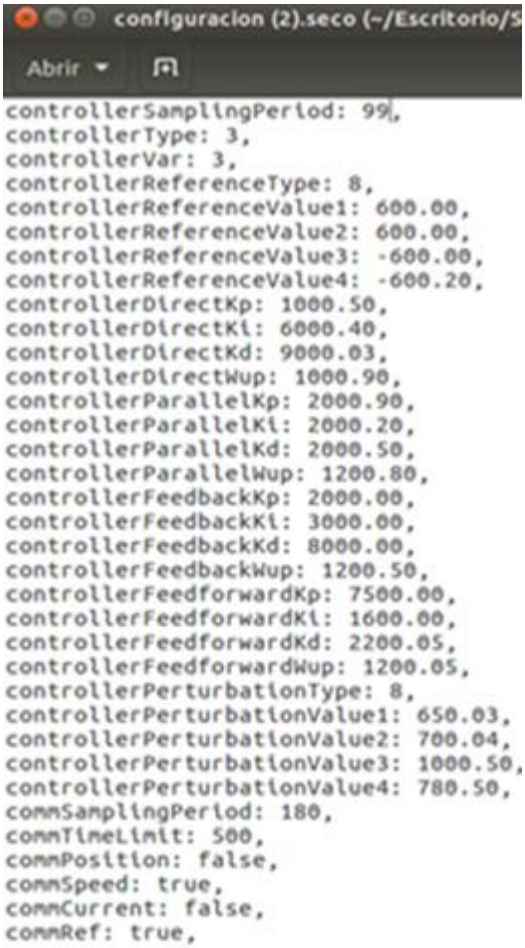

Figura 22 Ejemplo fichero de configuración

3.1.7. Implementación de la carga de datos de configuración desde un fichero

A través de un input de tipo file en el DOM se permite en HTML agregar un fichero para cargar la configuración. Para ello, se configura un escuchador de cambios al input, el cual al detectar un cambio en el elemento coge el fichero y comprueba que la extensión del nombre coincida con seco y asigna el fichero a un FileReader.

Con otro escuchador de eventos se espera a que el lector termine de leer el fichero. Seguidamente lanza la función que trata la información leída, con la que crea un array en el que en cada posición introducirá una línea (hasta un \n). Para cada posición del array comprueba si contiene el nombre de alguna de las variables esperadas y si es así coge su valor para cambiarlo posteriormente en el elemento adecuado del DOM.

La decisión de diseño de lógica adoptada es para que si se sube un fichero que no contiene información útil, éste no consiga estropear el funcionamiento de la aplicación porque al no coincidir con variables esperadas, dicha línea del fichero no hará nada salvo meterse en una posición de un array.

3.1.8. Pruebas

No se han realizado pruebas automatizadas para probar la carga y descarga de los datos de la gráfica: se ejecuta un experimento y de la gráfica obtenida se descarga el fichero CSV, se recarga la página y se carga dicho fichero en el input preparado para ello y se ve que se representa correctamente. Con ello se comprueba el funcionamiento correcto ante los formatos esperados y que no se ha estropeado la funcionalidad anterior.

Además se prueban los casos "no esperados":

- Subiendo un fichero con una extensión distinta se comprueba que emerge el prompt que indica que la extensión debe ser .CSV.
- Subiendo un fichero con extensión CSV pero que no cumple el formato deseado: se trata, se pinta una gráfica, pero se comprueba que el tratamiento no es el deseado, pero no bloquea la aplicación.

Todo ello implica que la aplicación tiene el comportamiento deseado para los casos tenidos en cuenta.

Para probar las funcionalidades de carga y descarga de la configuración del experimento, primero se comprueba que no genera conflicto con el funcionamiento anterior probando a enviar un experimento y recibiendo los valores pedidos, cargando unos datos para dibujar en la gráfica y probando a descargarlos de la ejecución. Después se prueba a cambiar las selecciones y descargar la configuración y tras volver a cambiarlas, cargar el fichero que se ha descargado y comprobar que se modifican automáticamente los valores de los elementos. De nuevo no se automatizan pruebas.

Para probar los casos "no esperados":

- Se intenta subir un fichero con otra extensión y emerge un prompt indicando la necesidad de ser un fichero con la extensión .seco.
- Se sube un fichero que a pesar de coincidir en extensión, no respeta el formato de la imagen anterior y en el cual hay texto plano que no coincide con ninguna variable del DOM y se comprueba que no genera cambios en los elementos ni cuelga el sistema, simplemente no hace nada de cara al usuario, lo cual coincide con el comportamiento esperado.

De nuevo, todo ello implica que la aplicación tiene el comportamiento deseado para los casos tenidos en cuenta y la integración con las funcionalidades previas.

Así se cierra el primer bloque de tareas y llegados a éste punto se podría hacer una entrega del sistema totalmente funcional y habiendo incrementado el valor del mismo ya que se han aumentado las funcionalidades que ofrece. Para la implementación y la documentación se han consultado las referencias [2-9].

3.2. IMPLEMENTACIÓN DE LA APLICACIÓN DE GESTIÓN

La siguiente tarea que se abordará es la aplicación de gestión que se ejecutará en un nuevo servidor. Primero se realizará un análisis de arquitecturas para el desarrollo de esta parte. A continuación se hará lo mismo con las tecnologías existentes en el mercado y las ventajas y desventajas de cada una de ellas para valorar cuál encaja mejor en el proyecto actual.

Finalmente con toda esa información se comentará el diseño a desarrollar y las conclusiones que han llevado a tomar dicha decisión. Ésta tarea no aporta valor funcional a la aplicación, pero es necesaria sobre todo por el peso educativo del proyecto. Por último y una vez esté decidido el diseño se pasará a implementarlo y probarlo.

3.2.1. Análisis de arquitecturas y comparación entre ellas

Introducción y definición

Antes de entrar a fondo a analizar las distintas opciones que se compararán para elegir la que mejor encaja en el proyecto, es importante comentar qué es la arquitectura de un software.

De acuerdo al Software Engineering Institute (SEI), la Arquitectura de Software se refiere a *"las estructuras de un sistema, compuestas de elementos con propiedades visibles de forma externa y las relaciones que existen entre ellos."*

El término "elementos" puede referirse a distintas entidades relacionadas con el sistema, pueden ser entidades que existen en tiempo de ejecución (objetos, hilos), entidades lógicas que existen en tiempo de desarrollo (clases, componentes) y entidades físicas (nodos, directorios). Por otro lado, las relaciones entre elementos dependen de propiedades visibles (o públicas) de los elementos, quedando ocultos los detalles de implementación. Finalmente, cada conjunto de elementos relacionados de un tipo particular corresponde a una estructura distinta, de ahí que la arquitectura esté compuesta por distintas estructuras.

La elección de la arquitectura es muy importante a la hora de diseñar un software ya que la manera en que se estructura un sistema tiene un impacto directo sobre su capacidad para satisfacer los requisitos. Algunos ejemplos de requisitos que dependen de la arquitectura elegida son el tiempo de respuesta del sistema, la usabilidad o la modificabilidad. Por ejemplo, un sistema estructurado de tal manera que una petición deba transitar por muchos componentes antes de que se devuelva una respuesta podría tener un desempeño (tiempo de respuesta) pobre. Por otro lado, un sistema estructurado de tal manera que los componentes estén altamente acoplados entre ellos limitará severamente la modificabilidad. Aunque la estructuración tiene un impacto mucho menor respecto a los requisitos funcionales del sistema. Por ejemplo, un sistema difícil de modificar puede satisfacer plenamente los requerimientos funcionales que se le imponen.

Finalmente, los diseños arquitectónicos pueden ser reutilizados para crear sistemas distintos. Esto permite reducir costes (económicos y temporales) y aumentar la calidad, sobre todo si dichos diseños han resultado previamente ser sistemas exitosos.

Una vez definido el concepto de arquitectura de un sistema software y su relevancia en el desarrollo, se debe comentar que todo diseño de arquitectura que se reutiliza constantemente termina formando parte de un catálogo de estilos arquitectónicos que recoge aquellas arquitecturas más usadas y las ventajas y desventajas de ellas.

Antes de entrar a estudiar algunos de éstos estilos es oportuno comentar que no se deben confundir los patrones o estilos arquitectónicos con los patrones de diseño ya que a pesar de ser ambas formas más o menos repetidas de enfrentarse a un desarrollo software, los patrones de diseño son de un nivel más bajo, más cercano al código específico de la aplicación, se mueven en el nivel de unas cuantas clases y los patrones arquitectónicos son más teóricos, más estructurales. Una vez se ha aclarado la diferencia y aunque no es un axioma, la mayoría de los arquitectos de software coinciden en que el catálogo de estilos que se ha mencionado previamente está compuesto por:

- Estilos Centrados en Datos
- Estilos de Flujo de Datos
- Estilos de Llamada y Retorno
- Estilos de Código Móvil
- Estilos de Capas Estratificadas

- Estilos Peer-to-Peer

Ahora se van a analizar algunos ejemplos que pertenecen a cada uno de éstos estilos:

Arquitecturas de Pizarra o Repositorio

Pertenecen a los estilos centrados en datos, también conocidas como Shared-Data o Data Warehouse. Se basan en que en el centro de la arquitectura se encuentra un almacén de datos y alrededor un conjunto de componentes que operan sobre éste. Sus accesos son frecuentes para realizar operaciones CRUD (Create, Read, Update, Delete).

Adicionalmente puede implementar la funcionalidad de avisar al consumidor de modificaciones en los datos y en dicho caso se denominaría Pizarrón o Blackboard y si es el consumidor (componente) el que realiza directamente la consulta se le denominará Repositorio.

Las principales ventajas de este estilo es que es una forma eficiente de compartir grandes cantidades de datos, pero los componentes no transmiten datos de unos a otros aunque sí deben compartir el modelo de los datos del repositorio. Además garantiza el desacoplo entre la producción de datos y el modo en que éstos vayan a ser consumidos y centraliza actividades como el respaldo de datos, la seguridad de los mismos y el control de acceso aunque cada componente podría tener sus propias políticas en cuanto a estas actividades se refiere, lo cual es una gran desventaja de ésta arquitectura. Por último se puede apreciar un esquema de la arquitectura en la Figura 23.

Figura 23 Ejemplo de arquitectura de pizarra

Tuberías y filtros

Pertenecen a los estilos de flujos de datos, es una arquitectura en la que los datos de entrada son transformados a través de un conjunto de componentes o módulos funcionales. Se podría describir como que los datos llegan a un filtro, se transforman y son pasados a través de tuberías al siguiente filtro.

Los filtros son independientes entre sí y no conocen la identidad de los demás, además pueden recibir y entregar datos a múltiples tuberías. Por otro lado, según la funcionalidad de un filtro en concreto, se puede permitir que éste inicie la transformación antes de terminar de leer la entrada, éste comportamiento dependerá del diseño funcional del filtro y no chocaría con las reglas de la arquitectura.

Las principales ventajas que proporciona esta arquitectura son:

- Reutilización de los filtros.
- Diseño intuitivo al pensar en secuencias de procesamiento de datos.
- Sencillez en la evolución y modificación del sistema al agregar, editar o eliminar transformaciones simplemente agregando, editando o eliminando filtros y/o “caminos”.

Como desventajas se pueden mencionar las siguientes:

- ➢ Debe existir acuerdo en el formato de los datos.
- ➢ Es una arquitectura que no casa con sistemas interactivos.
- ➢ Puede llegar a ser una arquitectura ineficiente tanto en cuanto podría llegar a hacer más de lo que debe o porque podría darse que los filtros repitieran comprobaciones.

Por último se puede apreciar un esquema de la arquitectura en la Figura 24.

