UNIVERSIDAD DE EXTREMADURA

ESCUELA POLITÉCNICA

GRADO EN INGENIERÍA INFORMÁTICA EN INGENIERÍA DE COMPUTADORES

TRABAJO FIN DE GRADO

Título: Desarrollo de un sistema de adquisición y tratamiento de datos para modelos hidraúlicos

Nombre: Alberto Iglesias Gonzalo

Septiembre, 2016

UNIVERSIDAD DE EXTREMADURA

ESCUELA POLITÉCNICA

GRADO EN INGENIERÍA INFORMÁTICA EN INGENIERÍA DE COMPUTADORES (GIIIC)

TRABAJO FIN DE GRADO

Título: Desarrollo de un sistema de adquisición y tratamiento de datos para modelos hidraúlicos

Autor: Alberto Iglesias Gonzalo.

Tutor: Pablo García Rodríguez.

Cotutores: Pablo Durán Barroso y José Ceballos.

Presidente: Antonio Manuel Silva Luengo.

Secretario: Jesús Torrecilla Pinero.

Vocal: Pilar Bachiller Burgos.

Abstract

En los últimos años la tendencia de todos los sectores industriales se han basado en la monitorización de sus sistemas, realizando así un control en tiempo real de todos los sensores del entorno. Actualmente, con la evolución de la tecnología sabemos que esos datos puedes ser tratados y almacenados para poder realizar un ecosistema más sostenible y eficiente, este proyecto da un paso más haya y unifica el control y almacenamiento de datos en un único aplicativo. Este proyecto será utilizado en el laboratorio de Ingeniería civil de hidrología por todos los alumnos, siendo útil para la elaboración de sus prácticas. Una vez unificado todo el proceso de muestreo y prueba, la aplicación podrá ser utilizada dentro de la Escuela Politécnica para monitorizar todo su sistema de abastecimiento de agua, pudiendo realizar un consumo de agua más eficiente y ampliando así el concepto de SmartPolitech o entornos inteligentes. Todo esto, unido a la utilización de software libre, lograría ser un entorno único, ya que conseguiríamos tener un sistema eficiente, sostenible y libre.

A mi familia y en especial a mi abuela. Gracias.

Contents

List of Tables

List of Figures

1 INTRODUCCIÓN

Este proyecto surge de la necesidad de construir y diseñar un sistema de recogida, almacenamiento y visualización de datos procedentes de una red de dispositivos que instalaremos en el laboratorio de hidráulica de la Escuela Politécnica. Dichos sensores se encargaran de recoger datos obtenidos del entorno, en nuestro caso, sensores de fluidos tales como: nivel, presión y caudal. El sistema de recogida de información debe ser capaz de adquirir, tratar, filtrar y almacenar dichos datos para su posterior análisis.

Este proyecto se enmarca dentro de una necesidad concreta, la actualización de los sistemas de medida del laboratorio. Es donde este proyecto lograría un paso en el sistema de recogida ya que, actualmente, la mediación se realiza anotando todas las medidas de forma manual. Ante tal necesidad, nace este proyecto. La idea principal del sistema consiste en poder ver en tiempo real la variación de los diferentes sensores, sin que el usuario se mueva del ordenador, ahorrando así infinidad de tiempo en la recogida de datos. Tales datos serán también almacenados para un posterior análisis.

El sistema de fluidos a monitorizar es utilizado frecuentemente por los estudiantes de ingeniería civil en todas sus especialidades. Con el nuevo sistema, los estudiantes podrán recoger muestras de manera más dinámica y rápida; solamente dándose de alta en la plataforma ya podrán iniciar sus pools de medidas. Para conseguir todos los requerimientos de la plataforma objetivo, será necesario un sistema completo de recogida robusto y estable, capaz de ser tolerante a fallos. De la misma forma, si el sistema detecta un fallo será capaz de lazar un evento, almacenarlo en el sistema y mostrando al usuario información de hora y descripción del error, para su posible recuperación y reparación.

La toma de medidas deberá ser exacta y precisa, ya que serán utilizadas para fines estadísticos y experimentales. Por ello, la calidad en la recogida de muestras deberá ser precisa y minuciosa. Todo ello, hará en un futuro un sistema predecible, pudiendo de manera autónoma ser un sistema de control de canales, tuberías y todos los sistemas

de fluidos actuales.

1.1 Internet de las cosas

Definimos Internet de las cosas[1], es un concepto propuesto por Kevin Ashton en 1999, en el cual, todos los objetos cotidianos estarían conectados a Internet. Según Cisco, IoT[2] es "el punto en el tiempo en el que se conectarían a Internet más cosas u objetos que personas". Podríamos saber el estado de cualquier objeto o cosa conectada, ya sea sensor o producto. Llevaría con ello un cambio en nuestro estilo de vida cotidiana, seguridad, compra de productos y también, por qué no decirlo, una nueva forma de control y monitorización de consumos de agua, electricidad, etc. La previsión de crecimiento de IoT es exponencial y estima que aproximadamente habrá más de 26 mil millones de dispositivos conectados.

Con esta tecnología, se pretende monitorizar el sistema de fluidos para obtener datos en tiempo real y poder hacer análisis predictivo. Se van a instalar sensores por todo el entorno del laboratorio, para poder así tomar todas las medidas posibles. Todo el conjunto de sensores, junto con el hardware que recoge y procesa todas las medidas, se denomina mota. Una mota se encarga de recoger los datos procedentes de los sensores, que en nuestro caso será un Arduino[3], y enviarlos a un servidor central, en nuestro caso utilizaremos la plataforma Arduino . Arduino se inició en el año 2006 como un proyecto para estudiantes en el Instituto IVREA, en Ivrea (Italia). Desarrollado como hardware libre, con costes totalmente irrisorios y una curva de aprendizaje bastante elevada. Las múltiples posibilidades de la plataforma la han convertido es un referente a nivel mundial dentro del IoT , convirtiéndola en la principal plataforma para desarrollo de prototipos del mercado.

La Escuela Politécnica, ha considerado que IoT será una de las principales tecnologías del futuro y por ello, la apuesta por este tipo de proyectos es siempre constante. Dentro de la escuela se han establecido espacios abiertos para el desarrollo

[1] Internet of things, https://www.wikipedia.com/wiki/Internet_of_things
[2] Internet of Things
[3] https://www.arduino.cc

de ideas de cualquier tipo y forma, entre ellas IoT. Dentro de Smart Open Lab[4] se desarrollan todo tipo de proyectos, incluido éste.

1.2 Software libre

La filosofía del software libre nació en 1985 encabezada por Richard Stallman[5] y la consecuente fundación Free Software Fundation[6].

El software libre fue la gran revolución de la informática, permitiendo a este ser copiado, modificado, estudiado y utilizado libremente bajo una licencia de tipo GPL (una entre muchas). Suele estar disponible de manera gratuita o bajo unos costes de distribución mínimos.

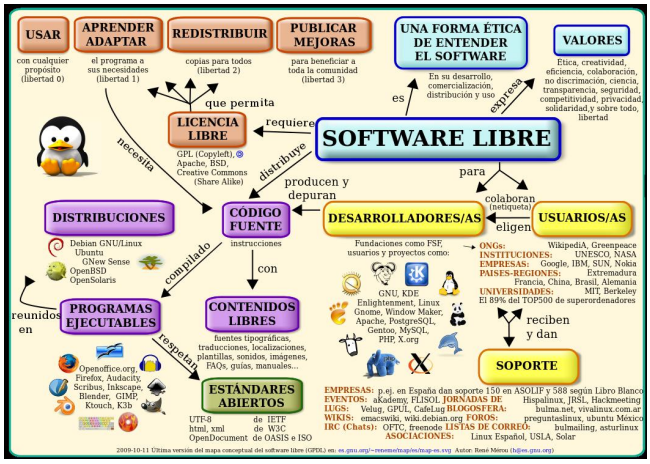


Figure 1: Mapa conceptual de Software libre.

[7] Los pilares fundamentales del software libre se basan en:

- Libre de uso bajo cualquier propósito.
- Libertad de estudiar cómo funciona el programa y modificarlo.
- Libertad de distribuir copias del programa.

[4] http://solepcc.unex.es/joomla/
[5] https:es.wikipedia.org/wiki/Richard_Stallman
[6] https:es.wikipedia.org/wiki/Free_Software_Foundation
[7] https://es.wikipedia.org/wiki/Software_libre#/media/File:Mapa_conceptual_del_software_libre.svg

- Libertad de mejorar el programa y hacer públicas esas mejoras a los demás.

Todo el software usado para la realización de este proyecto está basado en software libre, descartando toda opción privada, tanto para su desarrollo, como la implantación del mismo.

Arduino es un proyecto de hardware libre basado en una plataforma completa de hardware y software, compuesta por una placa que integra un microcontrolador y un entorno de desarrollo completamente libre, bajo uso el este tipo de licencias.

1.3 Almacenamiento NoSQL

El almacenamiento NoSQL[8] ha sido una de las grandes revoluciones de los últimos tiempos. La potencia de una base de datos de tipo NoSQL sobre grandes volúmenes de datos es increíble, pudiendo realizar consultas en milisegundos. Estos sistemas capaces de almacenar grandes volúmenes de datos crea un nuevo concepto llamado Big Data[9]. Estos datos componen un gran desafío para el mundo actual, ya que están dispuestos para ser analizados, comprendidos y entender dichos datos más allá de las herramientas tradicionales. Pero bien, ¿qué es Big Data y por qué se ha vuelto tan importante?

La tendencia del avance en la tecnología NoSQL ha abierto un nuevo enfoque sobre el valor de los datos y la toma de decisiones sobre los mismos, la cual es utilizada sobre grandes conjuntos de datos (estructurado, no estructurado o semi-estructurado). Esta toma de decisiones sobre grandes bases de datos relacionales demoraría demasiado tiempo y sería demasiado costoso. De esta manera, Big data aplica el mismo concepto sobre datos que no pueden ser procesados de manera tradicional. El porqué de todos estos datos tienen su origen en el uso exponencial de Internet en la primera década del siglo XXI, ya que el ser humano está constantemente generando información y cada vez más. Todos esos datos, son almacenados por la industria. Las compañías guardan datos de sus clientes, proveedores y todas las operaciones realizadas.

[8] https://es.wikipedia.org/wiki/NoSQL
[9] https://www.ibm.com/developerworks/ssa/local/im/que-es-big-data/

Dentro de la Escuela Politécnica, se está desarrollando un proyecto llamado SmartPolitech. Se enfrenta al gran desafío de analizar todos los datos procedentes de sensores, cámaras, etc. Se puede llegar a realizar análisis predictivo, analizando cada minuto todo tipo de parámetros: temperatura, consumo de electricidad y agua, pudiendo así tener un edificio sostenible, seguro y totalmente autónomo. Y no solamente eso, sino que será capaz de analizar cuanta gente hay dentro de las aulas, ocupación de aparcamientos, número de comidas vendidas y quien suele comer regularmente. No solamente se trata de almacenar todos esos datos, sino de saber tratar y darles forma. Según un estudio más del 90% deja de tener valor a partir del primer segundo después de su almacenamiento, por ello es necesario un sistema rápido y potente capaz de analizar y procesar ese volumen de datos.

1.4 Definición de objetivos

La definición de objetivos fue establecido tanto por mi director Pablo García, como por los dos subdirectores Pablo Durán Barroso y José Ceballos. El objetivo principal era claro, sensorizar y monitorizar el laboratorio de prácticas de ingeniería civil. Pero este objetivo principal se divide en dos bloques claramente diferenciados: requisitos no funcionales y funcionales.

Requisitos no funcionales

- Seguridad: deberá ser seguro sin peligro a caidas y a prueba de errores. Restringiendo el acceso a usuario no autorizados.
- Rendimiento: deberá tener un rendimiento óptimo, libre de sobre carga.
- Estabilidad: deberá ser estable, sin caídas del sistema ni cortes de red.
- Accesibilidad: el sistema deberá ser desarrollado de manera que todos los usuarios, independientemente de su discapacidad, puedan hacer uso de él.
- Escalabilidad: deberá ser un sistema escalable, libre de costes de rendimiento, ni sobre costes.

- Concurrencia: control de concurrencia.
- Interfaz: la interfaz deberá ser responsive, es decir, puede ser usada desde cualquier terminal.

Requisitos funcionales

- Implementación arquitectónica de mota para recogida de datos procedentes de sensores.
- Envío y recogida de datos a través de HTTP .
- Sistema de actualización de sketch vía web, sin que el usuario haga ninguna iteración con el dispositivo.
- Control de usuario:

  - Alta: se podrá realizar alta de usuarios en el sistema.

  - Modificación: modificación de todos los datos del usuario y permisos.

  - Baja: Se podrá realizar la baja definitiva del usuario en el sistema

  - Listado: Todos usuarios se mostrarán mediante una tabla, indicando los permisos y modificaciones sobre los mismo.
- Control y verificación de datos de entrada.
- Sistema de control de errores y modificaciones:

  - Control de usuarios modificados, se realizará mediante almacenamiento de evento.

  - Control de datos de entrada, se realizará mediante almacenamiento de evento.
- Sistema de almacenamiento en base de datos tipo NoSQL.
- Visualización de datos en gráficas.

- Logging de usuarios y control de acceso.

Todos estos objetivos serán detallados más adelante por casos de uso, con su requisito funcional que debe cumplir.

2 ESTADO DEL ARTE

Este capítulo está destinado a analizar la situación actual de la tecnología referente al trabajo propuesto, el objetivo principal del proyecto es la visualización de sensores de fluidos instalados en un entorno controlado. Para ello, existen dos partes diferenciadas, el uso actual de los sensores en entorno reales y sistemas de medición y, monitorización en entornos de circuitos de agua.

2.1 Sensores en entornos reales

Este tipo de sensores son de uso común en cualquier sistema de depuración de agua y tratamiento de aguas fecales, abastecimiento, etc. Como se ha indicado en el apartado anterior, se hará uso de sensores de caudal, presión y nivel enviando datos a nuestra plataforma web. Del mismo modo, su uso en entornos reales es similar, enviando datos en tiempo real al sistema de recogida. Estos sensores se suelen poner al principio y final de todos los sistemas mencionados anteriormente.

2.2 Sistema de medición

El sistema de monitorización por excelencia es SCADA[10][11], es un sistema que permite controlar y monitorizar procesos industriales desde lugares remotos, facilita la sensorización en tiempo real con dispositivos de campo, permite la recepción de toda la información que se genera en el entorno permitiendo su gestión e intervención.

Existen dos tipos de sistemas principales: de lazo cerrado y abierto. La principal diferencia entre ambos es la retroalimentación. Los sistemas de lazo cerrado funcionan de tal manera que, la salida vuelva al principio analizando la diferencia entre el valor obtenido y un valor de referencia, el actual. En los sistemas de lazo abierto en cambio, la salida se va ajustando hasta que el error sea 0.

[10] https://es.wikipedia.org/wiki/SCADA
[11] (Supervisory Control And Data Acquisition)

Cualquier sistema que necesite medir temperatura, presión, caudal, fuerza, entre otras, son normalmente de lazo cerrado. Los sistemas de lazo abierto no se comparan con una constante fija, cada ajuste determina una posición fija de un elemento de control, por ejemplo un actuador.

El siguiente esquema muestra un sistema típico SCADA, es un ejemplo básico de un sistema. Las áreas de acción de estos sistemas son[12]:

- Monitorizar procesos químicos.
- Gestión de producción.
- Mantenimiento.
- Control de calidad.
- Administración.
- Tratamientos de históricos.

[12]http://www.uco.es/grupos/eatco/automatica/ihm/descargar/scada.pdf

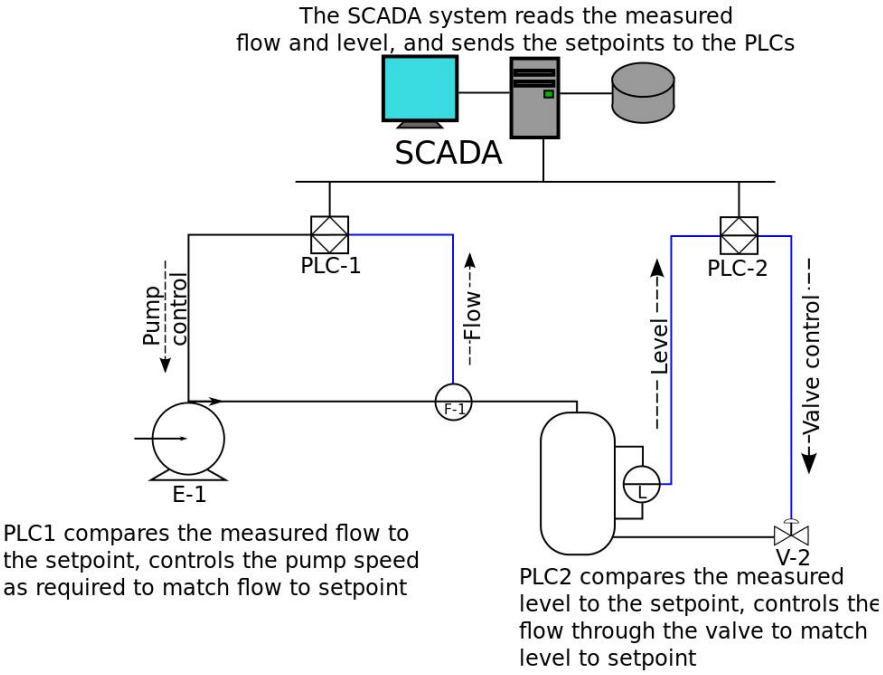


Figure 2: Sistema SCADA.

[13] Las funciones principales de estos sistemas son[14]:

- Obtención de datos, recoger, procesar y almacenar la información recibida.
- Supervisión, posibilita la observación de los datos obtenidos mediante un monitor en una interfaz intuitiva.
- Control, es la funcionalidad principal, haciendo posible la manipulación de todo tipo de actuadores conectados a él, ya sea directamente sobre el proceso que lo controla o sobre el sensor de la salida.

El paquete SCADA ofrece las siguientes prestaciones:

- Posibilidad de crear paneles de una alarma que, exigen la presencia del operador para reconocer una parada o situación de alarma, con registro de incidencias.

[13] https://upload.wikimedia.org/wikipedia/commons/0/0c/SCADA_schematic_overview-s.svg
[14] http://www.uco.es/grupos/eatco/automatica/ihm/descargar/scada.pdf

- Generación de históricos de señal de planta que, pueden ser volcados para su proceso sobre una hoja de cálculo.
- Ejecución de programas, que modifican la ley de control, o incluso el programa total sobre el autómata, bajo ciertas condiciones.
- Posibilidad de programación numérica permitiendo realizar cálculos aritméticos de elevada resolución sobre la CPU del ordenador, y no sobre la del autómata, menos especializado, etc.

Actualmente, SCADA está montado en el 100% de los sistemas de abastecimiento y tratamiento de aguas de Extremadura, haciendo este tipo de sistemas únicos y con un coste muy elevado de licencias.

3 TECNOLOGÍAS EMPLEADAS

El abanico de tecnologías a utilizar es bastante amplio, pasando por casi todos los lenguajes de programación. En primera instancia, cuando se presentó la propuesta de proyecto mi elección fue clara, utilizar una base de datos NoSQL tipo mongoDB[15], uso de Bpring MVC con webservice, AngularJS[16] Bootstrap[17] para el frontend. Utilicé un servidor tomcat para el despliegue del servidor, es el principal servidor de software libre.

Spring[18] basa su funcionamiento en inyección de dependencias, debido a esto me surgieron bastantes problemas al añadir librerías como Mongo o de webservice. Intenté durante horas dejar una beta estable pero fue imposible debido a errores de inyecciones. Me hubiese gustado realizar el proyecto con esta tecnología, ya que es la más demandada del sector informático.

Realicé todo tipo de pruebas y ejemplos que encontré por la web pero, fui incapaz de desarrollar una versión estable. Debido a las dos partes claramente diferenciadas, las tecnologías empleadas se explicarán por separado para una mayor comprensión y explicación. Por una parte la aplicación web y por otra el sistema de recogida de datos o mota.

3.1 Aplicación web

El desarrollo de la aplicación web es la parte más compleja, el desarrollo de la arquitectura de 3 capas hace que el nivel de abstracción y capacidad de programación sea amplia y con un alto nivel de recursos. De todas las tecnologías a utilizar, me he decantado por el uso funcional de todas y su rapidez de aprendizaje, nunca había utilizado ninguna de ellas anteriormente y es por ello el reto que esto propone.

[15] https://www.mongodb.org
[16] https://angular.io
[17] http://getbootstrap.com
[18] https://spring.io

3.1.1 Persistencia de datos

Para la persistencia de los datos obtenidos, se ha elegido una base de datos tipo NoSQL. Dentro del amplio abanico de SGBD se optó por una bases de datos MongoDB. Una bases de datos NoSQL tiene las siguientes características:

- Consistencia eventual: No se implementan mecanismos rígidos de consistencia como en las bases de datos relacionales, donde se confirma el cambio de los datos actualizados a los nodos replicados.
- Escalabilidad horizontal: puedes aumentar el rendimiento del sistema simplemente aumentando el número de nodos, sin necesidad de indicar al sistema ninguna otra operación, solamente indicando el número de nodos disponibles.
- No generan cuellos de botella: el problema de las bases de datos SQL es que se tiene que transcribir las secuencias, por lo tanto, necesita un nivel de ejecución mayor que una de tipo NoSql.
- Estructura dinámica: los datos no tienen una definición de atributos fija, es decir, cada registro puede tener una información diferente cada vez, pudiendo así almacenar sólo atributos que interesen para ella y facilitando el polimorfismo de los mismos. Aumenta el rendimiento ya que no existen los JOIN para obtener los datos, pues éstos están directamente en el mismo documento.

Debido a estas características hacen que esa fundamental en el sistema objetivo, ya que el nivel de datos para almacenar es bastante amplio, sin cuello de botella y con posibilidad de escalabilidad fácilmente. La base de datos MongoDB está basada en JSON, en documentos. Almacena información con clave única, pero con la diferencia que el valor de un fichero pueda ser entendido.

Seguidamente, paso a explicar el porqué de todas las ventajas de estas bases de datos en relación con una SQL.

