



UNIVERSIDAD DE BURGOS  
ESCUELA POLITÉCNICA SUPERIOR  
Grado en Ingeniería en Informática



TFG del Grado en Ingeniería Informática

**Aplicación web para la  
visualización y búsqueda de datos  
almacenados en dispositivo con  
linux embebido.**



Presentado por Rodrigo Martínez Bravo  
en Universidad de Burgos — 2 de julio de 2017  
Tutor: Jesús Enrique Sierra García

---

# Índice general

---

Índice general	I
Índice de figuras	IV
<b>Apéndice A Plan de Proyecto Software</b>	<b>1</b>
A.1. Introducción	1
A.2. Planificación temporal	1
Sprint 1: 01/02 – 13/02	1
Sprint 2: 23/02-20/02	2
Sprint 3: 20/02-27/02	2
Sprint 4: 27/02-13/3	3
Sprint 5: 13/03-03/04	3
Sprint 6: 03/04-10/04	4
Sprint 7: 10/04-17/04	5
Sprint 8: 17/04-1/05	6
Sprint 9: 1/05-20/05	6
Sprint 10: 20/05-2/07	7
A.3. Estudio de viabilidad	7
Viabilidad económica	7
Viabilidad legal	10
<b>Apéndice B Especificación de Requisitos</b>	<b>11</b>
B.1. Introducción	11
B.2. Objetivos generales	11
Traductor de tramas a variables informativas	11
Visualización de variables	11
Cursores	12
Manejo de ventana temporal	13
Diezmado de datos	13

Filtraje de datos . . . . .	13
Búsqueda de datos . . . . .	13
Vistas . . . . .	14
B.3. Catalogo de requisitos . . . . .	14
Requisitos funcionales . . . . .	14
Requisitos no funcionales. . . . .	14
B.4. Especificación de requisitos . . . . .	15
B.5. Casos de uso . . . . .	23
<b>Apéndice C Especificación de diseño</b>	<b>24</b>
C.1. Introducción . . . . .	24
C.2. Diseño de datos . . . . .	24
Configuración . . . . .	24
Biblioteca . . . . .	25
Cargador de ficheros . . . . .	26
C.3. Diseño procedimental . . . . .	26
Comportamientos de la tabla y botones asociadas. . . . .	26
Interfaz gráfica de las vistas gráficas. . . . .	27
Búsqueda y filtrados. . . . .	27
C.4. Diseño arquitectónico . . . . .	28
<b>Apéndice D Documentación técnica de programación</b>	<b>29</b>
D.1. Introducción . . . . .	29
D.2. Estructura de directorios . . . . .	29
Carpeta raíz. . . . .	29
Datalogger . . . . .	30
CSS . . . . .	30
JS . . . . .	30
D.3. Manual del programador . . . . .	31
PhpStorm . . . . .	31
Git . . . . .	34
D.4. Compilación, instalación y ejecución del proyecto . . . . .	36
Realizar cambios en el repositorio. . . . .	44
Para ejecutar el proyecto . . . . .	47
D.5. Pruebas del sistema . . . . .	47
<b>Apéndice E Documentación de usuario</b>	<b>49</b>
E.1. Introducción . . . . .	49
E.2. Requisitos de usuarios . . . . .	49
E.3. Instalación . . . . .	49
E.4. Manual del usuario . . . . .	49
Pantalla de inicio . . . . .	49
Fast Graph . . . . .	53
Añadir y modificar vistas . . . . .	55

*ÍNDICE GENERAL* III

Importar y exportar vistas. . . . .	58
Creación de gráficas sin Fast Graph. . . . .	60
Configuración del XML. . . . .	61
Cursores. . . . .	63
Utilizar la ventana temporal. . . . .	64
Utilizar búsquedas y filtros . . . . .	66

**Bibliografía** **70**

---

# Índice de figuras

---

A.1. El precio que se nos ofreció para usarlo en hardware fue de 80 euros al año por dispositivo. . . . .	10
B.1. Requisito funcional 1 . . . . .	15
B.2. Requisito funcional 2 . . . . .	16
B.3. Requisito funcional 3 . . . . .	17
B.4. Requisito funcional 4 . . . . .	18
B.5. Requisito funcional 5 . . . . .	19
B.6. Requisito funcional 6 . . . . .	20
B.7. Requisito funcional 7 . . . . .	21
B.8. Requisito funcional 8 . . . . .	22
B.9. Requisito funcional 9 . . . . .	22
B.10. Casos de uso del proyecto . . . . .	23
C.1. Diagrama de clases de configuración . . . . .	25
C.2. Diagrama de clases de biblioteca . . . . .	25
C.3. Diagrama de clases del fileloader . . . . .	26
C.4. Diagrama clases tabla y botones asociadas . . . . .	27
C.5. Diagrama de clases vistas gráficas . . . . .	27
C.6. Diagrama clases Búsqueda y filtrados . . . . .	28
C.7. Diagrama clases general . . . . .	28
D.1. Pagina oficial de PHPStorm . . . . .	32
D.2. Paso 1 de la instalación . . . . .	32
D.3. Paso 2 de la instalación . . . . .	33
D.4. Paso 3 de la instalación . . . . .	33
D.5. Página de descarga de git . . . . .	34
D.6. Paso 1 de instalación de git . . . . .	34
D.7. Paso 2 de instalación de git . . . . .	35
D.8. Paso 3 de instalación de git . . . . .	35