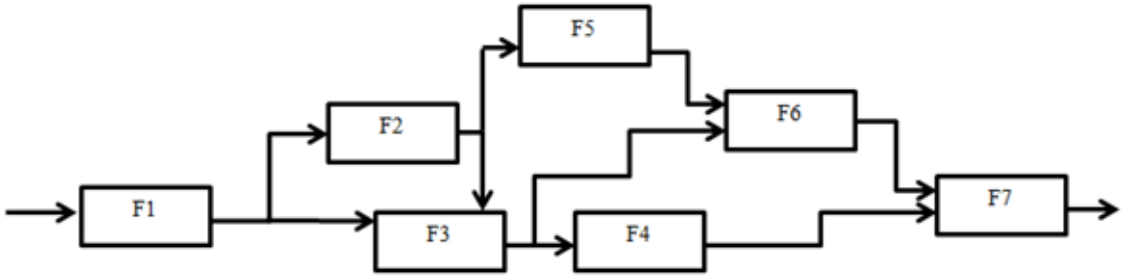

Figura 24 Esquema de arquitectura de tuberías y filtros

Orientadas a Objetos

Pertenecen a los estilos de llamada y retorno. Se diferencian porque los componentes del sistema encapsulan los datos y las operaciones que se deben realizar para manipularlos. Dichos componentes se conocen como objetos y la comunicación y coordinación entre ellos se realiza a través del paso de mensajes. Ésta arquitectura permite el desarrollo de conceptos como la herencia o el polimorfismo entre otros.

Podría destacarse como ventajas de éste diseño:

- ➢ Los objetos tienen implementaciones totalmente desacopladas de modo que cambiar la implementación de uno de ellos no afecta a los demás y promueve la reutilización de los mismos.
- ➢ Facilita la relación de lo virtual con la realidad ya que muchos objetos pueden representar entidades de la realidad, éste hecho hace que sea más fácil entender la estructura del sistema.

En cambio, la principal desventaja eso

- ➢ Para poder usar los servicios que ofrecen otros componentes se deben conocer los nombres de la interface de otro objeto. Y los cambios de las interfaces afectan a todos los objetos que las usan.

Objetos distribuidos

Se trata de una arquitectura que particulariza la anterior para sistemas distribuidos por lo que también pertenece al estilo de llamada y retorno. También tiene en cuenta conceptos de otras arquitecturas de llamada y retorno como pueden ser las estructuradas en capas de las que heredan el concepto de servidor como la capa que en un principio provee servicios a las llamadas de un cliente pero adaptándolo a que también ésta puede solicitar servicios al cliente.

El sistema se compone de objetos que proveen y usan servicios entre ellos. En éste diseño, se pueden distribuir los objetos entre distintas localizaciones en una red y comunicarse entre ellos a través de *middleware (*Se define como software diseñado para gestionar la comunicación y el intercambio de datos entre componentes ya que éstos pueden estar implementados en distintos lenguajes, ejecutarse en distintos procesadores, usar diferentes modelos de datos, representar información de manera distinta o usar distintos protocolos de comunicación. Generalmente se compran comercialmente ya que son de propósito general*)*.

La principal desventaja de esta arquitectura es que es más compleja de diseñar que aquella de la que hereda conceptos que es conocida como arquitectura en capas y será explicada a posteriori.

Arquitecturas en capas

De nuevo pertenecen al estilo arquitectónico de llamada y retorno pero los sistemas con ésta arquitectura deben reflejar la estructura lógica de la aplicación. Una forma posible es la mostrada en la Figura 25.

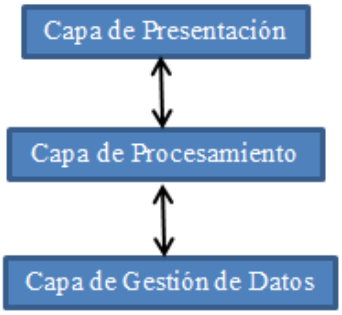

Figura 25
Esquema arquitectura en tres capas

La capa de presentación es aquella en la que se presenta la información al usuario e interacciona con el mismo. En la de procesamiento se encuentra la lógica de la aplicación y en la de gestión de datos se realizan las operaciones con las bases de datos y los archivos. En sistemas distribuidos ésta separación permite la distribución de cada capa en una máquina distinta.

También se puede hacer sobre dos niveles si el sistema es suficientemente simple y en ese caso se la suele conocer como arquitectura **Cliente-Servidor**. Para ello las dos capas inferiores (procesamiento y gestión de datos) se unifican en una única capa denominada servidor; por tanto la estructura queda como CLIENTE←→SERVIDOR, aunque puede estar compuesta por varios clientes que hacen peticiones y varios servidores idénticos que proporcionan el procesamiento y la gestión de los datos. En este planteamiento el cliente se denomina: *"cliente fino"*; pero existe una variante denominada *"cliente grueso"* en la que el servidor sólo realiza la gestión de los datos y el procesamiento se unifica con la capa de presentación formando el cliente. Cabe resaltar que la distribución en dos niveles puede provocar problemas de escalabilidad y rendimiento en el caso del cliente fino y de gestión del sistema en el caso del cliente grueso. Pero estos problemas se solventan usando el la arquitectura en tres niveles que es más escalable, reduce el tráfico en la red y su capa intermedia es fácilmente actualizada al estar localizada centralmente. En la Figura 26 se muestra un ejemplo de la estructura de un sistema bancario por internet que utiliza tres capas.

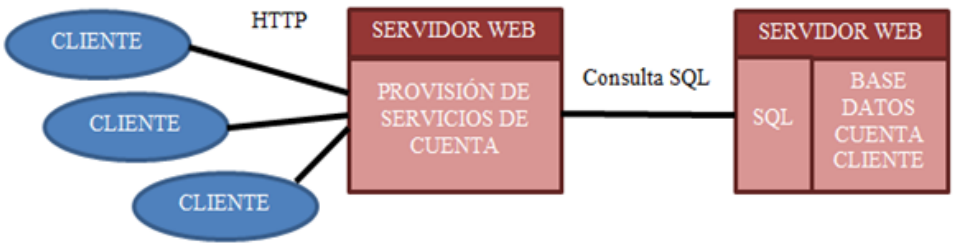

Figura 26 Ejemplo de sistema bancario con arquitectura en tres capas

MVC – Modelo Vista Controlador

De nuevo pertenece al estilo de llamada y retorno e inspirado por las arquitecturas de capas, especialmente pensado para sistemas donde se requiere el uso de interfaces de usuario y ante la

necesidad de separar los conceptos surge con la idea de separar el código en tres capas acotadas por su responsabilidad: Modelos, Vistas y Controladores; así quedan separadas la interfaz de usuario, la lógica de control y los datos de la aplicación.

- El modelo contiene una representación de los datos que maneja el sistema, su lógica de negocio y sus mecanismos de persistencia.
- La vista o interfaz de usuario compone la información que se envía al cliente y los mecanismos de interacción con éste.
- El controlador actúa como intermediario entre el Modelo y la Vista gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno.

El esquema mostrado en la Figura 27 representa la interacción entre las distintas capas.

Figura 27 Interacciones en una arquitectura Modelo-Vista-Controllador

De llamada y retorno

Aunque es un estilo arquitectónico y se han explicado ya algunas de ellas que pertenecen a ésta forma de diseñarlas, al haber tantas de ellas se ha considerado oportuno comentarlo de manera genérica. Su principal característica es que se preocupan por el flujo de control entre subsistemas. Permite una estructura fácil de modificar y ajustar a escala. Se subdividen en dos tipos:

- **Arquitecturas de programa principal o de control centralizado**. El axioma en el que se basa su funcionamiento es que uno de los subsistemas controla al resto. Dentro de ellos se pueden diferencias otras dos clases:

  - *Llamada-Retorno puro*: La delegación de tareas conlleva una llamada y una parada a espera de respuesta (retorno) del componente llamado para continuar. Responde al esquema mostrado en la Figura 28.

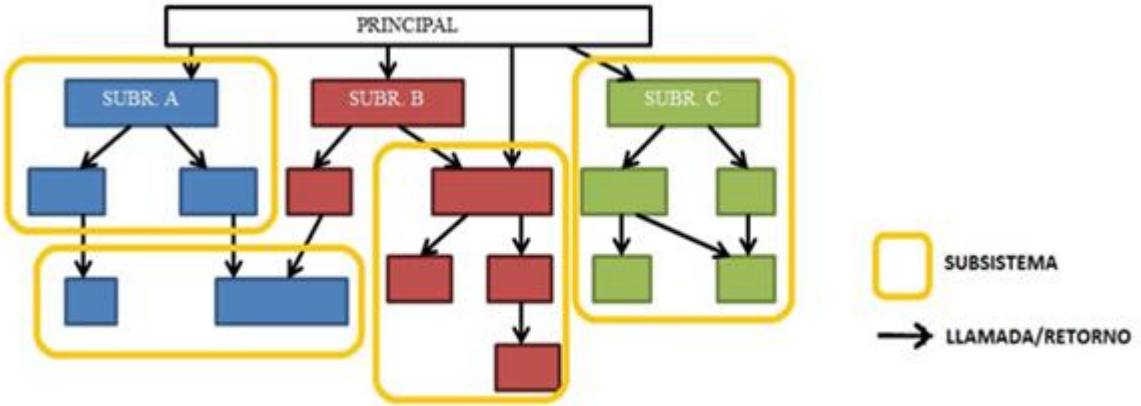

Figura 28 Esquema arquitectura de control centralizado

- *Modelo Gerente:* Un componente (o subsistema) es el gerente del sistema y controla a los otros procesos del sistema. Responde al esquema de la Figura 29.

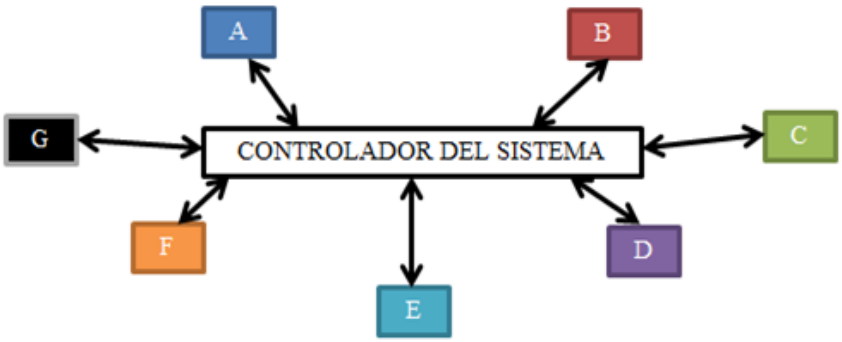

Figura 29 Esquema arquitectura de modelo gerente

➢ **Arquitecturas de llamada de procedimiento remoto o de control basado en eventos.** A diferencia de los modelos centralizados en los que las decisiones suelen basarse en alguna variable de estado del sistema, en éstos el control se basa en eventos generados externamente. Como ejemplo se puede mencionar el *"broadcast"* en el que los eventos se transmiten a todos los subsistemas. Aunque este subtipo es discutido que pertenezca a este estilo arquitectónico por parte de la comunidad ya que algunos de ellos consideran que pertenece más bien al estilo Peer-to-Peer. No hay argumentos esgrimidos por ninguna de las dos corrientes de pensamiento que pesen más que los de la contraria por eso se ha optado por incluirlo en una de las posibilidades y comentar la disputa existente.

De Código Móvil

Este estilo arquitectónico trata de mostrar la posibilidad de que el código se ejecute en distintas máquinas e incluso en máquinas virtuales para tener repartido el peso de procesamiento o de almacenaje o para acortar los tiempos de respuesta a peticiones. No es excluyente con los demás, al contrario, es un estilo que aborda una posibilidad de almacenaje o ejecución del código que complementa cualquiera de las otras arquitecturas. Es la forma de trabajar con los conocidos como sistemas distribuidos.

De Capas Estratificadas

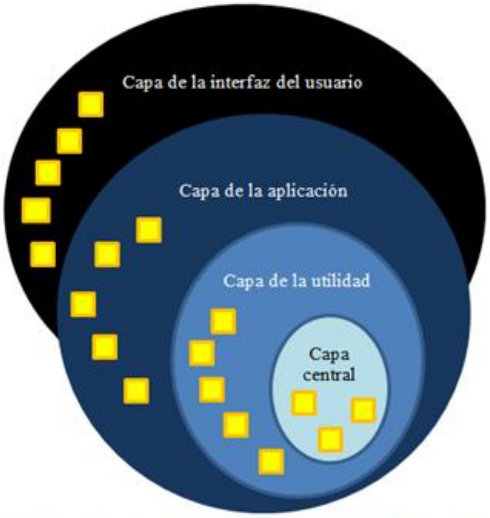

Figura 30 Esquema de arquitectura estratificada

Es un estilo inspirado en el de llamada y retorno de capas, pero su estructura se basa en que cada una de ellas realiza operaciones que progresivamente se aproximan al cuadro de instrucciones de la máquina. En la capa externa los componentes sirven a las operaciones de interfaz de usuario; en la interna, en cambio, son operaciones de interfaz del sistema. Las capas intermedias proporcionan utilidad y funciones del software de aplicaciones. También son conocidas como arquitecturas de capas jerárquicas y se subdividen en dos tipos:

➢ Modelo estricto en el que una capa sólo utiliza la capa inmediatamente inferior.