Desde hace décadas las bases de datos relacionales son las más utilizadas del mundo, básicamente todas las plataformas, web e industria con base en Internet usaba una base de datos SQL. Estas bases de datos aportan una solidez y consistencia para el almacenamiento y acceso a datos. Con el crecimiento de la web, el número de usuarios fue exponencial convirtiendo el estilo de vida social y comercial a nivel mundial, las redes sociales cumplen un papel fundamental es el sector NOSQL. Es donde este tipo de base de datos tienen su mayor potencia, debido al nivel de datos proporcionados por los usuarios.

En las redes sociales es donde una característica principal ofrece mayor potencia, la escalabilidad. Con un número de usuarios elevado, las bases de datos relacionales solamente se pueden escalar verticalmente, introduciendo más recursos al sistema. Incrementando los recursos de la máquina e incrementan también los costes. Pero no solamente eso, la relación entre rendimiento/coste no se mantiene sino que empeora, volviéndose totalmente ineficaz.

Una bases de datos relacional cumple una serie de características que hacen totalmente incompatible con el sistema objetivo, se denomina ACID[19].

- Atomicidad: todas las operaciones se tienen que completar de manera atómica, es decir, si falla una deben deshacerse todas las anteriores.

- Consistencia: Garantiza que todo lo que empieza debe acabar, esto es, integridad. Solamente se cumplen aquellas operaciones que cumplen con las reglas de las bases de datos.

- Aislamiento: todas las operaciones se ejecutan de manera aislada, garantizando así que dos operaciones sobre el mismo dato se ejecutan de manera independiente, sin errores.

- Durabilidad: Los datos deben persistir en el tiempo, una vez realizada la operación los datos serán persistidos aunque el sistema falle o caiga.

[19]https://es.wikipedia.org/wiki/ACID

A continuación mostramos ejemplos de usos de base de datos relacionales:

- Desarrollo web: uso de plataformas online, blog,etc. El uso de jerarquías de datos de usuarios y lógica no sea de gran dificultad.
- Negocios: Poder estructurar la información de sus clientes, facturas, consumo, etc.
- Empresas: proporcionando software a medida con un modelo de datos rígidos, con un sistema consistente de datos.

Con todo esto, queda totalmente descartado un sistema relacional. Las características de un sistema NoSQL ejerce mayor potencia en nuestro sistema, ofreciendo una disponibilidad de datos mucho mayor. Su rapidez en la ejecución de operaciones ofrecen una mayor potencia, pudiendo escalar horizontalmente el sistema, es decir, incluir más máquina pudiendo así duplicar la velocidad.

3.1.2 Lógica de negocio

Después de deliberar todas las tecnologías utilizadas para el desarrollo web, la preferencia fue el uso de Python. Fue una decisión difícil, para mí es una tecnología nueva, pero afronté el reto. Python es un lenguaje de desarrollo creado en 1991, por Guido van Rossum[20]. Al ser un lenguaje de interpretación cuyas síntesis favorece al código legible la curva de aprendizaje es bastante amplio. Es un lenguaje multiparadigma que soporta programación orientada a objetos, imperativa y funcional. Algunas de las siguientes características[21] de Python son:

- Simple: es un lenguaje simple y minimalista. El pseudo-código natural de Python es una de sus grandes fortalezas.
- Fácil de aprender: es el lenguaje recomendado para iniciarse en el mundo de la programación.

[20] https://es.wikipedia.org/wiki/Guido_van_Rossum
[21] http://dev.laptop.org/ẽdsiper/byteofpython_spanish/ch01s02.html

- Lenguaje de alto nivel: nunca debes preocuparte del manejo de memoria.
- Portable: al ser un lenguaje de open source, ha sido portado para diversas plataformas, funcionará igual en distintas plataformas sin requerir cambio alguno.
- Interpretado: no hay aparición de compiladores, sólo ejecuta el programa desde el código fuente. Internamente, Python traduce el código fuente en una forma llamada bytecodes y, seguidamente, en lenguaje que tu computadora pueda entender y lo ejecuta.
- Orientado a objetos.
- Ampliable: puedes escribir una pieza de código en C y después combinarla con tu programa Python.
- Incrustado.
- Librerías extendidas: el pool de librerías de Python es muy amplio, existen librerías desde criptografía, imágenes, WEB, etc.

Todas estas características hacen que el uso de este lenguaje para el proyecto objetivo sea excepcional. Dentro de Python, existe un framework llamado Django. Este framework fue creado por el desarrollo de aplicaciones web con unos conceptos claros:

- Desarrollo rápido.
- Rápidamente escalable.
- Seguro.

Estas características hacen que la elección por parte de la comunidad sea clara y directa. Respeta el patrón de diseño MVC (modelo-vista-controlador), el patrón por excelencia de la arquitectura web.

Django hace que el mundo de desarrollo web sea bastante más sencillo, al igual que el desarrollo de aplicaciones web o aplicación de servicios. El framework incluye: un servidor propio, un sistema de generación de código y plantillas, sistema de log´s, bases de datos propia (SQLlite), conexiones con diferentes bases de datos, control de usuarios, etc.

Todo esto hace que su uso sea actualmente el que mayor crecimiento tiene sobre la comunidad, facilita la labor tan tediosa para el desarrollador como son la seguridad, conexiones a bases de datos, etc.

3.1.3 Presentación de datos

El desarrollo de la capa de presentación tiene menos posibilidades que la capa de negocio, ya que el leguaje usado en la mayoría de las aplicaciones web es HTML, CSS y Javascript.

Se hará una especificación un poco más detallada de estos lenguajes, se detallan a continuación por sectores:

Maquetación de información HTML5

HTML es la quinta versión del lenguaje básico de marcado de la WWW y trae grandes diferencias con respecto a su versión anterior:

Etiqueta	Atributos de la etiqueta	Comentarios
<!-- -->	Estándar o ninguno	
<!DOCTYPE>	Estándar o ninguno	
<a>	href I target I rel I hreflang I media I type	Atributo Añadido: *media* Atributo cambiado: Target
<abbr>	title	
~~<acronym>~~		Etiqueta Eliminada
<address>	Estándar o ninguno	
~~<applet>~~		Etiqueta eliminada
<area>	Estándar o ningunos	
<header>	Atributos globales	Nueva etiqueta
<hgroup>		hgroup se añadió a la especificación HTML5, pero ahora está obsoleta.[4] Usar <header>
<hr>	Ninguno	Etiqueta cambiada
<html>	Estándar o ninguno	
<i>	Ninguno	Etiqueta cambiada
<iframe>	Estándar o ninguno	
<img>	Estándar o ninguno	
<input>	accept I alt I auto-complete I autofocus I cheked I disabled I form I formaction I formenctype I formmethod I formnovalidate I formtarget I height I list I max I maxlength I min I multiple I name I pattern I placeholder I readonly I required I size I src I step I type I value I width	Etiqueta cambiada: Añadidos 13 elementos a type
<ins>	Estándar o ninguno	
~~<isindex>~~		Etiqueta eliminada
<kbd>	Estándar o ninguno	
<label>	Estándar o ninguno	
<legend>	Estándar o ninguno	
	Estándar o ninguno	
<link>	Estándar o ninguno	
<mark>	Atributos globales	Nueva etiqueta

Figure 3: Diferencia HTML4 vs HMTL5.

22

[22] https://es.wikipedia.org/wiki/HTML5

<kbd>	Estándar o ninguno	
<label>	Estándar o ninguno	
<legend>	Estándar o ninguno	
	Estándar o ninguno	
<link>	Estándar o ninguno	
<mark>	Atributos globales	Nueva etiqueta
<map>	Estándar o ninguno	
<menu>	Estándar o ninguno	
<meta>	Estándar o ninguno	
<meter>	high I low I max I min I optimum I value	Nueva etiqueta
<nav>	Atributos globales	Nueva etiqueta
~~<noframes>~~		Etiqueta eliminada
<noscript>	Estándar o ninguno	
<object>	Estándar o ninguno	
	Estándar o ninguno	
<optgroup>	Estándar o ninguno	
<option>	Estándar o ninguno	
<output>	form	Nueva etiqueta
<p>	Estándar o ninguno	
<param>	Estándar o ninguno	
<pre>	Estándar o ninguno	
<progress>	max I value	Nueva etiqueta
<base>	Estándar o ninguno	
~~<basefont>~~		Etiqueta eliminada
<bdo>	Estándar o ninguno	
~~<big>~~		Etiqueta eliminada
<blockquote>	Estándar o ninguno	
<body>	Estándar o ninguno	
 	Estándar o ninguno	
<button>	Estándar o ninguno	
<canvas>	height I width	Nueva etiqueta
<caption>	Estándar o ninguno	
~~ ~~		Etiqueta eliminada
<cite>	Atributos globales	Etiqueta cambiada
<code>	Estándar o ninguno	
<col>	Estándar o ninguno	
<colgroup>	Estándar o ninguno	
<datalist>	Atributos globales	Nueva etiqueta

Figure 4: Diferencia HTML4 vs HMTL5

He extraído una tabla de diferencia para poder visualizar todos los cambios. Todas las diferencias eran necesarias para el desarrollo web, siendo aceptado y con buena crítica por la comunidad, ya que era ampliamente demandado.

Estilo y Diseño CSS

Para hacer que la información sea más atractiva para el usuario, se ha utilizado el framework por excelencia desarrollado por Twitter, Bootsprat 3[23]. Bootsprat[24] fue liberado por Twitter en agosto de 2011 y desde entonces no ha parado de crecer, tanto a nivel de usuarios de comunidad, como de uso. Tiene una licencia MIT (Apache License 2.0) lo cual hace su uso libre, pero mencionando al creador en todos los proyectos usados. Es el proyecto más popular en GitHub[25], lo cual hace que seduzca con mayor rapidez a los desarrolladores. Unas de las características principales de bootsprat son las siguientes:

- Sencillo y ligero.
- Responsive design.
- Arquitectura basada en Less.

Bootstrap se basa en un sistema de rejillas de doce columnas para crear el layout de una página web. Usa la técnica de CSS conocida como media queries para ajustar automáticamente el ancho de cada columna con acuerdo al tamaño/resolución de la pantalla. Además, dispone de una serie de clases para poder especificar una serie de grids (celda) . Permite también hacer offseting, desplazamiento lateral y horizontal.

Javascript

El código JS es quien da vida a nuestra aplicación web, es el encargado de darle dinamismo y vitalidad a nuestra plataforma. La aplicación cliente tiene la mayor potencia en el código JS, ya que es quien realiza todas las peticiones al servidor, navegaciones, búsquedas... Dentro del amplio abanico de framework, se ha tomado

[23] http://getbootstrap.com
[24] http://getbootstrap.com
[25] https://github.com/twbs/bootstrap

la decisión de utilizar jQuery[26] y AngularJS[27]. jQuery fue lanzado en 2006 con doble licencia de uso MIT license y Licencia General Pública GNU. Ofrece una gran cantidad de funcionalidad y bibliotecas que en código JS básico requerirían mucho más código. Algunas de las características principales de jQuery son:

- Acceso inmediato al DOM.
- Manipulación de estilos.
- Asincronía (AJAX).
- Animaciones.
- Compatibilidad de navegadores.

La aparición de jQuery hizo más fácil el desarrollo de cliente web, ofreciendo una mayor versatilidad y funcionalidad. La aparición de este framework ofreció una infraestructura limpia y ágil con la que los desarrolladores pueden crear aplicaciones complejas en el lado cliente web. La gran potencia radica en el uso de selectores. Vemos un ejemplo en el siguiente código:

```
<div id="capa">Pon el ratón encima de esta capa</div>
<div id="mensaje" >Has puesto el ratón encima!!</div>

$("#capa").mouseenter(function(evento){
   $("#mensaje").css("display", "block");
});
$("#capa").mouseleave(function(evento){
   $("#mensaje").css("display", "none");
});
```

Figure 5: Ejemplo jQuery

Como podemos apreciar en el código, la funcionalidad es simple. El acceso con selectores es bastante fácil, con el selector se accede a través del DOM a las propiedades habilitando toda posibilidad de modificación.

[26] https://jquery.com
[27] https://angularjs.org

Por otra parte, toda la potencia y lógica del cliente web radica en el uso de AngularJS. AngularJS es un framework javascript de código abierto desarrollado por Google y escrito en javascript que trabaja en el lado del cliente y nos permite desarrollar aplicaciones web de manera dinámica utilizando las tecnologías de HTML y CSS. Puede usar los patrones de diseño tanto MVC como MVVM. AngularJS está apoyado por Google y cada día más desarrolladores están adoptándolo, lo que nos da una idea del prometedor futuro de la librería. AngularJS nos permite tener un código más limpio y ordenador, evitando así tener código Javascript dentro del propio HTML. Es compatible con otros frameworks como jQuery, aunque tiene una filosofía contraria, ya que angular no es una librería de acceso al DOM.

¿Qué es angular JS?

Como se ha explicado anteriormente, angular ofrece considerables cambios para los desarrollos de código javascript, pudiendo llevar parte de la lógica al cliente eliminando al servidor cargar de trabajo. En la siguiente imagen se muestra la arquitectura de angular que se explicará a continuación.

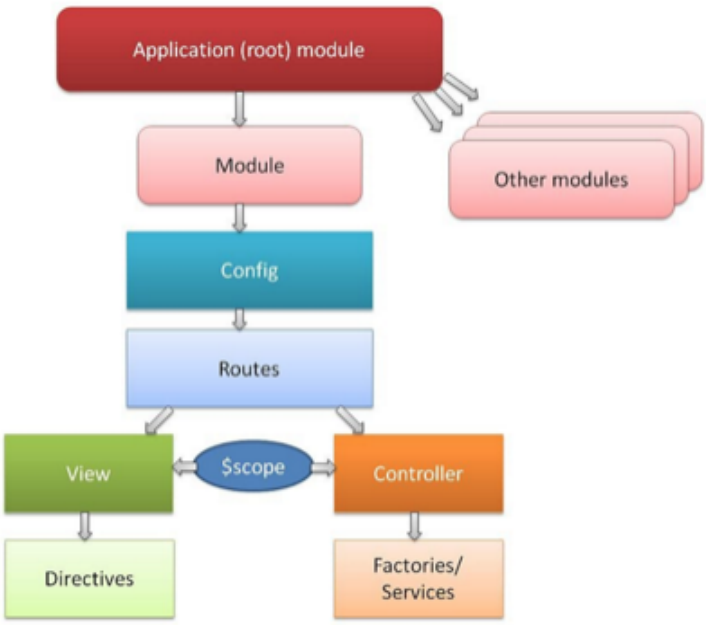


Figure 6: Arquitectura AngularJS

Como se muestra en la imagen, angular proporciona un patrón y una arquitectura diferente en el lado del cliente, separando así la vista por un lado y los controladores por otro, aunque están unidos por un scope. La arquitectura se divide en los siguientes bloques:

- Application: Se define una única aplicación angular para cada caso de uso o pantalla.
- Module: se define uno o varios para las diferentes partes de la aplicación de angular. Utilizado a modo contenedor para los diferentes servicios que puede ofrecer angular. Véase controladores, servicios, filtros, etc.
- Config: proporciona una configuración básica sobre la aplicación de angular. Por ejemplo, para la realización de un SPA (single page application) modificamos el config para cuando cambie la URL y cambie una plantilla de HTML u otra, indicando también que controlador que será el propio en cada HTML.
- Routes: usado para cargar la vista e indicar el controlador de dicha vista, también es usado para cambiar de vistas entre la diferente aplicación de angular. Usado también para realizar SPA, o para navegación hacia otras pantallas de nuestro aplicativo.
- View: vista de nuestro HTML general de nuestra aplicación, podremos tener tantas vistas diferentes como se crean oportunas, cada una con su controlador asignado.
- Controller: Controla la vista y es donde residirá la lógica de la aplicación que queramos proporcionar con código Javascript. Promueve los datos entre el servidor y la vista. Por lo tanto, hace un mejor uso de los mismos teniendo un control de ellos.
- Scope: la vista y el controlador están unidos mediante un scope, ésta es la parte más importante de angular, la unión de los datos entre el controlador y la vista.

- Directives: Cuelgan de la vista proporcionando directivas de angular en la parte del HTML, proporcionando así la relación entre el controlador y la vista, siendo un elemento de marcaje del DOM.

- Factories/service: cuelgan del controlador y proporcionan servicios, como por ejemplo GET, POST, PUT y DELETE. Usa inyección de dependencias, sustituyendo a los objetos. Se pueden crear tantos servicios como creamos convenientes para que se ajuste a la funcionalidad de nuestra aplicación, dando una lógica en el lado del cliente, evitando la programación spaghetti.

3.1.4 Testing

El desarrollo de test unitarios se ha desarrollado mediante mocking. Los mock permiten simular el comportamiento real que debería tener la lógica de la aplicación, sin necesidad de ser ejecutada. Los test son una de las fases más importantes de la ingeniería, ya que proporcionan gran versatilidad para probar la aplicación antes de hacer subidas a producción. Las pruebas desarrolladas serán: pruebas funcionales, de integración y de unidad. Pruebas funcionales: se realizan mediante pasos indicando las ejecuciones que tienen que realizar para finalizar la prueba funcional, constan de una entrada y una salida por cada paso. Pruebas de integridad: acoplamiento de componentes que no han sido desarolladas en este proyecto. Pruebas de unidad: se prueba el módulo de manera individual, por lo tanto, si el comportamiento falla, el test dará un error de ejecución. El uso de mock se ha implementado en las pruebas de unidad usando la librería de MagicMock que nos ofrece al API de Python en la versión 3.2.

3.2 Sistema de recogida de datos

En la recogida de datos, nuestra preferencia siempre ha sido la utilización de hardware libre. Dentro de todas las posibles plataformas, la joya de la corona es Arduino. Arduino es una placa de prototipo básico, la cual nos ofrece todo tipo de posibilidades

desde monitorización básica, hasta robots inteligentes. Voy a detallar brevemente todas las placas de Arduino disponibles con sus especificaciones técnicas y seguidamente se explicará el porqué de la placa resultante. Arduino tiene la misma problemática-ventaja que Linux, la libre distribución y el gran número de versiones que tiene. Existen un gran número de modelos oficiales y no oficiales para todos los tipos de requisitos.

ARDUINO UNO

Es la placa más extendida de todas y la primera que salió al mercado, es la placa de referencia e iniciación al mundo de arduino. Se basa en:

- Procesador Texas instrument Sitara AM335x a 1 Ghz.
- 512 MB de DDR3L.
- Microcontrolador Atmel ATmega320 de 8 bits a 16Mhz.
- 32 Kb de flash.
- 2.5 KB de SRAM.
- 14 pines digitales y 6 pines analógicos.

ARDUINO TRE

Es la primera placa construida en EEUU. Consigue hasta cien veces más rendimiento que la placa UNO o Leonardo, soportando un sistema basado en Linux. Tiene las siguientes características técnicas:

- Microcontrolador Atmel ATmega320 de 8 bits a 16Mhz.
- 32KB de memoria flash.
- 2KB de memoria SRAM.
- 1KB de memoria EEPROM.
- 14 pines digitales y 6 analógicos.

GENUINO 101

Es conocida dentro de EEUU como Arduino 101 y fuera como genuino, fue presentada en Roma en el Opening Conference at marker Faire. Tiene un precio de treinta euros y una características similares a arduino. Las características son las siguiente:

- Microcontrolador Atmel ATmega320 de 8 bit a 16Mhz.
- Módulo intel Curie, Intel Quark de 32 bit.
- 80KB SRAM.
- 385 KB de memoria flash.
- DSP.
- Bluetooth.
- Acelerómetro y giroscopio.

ARDUINO ZERO

Tiene un aspecto similar al UNO. En vez de un microcontrolador Atmel ATmega de 8bit, el zero contiene uno más potente Atmel SAMD21 de 48Mhz de 32 bit. Esta placa está destinada para usuario que arduino UNO se pueda quedar corto en procesamiento. Otras características que contiene son:

- 256 KB de memoria flash.
- 32 KB de memoria SRAM.
- 16KB de EEPROM.
- 14 pines digitales y 6 analógicos.

ARDUINO YUN

Basa su arquitectura en un microcontrolador ATmega32 y un chip Atheros AR9331 que controla el USB, puerto micro-SD y red Ethernet/WiFi. Se comunican mediante un puente. El procesador atheros soporta una distribución Linux llamada Linino. La

estructura es similar al arduino UNO pero con capacidades nativas para conexiones digitales. Contiene también la siguiente característica:

- Microcontrolador Atmel ATmega320 de 8 bits a 16Mhz.
- Microcontrolador ATmega32.
- Atheros AR9331.
- 20 pines, 7 pueden ser PWN y 12 analógicos.
- 32 KB de memoria flash.
- 2,5 KB de memoria SRAM.
- 1 KB de EEPROM.
- 64 MB de RAM y 16MB de flash para soportar Linux.

ARDUINO LEONARDO

Es una placa muy similar al UNO, pero este microcontrolador puede manejar 20 pines digitales. La estructura es bastante más compacta que el UNO, contiene un mini USB y los pines vienen compactos en la placa, lo que hace poder trabajar con ella en distintos entornos de desarrollo. Tiene las siguientes características:

- Microcontrolador ATmega32u4 de bajo consumo a 16Mhz.
- 32KB de memoria flash.
- 2,5Mb de SRAM.

ARDUINO DUE

Es el todoterreno de los Arduino y una de las más potentes, tiene una potencia de cálculo bastante superior a sus predecesores, por ello es idóneo para proyectos con gran capacidad de procesamiento. Tiene un core de 32 bit lo que permite operaciones de 4 bytes en un solo ciclo de reloj. Además tiene las siguientes características técnicas:

- Microcontrolador Ateml SAM3X8E Cortex M3 de 32 bit a 84 Mhz.
- Memoria SRAM de 96kb.
- 512 Kb de memoria flash.
- 54 pines digitales y 12 analógicos.

ARDUINO MEGA

El nombre proviene del controlador que lo maneja, es muy similar al Arduino Due pero tiene una arquitectura AVR en vez de una ARM. Tiene las siguientes características:

- Microcontrolador ATmega32u4 de bajo consumo a 16Mhz.
- Microcontrolador ATmega2560 a 16Mhz a 5V.
- SRAM de 8MB.
- 4KB de EEPROM.
- 256 KB de flash.
- 54 pines digitales, 16 de ellos analógicos.