D.9. Paso 4 de instalación de git . . . . .	36
D.10. Primera imagen al abrir la aplicación . . . . .	36
D.11. Carga de PHPStorm . . . . .	37
D.12. Carga de repositorio en PHPStorm. . . . .	38
D.13. Carga de repositorio en PHPStorm. . . . .	38
D.14. Advertencia de carga repositorio. . . . .	39
D.15. Selección interprete PHP . . . . .	40
D.16. Carpeta con contenido de PHP 5.6 . . . . .	41
D.17. Selección del interprete CLI. Paso 1 . . . . .	42
D.18. Selección del interprete CLI. Paso 2 . . . . .	42
D.19. Selección del interprete CLI. Paso 3 . . . . .	43
D.20. Aspecto final configuración del proyecto PHPStorm. . . . .	44
D.21. Vista general desde PHPStorm. VCS. . . . .	45
D.22. Progreso barra control de versiones . . . . .	45
D.23. Pantalla de login de gitHub . . . . .	46
D.24. Mensaje de éxito en VCS . . . . .	46
D.25. Commit en github del nuevo commit realizado desde github . . . .	47
D.26. Commit en github del nuevo commit realizado desde github . . . .	47
E.1. Pantalla de inicio del proyecto . . . . .	50
E.2. Selección de archivos . . . . .	51
E.3. Salir pantalla de inicio . . . . .	52
E.4. Pantalla de carga . . . . .	53
E.5. Pantalla Fast Graph . . . . .	53
E.6. Pantalla de confirmación . . . . .	54
E.7. Pantalla de confirmación . . . . .	54
E.8. Opciones de variable . . . . .	55
E.9. Cambio de nombre de variable. . . . .	56
E.10. Creación de nueva vista de variable. . . . .	56
E.11. Resultado creación variables. . . . .	57
E.12. Cambio de autoajuste de ejes. . . . .	58
E.13. Resultado quitando una variable. . . . .	58
E.14. Opción de exportar vista. . . . .	59
E.15. Opción de importar vista. . . . .	59
E.16. Solución al importar una vista. . . . .	60
E.17. Opciones gráficos. . . . .	60
E.18. Creación variable XY. . . . .	61
E.19. Solución gráfica XY. . . . .	61
E.20. Cambiar XML. . . . .	62
E.21. Único resultado en variables. . . . .	62
E.22. Resultado con único resultado. . . . .	63
E.23. Resultado con único resultado. . . . .	64
E.24. Ventana temporal: en rojo zoom izquierdo, azul derecho. . . . .	64
E.25. Zoom con ventana temporal. . . . .	65

E.26. Modo reproducción. . . . .	66
E.27. Filtro en la parte inferior. . . . .	67
E.28. Búsqueda con zoom automático. . . . .	68
E.29. Asignación cursores con el cursor de búsqueda. . . . .	69

## *Apéndice A*

---

# Plan de Proyecto Software

---

### **A.1. Introducción**

En esta sección voy a detallar como se fue planificando todo el proyecto. Como gestor de tareas he utilizado sprintometer, donde he ido apuntando en detalle las tareas de cada sprint, así como la duración de estas. Como he escrito en la memoria de este proyecto, he utilizado la metodología SCRUM [5] como metodología ágil. Esa metodología ágil es adaptativa, por lo que no es totalmente estricta. El objetivo de cada sprint era conseguir una funcionalidad nueva, la cual se pudiera ver y enseñar al cliente (tutor). Las iteraciones entre cada sprint duraron entre 1 y 3 semanas. A continuación especificaré cada fase de cada sprint.

### **A.2. Planificación temporal**

Como hemos mencionado en el apartado de introducción, especificaremos los periodos y las tareas realizadas en cada sping. Este proyecto lo hemos dividido en 10 sprints que detallaremos a continuación.

#### **Sprint 1: 01/02 – 13/02**

##### **Tareas**

- Reuniones con tutor para especificación de requisitos.
- Estudio de herramienta para realizar el proyecto.
- Prototipo básico en papel de la herramienta.



## Backlog

Esta iteración consta básicamente de negociaciones con el tutor y obtención de información para realizar el proyecto. En primer lugar, se nos presenta el proyecto, en el cual sólo teníamos una breve descripción, y desarrollamos las ideas para saber qué estamos buscando y qué queremos hacer. En principio el proyecto iba a ser programado únicamente en PHP, pero investigamos que se tenían que procesar una cantidad muy alta de datos, y que PHP no es una opción viable por lo que se decide realizar el proyecto en javascript, y que sea la máquina cliente la encargada de realizar todo el procesamiento de ficheros.

## Sprint 2: 23/02-20/02

### Tareas

1. Creación de clases de extracción de datos.
2. Aplicación prototipado con morris.js

## Backlog

En esta fase del proyecto se realiza un desarrollo primitivo de las clases de extracción de datos con un ejemplo totalmente inventado (aleatorio) por nosotros al disponer de ejemplos reales. Entre los ejemplos creados, éstos son cargados por ruta absoluta por javascript por lo que aún no es necesario un servidor PHP. Estos ficheros cargados de forma manual por javascript son:

- Ficheros con tramas de datos.
- XML que contiene la disposición de los datos.

También creamos la página principal index.html, en la cual se va a desarrollar de forma gráfica todo el proyecto. El tutor (cliente) ya puede ver la extracción y la disposición de los datos en el modo de depuración

## Sprint 3: 20/02-27/02

### Tareas

Realizar correcciones en la forma de extracción de datos.

## Backlog

- Documentar los requisitos.
- Reunión con el cliente (tutor) para mostrar los resultados.

En este sprint, básicamente consistió en la validación y corrección de datos de la parte crítica del proyecto que sería la de los datos. También se desarrollan y generan nuevos requisitos con respecto a los iniciales para dar fuerza al proyecto. Donde destacan toda la funcionalidad añadida a las gráficas deseada por el cliente. Entre la detección de posibles problemas, comprobamos que la aplicación funciona de forma asincrónica, y hay que forzar sincronismos para que funcione correctamente (carga de ficheros).

### **Sprint 4: 27/02-13/3**

#### **Tareas**

- Graficar datos.
- Añadir spinners entre las distintas pantallas de la aplicación.

#### **Backlog**

- Añadida carga de ficheros por ajax de forma sincrónica.
- Añadidos listener a elementos HTML
- Restructuración de las clases de extracción de datos.

Entre las distintas herramientas para graficar, seleccionamos la librería más completa: Highcharts que cumple varios requisitos a priori del proyecto, creando nuevas clases.