➢ Modelo relajado en el que se pueden "saltar" capas.

Sus principales ventajas son que aportan facilidad de comprensión, mantenimiento, reutilización y portabilidad.En

cambio, como desventajas se puede decir que el diseño en capas a veces es complicado ya que no siempre un sistema se puede estructurar en diferentes niveles de abstracción desde los requisitos y que el rendimiento puede verse afectado por la interpretación de los comandos en múltiples niveles.

Se puede ver un esquema general de la estructura de las arquitecturas estratificadas en la Figura 30.

Orientadas a servicios (SOA)

Pertenece al estilo arquitectónico Peer-to-Peer y nace del deseo de algunas organizaciones de hacer su información accesible a otros programas y de descomponer su lógica de negocio en pequeñas funcionalidades más sencillas de manejar y mantener denominadas servicios. Así surge el concepto de interface de servicio web.

Un servicio web es una representación estándar para algún recurso computacional o de información que puede ser usado por otros sistemas. Para que se considere que se usa el estándar, el servicio debe ser independiente de las aplicaciones que lo usen y deben permitir que se construyan aplicaciones a base del uso de distintos servicios.

Aunque existen varios modelos de servicios distintos, conceptualmente todos funcionan según el modelo de la Figura 31.

Figura 31 Esquema de arquitectura SOA

Así lo que se tendrá es una plataforma transversal formada por un inventario de servicios de forma que no se solventan las necesidades cambiantes del negocio creando nuevas aplicaciones sino combinando diferentes servicios.

Las principales ventajas que ofrecen estos diseños son que las aplicaciones pueden ser más pequeñas y compactas además de reactivas y con la capacidad de adaptar su operación de acuerdo a su medio mediante el cambio de enlaces a servicios durante la ejecución de la misma.

Orientadas a recursos (AOR)

Las principales desventajas de SOA es la cantidad de especificaciones que requiere para que un servicio sea considerado como tal y que conocer los servicios es necesario para poder usarlos y dado que la Web es ilimitada por naturaleza es imposible conocerlos todos siempre, esto hace que a veces dicha arquitectura sea menos accesible.

Por ello nacen soluciones como la arquitectura orientada a la web para hacerla más ligera restringiendo los servicios a los que usen REST (REpresentational State Transfer) antes que homólogos como SOAP.

Pero no se quedó ahí la lucha contra SOA, se desarrolló como cambio conceptual radical para el desarrollo web la AOR bajo la premisa de que el concepto de servicio en la Web es inadecuado ya que es imposible descubrirlos y mantener un catálogo. Como alternativa, AOR propone la abstracción del recurso y define la web como una colección de recursos los cuales son astronómicamente diversos y no requieren de un inventario, el cual tampoco sería posible ya que no podría mantenerse.

Cada recurso obedece a un protocolo con tres aspectos destacables:

- ➢ Conoce cómo se representa a sí mismo al consumidor.
- ➢ Sabe cómo hacer una transición de un estado a otro.
- ➢ Sabe cómo autodestruirse.

La parte clave es que pueden ser descubiertos, y una vez que son descubiertos pueden representarse a sí mismos. No hay un requerimiento de conocimiento previo del recurso para establecer una conversación.

Es preciso comentar que es una arquitectura completamente basada en REST y aprovecha sus ventajas en cuanto a simplicidad, conocimientos técnicos mínimos y URI (Uniform Resource Identifier) para cada recurso. La única desventaja es que es una arquitectura que se limita a la Web y aunque existen desarrollos en otras plataformas aún no son maduras para competir en ellas con SOA.

Otra tendencia que se puede observar en el desarrollo para la Web es que como XML puede llegar a generar mensajes pesados y cada vez más se encuentran grandes plataformas construidas exclusivamente en el uso de distintos servicios web y tienen su mayor peso en el intercambio de mensajes para éste fin, se está pasando a usar el famoso formato JSON (JavaScript Object Notation) que es mucho más ligero y encaja a la perfección con cualquiera del resto de decisiones arquitectónicas que se tomen, aunque tiene mayor presencia, hasta el momento, trabajando con sistemas REST. Además de que XML y JSON pueden convivir en la misma aplicación por lo que la migración de uno a otro o su elección en función de cada necesidad de comunicación es una decisión independiente y que no condiciona el funcionamiento del sistema.

Para la documentación en esta parte se han consultado las referencias [10-23].

3.2.2. Análisis de tecnologías y comparación entre ellas

Se van a analizar distintas posibilidades para elegir aquella tecnología que mejor encaja en el proyecto, pero ante el amplio abanico que existe, sólo se valorarán como posibilidad aquellas tecnologías que son conocidas previamente o que por su similitud a éstas posean una sencilla curva de aprendizaje.

Éste es el motivo por el que opciones de lenguaje como Ruby, Python, .NET o C# se han descartado y la primera decisión está en la elección entre Java, C, PHP o NodeJS.

El primero de los cuatro en ser descartado es C ya que no es un lenguaje orientado al desarrollo web, no entra dentro de las denominadas “tecnologías de servidor” y aunque es posible hacer un desarrollo usándolo, complica la compatibilidad, mantenimiento posterior e incluso el propio desarrollo ya que en el momento en el que exista un error, fallo o duda no habrá tanta documentación accesible sobre el mismo para la consulta haciendo que la implementación sea más pesada y menos estandarizada.

Para poder seguir decidiendo se va a explicar un poco de cada una de las otras tres opciones.

La primera será NodeJS y lo primero es que no es un lenguaje de programación, realmente es un entorno para tiempo de ejecución, multiplataforma, de código abierto y para la capa de servidor. Está basado en el motor V8 de Google (intérprete ultra-rápido de JavaScript escrito en C++) y usa un modelo de programación orientado a eventos (con la particularidad de que en lugar de presentar un bucle de eventos como una librería lo presenta como un entorno sin bloqueos por llamadas; de manera similar a como lo hace JavaScript en el navegador).

Una de las ventajas de usar Node para el desarrollo es su gran capacidad para manejar decenas de miles de conexiones simultáneas y aún bajo esta situación, al tratarse de la implementación de un API REST que hace que el peso de los mensajes sea de una pequeña cantidad de texto, el volumen de tráfico generado no es alto y una máquina probablemente podría manejar todas las demandas incluso del servicio web más solicitado posible.

Otra de sus grandes ventajas es que provee gran cantidad de módulos para ampliar funcionalidades como las de trabajar con MySQL o con WebSockets entre otras. Además presenta Node Package Module (NPM) que integra y administra los módulos que se estén usando, maneja automáticamente las dependencias facilitando la programación en éste entorno.

Todo esto hace a Node una solución que aporta una gran escalabilidad al proyecto. Además, es ampliamente extendida en la comunidad de programadores de back-end que es ideal para utilizarlo cuando se necesitan hacer muchas cosas al mismo tiempo y por supuesto gracias a la capacidad de conexiones persistentes con los navegadores para aplicaciones en tiempo real.

Además, poder utilizar el mismo lenguaje en cliente que en servidor (JS) facilita el desarrollo y el entendimiento de éste y que al estar basado en eventos compagina a la perfección con la forma de trabajar de las peticiones AJAX del cliente.

Lamentablemente, nada es perfecto y Node también tiene algunas desventajas como puede ser que proporciona una API bastante inestable que muchas veces rompe la compatibilidad hacia atrás de versión en versión; además no dispone de una librería estándar o de librerías en general ya que al no haber tenido años de popularidad no se han desarrollado bases suficientemente potentes y testadas. Sin olvidar que como siempre que algo se basa en lenguajes poco tipados, se generan demasiadas formas de programar y hace que sea difícil mantener los proyectos.

Con toda esta información se concluye que tampoco es un lenguaje apropiado para el proyecto sobre todo por el último argumento ya que uno de los requisitos es que sea sencillo de mantener y Node no aporta cumple este requisito. Por otro lado los beneficios que aporta como compensación no son realmente útiles en este sistema ya que la concurrencia existe pero de apenas unas decenas de conexiones simultáneas en el peor de los casos, situación para la que Node es demasiado “grande”.

Por el momento queda la decisión entre dos lenguajes potentes: Java y PHP, ambos lenguajes con un gran bagaje, muy testados, con mucha información y muy utilizados por lo que se van a tener que ver los pequeños detalles para poder elegir. De hecho basta con hacer una consulta en cualquier buscador para ver que el debate sobre cuál de los dos lenguajes es mejor para su uso en el desarrollo de portales web es habitual y de respuesta compleja.

De java se puede empezar contando que para su uso como lenguaje para desarrollo web se debe poner el foco en un sector concreto de todo el lenguaje: JSPs y Servlets. Además decir que se trata de un lenguaje orientado a objetos que desde 2007 se puede considerar software libre.

Actualmente es un lenguaje ampliamente demandado si nos basamos en la gráfica de la Figura 32 (donde se muestra información extraída de [24]) en la que se muestra junto a PHP según el índice Tiobe (mayor número de búsquedas en los buscadores más valorados, dichas búsquedas están ponderadas por distintos pesos en base al prestigio de cada uno de los 25 buscadores que se

tienen en cuenta) de Mayo de 2017. Si bien es cierto que estas búsquedas no disciernen entre aquellas que servirán para desarrollo de aplicaciones de índole distinto al web (como puede ser una aplicación de escritorio o una aplicación móvil para Android) y Java es un lenguaje que cubre casi todos los tipos de desarrollo por lo que es normal que reciba más búsquedas que PHP.

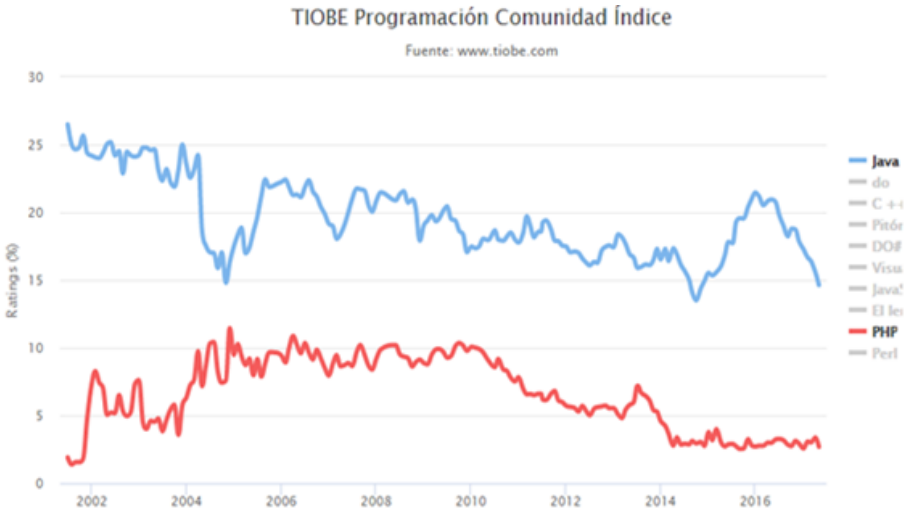

Figura 32 Evolución del índice Tiobe para Java y PHP

En cuanto a PHP comentar inicialmente que es un lenguaje interpretado en el servidor y desde 1995 es software libre. Es ampliamente usado en entornos de desarrollo web por su facilidad de uso, su integración con HTML y su versatilidad de uso en distintos Sistemas Operativos.

El lanzamiento de PHP5 en 2004 introdujo mejoras de rendimiento, de soporte a programación orientada a objetos y de soporte a conexión a bases de datos entre otras.