ARDUINO ETHERNET

Tiene unas características similares a un Arduino uno pero con un conector Ethernet integrador, la otra opción de trabajo sería comprar un UNO y un shield Ethernet para integrarlo. Características similares al UNO:

- Microcontrolador Atmel ATmega320 de 8 bits a 16Mhz.
- 32KB de memoria flash.
- 2KB de memoria SRAM
- 1KB de memoria EEPROM
- 14 pines digitales y 6 analógicos.

ARDUINO NANO

Tiene unas dimensiones reducidas, pero a pesar de eso no deja de tener toda la funcionalidad de una placa convencional. Se alimenta a través de un puerto mini-USB, está especialmente pensada donde el ahorro tanto de coste, como de espacio es realmente necesario. Tiene las siguientes características técnicas:

- Microcontrolador ATmega168 a 16Mhz.
- 16KB de flash.
- 1 SRAM.
- 512 bytes e EEPROM.

ARDUINO PRO

Está diseñada y construida por SparkFUn Electronic. A pesar de su nombre, no es una de las placas más potentes de toda la gama de Arduino. Se ha concebido como una de las placas para usuarios avanzados y bajo coste. Dispone también de las siguientes características:

- Microcontrolado ATmega168 o ATmega 328 de 8Mhz o 16Mhz.
- 16 o 32 KB de memoria flash.
- 2 Kb de SRAM.
- 512 de EEPROM.

ARDUINO MINI PRO

Es la hermana pequeña de la versión pro. Tiene las mismas características técnicas pero su tamaño es muy reducido, por lo que la hace una placa para tamaños reducidos.

ARDUINO MICRO

Diseñado por Adafruit, tiene bastante autonomía y un reducido tamaño. A pesar de su reducido tamaño tiene bastante potencia. Tiene las siguientes características técnicas:

- Microcontrolador ATmega32u4 de 16 Mhz.
- 20 pines digitales y 12 analógicos.

3.2.1 Plataforma empleada

Después de analizar todas las plataformas de Arduino, debemos analizar también nuestro sistema objetivo y hacer un análisis de los requerimientos del sistema. Haciendo un balance de costes y potencia, se ha decidido por una placa YUN. La decisión se ha tomado por las siguientes ventajas, tiene una distribución Linux, WiFi, Ethernet, microcontrolador ATmega, etc. La facilidad de trabajo que ofrece esta plataforma es enorme, ofreciendo la posibilidad de trabajar con Python, conexión SSH, servidor web y FTP. La hace una placa idónea para realizar toda la funcionalidad requerida para el proyecto. Otra de las posibilidades posibles fue la utilización de un Arduino UNO con 1 shield Ethernet, pero la potencia y versatilidad no es suficiente. En contrapartida, el coste de la placa YUN es bastante superior a todas las analizadas anteriormente. Lo cual, hace que la decisión haya sido difícil de tomar.

3.2.2 Arquitectura hardware

Arduino Yun tiene una arquitectura que lo hace tener una potencia bastante superior a las demás. Se detallan las siguientes especificaciones técnicas:

- Microcontrolador Arduino:

  - Microcontrolador: ATmega32u4.

  - Voltaje: 5V.

  - Voltaje entrada: 5V.

  - Pines digitales: 20.

  - Pines analógicos: 12.

  - Intensidad pin digital: 40mA.

  - Memoria Flash: 32KB.

  - SRAM: 2.5KB.

  - EEPROM: 1KB.

  - Velocidad de reloj: 16Mhz.

- Microprocesador Linux:

  -Procesador: Atheros AR9331.

  -Architectura: MIPS 400Mhz.

  -Voltaje: 3,3V.

  -Ethernet: IEEE 802.3 10/100Mbit.

  -WiFi: IEEE 802.11b /g/n.

  -USB: 2.0.

  -Lector tarjetas: micro SD.

  -RAM: 64 MB DDR2.

  -Memoria Flash: 16 MB.

  -SRAM: 2.5KB.

  -EEPROM: 1KB.

  -Frecuencia reloj: 16Mhz.

Arduino YUN posee el siguiente diseño que, a pesar de todas las posibilidades técnicas, tiene una estructura compacta.

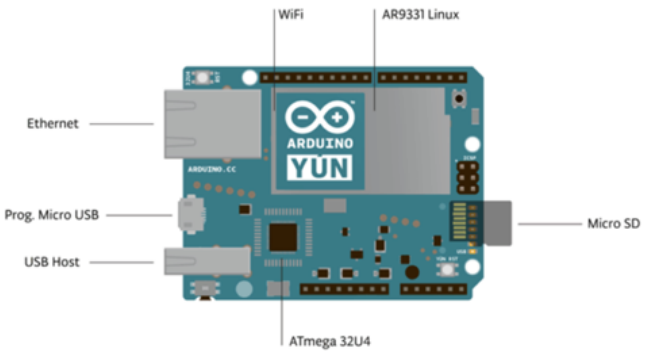


Figure 7: Arduino YUN.

Como podemos ver, su estructura es compacta. La placa posee un puente o bridge de comunicación entre el microprocesador Linux y el controlador ATmega. La potencia de red entre ambos hace el trabajo mucho más fácil habilitando la ejecución de comandos Linux, script de Python, etc. Lo que lo convierte en una robusta interacción.

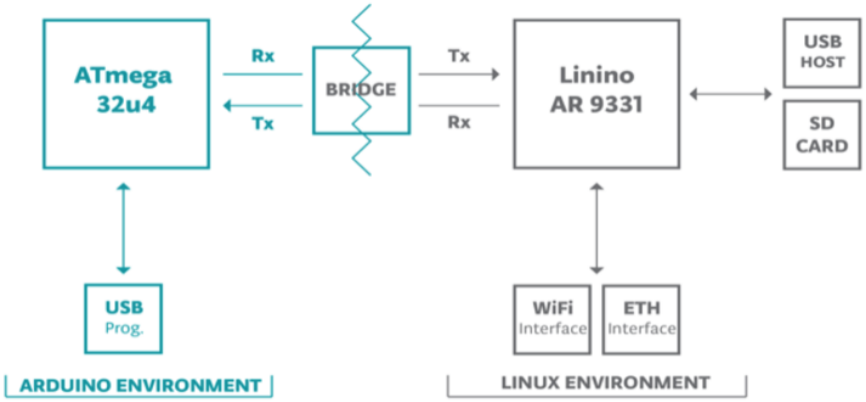


Figure 8: Arquitectura arduino YUN.

Como podemos apreciar en la imagen, el bridge hace de puente entre el Linino y el ATmega habilitando una comunicación bidireccional. La librería bridge facilita la comunicación entre los dos procesadores, proporcionando una interfaz de comunicación. Con el USB prog. conectado al ATmega para poder programar los sketch desde el IDE y conectado al Linino podemos ver conectados los módulos de conexión (WiFi y Ethernet) , el lector de tarjetas y el USB host. El arduino también posee botones de reset situados en los extremos del dispositivo, como se puede apreciar en la imagen.

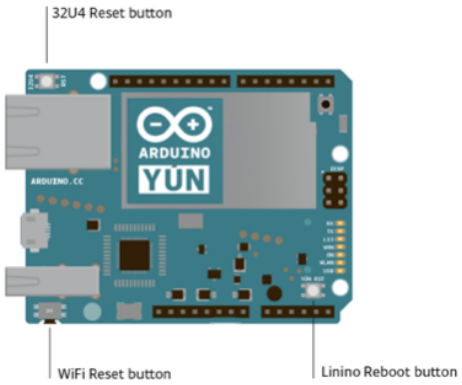


Figure 9: Reset arduino YUN.

3.2.3 Sensores

Es la parte principal de todo sistema de medida, la sensorización es utilizada en todos los campos desde aeronáutica hasta sistemas de riego, siendo parte indispensable para cualquier sistema actual que se requiera construir. Para la elaboración del proyecto objetivo se han utilizado tres tipos de sensores:

Sensor de nivel de fluidos

El sensor de nivel es el encargado de controlar el nivel de fluido que tenemos en el recipiente base. Para la realización del proyecto, hemos elegido el siguiente sensor.

Medidores de altura de fluidos: Standard eTape[28]. Tiene las siguientes características:

- Set de 4 medidor de altura de fluidos.
- Resistencia: 150 Ohmios.
- Ratio de potencia: 0.5 W.
- Tamaño mínimo: 1.0".
- Resolución: 0.01".

[28]http://milonetech.com/products/standard-etape

Figure 10: Sensor Nivel.

La siguiente gráfica muestra la diferencia de voltaje de salida dependiendo de la altura del agua.

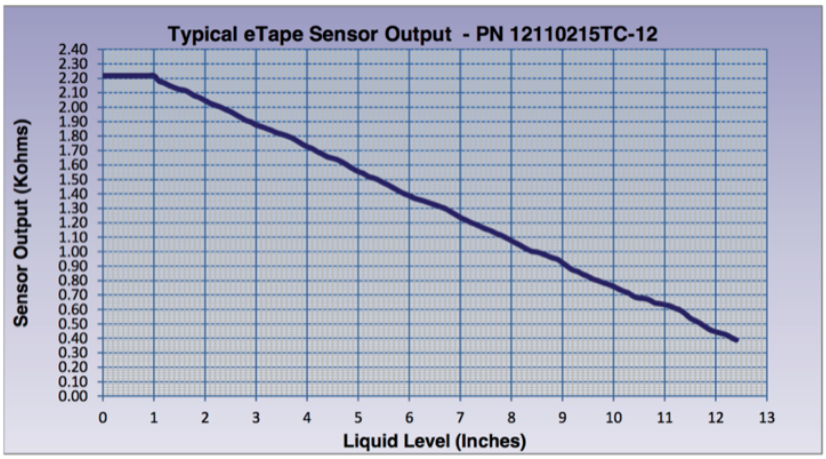


Figure 11: Salida voltaje vs Altura agua.

Sensor de presión

El sensor de presión será utilizado para controlar la presión de todo el circuito, se instalaran sensores de presión en diferentes puntos del sistema, controlando en todo momento las variaciones que estos tuvieran. Para la realización del proyecto se ha adquirido el siguiente sensor de presión.

Se denomina 0-1.2 MPa G1/4" Sensor de presión de líquidos y gases. Características

técnicas:

- Tensión de trabajo: DC 5V.
- Tensión de salida: 0,5 - 5V.
- Corriente de trabajo: 10mA.
- Rango de trabajo: 0-12 bares (1,2 MPa).
- Temperatura de trabajo: 0-85 C.

Figure 12: Sensor Presión

Sensor de caudal

El sensor de caudal será instalado al inicio del circuito, y será el encargado de medir el caudal en todo el sistema. Es un sensor de efecto hall, por lo tanto, se debe tener en cuenta en el desarrollo que éste enviará pulsos que debemos controlar. El sensor utilizado es el siguiente:
EL0432[29], es totalmente compatible con Arduino y por lo tanto, una ventaja. Características técnicas:

- Rango de trabajo: 0-5V.

[29]http://www.ebay.es/itm/CAUDALIMETRO-MEDIDOR-DE-CAUDAL-DE-AGUA-compatible-ARDUINO-desde-Espana-EL0

- Consumo máximo: 15mA a 5V.
- Temperatura máxima: 80º.
- Conexiones 1/4”.

Figure 13: Sensor Caudal

El cuadrilátero tiene la siguiente arquitectura:

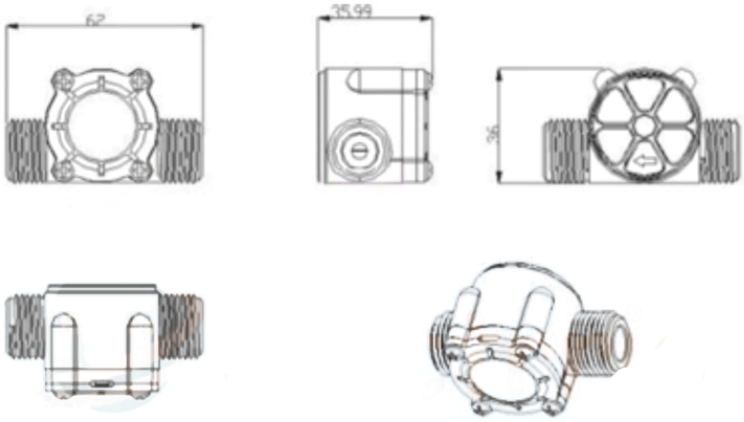


Figure 14: Arquitectura Caudal

Nota: Debido al tiempo de envío desde EEUU y los problemas con la empresa responsable en España de su compra, el sensor de nivel no se ha llegado a comprar. En sustitución y, de manera provisional, se utilizará un sensor de ultrasonido.

Figure 15: Sensor Ultrasonido

El módulo de ultrasonido HC-SR04, contiene las siguientes especificaciones:

- Voltaje de funcionamiento 5V.
- Corriente estática menos de 2mA.
- Ángulo del sensor menos de 15 grados.
- Distancia de detección 2cm - 450cm.
- Alta precisión de hasta 3mm.
- Peso 10g.

Después de realizar pruebas con el sensor de presión, hemos llegado a la conclusión que, el margen de presiones es bastante superior a los ofrecidos por el sistema. Por ello, se ha realizado la compra de otro sensor inferior, el MPX5010. Este sensor es utilizado también para sensorizar entornos con menor margen de presiones.

Figure 16: Sensor MPX5010

El método de obtención de presión es sencillo, se puede obtener tres tipos de presiones: absolutas, calibrada y diferencia[30].

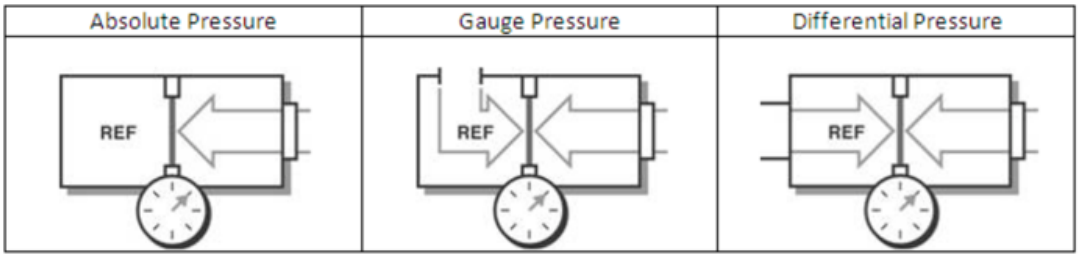


Figure 17: Arquitectura sensor MPX5010

El rango de presiones que soporta está entre los 10kPa y los 100kPa. Para la integración de todos los sensores en el circuito de agua se han comprado adaptadores y juntas específicas.

[30]http://circuits4you.com/2016/05/13/arduino-pressure-measurement/

4 ARQUITECTURA E IMPLEMENTACIÓN

El desarrollo de la arquitectura ha sido la parte más compleja de todo el proyecto, ya que la definición de código limpio y arquitectura limpia se ha llevado a su máxima expresión. La arquitectura de tres capas claramente diferenciadas y el patrón MVC hace que la complejidad de la plataforma sea uno de los puntos fuertes de dicho proyecto.
En el desarrollo de recogida de datos, la arquitectura es bastante más simple, ya que la arquitectura del Arduino YUN ofrece una gran ventaja para hacer la recogida de datos y envío hacia la plataforma. Aun así, se explicará de forma detallada la arquitectura de ambas por separado.

4.1 Aplicación web

Como se ha especificado en el apartado 3, la aplicación web la forman dos partes claramente diferenciadas: backend y frontend. El uso de las tecnologías empleadas también se ha detallado en el apartado anterior. Por lo tanto, se explicará todo lo relativo a colecciones y entidades usadas, tanto en persistencia, como en lógica de negocio.
Dispositivo empleado

- Dispositivo: MacBook Pro de 13”.
- CPU: intel core i5 de doble núcleo a 2,5Ghz.
- Memoria RAM: 16 GB.
- Disco duro: 256 GB de SSD.

Al usar una plataforma de desarrollo OS X, todos los comandos y paquetes utilizados son compatibles con Linux, usando el estándar POSIX. Para poder entender el

concepto de la arquitectura utilizada, debemos comprender de forma abstracta la arquitectura, se puede apreciar en la siguiente imagen la limpieza de dicha arquitectura.

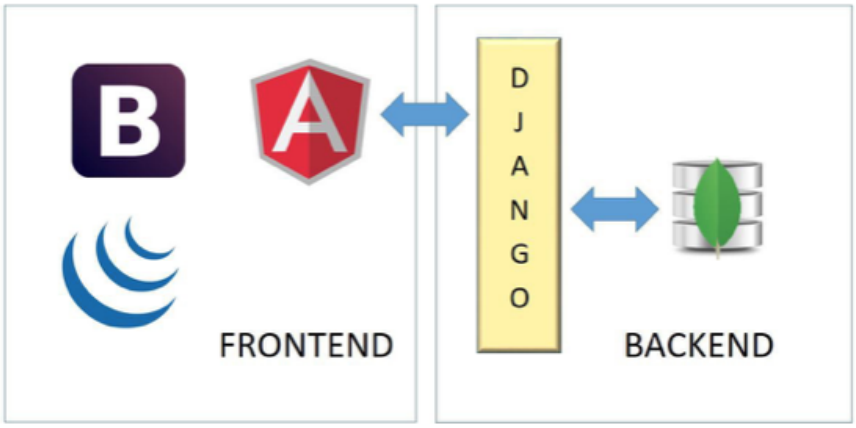


Figure 18: Arquitectura aplicación web.

Como se puede apreciar, hay capas claras y cada una de ellas hace solamente lo recomendado para su capa y uso. En este apartado se explicará cómo se instala, configuración y puesta en deploy todo lo relativo al backend y al frontend.

4.1.1 Backend

El backend es la parte más importante de cualquier plataforma web, es la encargada de procesar la información recibida del exterior, almacenarla o mostrarla a los usuarios mediante una interfaz. Se definen las distintas capas de abstracción que hay, donde reside gran parte de la ingeniería del software y donde se ha centrado gran parte de la lógica de este proyecto. Como podemos apreciar en la imagen anterior, el backend consta de dos capas: persistencia y lógica de negocio. El motor de persistencia utilizado es MongoDB, lo cual nos permite el almacenamiento de información tipo texto plano en formato JSON. En lógica de negocio Django, framework para creación de aplicaciones web basadas en Python. Pasamos a explicar detalladamente por separado todas las colecciones utilizadas y el modelo de datos necesarios para la realización de este proyecto.

PERSISTENCIA

El sistema gestor de bases de datos utilizado es MongoDB, con este gestor podemos almacenar todo tipo de datos. Para poder utilizar MongoDB, el primer paso que debemos hacer es la instalación y configuración de todos los parámetros de Mongo, en nuestra máquina o en el entorno de desarrollo que utilicemos.

Configuración e instalación:

1. Actualización del sistema gestor de paquetes “brew”, utilizando el comando:

```
brew update
```

Con esto, todos los paquetes de nuestro sistema quedarán actualizados.

2. Instalación de MongoDB. Una vez actualizado el sistema “brew”, procedemos a instalar Mongo con el siguiente comando:

```
brew install mongodb
```

3. Añadir los binarios al PATH, para que el sistema reconozca los comandos de Mongo debemos añadir al PATH las librerías, para añadir habría que ejecutar el siguiente comando:

```
export PATH=<mongodb-install-directory>/bin:$PATH
```

Nota: sustituir mongodb-install-directory por la ruta donde se han instalado los binarios.

Para poder ejecutar Mongo debemos ejecutar el siguiente comando:

```
mongod
```

Con esto, lanzamos el motor Mongo con el servicio por defecto y el puerto de escucha también por defecto.

Para poder entrar en MongoDB y hacer un uso de admin sobre nuestras colecciones de Mongo, debemos ejecutar el siguiente comando:

Dentro de la terminal, encontramos esto.

```
MacBook-Pro-de-srkapi:~ sr.kapi$ mongo
MongoDB shell version: 3.2.3
connecting to: test
Server has startup warnings:
2016-07-25T23:08:23.387+0200 I CONTROL  [initandlisten]
2016-07-25T23:08:23.387+0200 I CONTROL  [initandlisten] ** WARNING: soft rlimits
 too low. Number of files is 256, should be at least 1000
>
```

Figure 19: Acceso servicio mongoDB.

Accediendo a la terminal de Mongo, podemos ejecutar todos los comandos disponibles sobre nuestra DB. Para la configuración del proyecto se han utilizado los siguientes comandos:

- "use project" : con este comando creamos o accedemos a nuestra instancia de base de datos, en nuestro caso, 'project'.

```
> use project
switched to db project
>
```

Figure 20: Acceso a base de datos.

- "show collections" : muestra por pantalla todas las colecciones de nuestra instancia de Mongo. En nuestro caso, muestra todas las creadas por Django y las nuestras propias.

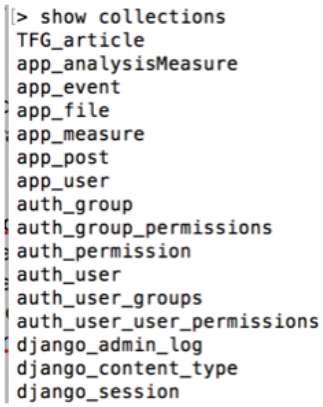


Figure 21: Show Collection.

Como podemos observar, hay un total de dieciseis colecciones. El objetivo de cada colección se define a continuación:

- tfg_article: colección para inserción de datos de prueba.
- app_analysisMeasure: usada para almacenar datos sobre intevalos de medias: hora inicio, hora fin y descripción de medida.
- app_event: guarda eventos de todo tipo, como por ejemplo: medidas negativas, modificación de usuarios, etc. Registra y almacena todo tipo de cambios del sistema, para poder así tener un sistema con un control de eventos claro.
- app_file: almacena el nombre de los ficheros y fecha de subida.
- app_post: colección de datos de prueba.
- app_user: registra todos los usuarios registrados en el sistema, con los siguientes datos: nombre, apellidos, edad, email, contraseña y si es admin o no.
- auth_group: creada por Django, almacena todos los grupos de usuarios.
- auth_group_permissions: creada por Django, almacena los permisos y roles para los grupos almacenados en la anterior colección.

- auth_permission: creada por Django, almacena los permisos de usuarios.
- auth_user: creada por Django, almacena los usuarios activos del sistema.
- auth_user_groups: creada por Django, almacena los grupos de usuarios del sistema.
- auth_user_userpermissions: creada por Django, registra los permisos de cada usuario.
- django_admin_log: creado por Django, sistema de admin log, se registran todos los accesos a la plataforma.
- django_contentype: creada por Django, almacena los modelos de datos.
- django_session: creada por Django, almacena las sesiones de usuarios activas.