Como habíamos adelantado en el sprint anterior realizamos una restructuración de las clases, para que podamos adaptar la lectura sincrónica de ficheros y poder adaptar los datos de la biblioteca a una biblioteca en particular.

En la carga de ficheros, realizamos una lectura sincrónica gracias a ajax forzando el sincronismo. Entre los distintos elementos, modificamos el prototipado anterior y creamos uno nuevo, donde habrá dos pantallas: una para seleccionar el fichero XML por defecto o uno nuevo, y otro para ver la gráfica.

Creamos un spinner intermedio mientras se procesa dicho fichero. También detectamos que hay casos anómalos donde hay picos de datos, que no corresponden a la realidad.

### **Sprint 5: 13/03-03/04**

#### **Tareas**

- Mapeo de directorios por PHP.
- Selección previa de ficheros en la pantalla de carga.

## Backlog

- Muestra spinner de carga más llamativo.
- Corrección automática de tramas.
- Mejora de interfaz gráfica.
- Detectamos que hay casos anómalos donde en los datos que se nos han pasado, no son correctos, y procedemos a repararlos, reparando la fecha de estos (el segundero no aumentaba correctamente).
- Procedemos a leer los ficheros de la máquina dinámicamente, (antes forzabamos la lectura dando las rutas absolutas) a través de PHP.
- Realizamos pequeñas correcciones a nivel visual y cargamos la carga inicial de la aplicación por otra donde podamos seleccionar con qué ficheros queremos extraer las bibliotecas (si seleccionábamos todos en principio la carga era muy alta).

En esta iteración detectamos las limitaciones de la librería de pago, y tenemos que dar un paso atrás y cambiar de librería gráfica por lo que nos supone en iteraciones futuras una carga más de trabajo.

## Sprint 6: 03/04-10/04

En esta iteración realizamos buena cantidad de los requisitos pedidos por el tutor, entre los cuales programamos los siguientes cambios, con su correspondiente explicación:

### Tareas

- Migrar de librerías a chartjs. Creación y modulación de las clases para poderse aplicar un graficado correcto.
- Añadir funcionalidad de un diezmado simple para que las gráficas vayan mucho más fluidas.
- Crear debajo del gráfico, un gráfico maestro.

## Backlog

- Mejorar las clases (más métodos) de opcionesGrafica y opcionesVariable del paquete de gráficas para poder aplicar a posteriori cambios en paneles de gráfica o paneles de variable.

- Añadir un atributo de grosor de línea para gráficas con muchos datos.
- Adaptar funcionalidad para que los datos sacados de la biblioteca funcionen con la nueva librería.
- Arreglar bugs añadiendo estilo al canvas. Arreglado bug del solapamiento de capas inyectando HTML. Añadida función de zoom gracias al plugin.
- Añadir funcionalidad para que cuando desplazemos el ratón por la gráfica te haga grande el punto más cercano y te grafique en un label su valor (funcionalidad que venía con la gráfica). Programada funcionalidad (no venía en la herramienta gráfica) para que cuando hagamos click podamos obtener el valor del eje x e y del punto más cercano.
- Añadir funcionalidad en la que te calcula el punto más cerca del click del ratón, y los puntos de las otras variables que están a la izquierda de dicho punto

### **Sprint 7: 10/04-17/04**

En esta iteración programamos distintas tareas a realizar como la anterior, en la búsqueda de cumplir todos los requisitos:

#### **Tareas**

- Creación de tabla.
- Añadidos botones para la gráfica temporal y una caja temporal que señalará el tiempo graficado

#### **Backlog**

- Funcionalidad a los clicks de la gráfica.
- Funcionalidad a los botones de selección display de la gráfica.
- Aplicada funcionalidad de reproducción en la ventana temporal.
- Arreglados algunos bugs en la reproducción.
- Arreglados bugs visuales/diezmado.

Como sucede en reuniones anteriores con el cliente (tutor) se observan fallos, y surgen nuevos cambios que tienen que ser solucionados en sprints posteriores.

**Sprint 8: 17/04-1/05**

Este sprint lo usamos básicamente para documentar y arreglar diversos bugs.

**Tareas**

- Documentación.
- Añadir funcionalidad de retroceder y avanzar en el modo reproductor

**Backlog**

- Arreglar bug cuando había 3 variables o más.
- Añadir funcionalidad de pintar puntero.
- Reparar bug cuando había 3 variables o más.
- Reparar bug en el que se necesitaban al menos 2 variables para mostrarse la tabla

**Sprint 9: 1/05-20/05**

El objetivo de esta iteración es acabar la aplicación. Para ello marcamos distintas tareas a realizar:

**Tareas**

- Creación de lógica de intervalos.
- Aplicación de una interfaz de filtros.
- Añadir filtros a las gráficas
- Añadir búsquedas a las gráficas.
- Importación de vistas a través de un JSON.
- Exportación de vistas a través de un JSON.
- Creación de gráficas XY.
- Creación de vistas para las gráficas y variables.

**Backlog**

- Creación de interfaces gráficas para las vistas.
- Corrección de errores, mejora gráfica.

**Sprint 10: 20/05-2/07**

El objetivo de esta iteración es preparar la entrega del proyecto. Esta iteración básicamente se basará en documentar el proyecto, mejorar algún detalle de la aplicación y reparar bugs. Se alarga hasta el 2/07 para mejorar la calidad de la documentación.

**Tareas**

- Acabar memoria de la documentación.
- Acabar anexos de la documentación.

**Backlog**

- Preparar formato físico para la entrega de la documentación.
- Arreglar bugs conocidos de la aplicación.
- Correcciones en el código y documentación de este.

**A.3. Estudio de viabilidad**

Este estudio se hace para ver si realmente es viable el desarrollo de la aplicación. El estudio se reparte en dos tipos de viabilidades:

- Económica: donde se considera si una vez realizado el proyecto se considera rentable o no, es decir, que si el beneficio que se puede sacar por la herramienta es mayor al coste del desarrollo.
- Legal: donde estudiamos si nuestra herramienta viola alguna legalidad en el transcurso del desarrollo del programa.