Es un lenguaje que da mucha confianza ya que es el utilizado en muchas de las páginas con mayor número de visitas del planeta y la cantidad de años que lleva siendo utilizado para este tipo de desarrollo, de hecho nació para facilitar algunas funciones de éste ámbito y todos sus avances se han centrado exclusivamente en el campo de la web.

Ahora que ya se han introducido ambos lenguajes, se van a comparar distintos aspectos que se consideran de especial relevancia para el desarrollo:

- **Modularización:** Observando la tecnología Java usada en cualquier portal se ve una clara estructura en la que se puede diferenciar el modelo MVC con sus diferentes módulos. En cambio, PHP se salta esta diferencia ya que todas las capas se implementan en el mismo fichero.

- **Tipado:** Java es un lenguaje fuertemente tipado, con muchas normas, lo que hace que sea ideal para trabajar con grupos de programadores ya que el propio lenguaje marca las pautas, pero la flexibilidad que aporta PHP hace que la programación sea mucho más intuitiva haciéndolo idóneo para proyectos más pequeños.

- **Mantenibilidad:** Generalmente un sistema con una estructura clara de componentes es más fácil de mantener a futuro por lo que debido a lo expresado en los puntos anteriores, Java ofrece mayor mantenibilidad sobre todo ante un aumento en la funcionalidad del proyecto (cantidad de código). Aunque el rigor en la forma de programar de un buen desarrollador PHP puede compensar esta debilidad.

- **Crecimiento del sistema:** Cuando se desarrolla un sistema se debe tener en cuenta que a futuro necesitará aumentar funcionalidades que pueden no ser implementadas por el mismo programador, para lo cual la estructura de nuevo aporta una guía para el mantenimiento de una estrategia de desarrollo que garantice el no empobrecimiento del código. De nuevo Java provee una mayor capacidad en este sentido y de nuevo en PHP se puede compensar esta debilidad con el rigor en la forma de programar.

- **Coste de desarrollo:** Siempre será menor en un proyecto PHP que en un proyecto Java ya que aporta una programación mucho más directa con resultados inmediatos además de que para programar un sistema en Java es necesaria mucha más preparación y experiencia. Por este motivo Java suele estar más orientado a proyectos de mayor magnitud ya que en ellos suele rentar mucho más la inversión necesaria.

- **Integración externa:** Considerando "integración externa" como el uso de herramientas, métodos y funcionalidades desarrollados por otros programadores y que son integrables en el sistema actual y siendo Java un lenguaje amplio y variado, supone una ventaja tanto en el momento del desarrollo como en la repercusión final de éste: existen multitud de Frameworks que facilitan los trabajos de los desarrolladores y aligeran los tiempos, adicionalmente existen "módulos" ya desarrollados de libre distribución disponibles para que los usen. En cambio, PHP al no ser estructurado no tiene tantas posibilidades sobre todo a nivel conceptual ya que la madurez que aporta ha llevado al desarrollo de algunos Frameworks que ayudan bastante pero que no guían tanto al desarrollador ya que el lenguaje como axioma tiene la libertad de estructura.

- **Seguridad:** En Java se deja en manos de servidores externos o de Frameworks como Spring Security, en cambio en PHP se realiza de una manera más manual, de primeras puede parecer que Java aporta ventaja en este sentido, pero como se encuentre una vulnerabilidad en el servidor o en el Framework, los desarrolladores en principio no pueden hacer nada para subsanarlo, en cambio si ellos han implementado la seguridad pueden actuar sobre el desarrollo realizado. Aunque en sentido contrario, la seguridad externalizada asegura que se valoran todos los ataques conocidos cosa que si se desarrolla manualmente no se puede garantizar.

- **Rendimiento:** PHP aporta mayor rendimiento ya que es mucho menos pesado que Java; esto se puede ver en la usabilidad y la sensación de rapidez que se ofrece al usuario.

- **Escalabilidad:** Al principio Java fue un gran competidor de PHP en cuanto a rendimiento ya que éste último sufría una gran pérdida de rendimiento al aumentar el número de usuarios del sistema, pero desde el lanzamiento de PHP5 la experiencia de los desarrolladores apoya que se ha solucionado el problema de PHP no habiendo gran diferencia entre los dos en éste aspecto.

Ninguno de los aspectos analizados hace que un lenguaje sea peor o mejor que el otro, simplemente cubren distintas necesidades de los proyectos de una forma u otra y hacen que el proyecto se adapte mejor, que el desarrollo sea más adecuado con las funcionalidades o que los costes se reduzcan pero en ambos lenguajes se puede programar el servicio que se busca en este proyecto sin problema. Se tome la decisión que se tome no se puede considerar errónea, solamente se podrá estar más o menos de acuerdo.

En la Tabla 3 se resumen los resultados anteriormente explicados y para la documentación necesaria en esta parte se han consultado las referencias [24-35].

CARACTERÍSTICA	PHP	JAVA
Modularidad	Débil	Fuerte
Tipado	Débil	Fuerte
Mantenibilidad	En base: menor	En base: mayor
Software Libre	SI	SI
Orientado a Objetos	NO	SI
Escalabilidad	Buena	Buena
Rendimiento	Mayor	Menor
Seguridad	Manualmente	Con herramientas
Integración externa	En base: menor	En base: mayor
Coste de desarrollo	Menor	Mayor
Crecimiento del sistema	En base: menor	En base: mayor
Abarca más campos que la web	NO	SI

Tabla 3

3.2.3. Conclusiones y diseño final

En cuanto a las arquitecturas se refiere se puede decir que en función de qué es aquello que "pesa" en el estilo hay distintas divisiones, por ejemplo: la arquitectura de código móvil tiene como principal característica la máquina física en la que corre, en cambio la de las capas estratificadas se basa es el tipo de código que se ejecuta en cada capa independientemente de en qué máquina física se esté ejecutando. Por lo tanto, en la realidad del desarrollo de un proyecto la arquitectura final suele ser una combinación de algunos de los estilos que se han explicado.

Por ese motivo y a la vista de requisitos de la aplicación, de su contexto y de las posibles arquitecturas, se considera que por el tipo de funciones que se espera que desarrolle el nuevo servidor (gestión de distintos recursos, servicio de autenticación y comunicación de datos con los distintos clientes para su posterior utilización) y el deseo de convivencia con la estructura existente, la arquitectura más apropiada corresponde a un estilo Cliente-Servidor en cuanto a que varios clientes (navegadores) harán solicitudes al servidor que contendrá las funcionalidades para la gestión y AOR en sentido de que no se va a trabajar orienta a objetos, si no que el intercambio con dichos clientes será de recursos. Además, puesto que es un sistema pensado en exclusividad para la web, la principal desventaja de la arquitectura AOR no afectará al proyecto.

También se decide que por el requisito de ligereza de la aplicación ante su necesidad de convivencia con el resto de servicios que aloja la máquina del departamento, se trabajará con mensajes JSON para la comunicación con el cliente.

Resumiendo, se va a realizar un API REST que se comunicará con el cliente a través de mensajes JSON para la gestión de los recursos (en concreto experimentos de usuario y usuarios para los administradores).

Por otro lado, para las tecnologías se ha considerado que teniendo en cuenta que como conclusión se puede decir que para sistemas web relativamente pequeños, con menores costes tanto económicos como de tiempo PHP tiene ventajas frente a Java, sobre todo con el uso de algún Framework que aporte estructura (mejora la mantenibilidad, la integración externa y el crecimiento del sistema). Como éste es el caso del proyecto que se está desarrollando, se elige este lenguaje.

Ahora falta valorar el Framework de PHP más apropiado para el desarrollo del API REST sobre todo para que permita asegurar la seguridad en el login (nótese que la definición de framework según Wikipedia es: "un marco de trabajo, es decir, un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar"; se podría resumir y simplificar diciendo que: "es un esquema (un esqueleto, un patrón) para el desarrollo y/o la implementación de una aplicación"). Hay que tener en cuenta que se van a analizar algunos pero existen muchísimos más y para documentarse se han consultado las referencias [36-41]:

- **Symfony:** Fue lanzado a finales de 2005 pero no llegó a ser realmente conocido hasta 2007. Es un framework MVC (pero modificando su forma de trabajar) para aplicaciones web que aporta:
  - ➢ Flexibilidad de desarrollo y a la hora de elegir un ORM ya que se adapta a muchos.
  - ➢ Sus componentes se pueden incorporar en proyectos mayores. Automatiza las tareas comunes permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación y por consiguiente reduciendo el tiempo de desarrollo de la misma.
  - ➢ Se utiliza en sitios webs de comercio electrónico de primer nivel por lo tanto aporta muchísima fiabilidad.
  - ➢ Es multiplataforma, se puede ejecutar tanto en Unix como en Windows.
  - ➢ Preparado y estable para desarrollar aplicaciones a largo plazo y adaptables.
  - ➢ Fácil de mantener al proporcionar código legible.
  - ➢ Rápido y consume menos memoria que sus competidores.

    Aunque como contras éste framework tiene:
  - ➢ Una vaga y poco referenciada documentación.
  - ➢ Aporta seguridad pero muy complicada de utilizar.
  - ➢ Se complica mucho la forma de parsear archivos.
  - ➢ Utiliza programación orientada a objetos y características como los espacios de nombres por lo que necesita al menos PHP 5.3 para funcionar.

- **Laravel:** Fue lanzado en 2011 influenciado por Ruby on Rails y ASP .NET MVC y en seguida se hizo muy popular debido a su elegancia y simplicidad a la hora de desarrollar código. Gran parte del mismo está formado por dependencias de Symfony entre otros ya que pretende aprovechar al máximo las ventajas de otros frameworks y de las últimas versiones del propio lenguaje.

  Además resaltan como pros para la elección de su uso:
  - ➢ Mucha documentación y de calidad,
  - ➢ Permite trabajar con el MVC tradicional aunque promueve el desarrollo de “Routes with Closures” para hacer el código más claro.
  - ➢ Característica adicional para el enrutamiento inverso.
  - ➢ Un CLI que comprende herramientas para tareas avanzadas y migraciones.
  - ➢ ORM (Eloquent) fácil de entender que permite la creación de relaciones de las bases de datos de forma simple. Aunque también dispone de otros recursos para interactuar con los datos como que el nombre de la clase coincida con el nombre de la tabla (igual que en Ruby on Rails) y una ruta para interactuar con el modelo.
  - ➢ Al ser modularizado permite que el código se reutilice sin problemas.
  - ➢ El motor de plantillas (para la vista) que utiliza acelera la compilación y hace que los usuarios puedan incluir nuevas características de manera sencilla haciendo un código más limpio y rápido gracias a que incluye un sistema de caché.

    En cambio, tiene como contras:
  - ➢ Es lento.
  - ➢ Se basa mucho en herencia, concepto que no es siempre sencillo de entender.
  - ➢ No tiene gran apoyo de la comunidad por lo que es difícil encontrar ayuda a la hora de resolver cualquier tipo de duda.
  - ➢ Los métodos encargados del enrutamiento inverso son complejos.

❖ **Wave:** Es un micro-framework (framework parcial, le faltan la mayor parte de las funcionalidades para desarrollar una aplicación completa, generalmente se centra en facilitar cómo recibir peticiones HTTP encaminándolas al controlador apropiado, recibiendo la respuesta del mismo y devolviendo una respuesta HTTP. A menudo están diseñados para el desarrollo de las API para otro servicio) que se construye sobre la arquitectura MVC, algunas de las ventajas que ofrece son:

- ➢ Está construido sobre una arquitectura API nativa
- ➢ Posee almacenamiento en caché por lo que aumenta la velocidad.
- ➢ Aporta un sistema para la gestión inteligente de imágenes y recursos ofreciendo aún una mayor velocidad a sistemas en los que éste tema tenga gran peso.
- ➢ Es compacto, sin bibliotecas opcionales por lo que tiene una huella pequeña y se desarrolla con la optimización en mente.
- ➢ Trabaja con objetos jerárquicos cargados dinámicamente.
- ➢ Devuelve XML, CSV, JSON, HTML, PHP nativo y otros formatos de datos posibles entre ellos ofrece la posibilidad de la salida de datos en formato comprimido.
- ➢ Sistema de control de usuarios y permisos.
- ➢ Soporte completo para Apache y NGINX.
- ➢ Mucha documentación para solventar posibles problemas encontrados en el desarrollo además de una guía de uso bastante completa.