Todos estos comandos son los básicos para poder realizar una serie de admin básico sobre nuestras coleciones.

- "db.createCollection(name, options)": creación de colecciones, las cuales son necesarias para almacenar los datos. Es donde residirán todos los datos que genere nuestro sistema.
- "db.collection.find()": extrae todos los datos insertados en una colección.
- "db.collection.remove()": vacía la colección.

CAPA DE NEGOCIO

Para la realización de toda la lógica de negocio, se ha utilizado un framework explicado anteriormente, Django. Con Django, se ha realizado tanto las conexiones a bases de datos tipo Mongo a la renderización de las páginas HTML.

IDE

Para la realización de toda la lógica se ha utilizado el IDE: pyCharm CE[31]. Es un

[31] https://www.jetbrains.com/pycharm/

IDE basado en IntelliJ, actualmente es bastante superior a los antiguos entornos de desarrollo. Contiene todo lo que un desarrollador necesita para hacer sus desarrollos más amenos y rápidos.

CONFIGURACIÓN E INSTALACIÓN

lo primero que tenemos que hacer para configurar e instalar Djandjo es tener instalado Python y para ello debemos seguir los siguientes pasos:

- 1. Descargar el ejecutable de esta URL
- 2. Instalar Python[32] con la confirmación y el launcher por defecto.

Seguidamente, debemos instalar todos los paquetes necesarios para Django. Se deben realizar los siguientes pasos:

- 1. Instalar git.
- 2. Realizar un git clone sobre este repositorio, ejecutar el siguiente comando:

```
git clone git://github.com/django/django.git
```

- 3. Instalar pip [33].

  3.1 Descargar el archivo get-pi.py[34].

  3.2. Ejecutar el comando:

```
'python get-pi.py'
```

- 4. Ejecuta el comando.

```
pip install -e django/
```

[32]https://www.python.org/downloads/mac-osx/
[33]https://pip.pypa.io/en/stable/
[34]https://bootstrap.pypa.io/get-pip.py

Con estos pasos ya tenemos django instalado en nuestro sistema.
También debemos descargar e instalar el driver de conexión Mongo. En nuestro caso se ha utilizado pymongo. Para instalar pymongo[35] se debe ejecutar el comando:

```
python -m pip install pymongo
```

Con todo esto ya tendríamos los paquetes necesarios para el desarrollo de nuestra aplicación web.
El siguiente paso constaría de las siguientes partes, por un lado la creación de un proyecto Django, la configuración del setting, ejecución del servidor, creación polls, URLs, templates, etc. Todos estos pasos se detallarán a continuación.

CREACIÓN DE PROYECTO

Si es la primera vez que se usa Django, deberás de tener cuidado con la configuración inicial. A saber, necesitará código autogenerado que se establece automáticamente por el proyecto Django. En todas las especificaciones y conexiones a bases de datos se establecen a través del setting. Es por ello, una de las grandes ventajas de Django para la realización de proyectos de este calado. Ejecutamos el siguiente comando para crear el proyecto Django:

```
django-admin startproject mysite
```

Se ejecutan todos los procesos para la creación del proyecto. Debería quedar con la siguiente estructura de directorios.

```
mysite/
    manage.py
    mysite/
        __init__.py
        settings.py
        urls.py
        wsgi.py
```

Figure 22: Estructura proyecto Django.

[35] https://api.mongodb.com/python/current/

CONFIGURACIÓN DEL SETTING

El settings.py contiene todas las librerías y configuraciones necesarias. El setting lo dividiremos en las siguientes partes.

- Conexión a database:

```
DATABASES = {
        'default' : {
              'ENGINE' : 'django_mongodb_engine',
              'NAME' : 'project'
           }
       }
```

- Directorios de templates:

```
TEMPLATE_DIRS = (
          os.path.join(BASE_DIR, 'templates'),
    )
```

- Instalador de paquetes:

```
 INSTALLED_APPS = (
     'django.contrib.admin',
     'django.contrib.auth',
     'django.contrib.contenttypes',
     'django.contrib.sessions',
     'django.contrib.messages',
     'django.contrib.staticfiles',
     'task',
     'rest_framework',
  )
```

- Configuración api REST:

```
REST_FRAMEWORK = {
    'DEFAULT_PERMISSION_CLASSES':     ('rest_framework'),
    'PAGE_SIZE': 10
}
```

El fichero setting.py de este proyecto se encuentra en esta ruta[36].

EJECUCIÓN DEL SERVIDOR

Django trae incluido en su gestor un servidor de apliaciones, para ejecutar el servidor debemos ejecutar el siguiente comando:

```
python manage.py runserver
```

Por defecto, se ejecuta en la ruta "127.0.0.1" por el puerto 8000. Para poder hacer accesible desde el exterior se debe ejecutar el siguiente comando:

```
python manage.py runserver 0:8000
```

CREACIÓN DE APP

Ahora ya somos capaces de ejecutar el servidor Django. Una aplicación escrita en Django consiste en un paquete Python, generalmente con un directorio de carpetas creadas por defecto. Pondremos el foco en ese directorio, ya que será el principal de nuestra aplicación. Para la creación de nuestro polls se debe ejecutar el siguiente comando:

```
python manage.py startapp app
```

Con esto se genera automáticamente el directorio para empezar nuestra app. Deberá tener la siguiente estructura:

[36] https://github.com/srkapi/TFGpython/blob/master/mysite/mysite/settings.py

Figure 23: Estructura aplicación Django.

Debemos crear también un fichero que contenga las vistas, lo llamaremos: 'view.py' y otro que contenga las URLs llamado: 'urls.py'.
Como hemos visto anteriormente, al crear el proyecto Django, también se generaba en fichero urls.py es el principal de la aplicación. Para poder redirigir hacia nuestra app. El fichero urls.py quedaría de la siguiente forma:

```
urlpatterns = [
              url(r'^app/', include('app.urls')),
              url(r'^admin/', admin.site.urls),
      ]
```

Como se puede ver, cuando la aplicación detecte la ruta /app/ incluirá todas las urls contenidas en el fichero app.urls.
Una vez definida la composición de las vistas, continuamos por la generación de vistas, es generada a través del fichero "views.py". Se explica mejor mediante un ejemplo, lanzamos el servidor con el comando visto anteriormente, seguidamente, accedemos a través del navegador a esta URL:

```
http://127.0.0.1:8000/app/
```

Como podemos observar, nuestro fichero de app.urls contiene la siguiente url a tratar.

```
url(r'^$', index, name='index')
```

El formato de url tiene tres parámetros, el primero contiene una expresión regular que, añadido al url principal, quiere decir que todos los que accedan a la ruta principal sin ningún parámetro serán renderizados por index, con el nombre index. La vista index la definimos en views.py de la siguiente forma.

```
@login_required(login_url='login/')
def index(request):
    dao = daoEvent.daoEvent()
    list = dao.getEvent(1)
    listUser = dao.getEvent(2)
    context = {'data': list, 'dataUser': listUser}
    return render(request, 'event.html', context)
```

Figure 24: Ejemplo vista Django.

En este caso, vemos que index es un método que tiene de entrada una request o petición, retornando un render con los siguientes parámetros: request, HTML a renderizar y contexto.
Esto sería un ejemplo básico de creación de una página sencilla en Django, es así como se ha construido toda la arquitectura web.
Como vemos en el ejemplo, la aplicación hace uso de DAO´s. Los DAO (Data Access Object) son elementos creados en la arquitectura para acceder a los modelos de dominio. Básicamente, utilizamos los DAO´s para poder acceder a Mongo o capa de persistencia, teniendo así una arquitectura limpia y esperando así la responsabilidad única de cada capa.

- DAO: Contiene la conexión a mongo y las operaciones básicas CRUD[37] sobre ella.

- Conexión con mongo: se ha utilizado pymongo (connection.py). Se han definido todas las colecciones que se han utilizado para realizar la aplicación.

[37] create, update, delete y read.

- Model: Objects creados con los mismos valores utilizados en cada colección de mongo. En nuestro caso son objetos sencillos llamados POJO. Éste sería un ejemplo de objeto POJO.

Como podemos ver en el objeto measure, contiene las características principales que puede tener una medida.

```
class Measure(object):

    def __init__(self, measure_, level_, volume_, date_):
        self.measure = measure_
        self.level = level_
        self.volume = volume_
        self.date = date_
```

Figure 25: Ejemplo modelo measure.

Básicamente se ha seguido la siguiente arquitectura software:

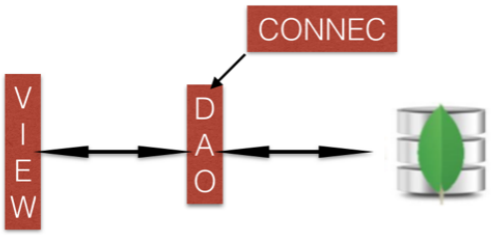


Figure 26: Arquitectura Software.

Se puede apreciar que la responsabilidad única es clara y concisa. Los DAO´s, son los únicos encargados de acceder a Mongo, transformando el JSON recibido en una lista de objetos de modelo y enviando solamente a la vista los objetos obtenidos por el DAO.

La vista es la encargada de renderizar el HTML con los objetos obtenidos, mostrando así al usuario los datos obtenidos. Esa es la manera principal de creación

de vistas pero, se ha creado una API REST con una librería de Django llamada REST_FRAMEWORK.

Con todo esto, quedaría explicado toda la fase de lógica de negocio, la fase de configuración e instalación y una explicación de cómo se crean las vistas.

4.1.2 Frontend

Tras la explicación de la lógica básica de negocio, una de las partes más importantes de la aplicación es la vista. En la vista se ha utilizado la tecnología más puntera actualmente.

MAQUETACIÓN

Como se ha definido anteriormente, el lenguaje de marcado utilizado es HTML5. Utilizado para la generación de páginas estáticas y estructuras definidas. Para definir todos los archivos HTML bajo la misma ruta, Django hace uso de plantillas o templates.

Las templates son definidas en el setting.py mediante una ruta única, en nuestro caso:

```
TEMPLATE_DIRS = (
    os.path.join(BASE_DIR, 'templates'),
)
```

Donde BASE_DIR es la ruta por defecto del proyecto. El archivo principal es index.html, sobre él se ha creado el menú lateral y la toolbar superior. Todos los archivos creados incluyen index.html, por lo tanto, todos contienen el mismo menú. El directorio tiene la siguiente estructura:

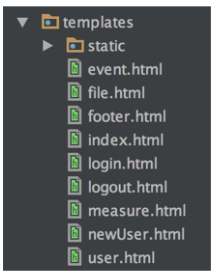


Figure 27: Estructura directorio template.

Dentro del directorio ‘static’, se encuentran todas las librerías utilizadas, ya sea boopstrap, jQuery, etc.En el fichero index carga todos los script necesarios desde esa ruta.

RENDERIZADO

Con Django, el renderizado de vistas se realiza a través de un *.html, en el cual se le pasa el contexto y lo renderiza en la vista. Se explicará ‘event.html’ visto en el ejemplo anterior.

```python
@login_required(login_url='login/')
def index(request):
    dao = daoEvent.daoEvent()
    list = dao.getEvent(1)
    listUser = dao.getEvent(2)
    context = {'data': list, 'dataUser': listUser}
    return render(request, 'event.html', context)
```

Figure 28: Ejemplo render Python.

Como se puede apreciar en la imagen, renderiza el archivo event.html pero antes es añadido el contexto, en este caso dos listas de eventos (user y measure).

Vemos el siguiente archivo event.html

```
{% include "index.html" %}

<div id="page-wrapper">
    <div class="row" >
            <!-- /.col-lg-8 -->
            <div class="col-lg-5" id="event">
                <div class="panel panel-success" >
                    <div class="panel-heading">
                        <i class="fa fa-bell fa-fw"></i> Event Measure
                    </div>
                    <!-- /.panel-heading -->
                    <div class="panel-body" id="bodyPanel">
                        <div class="list-group">
                            {%  for event in data %}
                                <a href="#" class="list-group-item">
                                    <i class="fa fa-gears fa-fw"></i> {{ event.description }}
                                    <span class="pull-right text-muted small"><em>{{ event.dateEvent }}</em>
                                    </span>
                                </a>
                            {% endfor %}

                        </div>
                        <!-- /.list-group -->

                    </div>

                </div>
            </div>
```

Figure 29: Ejemplo render HTML.

```
<!-
    <!-- Bootstrap Core CSS -->
    <link href="/static/bower_components/bootstrap/dist/css/bootstrap.min.css" rel="stylesheet">

    <!-- MetisMenu CSS -->
    <link href="/static/bower_components/metisMenu/dist/metisMenu.min.css" rel="stylesheet">

    <!-- Custom CSS -->
    <link href="/static/dist/css/sb-admin-2.css" rel="stylesheet">

    <!-- Morris Charts CSS -->
    <link href="http://netdna.bootstrapcdn.com/twitter-bootstrap/2.2.2/css/bootstrap-combined.min.css">

    <!-- Custom Fonts -->
    <link href="/static/bower_components/font-awesome/css/font-awesome.min.css" rel="stylesheet" type="text/css">
     <!-- jQuery -->
    <script src="/static/bower_components/jquery/dist/jquery.min.js"></script>

    <!-- Bootstrap Core JavaScript -->
    <script src="/static/bower_components/bootstrap/dist/js/bootstrap.min.js"></script>
```

Figure 30: Ejemplo import HTML.

Apreciamos que incluye el archivo “index.html”, esto quiere decir que, incluye todo el contenido en index.html y después añade nuestro archivo event.html. Observamos también el uso de llaves, estas son para indicar a Django que tiene que serializar, vemos como realiza un recorrido sobre una lista data, antes añadida en el contexto en views.py. Es un ejemplo claro de cómo hace uso Django de los render.

DISEÑO Y ESTILOS

Se ha utilizado el framework creado por Twitter, bootstrap. Lo primero que debemos hacer al crearnos una página con Bootstrap, es buscar una plantilla acorde a nuestras necesidades, hemos utilizado SB ADMIN 2[38]. Esta plantilla proporciona una interfaz típica de administrador, lo cual se ajusta perfectamente a nuestra necesidad.

Seguidamente, debemos importar todos los estilos a nuestro index.html, para poder hacer uso de todas las características y estilos.

Una vez importado todas las librerías, ya podemos hacer uso de todas las características y componente. Una de las principales clases en Bootstrap son los ‘row’, se encargan de dividir una fila en 12 partes iguales. Podemos ver un ejemplo en la siguiente imagen.

[38]https://startbootstrap.com/template-overviews/sb-admin-2/

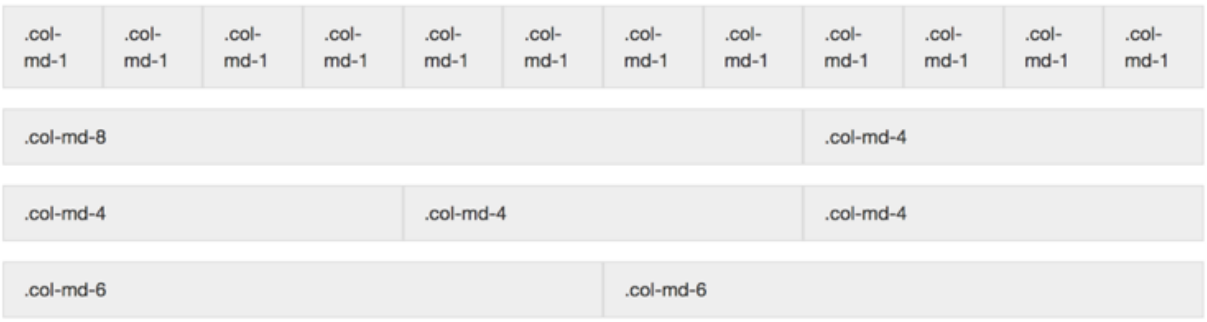


Figure 31: Estructura Bootstrap.

Éste es un ejemplo de cómo se estructuran las páginas con Bootstrap, indicando el md tendremos un tamaño u otro. También se puede hacer Offsetting indicando en la clase el valor que queremos "col-md-offset-* ".
Estas son dos de las características principales de Bootstrap, pero también permite crear botones, formularios, gráficas, tablas, menús desplegables, etc.

En nuestro caso, el menú se encuentra la página 'index.html', contienen todos los enlaces para realizar las navegaciones sobre la aplicación web. Como todos los archivos incluyen el archivo index todas las páginas tendrán el mismo menú y la misma barra superior. Unicamente se modifica el contenido de la pantalla central. Una gran ventaja de Bootstrap, es la responsividad. Una vez indicada la clase a utilizar nos olvidamos del tamaño de pantalla, ya que bootstrap es responsive, pudiéndose utilizar hasta en dispositivos móviles.

JAVASCRIPT

El uso de javascript es la principal potencia del cliente web, es donde reside todo el dinamismo. Se ha utilizado 3 librerías principales: jQuery, AngularJS y FusionChar. Se definirá el uso de cada una y para qué se ha utilizado.
El uso de jQuery es bastante amplio en toda la aplicación, ya que Bootstrap hace uso de él para la realización tanto de menús, como tablas. Un uso específico de jQuery es la creación de pop up, lo vemos en el siguiente código:

```
$(function() {
    $( "#dialog" ).dialog({
      autoOpen: false,
      show: {
        effect: "blind",
        duration: 1000
      },
      hide: {
        effect: "explode",
        duration: 1000
      }
    });

    $( "#opener" ).click(function() {
      $( "#dialog" ).dialog( "open" );
    });

     $( "#send" ).click(function() {
        descrFrom = document.getElementById("descr").value
        $( "#dialog" ).dialog( "close" );

    });
```

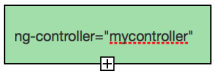
Figure 32: Ejemplo jQuery.

Éste es un ejemplo sencillo de uso de jQuery en nuestra aplicación web, para hacer uso de jQuery debemos iniciar con '$' y seguidamente entre paréntesis el selector. El empleo de jQuery en nuestra aplicación es principalmente para realizar animaciones, por lo tanto, es bastante fácil de mantener e implementar. Por otro lado, tenemos Angular JS. Su uso hace que nuestra aplicación de lado cliente tenga suficiente potencia, realizando algunas de las siguientes operaciones: realizar peticiones al servidor, recogida de datos desde Firebase, realización de SPA, etc.

Lo primero que debemos conocer de AngularJS es su arquitectura, ya que difiere ampliamente de todos los framework anteriormente conocidos. Por un lado, debemos conocer todas las directivas de angular. Las directivas van incluidas en el código HTML y AngularJS la reconoce y hace uso de ellas. Seguidamente, el uso de controladores. Los controladores realizan la parte importante de las operaciones de angular, residiendo la lógica de negocio, peticiones a servicios, etc. Para crear una aplicación de AngularJS lo primero que debemos hacer es iniciar el contexto de angular con la siguiente directiva:

```
'ng-app="myapp"'
```

Con esto, ya tendríamos iniciado la aplicación de AngularJS. Lo siguiente que debemos hacer es crear los controladores con la siguiente directiva:



Debemos tener claro donde poner la directiva, si la ponemos en un 'div' solamente tendrá utilidad y visibilidad el controller dentro de dicho div. Para tener mayor potencia se declaran al inicio de cada página, evitando confusión y creando un único controlador por funcionalidad. Accedemos al controlador, para ello en código JS debemos indicarlo de la siguiente forma:

```
var app = angular.module("myapp", []);
app.controller("mycontroller", function($scope,$http) {
```

Figure 33: Module angularJS.

Como vemos en la imagen, obtenemos el módulo de aplicación y seguidamente indicamos que controlador queremos crear. Dentro de dicho controlador reside toda la lógica.
Este es un ejemplo de uso de angular en nuestra aplicación. Algunas de las directivas usadas son las siguientes:

- ng-app: inicia aplicaciones angular.
- ng-controller: indica el controlador
- ng-model: databiding, pudiendo ser modificado desde el controlador por el $scope.
- ng-click: sustituye al antiguo onclick de JS.
- ng-init: inicia el valor de una variable del cope
- ng-repeat: similar a un bucle for, itera sobre el contenido de una lista.
- ng-change: sustituye al antiguo onchange de JS.

- ng-show/ng-hide: permite mostrar y ocultar información según las condiciones que indiquemos.

- ng-bind: tiene la misma funcionalidad que las llaves {{}}.

AngularJS también permite crear tus propias directivas, esta funcionalidad tiene mucha potencia ya que encapsula mediante una única directiva una template o HTML, un controlador y un scope asociado.
Podemos ver la potencia con el siguiente ejemplo[39]:

```
//Aquí creamos la directiva
app.directive('miCliente', function() {
  return {
    template: 'Nombre: {{cliente.nombre}} Dirección: {{cliente.direccion}}'
  };
});
```

Figure 34: Directive angularJS.

En la parte de la vista, ya podríamos usar la directiva como vemos en la siguiente imagen.

```
<body>
   <div ng-controller="MiControlador">
      <div mi-cliente></div>
   </div>
</body>
```

Figure 35: Ejemplo Angular 1.

Para no mezclar el código JS y el HTML en caso de renderizar dichas directivas, AngularJS hace uso de templateURL para dar solución a dichos problemas. Vemos el uso en la siguiente imagen.

[39]https://frontendlabs.io/2287–aprendiendo-directivas-en-angularjs

```
app.directive('miCliente', function() {
  return {
    templateUrl: 'cliente.html'
  };
});
```

Figure 36: Ejemplo Angular 2.

Dentro de la directiva existe una propiedad llamada restrict, indica restricciones que podemos incluir en dicha directiva. También podemos definir el nombre del scope a la directiva como vemos en la imagen siguiente:

```
app.directive('miCliente', function() {
  return {
    restrict: 'AE',
    scope: {
      clienteDinamico: '=cliente'
    },
    templateUrl: 'cliente.html'
  }
});
```

Figure 37: Ejemplo Angular 3.

Ésta es la parte principal de AngularJS, pero existen otras características como son, el uso de factorías.

Las factorías son contenedores de código que son usados por los controladores, son similares a los services. Lo común es que devuelvan objetos JS, dichos objetos son obtenidos de llamadas a servicios REST.