**Viabilidad económica**

Es el apartado, donde se desarrolla el análisis económico. Como hemos mencionado antes, consideramos el beneficio que se puede sacar por la herramienta es mayor al coste del desarrollo.

**Análisis de costes.**

Coste de personal: En este proyecto hemos considerado que ha trabajado una persona durante 4 meses: Febrero, Marzo, Abril, Mayo. Considerando que se han trabajado 6 horas diarias, 20 días del mes, y que el desarrollador de este proyecto, es un programador junior sin experiencia, el cual puede cobrar 12 euros/hora. El coste derivado por el desarrollo de este proyecto es el siguiente:

$$12\text{euros/hora} * 6\text{horas/día} = 72\text{euros/diarios}$$

$$72\text{euros/día} * 5\text{días/semana} = 360\text{euros/semanales}$$

$$360\text{euros/semana} * 4\text{semanas} = 1440\text{euros/mes}$$

$1440\text{euros/mes} * 4\text{meses} = 5760$  euros de salario bruto por el equipo de desarrollo.

A Esto habría que sumarle por parte del empleador los costes derivados de la seguridad social, que dependerán de cada caso.

**Costes de hardware**

El coste aproximado del equipo con el que se ha desarrollado el trabajo es de 500 euros. (Ordenador de sobremesa con intel i5, 6 GB Ram, 256 GB SSD). Por tanto aplicando la siguiente formula y que según la ley de impuesto de sociedades de España, la vida útil del hardware es de entre 4-8 años, y nosotros estimamos que son 6 años, obtenemos que:

$$\text{Amortización} = (V_0 - V_r) / \text{Vida útil}$$

$$\text{Amortización} = 500\text{euros} / (12 * 4)\text{meses} = 10,42\text{euros/mes}.$$

Como la duración de este proyecto a sido de 4 meses, el coste total del hardware ha sido:

$$\text{Coste de hardware} = 10,42\text{euros/mes} * 4\text{meses} = 41,68 \text{ euros brutos}$$

**Costes de software**

Este apartado es el coste de todas las licencias de software utilizados para la realización del proyecto. Contemplando que se han utilizando licencias de estudiantes, MIT, y gratuitas, salvo en Microsoft Windows, y que este tiene una licencia comercial de 120 euros.

El coste total respecto al software es de:

$$\text{Amortización} = (V_0 - V_r) / \text{Vida útil}$$

$$\text{Amortización} = 120\text{euros} / (12 * 4)\text{meses} = 2,5\text{euros/mes}.$$

Como la duración de este proyecto a sido de 4 meses, el coste total del software ha sido:

Coste de software =  $2,5 \text{ euros/mes} * 4 \text{ meses} = 10$  euros brutos.

### Coste total

Teniendo en cuenta estos factores (recordamos que el gasto de la seguridad no lo hemos estimado porque depende de cada caso, pero se tiene que contar igual):

Coste total bruto =  $5760 \text{ euros} + 10 \text{ euros} + 41.68 \text{ euros} = 5811.68 \text{ euros}$

Suponiendo que el coste de la seguridad social de esa persona se estime de la siguiente forma:

TIPOS DE COTIZACIÓN RÉGIMEN GENERAL EJERCICIO 2017 -  
ORDEN ESS/106/2017, de 09/02 (BOE del 11/02)

Para el empresario:

23.60 por ciento en contingencias comunes.

0.60 por ciento en formación profesional.

7.70 por ciento en concepto de desempleo.

Total 31.9 por ciento

Por tanto el coste total bruto real =  $5760 \text{ euros} + (5760 * 0,319) \text{ euros} + 41,68 + 10 = 7649,12$  euros

### Análisis coste-beneficio.

En este apartado, estimamos los beneficios de la empresa que utilice esta herramienta. Suponiendo que instalando esta herramienta en cada caja negra de cada máquina, y suponiendo que cada instalación contará como una licencia.

También suponiendo que la empresa propietaria estime que el producto añadido de tener esta herramienta de monitorización es de 5 euros al año por cada máquina (la librería hightstocks cobraba 80 euros su licencia anual). Y suponiendo que esta aplicación se instale en una flota de 1000 robots obtendremos los siguientes números:

Ingresos por cada unidad = 5 euros / año.

Número de máquinas esta aplicación = 1000.

Ingresos anuales =  $1000 * 5 = 5000$  euros anuales

Recuperación inversión =  $7649.12 / 5000 = 1.53$  años para recuperar la inversión la empresa.

A partir del año y medio aproximadamente, la empresa compradora empezará a generar beneficios con la aplicación.



## Viabilidad legal

Este apartado fue un problema en nuestro proyecto. En principio, la aplicación se iba a desarrollar en Hightstocks. Una de las razones de usar esta librería respecto a las demás es que muchos de los requisitos negociados con el tutor se cumplían.

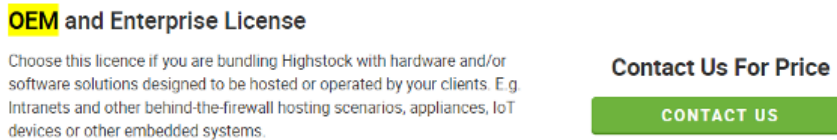


Figura A.1: El precio que se nos ofreció para usarlo en hardware fue de 80 euros al año por dispositivo.

Aunque la licencia fuera gratuita si lo usara de forma no comercial, no podría instalarlo en ninguna máquina ni probarlo por políticas de la empresa, ya que la propiedad del dispositivo es de la empresa.

Por ello, me puse en contacto con la empresa Highsoft, para ver que precios ofrecían por cada licencia. El precio que me calcularon ellos fue de 80 euros /año por cada máquina que se instalara, con un mínimo de 30 máquinas.

Cómo no se trataba de un proyecto de empresa, y el objetivo del proyecto era que funcionara en una máquina real, por lo que se decidió buscar otro tipo de librería que cumplieran los requisitos del proyecto, y se seleccionó Chartjs.org que contaba licencia MIT, lo cual implicaba que podía dar uso y disfrute de la librería gratuitamente con muy pocas limitaciones.