Como partes en contra:

- ➢ Tiene una pronunciada curva de aprendizaje.
- ➢ Sólo funciona sobre PHP 5.3 o superior.
- ➢ La documentación de la API se genera automáticamente pero no es de alta calidad.

❖ **Slim:** Micro-framework exclusivo para el diseño de APIs REST. Nace en 2011 y en principio para trabajar con Apache y MySQL. Algunas de sus ventajas son:

- ➢ Reducido tamaño, es un proyecto muy compacto.
- ➢ Excelente documentación que hace que la curva de aprendizaje no sea especialmente pronunciada.
- ➢ Sintaxis muy simple: solamente tenemos que instanciar la aplicación, asociar desde el router las peticiones HTTP con funciones anónimas y poner dicho router en marcha.
- ➢ No necesita levantar una infraestructura con módulos y opciones que no se van a utilizar.
- ➢ Encaja a la perfección como backend de una aplicación en Backbone.
- ➢ Fácilmente mantenible debido a su sencillez y a la estructura basada en MVC que posee.
- ➢ Su poco peso aporta gran velocidad y poco uso de memoria para las aplicaciones con el volumen adecuado para su uso.
- ➢ Renuncia a los controladores y componentes abstractos a pesar de estar basado en MVC para hacerlo más simple y fácil de usar.
- ➢ Trabaja con caché para aportar rapidez, además de disponer de un cifrado seguro de cookies.
- ➢ Aporta una configuración sencilla y un sistema para el tratamiento de mensajes flash.

Algunas de sus desventajas son:

- ➢ Aunque tiene funciones para la autenticación no son muy extensas.
- ➢ Necesita versiones iguales o superiores a PHP 5.3 ya que usa espacio de nombres.
- ➢ No es apropiado para una aplicación con gran volumen de rutas o cientos de objetos, en dicho caso es mejor valorar un framework completo.

A la vista de las características vistas, se puede decir que los frameworks completos son demasiado pesados para las funciones que se quieren desarrollar en el nuevo servidor, por lo tanto se debe elegir entre los dos micro-frameworks analizados: Slim y Wave.

Como punto fuerte de Wave para nuestro sistema se tiene que proporciona un sistema de usuarios y permisos, pero aunque Slim no lo proporciona y habría que gestionarlo como un recurso más de la API REST, su curva de aprendizaje es más suave y su sintaxis más simple lo que facilita el trabajo ya que ambos frameworks se deben aprender desde el principio.

Además Slim es más sencillo de mantener sin conocimientos por ser más legible y esto es una ventaja importante a la hora de poner el código en marcha en un departamento no especializado en programación. Por tanto se decide trabajar con Slim.

3.2.4. Elección y desarrollo de la Base de Datos

La elección de la tecnología para la base de datos viene condicionada por el gran número de proyectos que hechos sobre Apache y con Slim (PHP) utilizan MySQL ya que los tres encajan a la perfección.

Antes de entrar en el desarrollo de la base de datos y el diseño de los datos, debido a las decisiones de diseño tomadas hasta el momento, se va a instalar un entorno en el que trabajar con las tres tecnologías y debido a que se trabaja sobre Linux, se instalará XAMPP que es un paquete de instalación independiente de plataforma que consiste en el sistema de gestión de bases de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script: PHP y Perl. Aunque a partir de la versión 5.6.15 cambió de MySQL a MariaDB que es un fork de MySQL con licencia GPL. Para instalarlo se debe seguir los pasos del anexo A.

Ahora que se dispone de todo el entorno para empezar a trabajar en la nueva parte de la aplicación, se debe pensar qué se va a necesitar almacenar en la base de datos y cómo, para lo que se recuperan las funcionalidades que se desean que esta parte aporte: gestión de usuarios y experimentos; por lo tanto, se van a necesitar almacenar éstas dos "entidades". Por consiguiente, la base de datos debe contener dos tablas: una por entidad a almacenar.

La entidad experimento tendrá un id autogenerable, un nombre, un id de usuario que es una "foreing key", un origen (indicará si el experimento ha sido asignado (1) o es el usuario quien lo ha creado (0)), fecha de modificación, fecha de ejecución, número de ejecuciones y una columna por variable teórica de configuración del experimento salvo para aquellos valores que son fijos con el planteamiento actual de los mismos (DirectWup, por ejemplo).

Por su parte, la entidad usuario contará con un id autogenerable, nombre, email, contraseña, DNI, rol, flag de activación y la fecha de creación.

3.2.5. Desarrollo del código de la aplicación

Lo primero que hay que decir es que a partir de la instalación de XAMPP, la ruta que alojará los ficheros tanto de la parte de cliente como de la parte de servidores es: */opt/lampp/htdocs* por lo que se pasa la carpeta que contiene serverSECO y la parte web de cliente con todas las nuevas funcionalidades a dicha ruta.

A continuación se crea una nueva carpeta dentro de la ruta donde se alojarán los ficheros del nuevo servidor del API REST. Se descarga Slim, se descomprime la carpeta y se copia dentro de la carpeta sobre la que desarrollar la API.

Los dos primeros ficheros a desarrollar son aquellos que se pueden reutilizar para cualquier proyecto ya que se encarga de la conexión a base de datos MySQL (Conexión.php) y de la codificación UTF8 (funciones.php).

El siguiente fichero implementado (api.php) es el que gestiona cómo actuar ante peticiones de diferentes rutas: /usuario, /usuarios_modificar/:id y /experimentos_modificar/:id y además es el fichero que pone en funcionamiento la api con la sentencia *run()*.

Al ir programando se infiere la necesidad de programar en JS funciones que atenderán distintas necesidades, para ello se crea el fichero main.js con las funciones: subirFichero, eliminarAll, cargarExperimento, comprobarEmailExiste, registrarUsuario, eliminarUsuario, modificarUsuario, modificarUsuarioBD, asignarExperimento, asignarExperimentoBD, modificarExperimentoBD, nuevoAsginar, cargarUsuarios, user_cargarExperimentos, nuevoExperimento, cargarExperimentoUsuarios, modificarExperimentoBDUsuario, insertarExperimentoBDUsuario, eliminarExperimento y ejecutar.

Pero el concepto de Slim y PHP es que estas funciones accedan a la base de datos a través de llamadas AJAX a ficheros PHP que almacenarán funciones que realizarán las consultas (dicha forma de trabajar se puede ver reflejada en un ejemplo en la Figura 33), por lo que se desarrollan también para este fín los ficheros asignarExperimento.php, comprobarEmail.php, comprobarNumExp.php, eliminarAll.php, eliminarExperimento.php, eliminarUsuario.php, getExperimentos.php, modificarExperimento.php, modificarUsuario.php, registro.php y subirAdjunto.php.

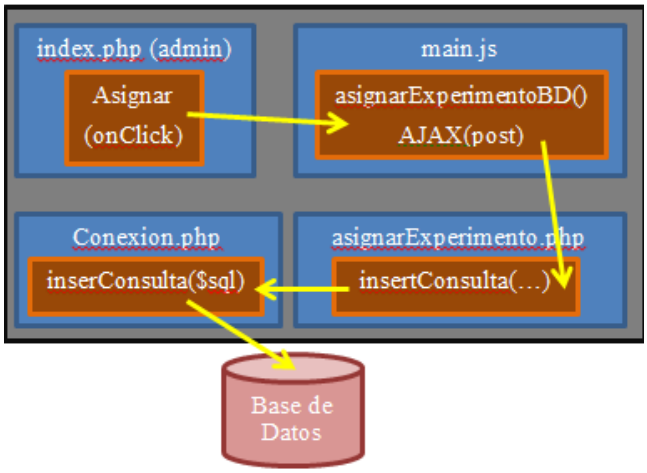

Figura 33
Ejemplo de interacción para la asginación de un experimento

Además, todo portal web desarrollado con php necesita de la implementación del fichero index.php que representa la estructura de un documento html con su head, su body y demás etiquetas e intercala funciones de PHP con <?php ?>. En él se tramita que para acceder a la aplicación se necesita sesión y a través de la librería *toogle* se definen áreas modales que son visibles en función de acciones del usuario haciendo así que con un único fichero que contiene todas las funcionalidades para los diferentes usuarios, se proporcione una estructura lo más limpia posible y ligera a su vez.

A la hora de implementar la creación de experimentos por parte de un usuario se ha desarrollado con un límite de 5 por usuario para asegurar que la base de datos no crezca demasiado y teniendo en cuenta que siempre se pueden ejecutar experimentos configurándolos en el momento y desde un fichero sin necesidad de pasar por la base de datos ni machacar los que almacenados en ella se tuvieran. Este valor se puede modificar, se consideró tratarlo como una variable global pero puesto que no se utiliza más que en la comprobación que se realiza en el fichero comprobarNumExp.php se decidió introducir manualmente.

Con todo esto queda en funcionamiento el nuevo servidor con el servicio de autenticación incluido.

Aclarar también que es importante el mantenimiento de las rutas a la hora de poder acceder a la parte de cliente: tanto en el fichero main.js, dentro de la aplicación ejecutar en la última línea de código, como en el fichero index.php, al comprobar la existencia de la sesión tipo user en el hipervínculo, se debe acceder a la ruta del fichero index de la parte web.

Ante cómo se ha probado la implementación, se ha hecho manualmente cada funcionalidad que se iba añadiendo y que la integración de ésta con el resto no afectaba al comportamiento final; además y como es habitual en programación se ha pedido a un compañero que no conoce el código que pruebe la aplicación como si de un usuario final se tratase intentando buscar fallos.

3.3. ADAPTACIÓN DE LA CAPA DE CLIENTE PARA EL FUNCIONAMIENTO CON LA APLICACIÓN DE GESTIÓN.

En esta sección se va a modificar parte del código del primer área para que pueda recibir los datos de la aplicación de gestión para concluir el correcto funcionamiento de los requisitos.

Para que funcione la "carga" de experimentos en la interfaz de cliente que se hace a través del almacenaje que proporciona el navegador, se necesita modificar el fichero index.html de la parte de cliente para que reciba y muestre los nuevos valores, para ello se crea una función de JavaScript al final del documento en la que con un condicional se comprueba si *parsing* es true y si es así es porque hay información en el almacenaje local y variable por variable se va recuperando el valor almacenado y asignando al identificador correspondiente.

Además, para evitar un ataque por sobrecarga de peticiones anónima se pretende proteger el acceso a la denominada parte de cliente teniendo que autenticarse previamente en el servidor de gestión que se acaba de implementar para poder acceder a ésta página. Para ello hay que convertir el fichero index.html a index.php para poder insertar código php y que se ejecute dentro del archivo.

A continuación nada más entrar en el cuerpo del código html se inserta un fragmento php en el que se recupera la sesión del navegador y si no existe sesión carga un aviso de que no se tienen permisos para acceder a la página, que se debe autenticar y el enlace para acceder a la página para autenticarse y si existe carga todo el antiguo fichero dentro del caso del condicional así sólo con una sesión se puede acceder al sistema y si se exceden las peticiones se puede rastrear más fácilmente quién ha realizado el ataque.

Las pruebas se realizan manualmente intentando acceder por enlace y por url y se deniega el acceso salvo que haya una sesión válida iniciada en el navegador correspondiente.

Además en este apartado se ha analizado la gestión de colas que provee el sistema porque se creía que no estaba haciendo correctamente su trabajo y se iba a necesitar gestionar. Como se muestra en la Figura 34, se creía que tras bloquear la entrada de peticiones y empezar a encolar en el instante A, se desbloquearía y aceptaría peticiones desde el momento B o C, pero tras hacer pruebas de acceso simultáneo, de peticiones de un breve tiempo de ejecución durante el envío de datos del servidor a la aplicación de cliente y analizar los mensajes que intercambian la placa,

serverSECO y la parte web se llega a la conclusión de que el sistema de gestión de colas funciona perfectamente siguiendo no desbloqueando la entrada de peticiones hasta el instante D.