También son usadas para transferir objetos entre diferentes controladores, ya que solamente se instancia una vez en toda la aplicación de angular. Para definir las factorias debemos seguir el siguiente ejemplo[40]:

[40]http://www.desarrolloweb.com/articulos/factorias-factory-angularjs.html

```
.factory("descargasFactory", function(){
    var descargasRealizadas = ["Manual de Javascript",
```

Figure 38: Ejemplo Angular 4.

Ya tendríamos creada una factoría, para poder inyectarla en los controller solamente debemos indicarla en la cabecera del function.

```
.controller("appCtrl", function(descargasFactory){
    var vm = this;
```

Figure 39: Ejemplo Angular 5.

Con todo esto quedaría detallado el uso de AngularJS en nuestra aplicación web. Como hemos indicado anteriormente, se ha hecho uso de una librería para realización de gráficas. Fusioncharts[41], es usada para creación de todo tipo de gráficas debido a su simplicidad de uso. En nuestro caso, se ha usado para creación de gráficas en 2D y usada para visualización de medidas.

4.1.3 Real Time Firebase

Una ventaja de nuestra aplicación web es la visualización de datos en tiempo real sin tener que realizar peticiones al servidor, para ello hacemos uso de un servicio externo llamado Firebase[42]. Firebase, fue comprado por google en 2015 e integrado en sus sistemas de Google cloud. La plataforma de Firebase nos ofrece visualización de todo tipo de datos en multiplataforma, ya sea Android, iPhone o web. Estás son algunos de los servicios ofrecidos por la plataforma:

- Database: nos ofrece servicio de base de datos, utiliza una base de datos no relacional. Actualizando en todo momento todos los dispositivos asociados al

[41] http://www.fusioncharts.com
[42] https://www.firebase.com

dataset. La tecnología que usa para llevar a cabo esto son websocket. Éste es un ejemplo de uso en nuestra aplicación web.

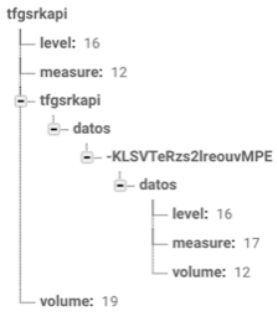


Figure 40: Nodo proyecto firebase.

Como vemos, Firebase crea un dataset con los valores que hemos indicado. Pudiendo modificar los nodos que queramos accediendo a través de una URL.

- Autenticación: Firebase proporciona un sistema de control de usuarios, ahorrando así crear y diseñar nuestro propio sistema. Tienen incluidos todos los sistemas de login más utilizados: Facebook, Twitter, Github, Google y sistema tradicional de correo.
- Carpetas: permite guardar y almacenar archivos enviados por los usuarios.
- Hosting: ofrecer servicio de hosting, proporcionando un sistema gratuito, pero también puedes ofrecer un sistema externo.
- Crash: cuando una aplicación de Android falla, Firebase registra dicho error y envía una notificación de dicho error.
- Notificaciones: servicio de envío de notificaciones a todos los usuarios de la aplicación.

Para este proyecto solamente hemos utilizado el servicio de dataset, pero los demás servicios pueden ser utilizados para implementaciones futuras.

Firebase está diseñado para aplicaciones multiplataforma, por lo tanto se puede usar en Android, IOS o web. Usaremos una librería de AngularJS para poder obtener los datos de nuestro servicio de dataset.
Para lograr acceder a nuestra aplicación de Firebase solamente necesitamos esta línea de código:

```
var myFirebaseRef = new Firebase("https://tfgsrkapi.firebaseio.com/");
```

Con esto, ya podríamos acceder a todos los datos que tenemos en nuestro dataset, crear usuarios,etc.
Accedemos al nodo donde se encuentran los datos y los mostramos por pantalla.
Al utilizar websocket no hay necesidad de recargar la pantalla para obtener los datos, ya que son enviados directamente desde Firebase a nuestro navegador web. Ésta es una gran ventaja, ya que se pueden mostrar los datos al usuario en tiempo real.

4.2 Recogida de datos

El sistema de recogida utilizado es un Arduino YUN, ya se ha definido en el capítulo anterior su arquitectura. Lo primero que debemos hacer cuando nos encontramos un Arduino YUN es obtener su dirección IP, pudiendo así acceder mediante SSH al sistema Lilino. La plataforma genera su propia red WiFi para poder conectarnos a ella, una vez conectados, accedemos a su panel de control y podemos cambiar todos los valores WiFi y redes que deseemos. Para poder entrar al panel de configuración y conectarnos mediante SSH debemos seguir los siguientes pasos:

- 1. Nos conectamos a la red Linino.

- 2. Accedemos a la URL y nos encontramos el siguiente panel de control.

```
http://192.168.240.1/cgi-bin/luci/webpanel/homepage
```

Figure 41: URL acceso Linino.

Figure 42: Pantalla acceso Linino.

NOTA: la password usada es “doghunter”.

- 3. Obtenemos su dirección.

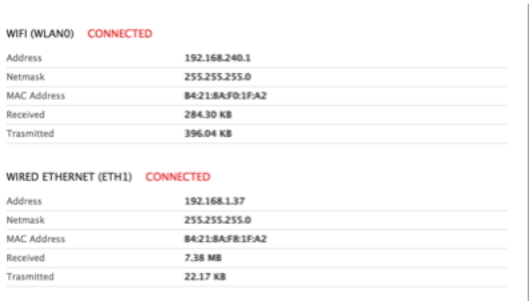


Figure 43: Pantalla ajustes Linino.

- 4. Copiamos la dirección IP, abrimos la terminal e introducimos el siguiente comando:

```
ssh root@192.168.1.37
```

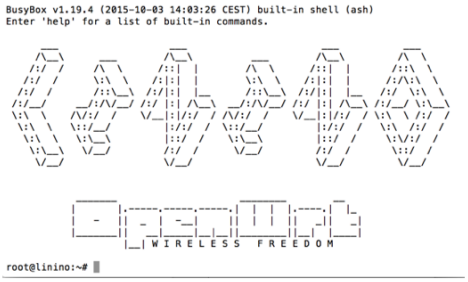


Figure 44: Acceso a Linino mediante SSH.

Con esto ya tendríamos acceso al Linino. El sistema trae instalado por defecto Python, PHP y servidor web. Por ello, hace muy accesible su uso para este tipo de proyecto.

CONFIGURACIÓN E INTALACIÓN

En el dispositivo de medida se ha instalado una serie de paquetes, para hacer uso de librerías y herramientas de transferencia de archivos.

El primer paso, es instalar el gestor de paquetes 'easy_pip', se deben seguir los siguientes pasos.

- 1. Descargar el archivo get_pip.py[43]
- 2. Ejecutar el comando.

```
python get-pip.py
```

Una vez instalado easy pip lo siguiente que debemos instalar es un servidor de FTP, para poder transferir archivos y facilitar la labor de desarrollo. Se debe ejecutar el siguiente comando.

```
opkg install openssh-sftp-server
```

Con esto, ya tendríamos listo nuestro arduino YUN para el desarrollo.

Como el Arduino YUN posee tan poca memoria, debemos hacer una expansión de su

[43] https://bootstrap.pypa.io/get-pip.py

memoria usando una tarjeta microSD. Arduino nos proporciona un sketch para estos casos. Seguiremos los siguientes pasos:

- 1. Descargar el archivo en la siguiente ruta[44].
- 2. Abrir el IDE Arduino, conectar el Arduino mediante un Ethernet, cargar el archivo en el microcontrolador y abrir el monitor serie.
- 3. Responder YES cuando aparezca la siguiente pantalla y comenzará el proceso de formateo de la tarjeta microSD.

Figure 45: Expansión memoria SD 1.

- 4. Responder yes cuando pregunte si tenemos el Arduino conectado a Internet.

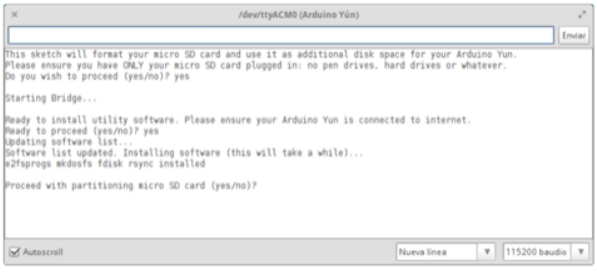


Figure 46: Expansión memoria SD 2.

- 5. Si todo va bien, nos preguntará si queremos continuar con el proceso, responderemos YES.

[44]http://arduino.cc/en/uploads/Tutorial/YunDiskSpaceExpander.zip

Figure 47: Expansión memoria SD 3.

- 6. El siguiente paso debemos introducir el tamaño de memoria que debe tener la partición de datos. Suele ser un 45% del tamaño total de la memoria.

Figure 48: Expansión memoria SD 4.

- 7. Comienza el proceso de particionado, este proceso puede tardar algo de tiempo.

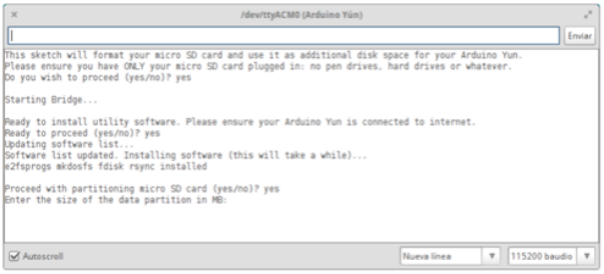


Figure 49: Expansión memoria SD 4.

- 8. Reiniciar el Arduino YUN y ejecutar el comando para comprobar el tamaño de la memoria.

```
df -h / /mnt/sda1
```

Si no ha encontrado ningún error, deberíamos tener un tamaño superior de memoria.

4.2.1 Arquitectura software Linino

Para poder enviar los datos a nuestra plataforma, hemos utilizado Python. Con Python tenemos la flexibilidad para poder recoger datos a través de la API REST creada en el Arduino, enviarlos al mismo tiempo al servidor web y, al mismo tiempo, enviarlo al servicio web Firebase.
El script de Python sigue la siguiente fases:

- 1. Leer el JSON generado por el sketch. Se puede leer directamente a través de esta URL.

```
http://192.168.240.1/arduino/data/1
```

- 2. Descodificar el JSON obtenido.
- 3. Enviar los datos al servicio externo Firebase, mediante una petición PUT.
- 4. Enviar los datos obtenidos a la plataforma, mediante una petición GET.
- 5. Ejecutar el archivo obtainData.py con el crontab de manera periódica.

Con esto ya tendríamos la estructura general para hacer la recogida y el envío de datos. Por otro lado, tenemos el sistema de actualización de sketch, también realizado en Python. Contiene la siguiente estructura:

- 1. Obtener el sketch de la ruta del servidor, mediando el comando wget.

- 2. Mueve el archivo a la carpeta repositorio.
- 3. Comprueba la fecha de modificación.
- 4. En caso de ser mayor la fecha de modificación, copia el fichero de repositorio a directorio producción.
- 5. Ejecuta los comandos para cargar el fichero .hex en el micro-contralador.
- 6. Ejecutar el archivo updateSketch.py con el crontab de manera periódica.

Con estos dos script tenemos creado toda la lógica importante contenida en el sistema Linino. Debemos tener en cuenta el tiempo de temporización, para ello, usaremos el crontab para programar la ejecución de los dos procesos. En del caso de script obtainData.py lo programaremos para ejecutarlo cada segundo, el de updateSketch.py cada hora. Para realizar la transferencia de archivos entre el arduino YUN y el PC de desarrollo usaremos un filezilla, ya que antes teníamos instalado un servidor FTP en el arduino. Otro punto a tener en cuenta son los sistemas de directorios, hemos creado 3 directorios para una mayor organización.

- Repositorio: almacenaremos los sketch procedentes del servidor web.
- Producción: almacena el sketch que existe actualmente en producción.
- Log: sistema de log con los datos obtenidos de los sensores.
- Error: almacena un error cuando el sistema encuentra algún error.

4.2.2 API REST arduino

Para recoger datos de los sensores de manera más correcta y eficiente, se ha construido una API REST en el sketch de Arduino. Los sketch de Arduino se componen de dos módulos principales: setup y loop.

- Setup: Configuración del microcontrolador, se especifican todos los parámetros de entrada, salida, inicialización de librerías, etc.

- Loop: se ejecuta de manera cíclica, cuando acaba la ejecución del método vuelve otra vez al inicio ejecutando una y otra vez todas las instrucciones.

Para realizar la API REST se han utilizado librerías externas: BridgeServer y BridgeClient. El uso de estas librerías es bastante sencillo, para poder crear nuestra API REST debemos seguir los siguientes pasos:

- 1. Aceptamos el servidor con el siguiente código:

```
BridgeClient client = server.accept();
```

- 2. Hacemos una comprobación si tenemos un cliente o petición web.
- 3. Leemos hasta el que encuentre un atributo '/' y comprobamos si la petición es correcta.
- 4. Obtenemos los datos de los sensores.
- 5. Creamos una respuesta con el cliente Bridge con el siguiente código:

```
client.print("prueba");
```

Con esto, tendríamos construida una sencilla API REST, pero una de las partes más importantes de sketch es la obtención de datos procedentes de los sensores. Como se ha explicado en apartados anteriores, necesitamos tres tipos de sensores, actualmente solamente se han implementado dos de ello, debido a la estructura del sistema. Estos sensores de agua son de difícil de utilización, ya que necesitamos un circuito de agua adaptado a los sensores utilizados. Explicaremos el desarrollo de los sensores a continuación.

CAUDAL

El sensor de caudal es el que más complejidad tiene, es un sensor de efecto hall. Quiere decir que, envía pulsos durante un intervalo de tiempo y debemos capturar dichos

pulsos y calcular el número de vueltas que ha realizado el sensor en un periodo X. Para poder capturar los datos procedentes del sensor seguiremos los siguientes pasos de construcción:

- 1. Crear una variable global para contar el número de impulsos recibidos.
- 2.Configurar en el módulo 'setup' para recepción de pulsos indicando el pin de entrada, el método a ejecutar y el tipo de señal. Se realizará con el siguiente código:

```
1.  attachInterrupt(digitalPinToInterrupt(2),
```

- 3. Crear módulo para obtener la medida. Como podemos ver en la siguiente imagen, ponemos a cero la variable global creada para contar el número de impulsos recibidos, habilitamos la entrada de interrupciones, realizamos un delay de un segundo y volvemos a deshabilitar las interrupciones. Con esto, ya tendríamos el número de vueltas que ha realizado el sensor de caudal, lo siguiente que realizamos es calibrar el sensor con los parámetros correspondientes.

```
float volumenSensor() {
  NbTopsFan = 0;
  sei();
  delay (1000);
  cli();
  Calc = (NbTopsFan * 60 / 5);
  Serial.print (Calc *10, DEC);
  Serial.print (" Litros/hor\r\n");
}

void rpm (){
  NbTopsFan++;
}
```

Figure 50: Módulo sensor caudal.

Ya tendríamos preparado el sensor de caudal, para hacer uso de él debemos llamarlo desde el método encargado de generar la respuesta al servicio REST.

PRESIÓN

Como se ha explicado en apartados anteriores, se realizaron pruebas con el sensor que se adquirió primeramente, pero los resultados no fueron concluyentes, se decidió comprar un sensor de una menor calidad el MPX5010. Es un sensor sencillo de utilizar, como se muestra en la siguiente imagen podemos observar como varía el voltaje en función de la presión obtenida. Por lo tanto, su uso es realizado por los puertos analógicos.

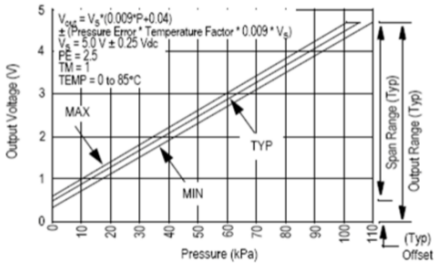


Figure 51: Voltaje salida sensor presión.

En la codificación de dicho sensor no se precisa configuración en el setup, ya que los valores de entrada son por el puerto analógico. Tras analizar el siguiente código, podemos observar como lee del pin analógico cero, le resta el offset de calibración y lo divide por 100 al ser kilo pascales.

```
void preasureSensor(){
  float sensorValue = (analogRead(A0)-SensorOffset)/100.0;
  // print out the value you read:
  Serial.print("Air Pressure: ");
  Serial.print(sensorValue,2);
  Serial.println(" kPa");
}
```

Figure 52: Módulo sensor presión.

Con todo esto, ya estaría explicada la codificación de todo lo referente a la obtención de datos y proceso de la petición REST.

5 INGENIERÍA DEL SOFTWARE

Este capítulo explicará todos los aspectos relativos a la ingeniería del software, explicando todos los casos de uso, especificación de requisitos, diagrama de clase, diagrama de flujo, test funcional, etc. Es uno de los apartados más importantes, ya que acredita una mantenibilidad del sistema para implementaciones futuras indicando al programador todas las funcionalidades afectadas.

5.1 Plan de ejecución

Para alcanzar todos los objetivos propuestos se ha fijado una ejecución por fases de ejecución, ejecutando tareas tales como: lectura de estudio para bases de datos NoSQL, seguimiento de tutoriales Django, administración de sistema, instalación de circuitos, desarrollo de software entre otros. También, se ha usado un control de versiones, GIT. Se ha utilizado un repositorio remoto para tener un backup del código siempre disponible.

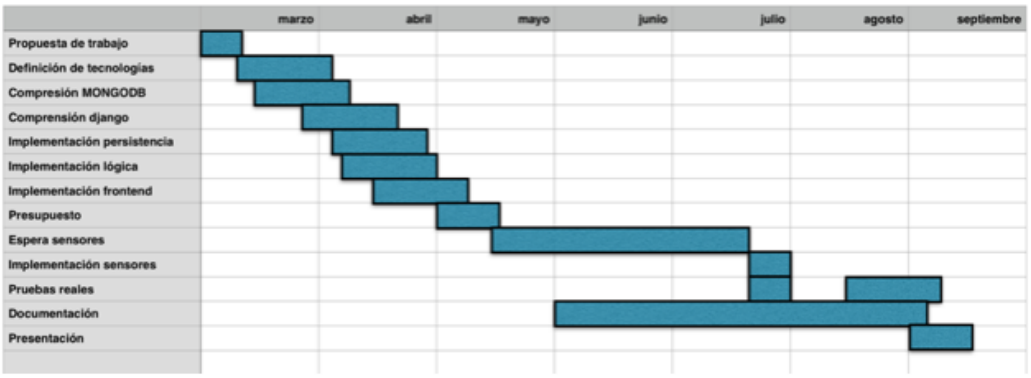


Figure 53: Plan de ejecuion.

Observamos que hay un periodo de espera bastante amplio en la recepción de sensores, la empresa responsable del pedido y envío no se hizo cargo. Se decidió la compra de los sensores a cargo personal por parte del alumno para la finalización del trabajo y de las fases de desarrollo del proyecto.

5.2 Metodología empleada

En la metodología no hubo duda, elegí la que actualmente es la más eficaz y productiva, SCRUM. Es una metodología denominada ágil, definiendo en un periodo de tiempo entre 2 o 3 semanas una serie de objetivos a cumplir denominados sprint, para documentar y desglosar dichos objetivos se ha utilizado la herramienta online Trello. Scrum es una metodología para trabajar en equipo, pero la he adaptado para trabajar solo. En teoría el SCRUM se realiza de la siguiente forma:

- Se realizan entregas parciales del producto final.
- Los requisitos son cambiantes, por lo tanto tendrás que adaptar los objetivos a los nuevos requisitos.
- Identificar los problemas del sprint anterior y poder solventarlos.
- Los sprint suelen ser de 2-4 semanas, en cada iteración del sprint se analizan los resultados obtenidos para una posible entrega al cliente.
- La planificación de los sprint se realiza en dos fases: selección de requisitos a cumplir y estimación de los requisitos.
- Al finalizar cada sprint se realiza una evaluación de las mejoras posibles, cosas a mejorar, cosas que dejar de hacer y hacer menos.

Estos son algunos de los pasos que he realizado en la ejecución de las fases, durante el desarrollo e implementación se ha utilizado también una técnica llamada pomodoro. Con esta se mejora la productividad exponencialmente, dividiendo el tiempo de desarrollo en intervalos de tiempo de 25min. realizando descanso de 5 min. cada ciclo de 4 se toma un descanso de 25 min. En estos 25 min. de trabajo la concentración debe ser máxima evitando todo tipo de distracciones con el exterior.
En menos medida, se ha intentado llevar a cabo una metodología TDD. Es un desarrollo basado en pruebas, realizando primeramente los test unitarios y seguidamente el desarrollo e implementación. Es un cambio bastante grande en el

Table 1: Requesito funcional 1

Número de requisito	1
Nombre de requisito	Pantalla login
Tipo	Interfaz, Restricción: Ninguna
Especificación del requisito	- Dos valores de entrada: user y pass. - Creación botones: 'login' y 'cancel' - Situación de la pantalla: central. - Colores de fondo: blanco - Color botón: rojo. - Color cuadro login: gris
Prioridad del requisito	Alta/Esencial

desarrollo tradicional de software pero trae grandes ventajas para el desarrollo, evita el uso de debugger, añadiendo calidad al desarrollo, permite al programador centrarse en la funcionalidad, validación de requisitos a cumplir, etc. Todo ellos, hacen que este tipo de metodologías sean de un alto valor técnico, mejorando así la calidad en los desarrollos de software.

El uso de metodologías ágiles mejora el proceso tradicional, realizando procesos evolutivos y generando distintas versiones del producto final. Ayuda a mantener el código limpio, con patrones de diseño totalmente adaptados a los requerimientos del sistema.

5.3 Definición de requisitos

Para la definición de requisitos se ha utilizado la plantilla oficial del IEEE830. En este proyecto se definen tres tipos de requisitos: interfaz, funcional y no funcional. Es una parte fundamental en la ingeniería del software, ya que a partir de aquí se desarrollarán todas las especificaciones e implementaciones del sistema.