La licencia de esta aplicación esta bajo una "licencia de creative commons reconocimiento-Sin obraDerivada 4.0 internacional", la cual no se permite que se compartan las adaptaciones de la obra, pero si que se permite la reproducción, distribución y comunicación publica de la obra, incluso para fines comerciales.

---

## Especificación de Requisitos

---

### B.1. Introducción

El siguiente anexo, tiene como objetivo mostrar las distintas características y necesidades funcionales que se tienen que desarrollar en la aplicación, así como las no funcionales. Dichos requisitos se reflejarán en un diagrama de casos de usos.

### B.2. Objetivos generales

A continuación los requisitos funcionales de la aplicación:

#### **Traductor de tramas a variables informativas**

La traducción se define en un fichero XML.

El formato tipo seria el siguiente:

*tramaxx1*

*variablexxx; bitInixxx; bitFinxxx; Formatorxxx*

*Variableyyy; bitInixxx; bitFinxxx; Formatorxxx.*

*tramaxx2*

Las variables pueden mapearse al menos en los siguientes formatos:

BOOL, UInt8, Int8, UInt16, Int16, UInt32, Int32.

Todas ellas con el formato littleEndian.

#### **Visualización de variables**

Tienen que poder abrirse varios ficheros a la vez.

El usuario podrá visualizar el listado de ficheros disponibles e indicará con qué ficheros quiere trabajar.

Al añadir cada ventana indicamos el tipo de representación entre:

1. Representación temporal
2. Gráficas tipo XY (2 Variables)

Por cada ventana de visualización se nos permite añadir varias gráficas y modificar ciertos parámetros en función del tipo de representación:

Representación temporal:

- Variable asociada
- Color de representación
- Escala
- Valor de referencia
- Tipo de gráfica: Curva, texto, cronograma...

Gráficas tipo XY:

- Variable asociada a X.
- Variable asociada a Y.
- Tipo de puntero de dibujo: color.

## **Cursores**

En cada ventana se pueden añadir 2 cursores. Los cursores se pueden mover con el ratón. Se indicará el valor de las variables en las posiciones que indiquen los cursores. Es interesante poder hacer operaciones con los valores de los cursores (sumas, restas...).

## Manejo de ventana temporal

Esto es valido para todos los tipos de gráficas no sólo para la representación temporal.

El usuario indicará cuanto tiempo quiere visualizar en la misma ventana (zoom horizontal) y el instante inicial de visualización.

Ambos valores podrán cambiarse con barra de desplazamiento.

Los datos se ajustarán para que puedan ser visualizados en el tamaño de la ventana sin sufrir deformación.

Modo película: botón PLAY, PAUSE, STOP y PASO A PASO. Pulsando en el botón play el instante inicial de la ventana temporal se irá incrementando hasta que se llegue al final de los ficheros. Con el botón PAUSE se parará. Con el botón STOP volverá al estado inicial del gráfico. Se podrá modificar la velocidad de reproducción.

## Diezmado de datos

Si el tamaño de la ventana es tal que hay mas datos que pixeles posibles para representar se realizará un diezmado de la señal, es necesario que la página vaya fluida por lo que el programador tiene que estimar hasta cuantos puntos puede graficar.

## Filtraje de datos

En las gráficas temporales, se deben poder aplicar filtros de datos.

Es decir, representar sólo aquellos tramos de las señales que cumplan ciertas condiciones.

Para cada gráfica se podrán seleccionar un conjunto de condiciones (en formato textual tipo SQL) que deben cumplir cada tramo de señal para que sea visualizado. Por ejemplo: Voltaje > 30 AND Temperatura < 15.

Sólo se visualizarán los tramos en los que se cumplan simultáneamente esas condiciones.

## Búsqueda de datos

Inspirado en el modo de filtraje de datos anterior.

Mediante la búsqueda de datos se puede localizar el siguiente o el anterior valor temporal en el que se cumplan un conjunto de condiciones (especificadas en un formato textual similar al SQL). Y se posicionara el cursor en ese valor. De no verse ese valor por el diezmado, se realizará un zoom automático.

## Vistas

Se crearán vistas con las variables y gráficas creadas, de tal forma, que se pueda graficar la misma variable con distintas opciones y poder asignar un nombre a cada variable, con las opciones asociadas a dicha variable y a dicha gráfica. Por ejemplo, si a esa variable queremos representarla de dos formas, con o sin escalado. Aparte de la creación de vistas, se deberán poder exportar e importar independientemente de los ficheros que se apliquen.

## B.3. Catalogo de requisitos

### Requisitos funcionales

- R1. Traducción a variables informativas
- R2. Creación de vistas.
- R3. Visualización de variables
- R4. Manejo de cursores y tablas
- R5. Manejo de ventana temporal
- R6. Búsqueda de datos.
- R7. Filtro de datos.
- R8. Importar vistas.
- R9. Exportar vistas.

### Requisitos no funcionales.

A continuación los requisitos no funcionales:

- La aplicación tiene que ser compatible con las versiones de navegador más utilizadas.
- La aplicación se tiene que poder ejecutar tanto en móvil como en ordenador con un diseño responsive.
- El formato del archivo de configuración de las variables informativas debe ser un xml.
- El formato de importación y exportación debe ser un JSON.
- El proyecto deberá funcionar con los minimos recursos posibles del datalogger.

- El proyecto se podrá expandir en el futuro, por lo que el código tiene que ser debidamente comentado.