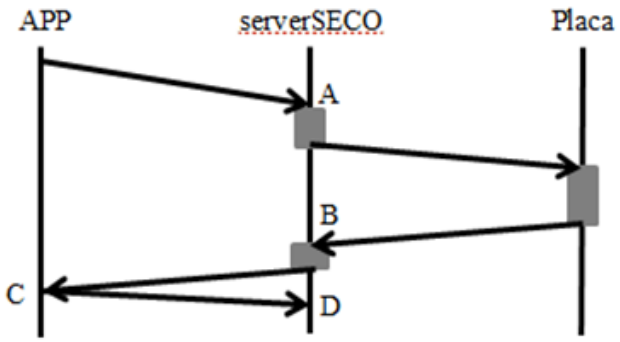

Figura 34
Intercambio de mensajes entre la aplicación y la placa

4. CONCLUSIONES Y LÍNEAS FUTURAS

4.1. PROBLEMAS ENCONTRADOS

Se pueden destacar cuatro grandes problemas encontrados a lo largo del desarrollo del proyecto.

El primero de ellos fue una gran pérdida de tiempo y esfuerzo intentando desarrollar la aplicación de gestión con un framework de Java denominado Spring, se trata de un framework completo, bastante pesado y con mucha necesidad de configuración en ficheros XML que llevó a que en algún punto la configuración de seguridad no funcionase y ante la imposibilidad de solventar el problema y el gran esfuerzo sobre todo temporal que llevaría solucionarlo y sin garantías de que se fuera a conseguir se decidió cambiar de lenguaje y de framework habiendo aprendido la lección de que para proyectos de menor envergadura un framework demasiado potente puede tener una curva de aprendizaje demasiado pronunciada y que Java es un lenguaje que requiere de mucha especialización y experiencia.

El segundo problema al que nos enfrentamos, ocurre que al comprobar que la gestión de colas funcionaba correctamente, los ficheros de peticiones distintas son iguales, sin serlo los experimentos, primeramente se consideró la posibilidad de que existiera un error de código en la configuración del diagrama en el tratamiento de los datos, pero finalmente se encontró que había sido un error a la hora de configurar el entorno, es importante que las rutas estén bien configuradas según se indica en el Anexo C.

El tercero de ellos se encontró durante las pruebas realizadas por un agente ajeno a quien desarrolló el código. Se trata de que al cargar un experimento en la parte de cliente desde la aplicación de gestión, se cargan correctamente todos los valores tratados en ésta del experimento, pero se machaca el valor fijo de la variable Controller Type poniendo éste a 0 –Test-. Y que tras cargar un experimento, se deben machacar los valores predefinidos ya que al acceder posteriormente a la parte de configuración de la aplicación básica desde otro usuario, se carga con los valores del experimento de la sesión anterior. Para solucionarlo se analizó el código y se observó, por un lado, que no se reseteaban las variables de sesión por lo que para solucionarlo al cerrar el condicional de reasignación se deben resetear con la sentencia: *“localStorage.setItem(“parsing”,””);*. Por otro lado, se sobreescribe siempre dentro del condicional el valor de la variable ControllerType a su valor fijo PID (1) y así se garantiza que no se modifica.

El último se trataba de la parte de seguridad de la aplicación ya que el cierre de sesión por time-out estaba delegado a la configuración de PHP de la máquina que lo alojase y aunque Slim garantiza protexxión frente a los ataques de inyección de SQL, no estaba considerada la opción de una autenticación hecha pero no aceptada haciendo que desapareciese la posibilidad de volver a autenticarse desde el navegador que hubiera realizado el ataque. Para solucionar esta parte, en el fichero index.php en los casos del condicional de sesión se añade un elemento *else* que redirecciona a la página inicial de la aplicación de gestión. Además al iniciar sesión se almacena en la sesión del navegador la hora de modificación de la sesión y cada vez que se intenta realizar alguna acción se comprueba el tiempo que ha transcurrido y si es mayor a 10 minutos se vacían los datos de sesión y se cierra y si no se actualiza la hora de última modificación y permite la acción.

4.2. LÍNEAS FUTURAS

Existen multitud de funcionalidades que se pueden añadir a la aplicación, como la visualización de estadísticas a raíz de las fechas y número de ejecuciones que se almacenan en la base de datos o aviso por correo electrónico al usuario de que se le ha asignado un experimento.

Además la aplicación requiere de un trabajo de diseño estético para adaptarse mejor al resto de aplicaciones de que dispone el departamento aportando así una visión más corporativa del mismo. Pero éstos son sólo algunos de los posibles puntos de evolución que tiene el proyecto en el punto en el que se entrega.

4.3. RESUMEN DE LA IMPLEMENTACIÓN

Para concretar sobre lo que se ha implementado, se puede concluir que se ha desarrollado una aplicación que permite el acceso remoto a los dispositivos de un puesto de laboratorio en el que se transmiten los parámetros de trabajo de un motor de corriente continua con la finalidad de recibir los valores de diferentes señales obtenidas de la ejecución del experimento.

Para ello se ha ido desarrollando la aplicación de manera que se pueden discernir dos versiones de la misma, una con funcionalidades relativas a la configuración y visualización de los resultados y otra (que se considera la completa) en la que para acceder a la anterior se ha desarrollado "delante" una aplicación de gestión que aporta el acceso autenticado, la gestión de los usuarios y sus experimentos y la principal funcionalidad para los experimentos de cargarlos desde la base de datos en la que pueden ser almacenados. Además dicha aplicación proporciona funcionalidades adicionales que facilitan el trabajo de gestión del personal como puede ser la asignación de experimentos a través de la base de datos a un usuario concreto como ejemplo.

La primera versión se basa en una aplicación iniciada por el departamento en Ansi-C, para ella se utilizan las herramientas de front-end HTML5, CSS3, JavaScript, HighCharts y JQuery.

Para el desarrollo de la aplicación se han analizado diversas arquitecturas, lenguajes y frameworks y tras errar en la decisión se acabó decidiendo desarrollar un API REST orientado a recursos en PHP con apoyo en el micro-framework Slim que está orientado a dicho tipo de implementaciones y facilita enormemente el desarrollo ocupándose de cuestiones como la seguridad.

4.4. CONCLUSIONES

Como conclusiones del trabajo encontramos que existen multitud de formas de desarrollar una aplicación al igual que muchos lenguajes y tecnologías con los que hacerlos y que la elección de unos u otros afecta a la calidad del proyecto, la concordancia con los requisitos sobre todo no funcionales y el esfuerzo invertido en el desarrollo del mismo.

La programación no es una ciencia exacta y al final la experiencia hace ver cuál es la forma de programar y las herramientas que mejor encajan con el desarrollador. Pero a pesar de ello, la tecnología avanza constantemente y es imposible conocer todo por lo que la capacidad de investigación y análisis para encontrar cómo resolver un problema son imprescindibles para la labor de implementación.

Por otro lado, el trabajo con un código ajeno es muy complicado, ya que entender la forma de trabajar de un compañero y por lo tanto, la estructura y finalidad del código es muy complicada y requiere de un gran esfuerzo de análisis y a veces lleva a errores que si no se está muy atento pueden verse en producción con los costes y responsabilidades que este hecho puede conllevar.

También ha quedado claro que una mala decisión de una tecnología o el desconocimiento de ella puede acarrear graves problemas de tiempo en el desarrollo de un software, por lo que el tiempo invertido en el análisis de requisitos y la investigación de qué tecnologías encajan mejor con dichas especificaciones es un tiempo muy rentable al final.

Por último valorar que el esfuerzo invertido en este proyecto al final ha resultado en una aplicación funcional que facilita el trabajo y la organización de los miembros del departamento y que al estar diseñada por piezas permite ir introduciéndola por funcionalidades para que el personal se habitúe a trabajar con ellas sin perturbar en exceso su labor.

5. ANEXOS

5.1. ANEXO A

INSTALACIÓN DE XAMPP

Para realizar una correcta instalación del paquete se debe:

1º Descargar el fichero .run de https://www.apachefriends.org/es/download.html y cambiarle los permisos al fichero para poder ejecutarlo sin ser super-usuario

2º Ejecutar el fichero con *./nombreDelFichero.run* y seguir los pasos del asistente de instalación. Tras finalizar la instalación se tendrá acceso al interfaz gráfico de XAMPP en el que se pueden levantar y parar los servicios de MySQL, Apache y FTP; si no hay ningún otro servicio en los puertos correspondientes se podrán levantar todos sin problema, en caso contrario habría que saber qué proceso está ocupando el puerto y "matarlo" y volver a intentarlo. (En posteriores conexiones a XAMPP puede no encontrar el interfaz gráfico, para lanzarlo puede agregar el panel de control o bien ejecutar en la terminal: *sudo /opt/lampp/administrador-linux(x64).run*)

3º En la mayoría de las versiones de XAMPP también se instala PhpMyAdmin, pero si en la elegida no fuese el caso, se debería instalar para facilitar la manipulación de la base de datos introduciendo los dos siguientes comandos en la terminal: *sudo apt-get update* y *sudo apt-get install phpmyadmin php-mbstring php-gettext* y posteriormente en el asistente de instalación se deberá configurar phpmyadmin. Tras haberlo hecho se deben habilitar las extensiones de PHP mcrypt y mbstring con el comando phpenmod y tras reiniciar el servicio de apache (*sudo service apache2 restart*) si se accede a la url http://localhost/phpmyadmin se podrán gestionar las bases de datos desde su interfaz gráfico.

5.2. ANEXO B

INSTALACIÓN DE APACHE

Para preparar el equipo para agregarle nuevo software se actualiza la situación actual con *sudo apt-get update*. A continuación, se instala el servidor con *sudo apt-get install apache2* y con *sudo servide apache2 restart* se reinicia el servicio.

Si todo ha ido bien en los pasos previos, al abrir un navegador y acceder a localhost deberá visualizarse la página que se muestra en la Imagen 21; si es así, la instalación ha sido correcta y a partir de ese momento los ficheros que se incluyan en la ruta */var/www/html/ (*)* relacionados con un *index.html* alojado en la misma ruta serán ejecutados y mostrados en localhost.

Aunque no se realiza por no ser objeto de este proyecto, la ruta por defecto de localhost en Apache se puede configurar en el fichero *000-default.conf* alojado en */etc/apache/*. Aunque siempre que se realicen cambios en la configuración del servidor éste deberá reiniciarse con *sudo service apache2 restart*.

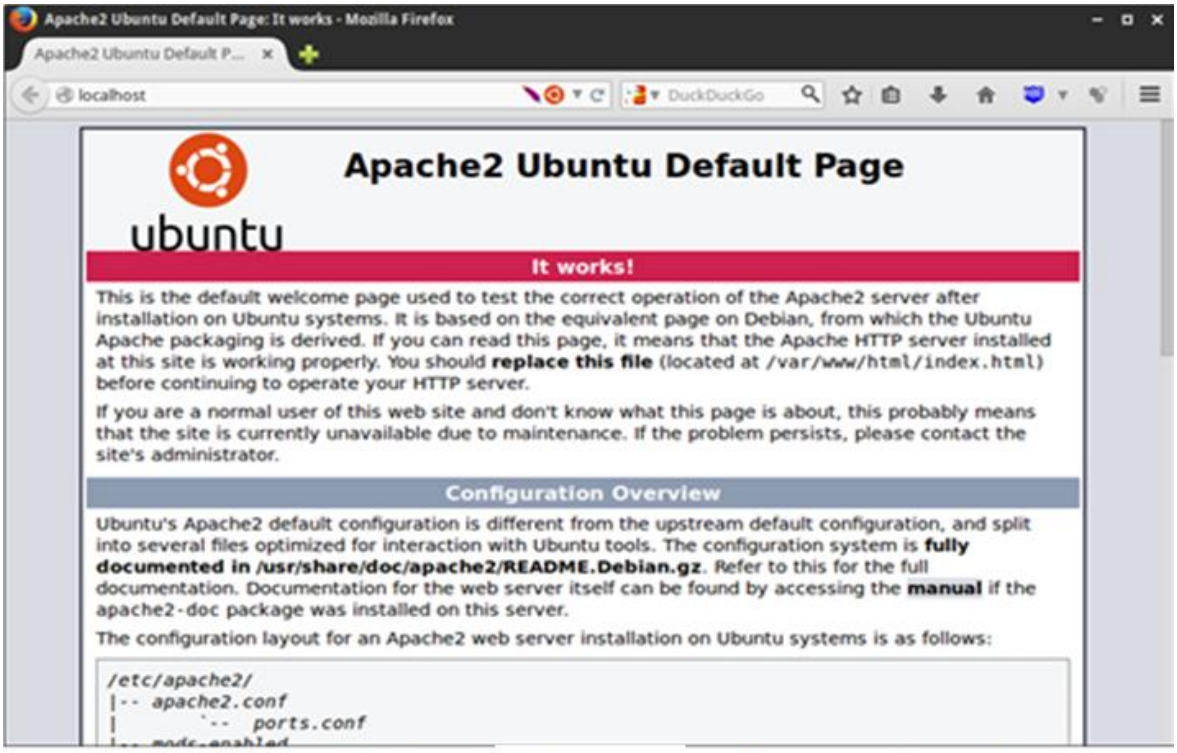

Figura 41 Captura de pantalla que muestra que Apache está bien instalado

*La ruta será similar en otras distribuciones pero todas las indicaciones proporcionadas en esta memoria son para Ubuntu 16.04 LTS.