REQUISITOS DE INTERFAZ

Table 2: Requisito funcional 2

Número de requisito	2
Nombre de requisito	Toolbar superior
Tipo	Interfaz, Restricción: Ninguna
Especificación del requisito	- Contendrá el logotipo de aplicación en la parte izquierda. - Menú desplegable en la parte derecha. - Botón 'logout', opción menú desplegable. - Botón mensaje.
Prioridad del requisito	Alta/Esencial

Table 3: Requisito funcional 3

Número de requisito	3
Nombre de requisito	Menú lateral
Tipo	Interfaz, Restricción: Mostrará las opciones de cada usuario.
Especificación del requisito	- Contendrá todas las opciones disponibles por el usuario. - Navegación a las pantallas principales. - Lista desplegables con las opciones disponibles. - Restricción de seguridad, solamente los admin podrán ver todas las opciones disponibles.
Prioridad del requisito	Alta/Esencial

Table 4: Requisito funcional 4

Número de requisito	4
Nombre de requisito	Footer Inferior
Tipo	Interfaz, Restricción: Ninguna
Especificación del requisito	- Únicamente aparece en la pantalla de inicio. - Contendrá el tipo de licencia usada.
Prioridad del requisito	Baja

Table 5: Requisito funcional 5

Número de requisito	5
Nombre de requisito	Pantalla eventos
Tipo	Interfaz, Restricción: Ninguna
Especificación del requisito	- Pantalla de inicio de la aplicación. - Contendrá dos módulos, uno por cada tipo de evento. - Mostrará los 5 últimos eventos recibidos. - Contendrá el menú lateral, toolbar superior e inferior.
Prioridad del requisito	Baja/Media

Table 6: Requisito funcional 6

Número de requisito	6
Nombre de requisito	Pantalla Sketch
Tipo	Interfaz, Restricción:Acceso únicamente para usuarios admin.
Especificación del requisito	- Opción disponible desde menú lateral. - Caja con nombre ‘drag and drop file here’ - Mostrar el archivo subido con su progresión. - Nombre de la cabecera: ‘upload file’ - Posibilidad de realizar ‘drag and drop’ sobre fichero. - Posibilidad de realizar subidas de manera tradicional.
Prioridad del requisito	Alta

Table 7: Requisito Funcional 7

Número de requisito	7
Nombre de requisito	Pantalla Graphics
Tipo	Interfaz, Restricción:Ninguna.
Especificación del requisito	- 3 Componentes de control para los tres valores: volume, pressure y level. - Parte superior pantalla principal. Colores: Azul, verde y naranja respectivamente. - Gráfica con los datos almacenados de caudal. - Row con botón para inicial un intervalo de medida. - Al realizar click sobre el botón, aparece un pop up con los descripción del intervalo y la fecha de medición. - Botones del pop up: send y clean. - El pulsar escape con el pop up en pantalla desaparece con efecto ‘cortinilla de estrellas’.
Prioridad del requisito	Alta

Table 8: Requisito funcional 8

Número de requisito	8
Nombre de requisito	Pantalla Lista Usuario
Tipo	Interfaz, Restricción:Acceso unicamente usuario admin
Especificación del requisito	- Contendrá todos los usuarios activos en el sistema, visualización mediante tabla. - Posibilidad de eliminar usuarios, pidiendo confirmación. - Posibilidad de actualizar usuarios, mediante formulario.
Prioridad del requisito	Alta

Table 9: Requesito funcional 9

Número de requisito	9
Nombre de requisito	Pantalla Nuevo Usuario
Tipo	Interfaz, Restricción:Acceso unicamente usuario admin
Especificación del requisito	- Contendrá un formulario con todas las características de un usuario. - Botón de confirmación.
Prioridad del requisito	Alta

Table 10: Requisito funcional 10

Número de requisito	10
Nombre de requisito	Login de acceso
Tipo	Funcional, Restricción:Acceso unicamente usuario admin
Especificación del requisito	- Creación de un login de acceso para usuarios activos en la aplicación. - Evitar a usuarios no autorizados el acceso a la plataforma. - Creación de sesiones activas.
Prioridad del requisito	Alta

Estos son todos los requisitos que nuestra interfaz requerirá, por ello, seguidamente se detallarán los requisitos funcionales y no funcionales que debemos cumplir para alcanzar los objetivos necesarios.

REQUISITOS FUNCIONALES

Son los principales que se deben cumplir, ya que marcarán los objetivos establecidos por la aplicación. Para ello, se fijarán una serie de funciones que deben realizar.

Table 11: Requesito funcional 11

Número de requisito	11
Nombre de requisito	Logout
Tipo	Funcional,Restricción:Ninguno
Especificación del requisito	- Creación de un logout, eliminando de sesión el usuario registrado. - Eliminar de sesión todos los parámetros del usuario.
Prioridad del requisito	Alta

Table 12: Requisito funcional 12

Número de requisito	12
Nombre de requisito	Servicio recogida datos arduino
Tipo	Funcional,Restricción:Ninguno
Especificación del requisito	- Creación de servicio para recibir todos los parámetros enviados por el Arduino. - Recibir todos los valores e insertarlos en la base de datos. -Controlar los valores negativos y almacenar los errores en la base de datos.
Prioridad del requisito	Alta

Table 13: Requisito funcional 13

Número de requisito	13
Nombre de requisito	Alta nuevos usuario
Tipo	Funcional,Restricción: Solamente accesible para usuarios admin
Especificación del requisito	- Creación de formulario con todos los datos requeridos. - Servicio de para recoger los datos mediante petición POST Almacenar los datos en base de datos. - Dar privilegios de admin a los usuarios que lo necesiten.
Prioridad del requisito	Alta

Table 14: Requesitio funcional 14

Número de requisito	14
Nombre de requisito	Listar,usuario
Tipo	Funcional,Restricción: Solamente accesible para usuarios admin
Especificación del requisito	- Listado de todos los usuarios activos en base de datos. - Obtener los usuarios de la base de datos - Requerimientos accesibles por petición GET. -Dar privilegios de admin a los usuarios que lo necesiten.
Prioridad del requisito	Alta

Table 15: Requisito funcional 15

Número de requisito	15
Nombre de requisito	Actualizar usuario
Tipo	Funcional,Restricción: Solamente accesible para usuarios admin
Especificación del requisito	- Deberá ser accesible desde la tabla de usuarios. - Después de hacer click, apareceré el formulario de actualización. - Envio de petición mediantes POST. - Actualizar el usuario requerido y modificar la base de datos.
Prioridad del requisito	Alta

Table 16: Requisito funcional 16

Número de requisito	16
Nombre de requisito	Eliminar usuarios.
Tipo	Funcional,Restricción: Solamente accesible para usuarios admin
Especificación del requisito	- Deberá ser accesible desde la tabla de usuarios. - Después de hacer click, aparecerá un pop up de confirmación. - Envío de petición mediantes POST. - Eliminación del usuario requerido y eliminar de la base de datos.
Prioridad del requisito	Alta

Table 17: Requisito funcional 17

Número de requisito	17
Nombre de requisito	Subida sketch.
Tipo	Funcional,Restricción: Solamente accesible para usuarios admin
Especificación del requisito	- Deberá ser accesible desde el menú lateral. - Recogida del fichero enviado por la aplicación web. - Modificación del nombre del fichero y almacenarlo con el nombre por defecto. - Confirmación de subida. - Almacenar el fichero en una ruta por defecto.
Prioridad del requisito	Alta

Table 18: Requesito funcional 18

Número de requisito	18
Nombre de requisito	Visualización de datos de tiempo real.
Tipo	Funcional,Restricción: Ninguna.
Especificación del requisito	- Visualización de todos los datos procedentes del servicio externo Firebase. - Creación de tres módulos para visualización de los datos.
Prioridad del requisito	Media

Table 19: Requisito funcional 19

Número de requisito	19
Nombre de requisito	Visualización de datos de base de datos.
Tipo	Funcional,Restricción: Ninguna.
Especificación del requisito	- Visualización de todos los datos procedentes de base de datos. - Visualización de datos mediante gráficas.
Prioridad del requisito	Media

Table 20: Requisito funcional 20

Número de requisito	20
Nombre de requisito	Creación de intervalos.
Tipo	Funcional,Restricción: Ninguna.
Especificación del requisito	- Creación de intervalos almacenando fecha inicio, fecha fin y descripción. - Envió de una petición POST con los parámetros recibidos. - Almacenar en base de datos el intervalo de medición.
Prioridad del requisito	Media

5.4 Casos de usos

Los casos de uso son son todas las acciones o pasos que deberán hacer los usuarios o actores para completar una acción. Es una secuencia de acciones o intervenciones entre un sistema y sus actores en respuesta a una acción del usuario. Las acciones sirven para indicar el comportamiento del sistema ante las acciones del usuario o los eventos que pueda lanzar el propio sistema. En nuestro caso tenemos tres tipos de actores: user, admin y system.

ACTOR USER

Es el actor que hará uso de la aplicación sin privilegios de administrador, no tendrá las opciones de menú de usuario y sketch, por lo tanto, todas las funcionalidades de esos casos de uso estarán capadas para el actor user. Los siguientes casos de uso serán los que pueda realizar:

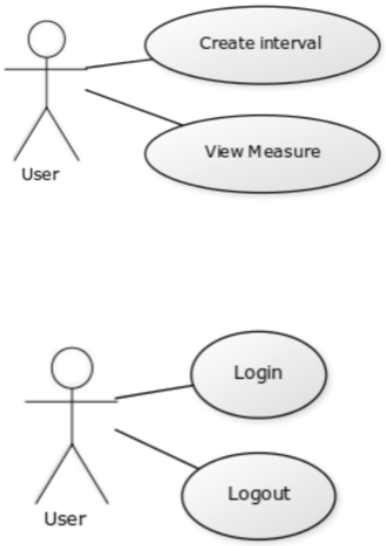


Figure 54: Casos de uso actor User.

Estos son los cuatro casos de uso que podrá hacer el usuario normal, en primera instancia solamente podrá hacer login y logout si el usuario está registrado en la aplicación. Seguidamente, solamente le aparecerán las opciones de crear intervalo y visualización de medidas.

ACTOR ADMIN

Este actor es el usuario avanzado que tendrá la capacidad de controlar todos los privilegios sobre la aplicación y los usuarios de la misma. Tendrá la capacidad de poder modificar todos los permisos y actualizar el sketch cargado en el Arduino, aparte podrá acceder también a todas las funcionalidades que tienen los usuarios normales. Estos son los Casos de uso del usuario admin:

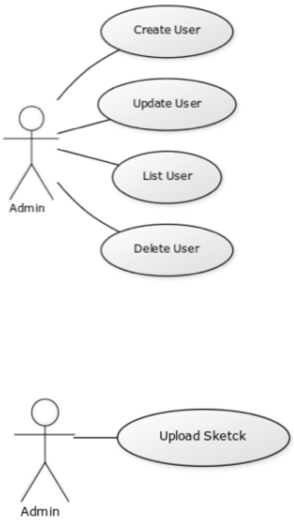


Figure 55: Casos de uso actor Admin.

Como podemos observar, este actor tiene mayor responsabilidad sobre la aplicación. Ofreciendo así la capacidad de administrar completamente la aplicación y sus usuarios.

SYSTEM

El actor system no es un actor humano, ya que es el mismo sistema el encargado de ejecutar los casos de uso. El sistema consta de dos partes: servidor y mota de sensorización. Por esto debemos separar por partes estos dos sistemas. Los siguientes casos de uso muestran los casos de uso a ejecutar por el servidor de aplicaciones:

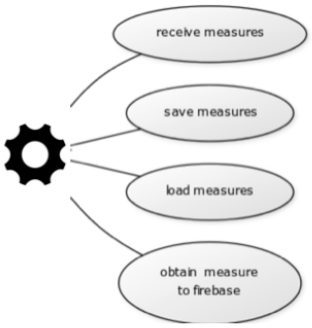


Figure 56: Casos de uso server.

Estas son las cuatro acciones que el servidor debe realizar. También son las principales funciones de la aplicación, por lo que deja con poco margen de maniobra a los dos actores anteriores.

Por otro lado tenemos al Arduino, el YUN es el encargado de enviar los datos obtenidos de los sensores, enviarlo a nuestra plataforma web y a Firebase. Por lo tanto, estos son sus casos de uso:

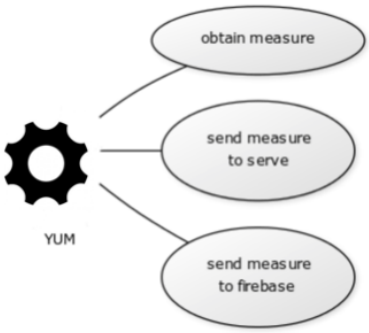


Figure 57: Casos de uso YUN.

Con esto, ya estarían todos los casos de uso.

5.5 Diagrama de clases

Es el principal diagrama de todos, ya que indica toda la estructura y arquitectura software que se ha requerido en el sistema. El principal Lenguaje para visualizar el diagrama de clase se UML[45]. Es el lenguaje de modelado más importante de los sistemas software, es un lenguaje gráfico que permite visualizar, especificar, construir y documentar el sistema completo.

Para comprender mejor el diagrama de clases se procede hacer un diagrama por funcionalidad.

LOGIN

Lo primero que debemos hacer cuando accedemos a la URL de nuestra aplicación web es el login. Para ello el sistema de URL nos envía al view.py y trata la petición. Lo podemos observar en el siguiente diagrama.

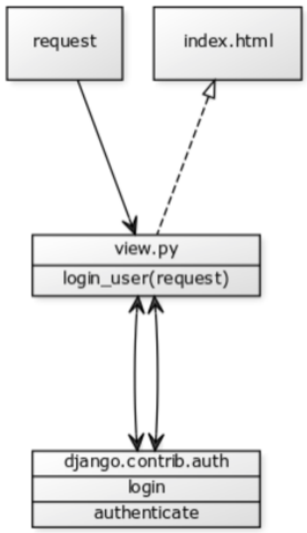


Figure 58: Diagrama clase login.

Como podemos observar en el diagrama, recibimos una petición request que es procesada en el método 'login_use'y en caso de que sea un usuario autorizado retorna al index.html.

[45]https://es.wikipedia.org/wiki/Lenguaje_unificado_de_modelado

LOGOUT

Para realizar el logout realizaremos lo mismos pasos que en el login. El método recibe una petición request, la procesa y envía una redirección. Lo podemos observar en el siguiente diagrama.

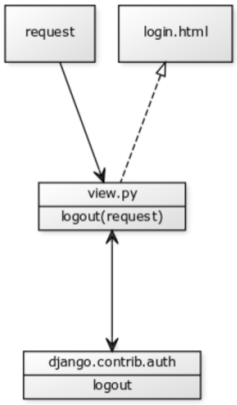


Figure 59: Diagrama clase logout.

CREATE INTERVAL

Esta funcionalidad está accesible desde el menú lateral, los intervalos guardan la hora inicio, hora fin y la descripción del intervalo, para un posterior procesamiento de las medidas. Podemos observar el siguiente diagrama para una mayor comprensión de la funcionalidad.

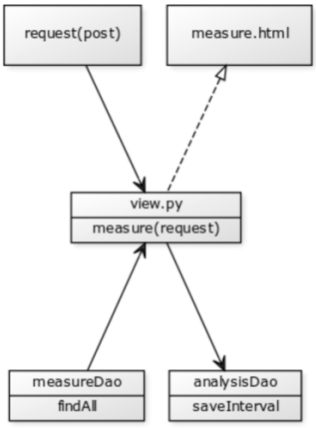


Figure 60: Diagrama clase interval.

Debemos enviar el intervalo en una petición POST, esta captura todos los parámetros enviados y los guarda en la base de datos través de un DAO. Seguidamente, obtiene todas las medidas de la base de datos y retorna una redirección hacía measure.html.

VIEW MEASURE

La funcionalidad está disponible a través del menú lateral, para ellos debemos acceder a través de una petición GET sobre la misma URL que la funcionalidad anterior. Lo podemos observar en detalle en el siguiente diagrama.

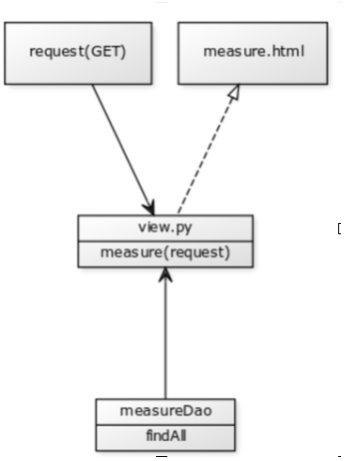


Figure 61: Diagrama clase measure.

CREATE USER

Solamente tendrán acceso los usuarios admin, esta funcionalidad es accesible desde el menú lateral. Debemos rellenar un formulario que será validado con jQuery y enviado seguidamente al servidor mediante una petición request. Lo observamos en el siguiente diagrama.

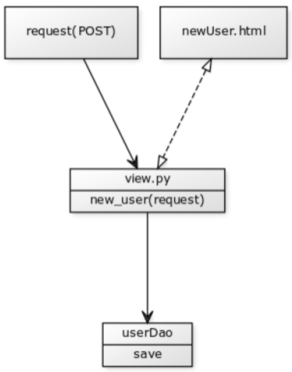


Figure 62: Diagrama clase create user.

UPDATE USER

Funcionalidad disponible desde el menú lateral, permite actualizar un usuario del sistema otorgando privilegios de administrador si fuera necesario, modificando todos los valores que se requieran de un usuario. Observamos todos los detalles en el siguiente diagrama.

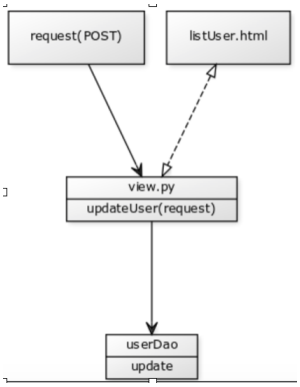


Figure 63: Diagrama clase update user.

LIST USER

Se accede mediante el menú lateral por una petición GET. Lista todos los usuarios almacenados en la plataforma, estén activos o no. Pudiendo desde la misma pantalla editar o eliminar usuarios. Se muestran los detalles en el siguiente diagrama.

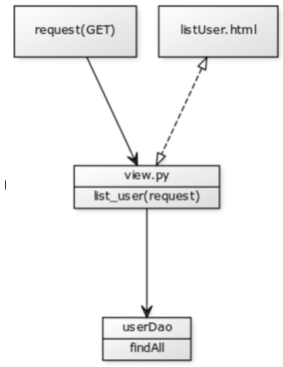


Figure 64: Diagrama clase list user.

DELETE USER

Permite eliminar de forma lógica al usuario, revocando todos los permisos que tengan sobre el sistema. La funcionalidad es accesible desde el listado de usuarios. Se muestra el detalle en el siguiente diagrama.

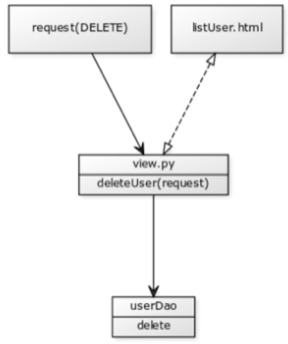


Figure 65: Diagrama clase delete user.

UPLOAD FILE

Permite la subida de los fichero .ino a la plataforma, para que haga su posterior carga en

el Arduino YUN. Esta funcionalidad está disponible desde el menú lateral, se observan todos los detalles en el siguiente diagrama.

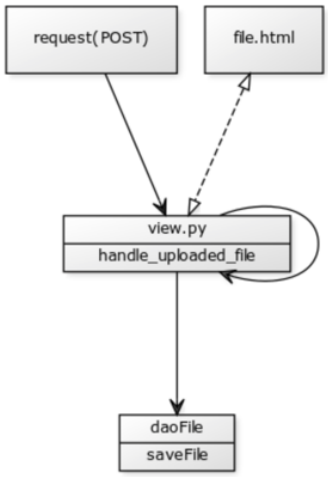


Figure 66: Diagrama clase upload file.

RECEIVE & SAVE MEASURE

Es la principal función del servidor, encargado de tener siempre disponible un servicio encargado de procesar todas las medidas obtenidas por la mota o Arduino YUN. Recibe una petición POST y guarda todos los datos obtenidos de los sensores.

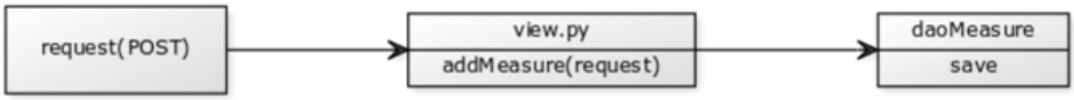


Figure 67: Diagrama clase receive and save measure.

Estas son todos los diagramas que debemos tener en cuenta para un mejor entendimiento de la aplicación web. Mejorando el mantenimiento de la misma y añadiendo calidad al desarrollo de software.

5.6 Diagrama de secuencia

El diagrama de secuencia muestra el flujo de ejecución que debería de tomar para realizar la funcionalidad establecida, incluye también todas las opciones y alternativas que debe tomar. Muestra la interacción de un conjunto de objetos a través del tiempo y cada diagrama representa un caso de uso, contiene los detalles de implementación del escenario y los mensajes del intercambio entre objetos de modelo.

Los siguientes diagramas muestran la implementación de todos los casos de uso con sus opciones de implementación.

LOGIN

El siguiente diagrama muestra la ejecución del login, como podemos observar el usuario o actor ejecutar una petición request contra el servidor, esto lo autentica con otra las librerías de Django y retorna una redirección en caso afirmativo o negativo.

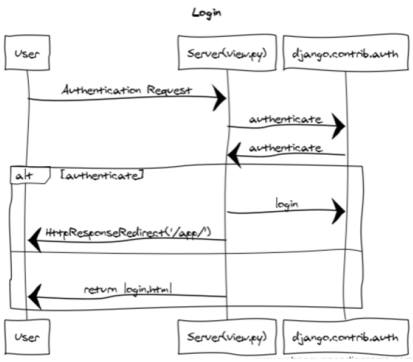


Figure 68: Diagrama secuencia login.

LOGOUT

Tras el login representamos el logout, este diagrama es similar al anterior. El usuario o actor envía una petición request para salir del sistema, éste recoge la petición y realiza el logout contra Django retornando una re-dirección hacia el login.

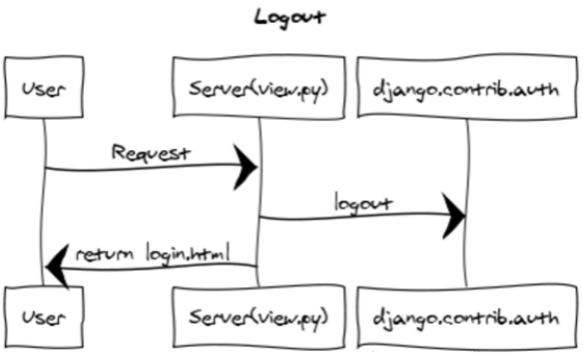


Figure 69: Diagrama secuencia logout.