#### B.4. Especificación de requisitos

RF-1	Traducción a variables informativas	
Version	V1.0	
Autor	Rodrigo Martínez Bravo	
Descripción	El programa tiene que traducir las tramas de los ficheros para traducir la aplicación	
Precondición	Se tienen que tener los archivos de las tramas y los archivos de configuración en las carpetas correspondientes	
Secuencia normal	Paso	Acción
	1	Mapear los archivos de tramas
	2	El usuario tiene que escoger que archivos quiere cargar.
	3	Cargar en memoria los archivos de tramas
	4	Cargar en memoria el XML
	5	Parsear el archivo XML y obtener el objeto Configuración
	6	Aplicar por cada línea de trama de cada fichero el objeto Configuración
Postcondición	Se crea el objeto biblioteca con las variables informativas calculadas. Se da paso al área de graficado	
Excepciones		
Importancia	Alta	
Comentarios	Ninguno	

Figura B.1: Requisito funcional 1

RF-2	Creación de vistas	
Version	V1.0	
Autor	Rodrigo Martínez Bravo	
Descripción	El usuario crea vistas, es decir, configura que variables quiere ver, y configura dichas variables.	
Precondición	Biblioteca calculada.	
Secuencia normal	Paso	Acción
	1,1	El usuario selecciona que tipo de gráfica quiere, junto a las opciones que quiere
	1,2	Si se selecciona las vistas de graficado rápido, el usuario selecciona la gráfica que quiere, junto a las variables que va a graficar. Estas variables son extraídas de la biblioteca.
	2,1	Si se trata de una gráfica temporal, el usuario puede añadir y quitar variables al gusto personalizandolas. Estas variables son extraídas de la biblioteca.
	2,2	Si se trata de una gráfica XY, el usuario tendrá que escoger una variable a representar por cada eje de abscisas. Estas variables son extraídas de la biblioteca.
	3	Se guarda la vista y queda lista para el procesamiento dentro de la biblioteca.
Postcondición	Se crea el objeto biblioteca con las variables informativas calculadas.	
Excepciones		
Importancia	Media	
Comentarios	Si se trata de una vista de graficado rápido (fast graph) solo se realiza el punto 1.2 y 3	

Figura B.2: Requisito funcional 2

RF-3	Visualización de variables	
Version	V1.0	
Autor	Rodrigo Martínez Bravo	
Descripción	El usuario visualiza las variables en las gráficas	
Precondición	Vistas calculadas	
Secuencia normal	Paso	Acción
	1	Una vez que el usuario ha calculado las vistas y ha dado guardar. Si el sistema ha detectado que hay suficientes parámetros para que se pueda graficar algo, lo hace.
	2	Carga las opciones de las vistas, y extrae los datos de la biblioteca de las variables seleccionadas por las vistas.
	3	Una vez extraídos los datos, se convierten en formato de tal manera que la librería gráfica lo pueda leer. Se cargan también las opciones que se han escogido de las gráficas y variables en las vistas.
Postcondición	Se realiza un diezmado y se inyectan los datos a la librería gráfica	
Excepciones		
Importancia	Alta	
Comentarios	Ninguno.	

Figura B.3: Requisito funcional 3



RF-4	Manejo de cursores y tablas	
Version	V1.0	
Autor	Rodrigo Martínez Bravo	
Descripción	El usuario puede colocar hasta 2 cursores por la tabla. Una vez están situados los 2 cursores se visualiza una tabla con los valores más proximos a la izquierda.	
Precondición	Gráfica dibujada.	
Secuencia normal	Paso	Acción
	1	El usuario selecciona que cursor quiere dibujar. pinchando en el icono del cursor que se quedará activado
	2	Una vez seleccionado el usuario puede ir posicionando con el click izquierdo del ratón el cursor dentro de la gráfica.
	3	Una vez esté conforme el usuario, puede dejar de dibujar el cursor con el click izquierdo, desmarcando el icono del cursor., igual que lo había activado.
	4	Si se han dibujado 2 cursores, se calculará una tabla con las diferencias de los 2 cursores y los valores que tiene cada una.
Postcondición	Se dibujan 2 cursores y se inyecta por HTML una tabla.	
Excepciones		
Importancia	Baja	
Comentarios	Ninguno.	

Figura B.4: Requisito funcional 4

RF-5	Manejo de ventana temporal	
Version	V1.0	
Autor	Rodrigo Martínez Bravo	
Descripción	Debajo de la gráfica tendremos una gráfica más pequeña, también llamada gráfica maestra o ventana temporal.	
Precondición	Gráfica dibujada	
Secuencia normal	Paso	Acción
	1	El usuario podrá personalizar el zoom mediante los botones dispuestos para ello, o pinchando en los puntos donde se desea si están activados el desplazamiento izquierdo o derecho . La gráfica se recalculará con un diezmo.
	2	El usuario podrá reproducir la gráfica y podrá manipular la velocidad de reproducción. Incluso reproducir paso a paso adelante y hacia atrás.
	3	El usuario podrá volver a la posición inicial de la gráfica presionando el botón stop.
Postcondición	Se manipula el espacio temporal de las gráficas sin afectar a los datos que la contiene.	
Excepciones		
Importancia	Media	
Comentarios	Ninguno	

Figura B.5: Requisito funcional 5

RF-6	Búsqueda de datos	
Version	V1.0	
Autor	Rodrigo Martínez Bravo	
Descripción	El usuario a través de una consulta puede buscar datos.	
Precondición	Gráfica dibujada	
Secuencia normal	Paso	Acción
	1	El usuario introduce una consulta y da el botón buscar.
	2	Se calculan los intervalos que se cumple esa condición y se muestra con un cursor el valor en el que se cumple. Si no es el primer valor, se realiza un zoom.
	3	El usuario puede introducir nuevas consultas y partirán del punto donde estaba el cursor de la primera búsqueda. El usuario podrá buscar el siguiente valor o el anterior.
	4	Si lo desea el usuario puede convertir el cursor de búsqueda en un cursor. Del RF-5
Postcondición	Ninguna.	
Excepciones		
Importancia	Media	
Comentarios	Si se trata de una vista de gráfico rápido (fast graph) solo se realiza el punto 1.2 y 3	