5.3. ANEXO C

MANUAL DE ADMINISTRACIÓN

El software resultado del presente proyecto tiene dos modos de utilización por lo que este manual se va a dividir en dos secciones: la configuración e instalación de la versión libre de gestión y las de la versión completa.

A) Configuración e instalación sin aplicación de gestión

Las **funcionalidades** que proporciona esta versión son:

- Servidor encargado de aportar comunicación bidireccional con la placa del laboratorio.
- Gestión de colas.
- Configuración de los parámetros de experimento para su envío y ejecución.
- Visualización del movimiento del motor.
- Representación gráfica de los resultados obtenidos tras la ejecución.
- Descarga de los resultados en diversos formatos.
- Carga de resultados almacenados en fichero CSV para su representación gráfica.
- Descarga de los valores de configuración de un experimento.
- Carga de los valores de configuración de un experimento desde un fichero.

Las **dependencias** que se deben satisfacer son:
apt-get, install, gcc, cmake, zlib1g-dev, libssl-dev

Se necesita también tener instalado el servidor web **Apache**, su instalación se especifíca en el Anexo B.

Además se necesita instalar **la librería de websockets** que se adjunta, para realizarlo se debe introducir en una terminal los siguientes comandos:

tar –xvzf libwebsockets-1.4.-chrome43-firefox-36.tar.gz
cd /PATH/LIBWEBSOCKET
mkdir build
cd build
cmake ../
make

(Si da un error la creación del directorio compartido, volver a ejecutar el make o crear en el nivel anterior un directorio compartido)

make install
ln –s /usr/local/lib/libwebsockets.so /usr/lib/libwebsockets.so
ln –s /usr/local/lib/libwebsockets.so.5 /usr/lib/libwebsockets.so.5

Ahora se pasa a configurar **el servidor** para la comunicación serverSECO, para ello:

cd /PATH/server
sudo su

(Insertar la contraseña de superusuario ya que sus permisos son necesarios para determinadas sentencias que posee el código)

make
./serverSECO

Para configurar e instalar la **parte web** se debe copiar el contenido de la carpeta web (directorios css, data, images, js, SOURCE_FILES y tmp y el fichero index.html) en la ruta /var/www/html/ o similar.

Si todo ha ido bien, en este punto estará corriendo el servidor haciendo posible la comunicación con la placa; al acceder a localhost desde un navegador se podrá visualizar la interfaz de usuario donde todas las funcionalidades están disponibles.

En cuanto a los **ficheros** para poder utilizar las funcionalidades cabe comentar que:

- El fichero para representar en la gráfica debe ser de extensión CSV y seguir la estructura en la que en la primera línea se debe iniciar con "Category" y separados por comas los nombres de cada línea de valores correspondiente a una variable a representar que deben entrecomillarse con comillas dobles; en las siguientes líneas se encontrarán en primer lugar el valor del eje de abscisas y los siguientes corresponderán al valor del eje de ordenadas de cada variable para dicho valor de abscisas.

- Por su parte el fichero de configuración debe ser de extensión SECO y su estructura debe ser que en cada línea aparezca el nombre de la variable seguido de dos puntos y un espacio y su valor seguido de una coma salvo en el caso de la última línea en la que dicha coma se omitirá.

Para terminar comentar **posibles puntos a modificar** que puede que se necesiten:

- Variables WEBSITE_JSON_DATA y WEBSITE_TMP_DATA del fichero common.h son las rutas donde se almacenarán los ficheros de donde la gráfica tomará los valores de las variables, deben ser las rutas de la carpeta data y tmp de la parte web que en principio deben estar en /var/www/html/data/ y /var/www/html/tmp/ respectivamente pero según la estructura de la máquina que aloje la aplicación puede variar.

- Variables de la placa BOARD_SOCKET_ADDR y BOARD_SOCKET_PORT del fichero common.h que son la dirección IP y el puerto de la tarjeta física del laboratorio que si llegasen a cambiar se deben modificar para mantener la comunicación.

- Dirección sobre la que se dispara el websocket en el fichero websocket.js en la línea 9 donde se instancia el websocket, el primer parámetro del constructor debe coincidir con la dirección IP contra quien se lanza el websocket y en el puerto 9000.

- Las extensiones que se acepta en el fichero para la gráfica se configura en el fichero tratamientoCSV.js en la condición de la línea 14 y las de la configuración en el fichero tratamientoConfiguración.js en la condición de la línea 13. Y la extensión con la que se descarga el fichero de configuración se declara en el archivo configuracion.js en la línea 142 junto con el nombre que tendrá el fichero descargado.

B) Configuración e instalación completa

Las **funcionalidades** que se añaden en esta versión son:

- Necesidad de autenticación para acceder a la aplicación.
- Perfil de administrador que puede gestionar los usuarios y sus experimentos.
- Posibilidad para el administrador de añadir a la vez un grupo de usuarios desde un archivo.
- Posibilidad para el administrador de borrar a la vez todos aquellos usuarios con rol de usuario.
- Posibilidad para el administrador de asignar un experimento a un usuario.
- Perfil de usuario que puede gestionar sus experimentos en la base de datos.
- Perfil de usuario que puede acceder a todas las funcionalidades de la versión anterior.
- Perfil de usuario que puede cargar la configuración de un experimento desde el gestor a la aplicación para ejecutarlo.

Se necesita también tener instalado el servidor web **Apache,** el gestor de base de datos **MySQL** y un intérprete de **PHP**, se ha decidido para ello instalar el paquete **XAMPP. S**u instalación se puede ver en el Anexo A.

También se deben satisfacer las dependencias del punto anterior e instalar la librería de websockets como en él se indica.

Para instalar la **parte web** y **el servidor** para la comunicación serverSECO se debe copiar la carpeta BASICO íntegra (con la carpeta server y la carpeta web) a la ruta /opt/lampp/htdocs/. Para arrancar el servidor se deben realizar los pasos indicados al respecto en el apartado anterior pero teniéndolo ubicado en esta ruta.

Para contar con la nueva **aplicación de gestión** se debe copiar la carpeta GESTSECO que la contiene a la ruta /opt/lampp/htdocs/.

Por último, para arrancar la **base de datos**, se debe acceder a phpmyadmin a través de localhost/phpmyadmin y en él, a la izquierda en New crear un schema de nombre vero_bd. Tras seleccionarlo, se debe pulsar Import en las pestañas de la zona central y seleccionar el fichero que se adjunta como vero_bd.sql.

Con todo configurado, en XAMPP se inician todos los servicios y si todo ha ido bien, en este punto estará corriendo el servidor y la comunicación con la placa es posible; al acceder a localhost/GESTSECO desde un navegador se podrá visualizar la interfaz de la aplicación de gestión preparada para elegir autenticarse y tras ello acceder a todas las funcionalidades descritas.

En cuanto a los **ficheros** para poder utilizar las funcionalidades cabe comentar que:

- El fichero para añadir a varios usuarios a la vez debe ser de extensión TXT y cada una de sus líneas debe responder a una de las siguientes estructuras (pueden estar combinadas y si no responde a ninguna de las dos, se ignorará):
  - Nombre;correo_electrónico;contraseña;DNI
  - Nombre;correo_electrónico;DNI

  En el segundo caso, se tomará como contraseña para el usuario también su DNI.

Para terminar comentar **posibles puntos a modificar** que puede que se necesiten además de los comentados en el apartado previo:

- La seguridad que se aplica sobre la página principal del envío de los parámetros y la visualización de los resultados a estar logado se declara en el fichero index.php de la carpeta web entre las líneas 39 y 48, bastaría con suprimirlas junto con la llave de cierre del else de las líneas 516-518 para que se pudiera acceder sin tener que pasar por la aplicación de gestión.

- La limitación del número de experimentos por usuario se encuentra especificada en el fichero main.js de GESTSECO en la condición de la línea 983; no se ha tomado como variable global ya que solamente se utiliza en dicha comprobación.

- En cuanto a las rutas de direccionamiento también hay que tener especial precaución si se deciden alojar las carpetas en ubicaciones distintas, en el fichero main.js se redirecciona a la página de inicio de la aplicación básica en la línea 1225 al abrir una nueva pestaña, su ruta debe coincidir con la del archivo index.php de la parte web; además en el index.php de GESTSECO en la línea 97 también se redirecciona a dicho fichero, por lo que también se debe asegurar que la ruta es la correcta.

- La carpeta fichero_usuarios de la carpeta de GESTSECO es en la que se suben los ficheros con los usuarios, es un detalle que puede ser interesante si se quieren recuperar.