ADD MEASURE

Es la funcionalidad principal del servidor, permite desde el exterior insertar medidas en la base de datos. Recibe una petición POST con los datos a insertar, procesa los mismos y los inserta en la base de datos en caso de que sean correctos.

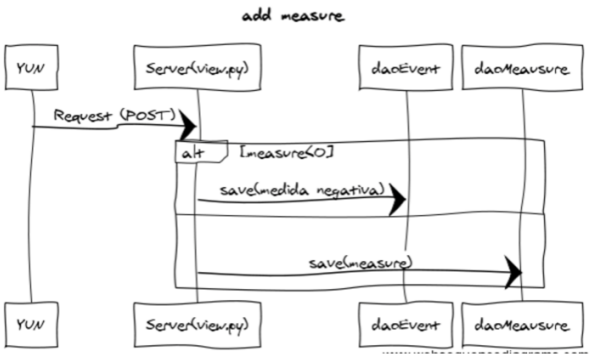


Figure 70: Diagrama secuencia añadir medida.

MEASURE

Funcionalidad accesible desde el menú lateral, permite visualizar todas las medidas insertadas en la base de datos, crear intervalos y visualización de los datos en tiempo real. Sigue el siguiente flujo de ejecución.

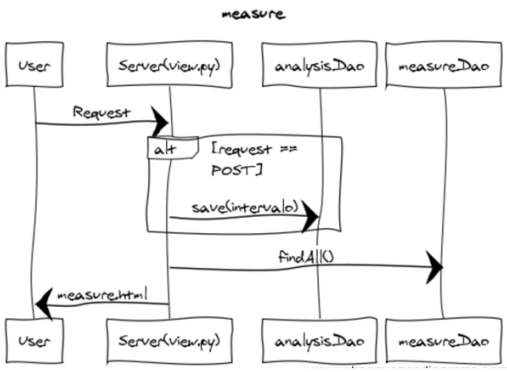


Figure 71: Diagrama secuencia measure.

ADD USER

Funcionalidad accesible desde el menú lateral, mediante una petición GET se obtiene un formulario que, una vez rellenado, se envía por POST sobre la misma URL. Éste identifica si la petición es POST y guarda el usuario en la base de datos.

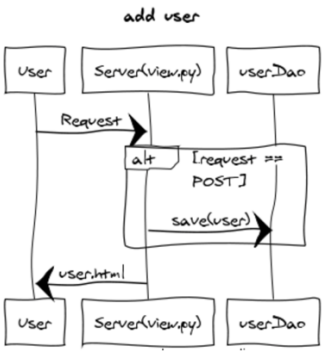


Figure 72: Diagrama secuencia add user.

LIST USERS

Funcionalidad accesible desde el menú lateral, renderiza un HTML después de procesar una petición GET, obteniendo todos los usuarios de la base de datos y pasando el listado a la vista. Se muestra la ejecución en el siguiente diagrama:

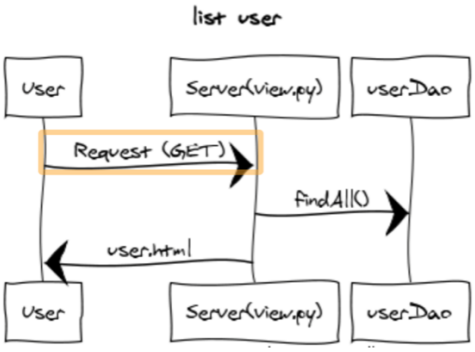


Figure 73: Diagrama secuencia list users.

UPDATE USER

Funcionalidades accesibles desde la vista list user, permite actualizar un usuario elegido modificando todas las características del usuario. Se realiza metiendo una petición PUT, se visualiza el flujo de ejecución en el siguiente diagrama:

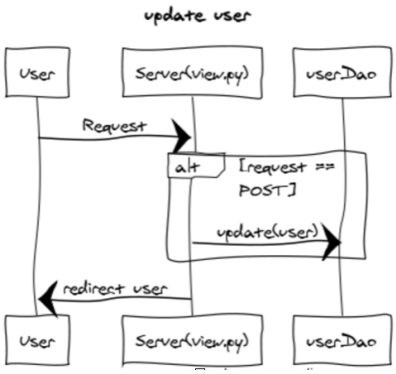


Figure 74: Diagrama secuencia update user.

DELETE USER

Funcionalidad accesible desde la vista de list user, permite borrar un usuario de forma lógica. El usuario envía una petición DELETE al servidor, éste procesa la petición inserta un evento de usuario eliminado y borra el usuario, se puede ver la ejecución en el siguiente diagrama.

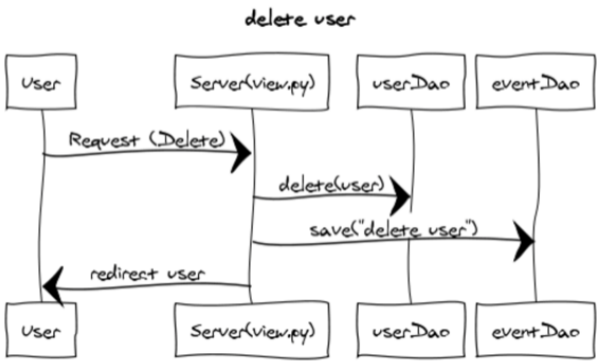


Figure 75: Diagrama secuencia delete user.

UPDATE FILE

Funcionalidad accesible desde el menú lateral, permite la actualización de sketch ejecutado en el Arduino. Se envía el fichero mediante una petición POST, el servidor elimina el fichero antiguo y guarda el fichero nuevo siempre en la misma ruta. Se puede visualizar la ejecución en el siguiente diagrama:

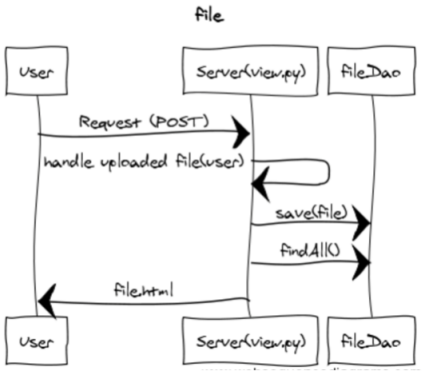


Figure 76: Diagrama secuencia update file.

Con todo esto ya estarían explicados todos los diagramas de secuencia, faltando solamente los procesos ejecutados en el Arduino.

5.7 Pruebas funcionales

Los test funcionales representan una fase importante de la ingeniería, es una prueba basada en la supervisión, ejecución y feedback de todas las funcionalidades diseñadas por la aplicación. Son pruebas específicas y concretas que sirven para probar y validar el software. En nuestro caso se realizarán por cada caso de uso.

NOTA: para la validación de todas la pruebas funcionales se deben completar todos los pasos de prueba.

LOGIN

CASE: 0001	PRE-REQUISITO	ACCIÓN	RESULTADO
	1. Usuario activo en el sistema		
PASO 1		Acceder a la url principal del sistema	Aparece la pantalla del login, contiene 2 input text y un button con nombre 'login'
PASO 2		Insertar un user y pass activo en el sistema y pulsar longin	Re-dirección a la pantalla de index.html

Figure 77: Test funcional login.

LOGOUT

CASE:0002	PRE-REQUISITO	ACCIÓN	RESULTADO
	1. Estar loguedo en el sistema.		
PASO 1		Pulsar sobre el menu desplegable superior derecho, 'logout'	Envio una redirección hacia la pantalla login

Figure 78: Test funcional logout.

CREATE INTERVAL

CASE: 0003	PRE-REQUISITO	ACCIÓN	RESULTADO
	1. Usuario activo en el sistema		
PASO 1		Acceder a la pantalla Graphics, pulsa sobre opción del menu lateral	Carga en pantalla todas los complementos del sistema
PASO 2		Pulsa sobre el botón verde contenido dentro del sector 'buttons'	Aparece un pop up con las opciones: init date, descripción, dos botones (clear y save).
PASO 3		Rellenar los datos pulsando el botón now y una descripción. Pulsar clear	Se vacían los datos del formulario
PASO 4		Rellenar los datos y pulsar el botón save	Desaparece el popup y el icono cambia de colo a naranja con una cruz
PASO 5		Pulsamos sobre el botón	Finaliza el intervalo de medida

Figure 79: Test funcional create interval.

VIEW MEASURE

CASE:0004	PRE-REQUISITO	ACCIÓN	RESULTADO
	1.Usuario activo en el sistema		
PASO 1		Acceder a la pantalla Graphics, pulsa sobre opción del menu lateral	Acceder a la pantalla Graphics, pulsa sobre opción del menu lateral
PASO 2		Aparece en el componente central todas las medidas insertadas	

Figure 80: Test funcional view measure.

ADD USER

CASE:0005	PRE-REQUISITO	ACCIÓN	RESULTADO
	1.Usuario admin activo en el sistema		
PASO 1		Acceder a la pantalla Users-> créate user, pulsa sobre opción del menu lateral	Acceder a la pantalla créate user con todas las opciones input text diponibles.
PASO 2		Rellenar todos los campos modificando los datos de pass y repeat pass	Pulsar submit, aparece un mensaje de error 'please enter the same value again'
PASO 3		Rellenar todos los campos requeridos	Pulser submit

Figure 81: Test funcional add user.

LIST USERS

CASE:0006	PRE-REQUISITO	ACCIÓN	RESULTADO
	1.Usuario admin activo en el sistema		
PASO 1		Acceder a la pantalla Users-> list user, pulsa sobre opción del menu lateral	Acceder a la pantalla créate user con todas las opciones input text diponibles.
PASO 2		Aparece una tabla con los usuario activos	

Figure 82: Test funcional list users.

UPDATE USER

CASE:0006	PRE-REQUISITO	ACCIÓN	RESULTADO
	1.Usuario admin activo en el sistema		
PASO 1		Acceder a la pantalla Users-> list user, pulsa sobre opción del menu lateral	Acceder a la pantalla créate user con todas las opciones input text diponibles.
PASO 2		Seleccionar un usuario de la tabla, sobre la columna update, pulsando el botón con el icono del lápiz	Aparece un formulario con los datos de usuario en el placeholder
PASO 3		Rellenar los campos del formulario y pulsar submit	Aparece el usuario modificado en la tabla

Figure 83: Test funcional update user.

DELETE USER

CASE:0006	PRE-REQUISITO	ACCIÓN	RESULTADO
	1.Usuario admin activo en el sistema		
PASO 1		Acceder a la pantalla Users-> list user, pulsa sobre opción del menu lateral	Acceder a la pantalla todo los usuarios activos o no en el sistema
PASO 2		Seleccionar un usuario de la tabla, sobre la columna DELETE , pulsando el botón con el icono papelera	Aparece un pop up de confirmación de eliminación.
PASO 3		Pulsamos yes	Aparece el usuario con el icono activo a false

Figure 84: Test funcional delete user.

UPLOAD FILE

CASE:0006	PRE-REQUISITO	ACCIÓN	RESULTADO
	1.Usuario admin activo en el sistema		
PASO 1		Acceder a la pantalla Sketch, pulsa sobre opción del menu lateral	Aparece el componente para subida de archivos
PASO 2		Realizar drap & drop con el fichero sketch.ino	Aparece el elemento de carga por porcentaje con el nombre del fichero
PASO 3		Finaliza la carga sin error	

Figure 85: Test funcional upload file.

6 RESULTADOS Y DISCUSIÓN

Las primeras muestras obtenidas en el laboratorio se consiguió una monitorización por puerto serial. Lo primero que conseguimos sensorizar fue el caudal, conectamos el sensor de caudal a la entrada de flujo del circuito tomando como referencia en antiguo indicador de caudal, tras varios intentos conseguimos visualizar las medidas por el puerto serie, como se muestra en la imagen siguiente.

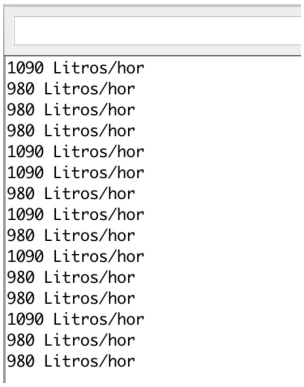


Figure 86: Visualización medidas puerto serie.

Tras visualizar las medidas por el puerto serie procedemos a la inserción en nuestra base de datos a través del servicio. Vemos que se ha realizado correctamente en la siguiente imagen.

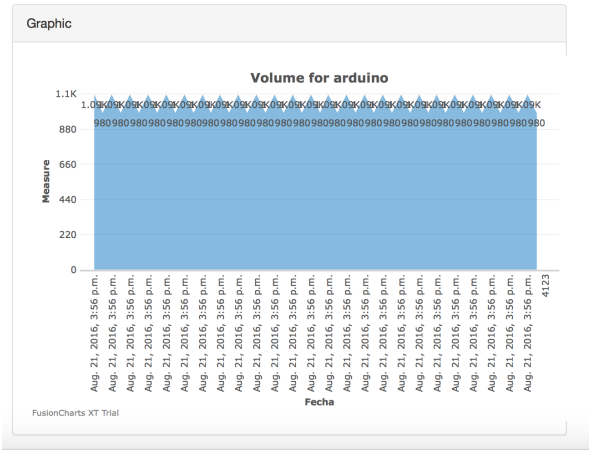


Figure 87: Medidas en MongoDB.

La visualización de medidas de caudal fue relativamente sencillo, lo siguiente que intentamos obtener fueron las medidas de presión. Utilizamos el primer sensor adquirido pero nos dimos cuenta de que la capacidad de este sensor era bastante superior a la ofrecida por el circuito. Como indicamos en apartados anterior utilizamos el sensor "0-1.2 MPa G1/4" Sensor de presión de líquidos y gases. Tras ver las características técnicas del sensor la tolerancia del sensor era la capacidad máxima del circuito, introdujimos presión al circuito y no ofrecía variación de voltaje. En esta fase de proyecto se decidió la adquisición de un segundo sensor de presión de menor margen de actuación, adquirimos el sensor MPX5010 como indicamos en el apartado anterior, este sensor es usado para medir presiones de aire y fluidos, lo instalamos en el circuito como se muestra en la siguiente imagen.

Figure 88: Instalación sensor presión.

Tras la instalación del sensor, procedemos a hacer la toma de medidas por puerto serie. Las primeras medidas fueron algo algo inestables ya que el rango de medida era bastante menor del esperado. Lo primero que debemos hacer para realizar una toma de medidas correctas es la carga del circuito, para ello debemos sacar todo el aire del circuito, las medidas tomadas en la carga es la siguiente.

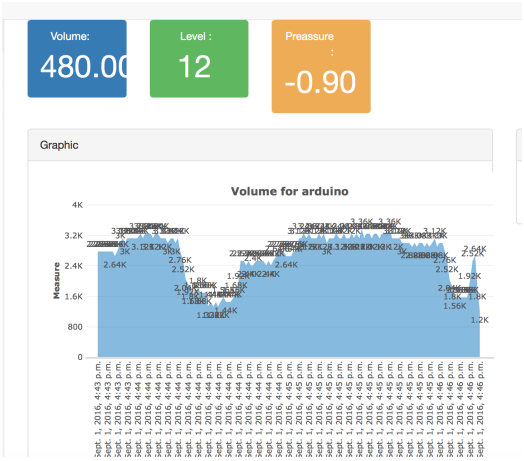


Figure 89: Carga circuito.

Tras la carga se decide hace una monitorización en tres puntos del sistema: salida de rotámetro, entrada rotrametro y inicio del circuito. Para ver el correcto funcionamiento de los sensores se decide tomar medidas en intervalos de caudal en 250, 500, 1000, 1500 y 2000. Analizando así la diferencia de presión en los diferentes puntos de medida.

SALIDA ROTÁMETRO

Instalamos inicialmente el sensor en la parte superior del rotametro, como se aprecia en la siguiente imagen.

Figure 90: Instalación sensor salida rotametro.

Las medidas son tomadas por puerto serie y seguidamente son insertadas en base de datos. Las siguientes gráficas muestran los datos obtenidos de caudal y presión.

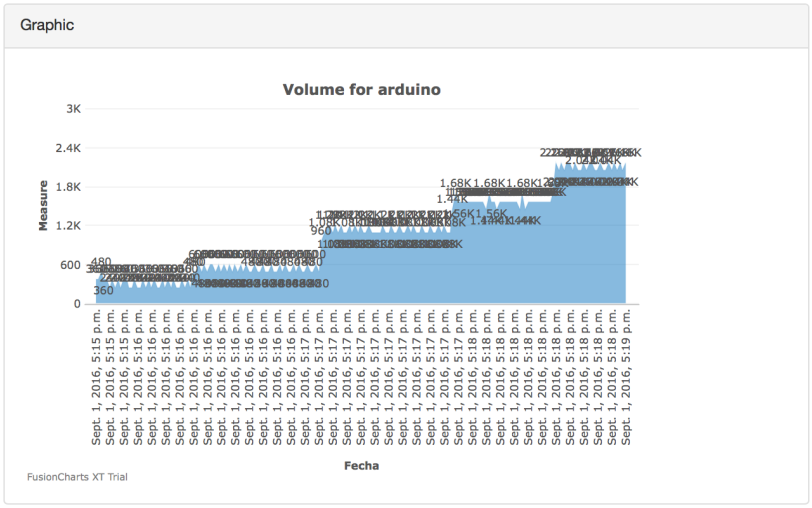


Figure 91: Caudal salida rotámetro.

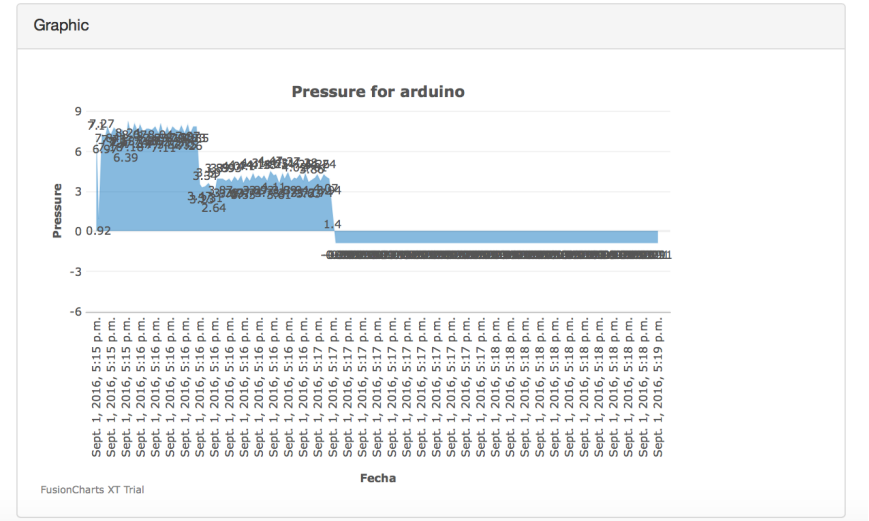


Figure 92: Presion salida rotámetro.

Como podemos apreciar en las imágenes, las medidas muestran un descenso de la presión según va aumentando el caudal mostrando así un rango de utilización entre los 250L/H hasta los 1000L/H, registra un descanso drástico de la presión obtenida cada

vez que aumenta el caudal del circuito.

ENTRADA ROTÁMETRO

En la entrada del rotámetro las medidas de presión deberían de ser superiores que las recogidas en el apartado anterior. En primer lugar, instalamos el sensor en el punto de muestra, lo podemos ver en la siguiente imagen.

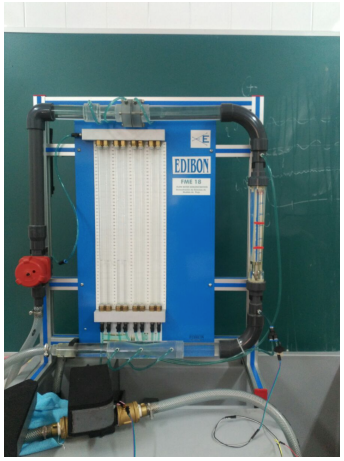


Figure 93: Instalación entrada rotámetro.

Tras las instalación iniciamos el circuito e iniciamos la toma de medidas y visualizamos los datos obtenidos, en las siguientes imágenes se puede ver el resultado.

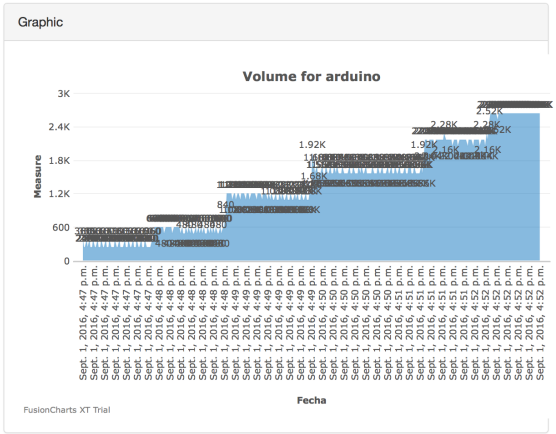


Figure 94: Caudal entrada rotámetro.

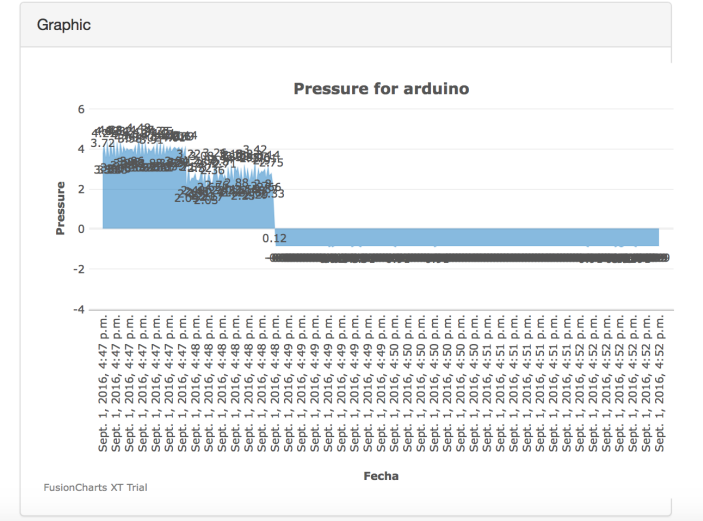


Figure 95: Presión entrada rotámetro.