Figura B.6: Requisito funcional 6

RF-7	Filtro de datos	
Version	V1.0	
Autor	Rodrigo Martínez Bravo	
Descripción	El usuario a través de una consulta puede buscar datos.	
Precondición	Gráfica dibujada	
Secuencia normal	Paso	Acción
	1	El usuario introduce una consulta y da el botón filtrar.
	2	Se calculan los intervalos que se cumple esa condición
	3	Los puntos que no cumplen la condición desaparecen de la gráfica
	4	El usuario podrá eliminar el filtro dando el botón eliminar.
Postcondición	Si no se elimina el filtro, no podremos realizar otras funcionalidades de la gráfica.	
Excepciones		
Importancia	Media	
Comentarios	Ninguno	

Figura B.7: Requisito funcional 7

RF-8	Importar vistas.	
Version	V1.0	
Autor	Rodrigo Martínez Bravo	
Descripción	El usuario importa vistas a través de un JSON	
Precondición	Biblioteca calculada. El archivo JSON no puede estar manipulado.	
Secuencia normal	Paso	Acción
	1	El usuario selecciona que ficheros quiere aplicar las vistas.
	2	El usuario carga un JSON de su directorio con las vistas que quiere aplicar.
	3	El sistema comprueba que contenga las variables que se quieren aplicar con la vista importada.
	4	Si se realizado con éxito la importación se le avisará al usuario
Postcondición	Se dibuja una gráfica	
Excepciones		
Importancia	Media	
Comentarios	Ninguno	

Figura B.8: Requisito funcional 8

RF-9	Exportar vistas.	
Version	V1.0	
Autor	Rodrigo Martínez Bravo	
Descripción	El usuario puede exportar las vistas actuales	
Precondición	Biblioteca y vistas calculadas.	
Secuencia normal	Paso	Acción
	1	El usuario selecciona la opción de importar.
	2	Se dispone un enlace para la bajada de las vistas en formato JSON
Postcondición	Se genera un archivo JSON que después podrá ser utilizado para importar vistas.	
Excepciones		
Importancia	Media	
Comentarios	Ninguno	

Figura B.9: Requisito funcional 9

## B.5. Casos de uso

A continuación mostramos los casos de usos del proyecto. En principio existe un único actor que podrá realizar todas las funcionalidades disponibles del proyecto.

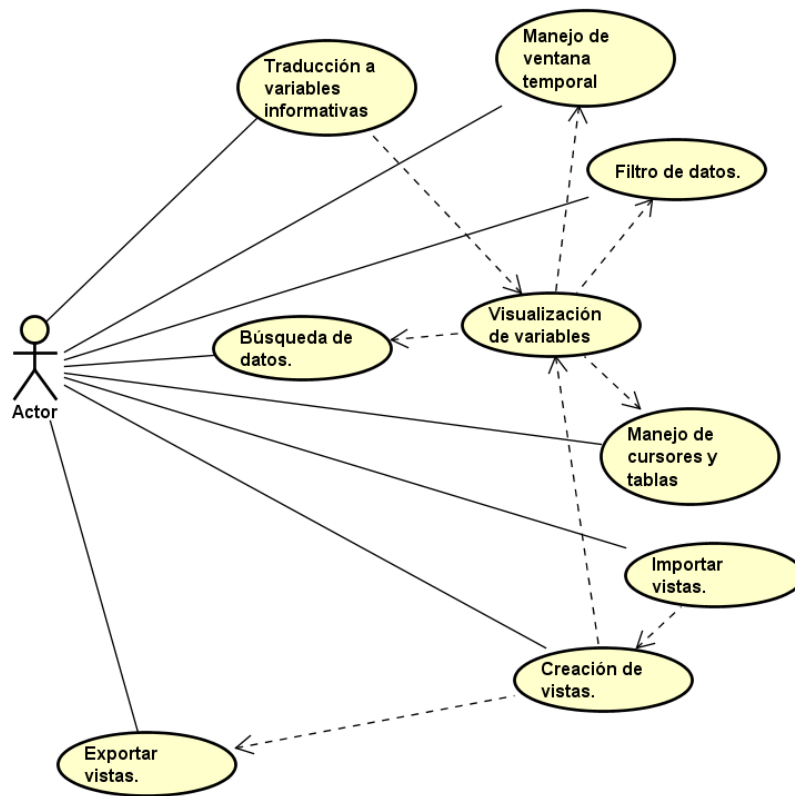


Figura B.10: Casos de uso del proyecto

---

## Especificación de diseño

---

### C.1. Introducción

En este apartado hablaremos del diseño del proyecto. Para ello nos basaremos en diagramas de clases en UML. Para facilitar legibilidad omitiremos ciertos métodos y atributos de las clases que a continuación vamos a mostrar.

### C.2. Diseño de datos

En primer lugar vamos a definir el diseño de los datos.

#### Configuración

Por un lado, tendremos las estructuras correspondientes a la creación del archivo configuración, que es creado a partir del parseo del XML. Este archivo de configuración está estructurado en distintas clases, que permiten una posterior modificación en el caso que el XML en el que está estructurado evolucione.