5.4. ANEXO D

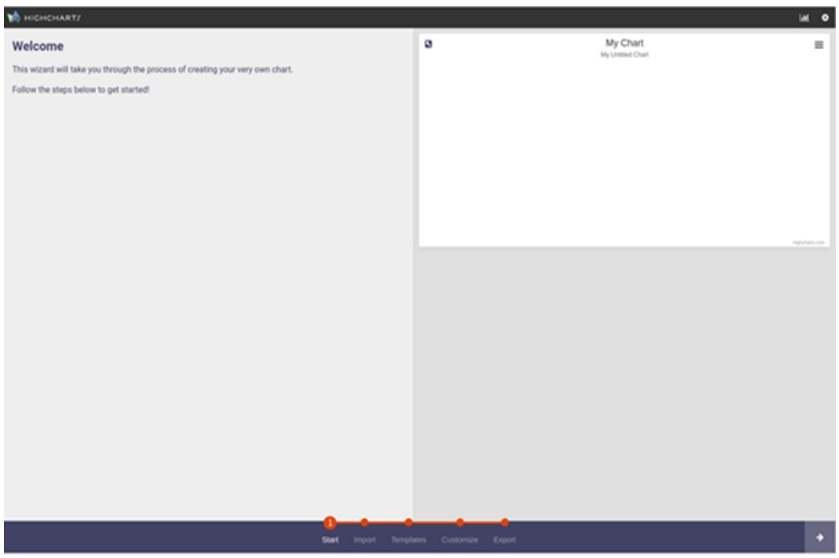

Figura 35 Capturas de pantalla de la aplicación de HighCharts

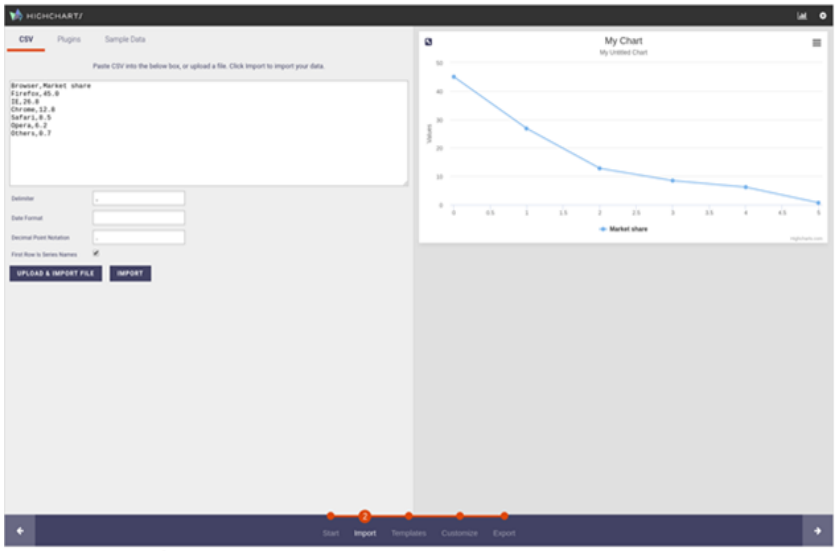

Figura 36 Capturas de pantalla de la aplicación de HighCharts

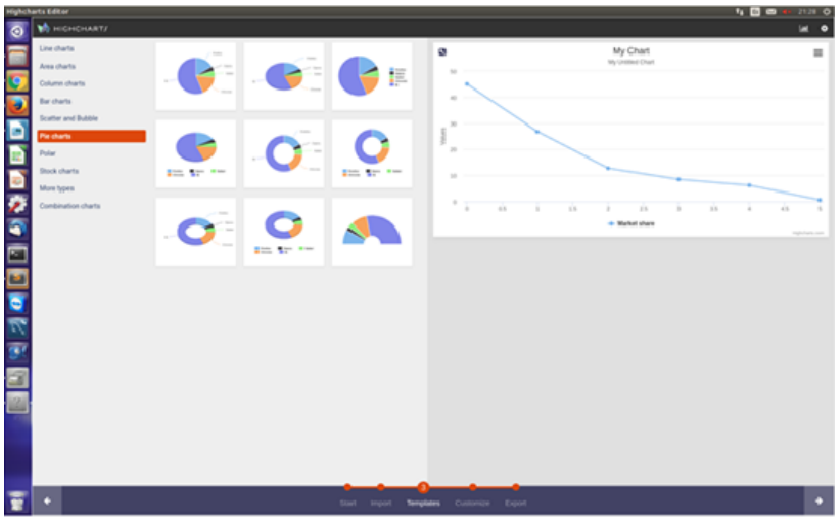

Figura 37 Capturas de pantalla de la aplicación de HighCharts

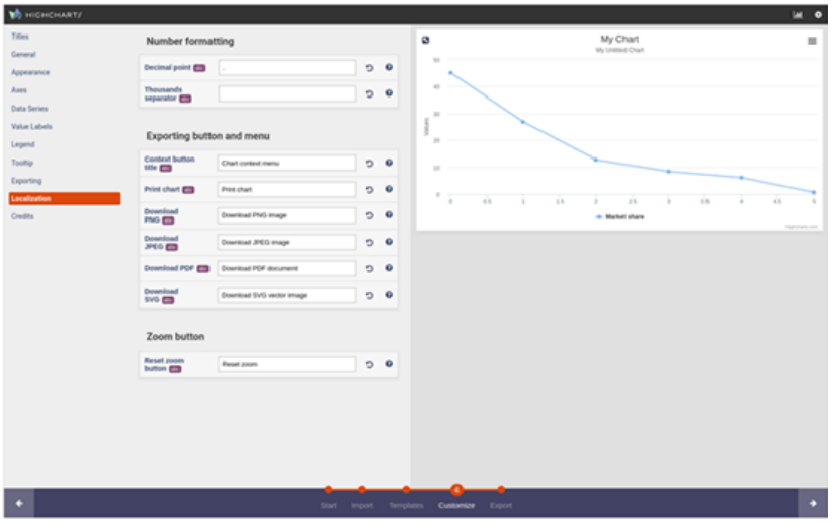

Figura 38 Capturas de pantalla de la aplicación de HighCharts

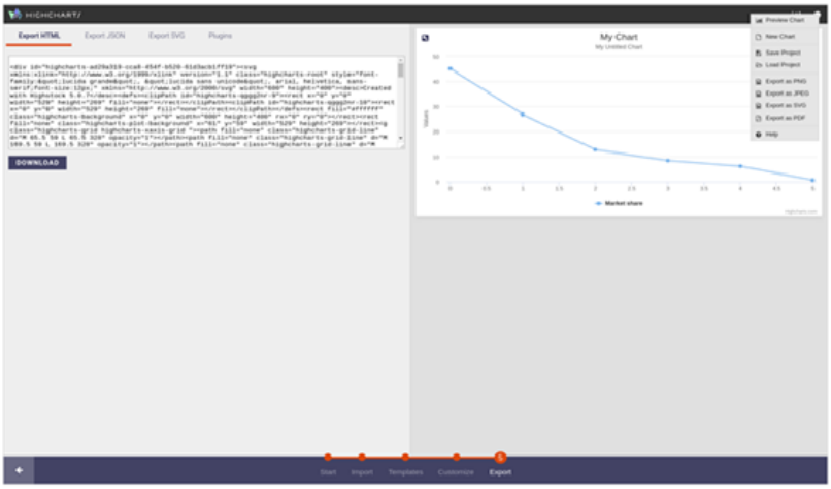

Figura 39 Capturas de pantalla de la aplicación de HighCharts

5.1

6.1

```
//Function to draw a chart when the client load
$(function () {
  Highcharts.chart('container', {
      chart: {
        type: 'line',
        zoomType: 'xy'
      },
      title: {
        text: 'Measurements input from CSV',
      },
      yAxis: {
        plotLines: [{
          value: 0,
          width: 1,
          color: '#808080'
        }]
      },
      tooltip: {
        valueSuffix: ''
      },
      legend: {
        layout: 'vertical',
        align: 'right',
        verticalAlign: 'middle',
        borderWidth: 0
      },
      data: {
        //The data to draw is load here, result
        //The other positions will be interpret
        rows: resultado
      }
  });
});
```

Figura 40
Extracto de código del constructor del gráfico para representar la información del fichero CSV subido.

6. BIBLIOGRAFÍA

[1] Robolabo (2016). *Documentación TeleLabo.* Recuperado en 2016 de http://www.robolabo.etsit.upm.es/asignaturas/seco/apuntes/telelabo/introLabos.pdf

[2] Developer Mozilla (2016). *Documentación FileReader.* Recuperado en 2016 de https://developer.mozilla.org/es/docs/Web/API/FileReader

[3] Developer Mozilla (2016). *Documentación File.* Recuperado en 2016 de https://developer.mozilla.org/es/docs/Web/API/File

[4] Developer Mozilla (2016). *Blob.* Recuperado en 2016 de https://developer.mozilla.org/es/docs/Web/API/Blob

[5] W3 (2016). *Documentación sobre elemento a.* Recuperado en 2016 de https://www.w3schools.com/tags/att_a_download.asp

[6] W3 (2016). *Documentación sobre la carga de ficheros.* Recuperado en 2016 de https://www.w3schools.com/jsref/dom_obj_fileupload.asp

[7] JavaScripture (2016). *Documentación FileReader.* Recuperado en 2016 de http://www.javascripture.com/FileReader

[8] JavaScripture (2016). *Documentación Blob.* Recuperado en 2016 de http://www.javascripture.com/Blob

[9] HighCharts (2016). *Documentación HighCharts.* Recuperado en 2016 de https://www.highcharts.com/

[10] Wikia (2017). *Documentación AOR.* Recuperado en 2017 de http://es.dairelysprogramacion.wikia.com/wiki/Arquitectura_Orientada_a_Recursos_(AOR)

[11] Wikipedia (2017). *Documentación REST.* Recuperado en 2017 de https://es.wikipedia.org/wiki/Transferencia_de_Estado_Representacional

[12] Desarrollo Web (2017). *Documentación REST.* Recuperado en 2017 de https://desarrolloweb.com/articulos/que-es-rest-caracteristicas-sistemas.html

[13] SG Buzz (2017). *Documentación Arquitectura software.* Recuperado en 2017 de https://sg.com.mx/revista/27/arquitectura-software#.WTMz6dryi03

[14] GitBooks (2017). *Documentación REST.* Recuperado en 2017 de https://juanda.gitbooks.io/webapps/content/api/arquitectura-api-rest.html

[15] Slideshare –Landeta- (2011). *Documentación Estilos arquitectónicos.* Recuperado en 2017 de https://es.slideshare.net/landeta_p/2-2-estilos-arquitectonicos

[16] Slideshare –Mstabare- (2011). *Documentación Arquitecturas software.* Recuperado en 2017 de https://es.slideshare.net/mstabare/arquitecturas-de-software-parte-2

[17] Slideshare –Jose-Rob- (2009). *Documentación Arquitectura software.* Recuperado en 2017 de https://es.slideshare.net/jose_rob/diseo-de-la-arquitectura-del-software

[18] Facultad de ingeniería de la República de Uruguay (2017). *Documentación Arquitectura software.* Recuperado en 2017 de https://www.fing.edu.uy/tecnoinf/mvd/cursos/ingsoft/material/teorico/is05-ArquitecturaDeSoftware.pdf

[19] Slideshare –Melidevelopers- (2016). *Documentación REST.* Recuperado en 2017 de https://es.slideshare.net/melidevelopers/arquitectura-api-rest

[20] Ticbeat –Marcos Merino- (2014). *Documentación API.* Recuperado en 2017 de http://www.ticbeat.com/tecnologias/que-es-una-api-para-que-sirve/

[21] BBVA (2016). *Documentación REST.* Recuperado en 2017 de https://bbvaopen4u.com/es/actualidad/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos

[22] Youtube –Fernando Restrepo- (2015). *Documentación Estilos en arquitectura de software.* Recuperado en 2017 de https://www.youtube.com/watch?v=CVJeakuaAks

[23] Youtube –Programación y más- (2015). *Documentación API y Webservice.* Recuperado en 2017 de https://www.youtube.com/watch?v=SXZ0kDZbOas

[24] Tiobe (2017). *Documentación Índice Tiobe.* Recuperado en 2017 de https://www.tiobe.com/tiobe-index/

[25] Node JS (2017). *Documentación NodeJs.* Recuperado en 2017 de https://nodejs.org/es/about

[26] W3 (2017). *Documentación PHP.* Recuperado en 2017 de https://www.w3schools.com/php/

[27] Desarrollo Web (2017). *Documentación PHP.* Recuperado en 2017 de https://www.desarrolloweb.com/php/

[28] PHP (2017). *Documentación PHP.* Recuperado en 2017 de http://php.net/manual/es/intro-whatis.php

[29] IBM -Michael Abernethy- (2011). *Documentación NodeJS.* Recuperado en 2017 de https://www.ibm.com/developerworks/ssa/opensource/library/os-nodejs/

[30] Wiki –Hernán Darío Fernandez- (2004). *Documentación Java.* Recuperado en 2017 de daitesrc.wikispaces.com/Frameworks+para+Java

[31] Arquitectura Java -Cecilio Álvarez Caules- (2015). *Documentación Java.* Recuperado en 2017 de http://www.arquitecturajava.com/java-ee-7-vs-spring-framework/

[32] GejoTres (2015). *Documentación Java.* Recuperado en 2017 de http://www.gajotres.net/best-available-java-restful-micro-frameworks/

[33] Cacharreros de la Web –Sebastian Córdoba- (2017). *Documentación Lenguajes para programar en 2017.* Recuperado en 2017 de https://cacharrerosdelaweb.com/2017/01/siete-lenguaje-programacion-aprender-2017.html

[34] UmainClass (2017). *Documentación Lenguajes para programar en 2017.* Recuperado en 2017 de http://www.umainclass.com/2017/02/los-9-lenguajes-de-programacion-mas-demandados-para-este-2017.html

[35] Adictos al trabajo –Ruben Gómez López- (2014). *Documentación PHP y Java.* Recuperado en 2017 de https://www.adictosaltrabajo.com/tutoriales/php-vs-java/

[36] Django (2017). *Documentación Django.*Recuperado en 2017 de www.django-rest-framework.org

[37] Slim (2017). *Documentación Slim.* Recuperado en 2017 de https://www.slimframework.com

[38] Symfony (2017). *Documentación Symfony.* Recuperado en 2017 de https://symfony.com

[39] Laravel (2017). *Documentación Laravel.* Recuperado en 2017 de https://symfony.com

[40] Wave (2017). *Documentación Wave.* Recuperado en 2017 de https://www.waveframework.com

[41] PHP (2017). *Documentación Frameworks PHP.* Recuperado en 2017 de phpflow.com/php/restful-api-frameworks-for-php