Como podemos observar las medidas de presión difieren de las tomadas del apartado anterior, la resistencia obtenida por el sensor es bastante menor pasando de 7 a 3-4, observamos también que el rango de medidas es similar al anterior entre los 250L/H y los 1000L/H, las mediciones registran los mismos descenso de presión cuando se aumenta el caudal en el circuito.

INICIO CIRCUITO

El sensor se instala en la parte inicial del circuto, debería ser en teoría donde mayor presión se obtenga. Instalamos el sensor en la entrada como se muestra en la siguiente imagen.

Figure 96: Instalación sensor inicio.

Tras la instalación iniciamos la toma de medidas, visualizamos todos los datos obtenidos en las siguientes gráficas.

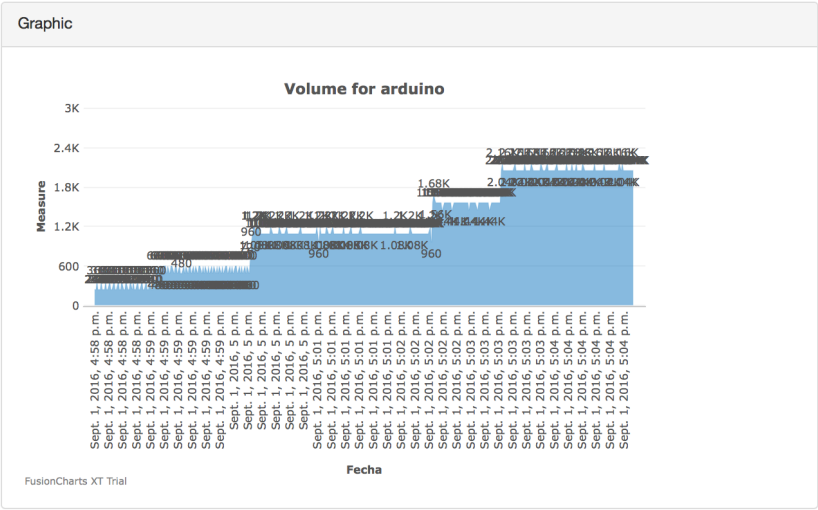


Figure 97: Caudal sensor inicio.

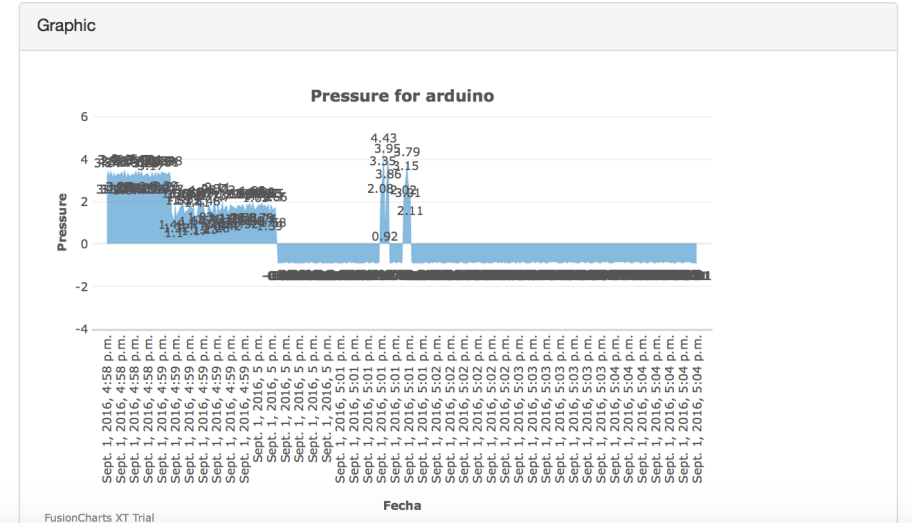


Figure 98: Presión sensor inicio.

Observamos como ha disminuido ligeramente la presión del sensor frente al instalado en la parte inferior del rotámetro, se ha quedado fijado en unos 3,5 de media frente al 3,9. El rango de utilización es similar a los anteriores entre 250L/H y los 1000L/H.

NOTA: para realizar todas las tomas correctamente se han elimando en cada prueba todas las medidas de base de datos anteriormente almacenadas.

Aparte de las gráficas en en la parte superior de la pantalla se pueden visualizar los datos en tiempo real, gracias a Firebase. Como podemos ver en la siguiente imagen se visualizan los datos sin tener que realizar una recargar de pantalla.

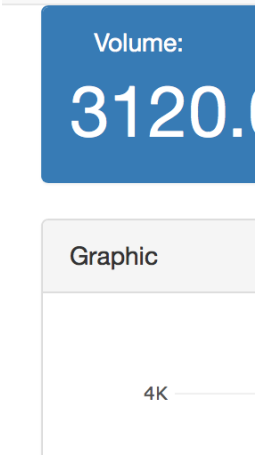


Figure 99: Tiempo real sensor caudal.

7 CONCLUSIONES Y LÍNEAS FUTURAS

En el siglo XXI el mundo se enfrenta a varios problemas: cambio climático, cambios sociales, de consumo, abastecimientos, hambrunas, sequías, etc. La tecnología, y en especial el análisis de datos, puede jugar un papel importante para afrontar tales problemas. Este proyecto es un pequeño grano de arena para ayudar a resolver dichos problemas y conseguir hacer un análisis predictivo en medición de flujos de agua, llegando a ahorrar en consumo de electricidad, agua, mantenimiento, etc. El uso de software libre está teniendo su mayor auge actualmente y está empezando a calar de forma drástica en las empresas y entidades públicas, por ello desde un primer momento se ha descartado el uso de cualquier lenguaje privado o tecnología con copyrigth. Todo proceso de desarrollo ha sido un gran reto, ya que no conocía ninguna tecnología utilizada y no había ningún precedente, ni proyecto base. Afronte el reto que me propuso Pablo y fuí tomando los objetivos necesarios para alcanzar los plazos previstos y funcionalidades requeridad.

Por un lado, los sistemas NoSQL han marcado un antes y un después en el almacenamiento de datos, pudiendo llegar a crear sistemas nunca antes conocido debido a las limitaciones de las bases de datos tradicionales, para realizar toda la fase de persistencia de datos tuve que hacer un estudio previo de su funcionamiento y su potencia frente a las bases de datos tradicionales, ya que eran totalmente desconocidas para mí.

Seguidamente, la potencia y sencillez de Django con Python posibilita que el desarrollo de este tipo de proyecto se realize de forma rápida y dinámica, actualmente Django posee una de las curvas de aprendizaje más altas de todos los framework conocidos para el desarrollo web, esto unido al uso de Python hace que el control de bug sea mínimo en todo el desarrollo.

LÍNEAS FUTURAS

Este proyecto puede ser el inicio de una interesante línea de investigación, no sólo dentro del ámbito hidráulico sino dentro de la Escuela Politécnica. El desarrollo fue orientado para dispositivos Arduino y por lo tanto, hace de esta plataforma algo único dentro de la Escuela Politécnica. Tras el desarrollo de toda la plataforma, surgen varias líneas interesantes que cabe destacar:

- Desarrollo de dispositivos móviles: al usar la plataforma externa de Firebase tenemos acceso directamente desde cualquier dispositivo, ya que esta plataforma está diseñada para ser multiplataforma.

- Escalabilidad del sistema: para el desarrollo de este proyecto se ha utilizado únicamente una base de datos mongo sin replicación, el uso de bases de datos replicadas hace que su potencia ante análisis de datos masivos se realice en la mitad de tiempo.

- Red de dispositivos: para el desarrollo del proyecto hemos utilizado únicamente un Arduino YUN. En un futuro, cuando el sistema escale, se hará uso de más dispositivos conectados y, por lo tanto, tendremos que habilitar la plataforma para tal uso, aunque ya está orientada para tal uso.

- Red de actuadores: la potencia de Arduino no solamente radica en su monitorización, también tiene la posibilidad de poder actuar dependiendo de la información extraída.

AGRADECIMIENTOS

Lo primero que debo hacer en este apartado es agradecer a Pablo García por hacer posible este proyecto. También quiero agradecer a ese grupo de amigos que hemos estado juntos desde primero, solemos llamarnos LBDP (la banda del patio) y en especial a Daniel Llanos, mi compañero de batallas en esta travesía.

Mi agradecimiento también a Pablo Durán por aguantarme en el laboratorio y ayudarme a instalar todos los sistemas de medida.

Y por último, a mi familia. GRACIAS por apoyarme todos estos años en mi etapa en la universidad.

8 BIBLIOGRAFÍA

INTRODUCCIÓN

- IoT in five days - Antonio Linan Colina and Alvaro Vives and Antoine Bagula and Marco Zennaro and Ermanno Pietrosemoli - (Junio, 2016).

ESTADO DEL ARTE

- SCADA: Supervisory Control and Data Acquisition - (Julio, 2016).
- WIKI: https://es.wikipedia.org/wiki/SCADA - (Julio, 2016).
- Universidad de Cordoba. http://www.uco.es/investiga/grupos/eatco/automatica/ihm/descargar/scada.pdf - (Julio, 2016).

PERSISTENCIA

- The Little MongoDB Book -28 Mar 2011 - By Karl Seguin - (Marzo, 2016).
- Extracting Data from NoSQL Databases - Enero 2012- PETTER NÄSHOLM - (Marzo, 2016).
- MongoDB: The Definitive Guide, 2nd Edition, Kristina Chodorow - (Marzo, 2016).
- MongoDB in Action (Abril, 2016).
- WIKI - https://es.wikipedia.org/wiki/MongoDB
- Documentación ofical de Mongo - https://docs.mongodb.com - (Marzo, 2016).

DJANGO

- The Book Django http://www.djangobook.com/en/2.0/index.html - (Marzo, 2016).

- Django Documentation - https://media.readthedocs.org/pdf/django/1.9.x/django.pdf -(Abril, 2016).
- Two Scoops of Django: Best Practices for Django 1.8 - (Abril, 2016).
- HIGH PERFORMANCE DJANGO -Peter Baumgartner y Yann Malet - (Mayo, 2016).
- Documentación oficial - https://docs.djangoproject.com/en/1.10/

JAVASCRIPT

- ng-book 2, Ari Lerner - (Abril, 2016).
- Angular for the jQuery developer - http://www.ng-newsletter.com/posts/angular-for-the-jquery-developer.html
- Documentación oficial - https://docs.angularjs.org/tutorial - (Mayo, 2016).
- AngularJS in 60 Minutes - Dan Wahlin - (Abril, 2016).
- Mastering AngularJS Directives -Pascal Precht - (Mayo, 2016).
- jQuery Fundamentals - Rebecca Murphy - (Mayo, 2016).

ESTILO

- Twitter Bootstrap 3 Succinctly - Peter Shaw - (Mayo, 2016).

ARDUINO

- Comparación Arduino - http://comohacer.eu/analisis-comparativo-placas-arduino-oficiales-compatibles/ - (Junio, 2016).
- Documentación oficial - https://www.arduino.cc.

TRABAJO FIN DE GRADO

- Vecino, A. SmartPolitech: Soporte del almacenamiento de datos mediante Apache Cassandra. Septiembre, 2014.

9 Anexos

MANUAL DE USUARIO

AÑADIR USUARIO

Lo primero que debemos hacer para dar de alta un nuevo usuario es acceder a la aplicación, en el menú lateral pulsamos sobre la opción User-create user.

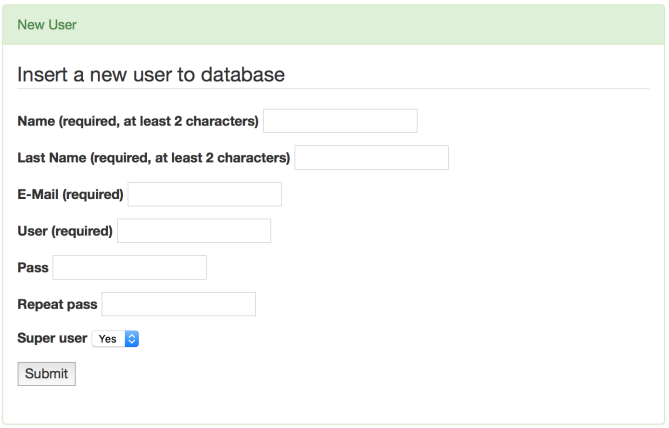


Figure 100: Manual usuario, new user.

Rellenamos el formulario con los datos del nuevo usuario y pulsamos submit.

ACTUALIZAR USUARIO

Accedemos al menú lateral de la aplicación, sección Users-list users. Aparecerá una tabla con todos los usuarios disponibles, en una columna de la tabla aparece un lápiz. Haremos click sobre el usuario a modificar y aparecerá un formulario en la parte inferior de la tabla, podemos observar el resultado en la siguiente imagen.

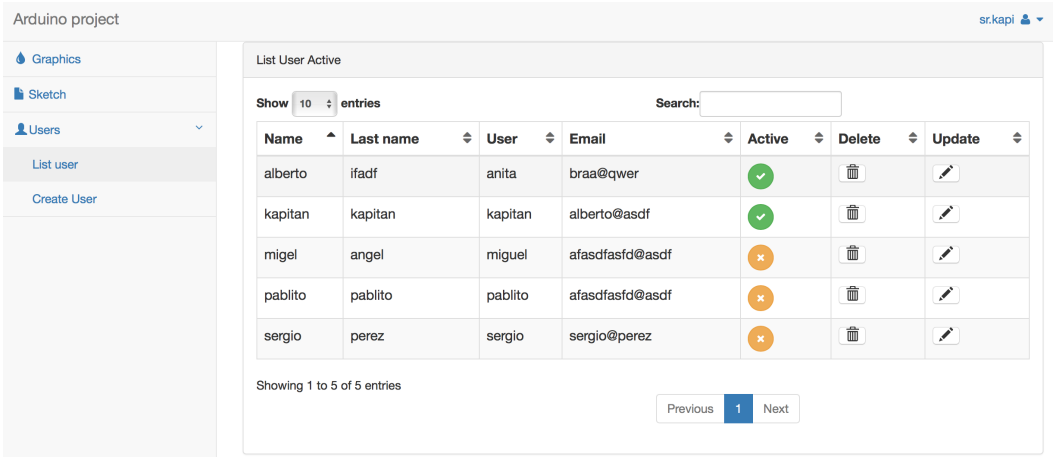


Figure 101: Manual usuario, list users.

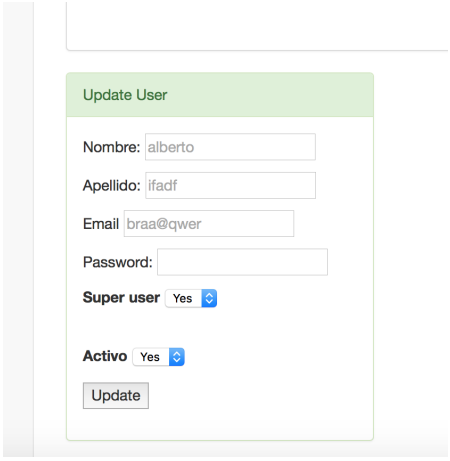


Figure 102: Manual usuario, list users.

BORRAR USUARIOS

Accedemos al menú lateral de la aplicación, sección Users-list users. Aparecerá una tabla con todos los usuarios disponibles, en una columna de la tabla aparece una papelera. Al hacer click sobre ella nos aparece un pop up preguntando que deseamos eliminar el usuario, podemos ver el resultado en la siguiente imagen.

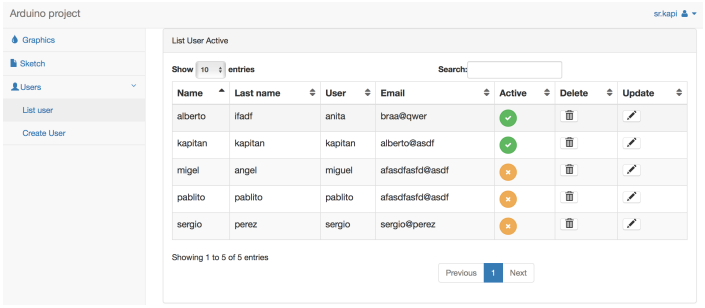


Figure 103: Manual usuario, list users.

VER MEDIDAS & CREAR INTERVALOS

La visualización de medidas y la creación de intervalos es la principal funcionalidad, disponible para todos los usuarios dados de alta en la aplicación. Está disponible desde el menú lateral, opción “Measure”. Al hacer click sobre la opción aparecen dos gráficas, caudal en la parte superior y presión en la inferior. Para crear intervalos debemos hacer click sobre el icono circular verde, nos aparece un pop up con dos opciones: fecha inicial y descripción, rellenamos las opciones disponibles y pulsaremos ‘send’. El icono que anteriormente estaba verde ahora es naranja, haremos click sobre él si deseamos terminar el intervalo y guardarlo en la aplicación. Podemos ver la funcionalidad en las siguientes imágenes.

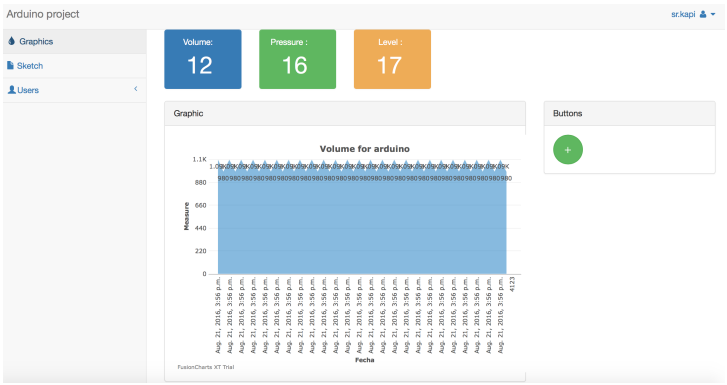


Figure 104: Manual usuario, measure.

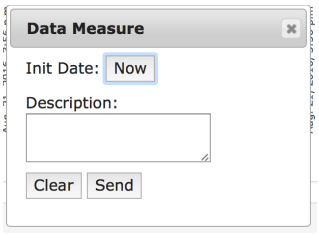


Figure 105: Manual usuario, pop up.

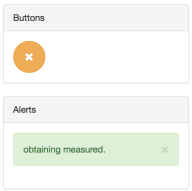


Figure 106: Manual usuario, botón finalizar.

ACTUALIZAR SKETCH

Lo primero que debemos hacer para actualizar el sketch es generar un fichero '.hex', para ello accedemos al IDE de arduino y generamos el fichero como binario compilado, pulsamos sobre la opción: programa-exportar binarios compilados. Una vez obtenido el binario accedemos a la aplicación, dentro del menú lateral opción 'Sketck' nos aparecerá una ventana donde realizaremos 'drap & drop' para subir el fichero a la plataforma web. Se puede visualizar la secuencia en las siguientes imágenes.

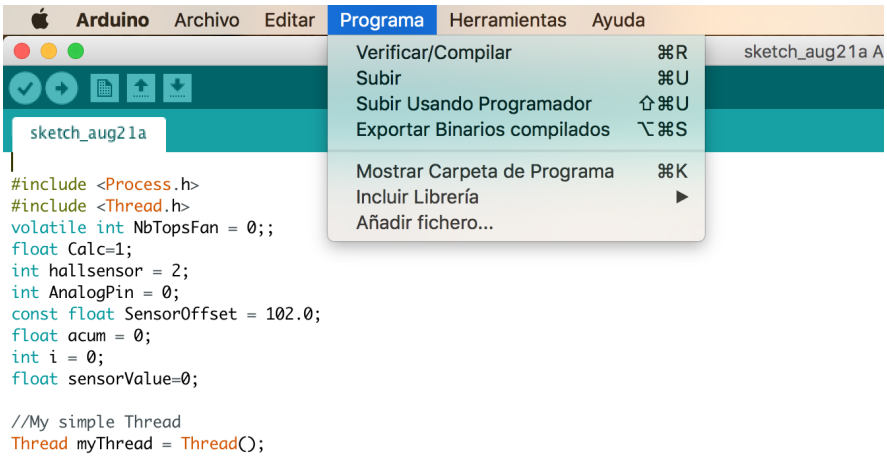


Figure 107: Manual usuario, obtención binario.

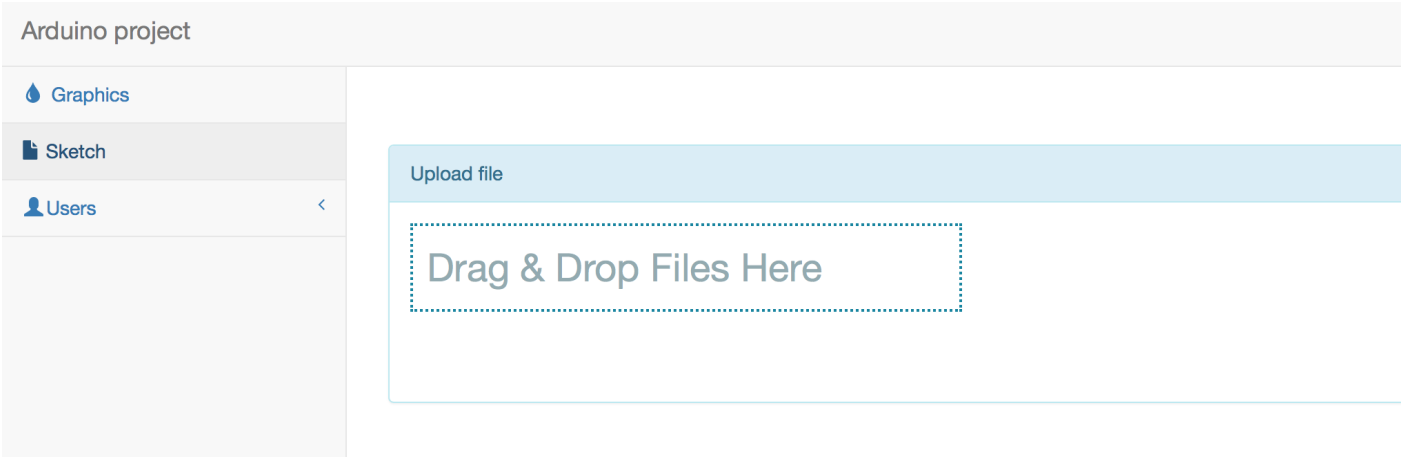


Figure 108: Manual usuario, sketch.