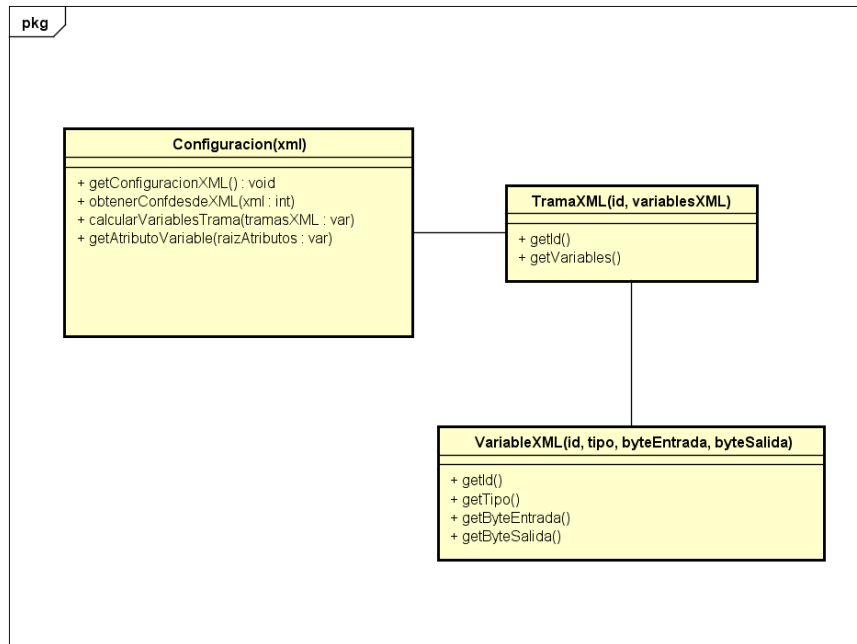


Figura C.1: Diagrama de clases de configuración

## Biblioteca

Por otro lado, tenemos la creación de la biblioteca aplicando el archivo de configuración antes mostrado. Para ello creamos una distinta estructurada en distintas clases donde guardaremos las distintas variables, como los datos creados en ellos.

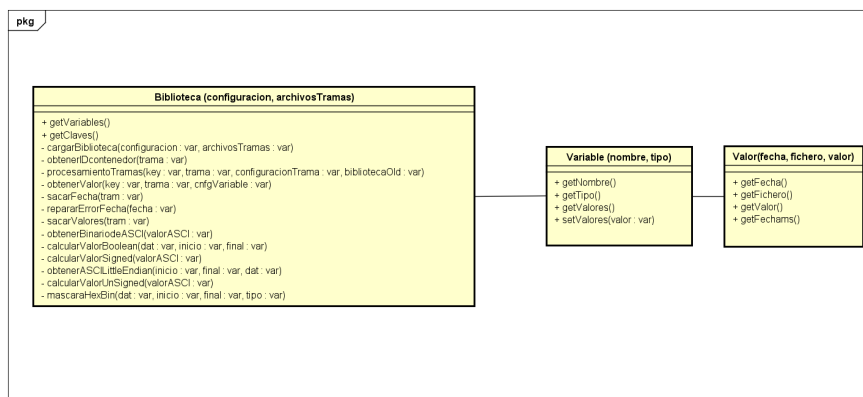


Figura C.2: Diagrama de clases de biblioteca



## Cargador de ficheros

Por último lugar tendremos el cargador de ficheros, que engloba todas las estructuras de datos anterior mencionadas.

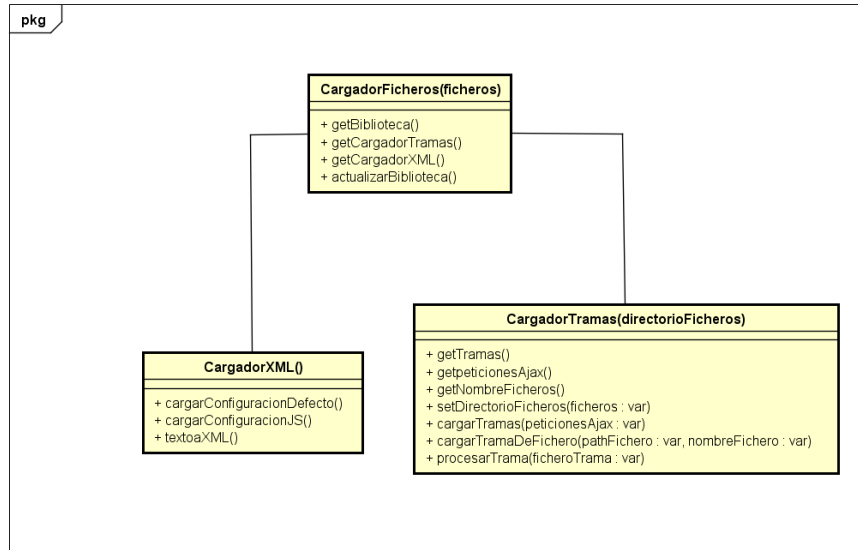


Figura C.3: Diagrama de clases del fileloader

## C.3. Diseño procedimental

En este apartado, vamos a realizar una especificación más detallada del funcionamiento de ciertas funcionalidades. Para ello diversificamos:

- Comportamientos de la tabla y botones asociadas.
- Interfaz gráfica de las vistas gráficas.
- Búsqueda y filtrados.

### Comportamientos de la tabla y botones asociadas.

En este caso, mostramos los diagramas de clase asociados a los comportamientos de los elementos de la gráfica así como su creación.

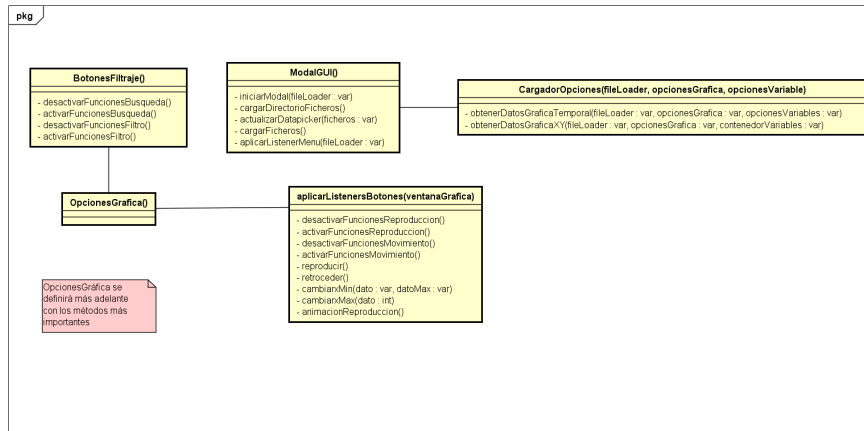


Figura C.4: Diagrama clases tabla y botones asociadas

## Interfaz gráfica de las vistas gráficas.

En este apartado mostramos los diagramas de clases correspondientes a la creación de la interfaz gráfica de las vistas (Creación y modificación de vistas). Todas ellas reutilizan el DOM del modal con un patrón decorador.

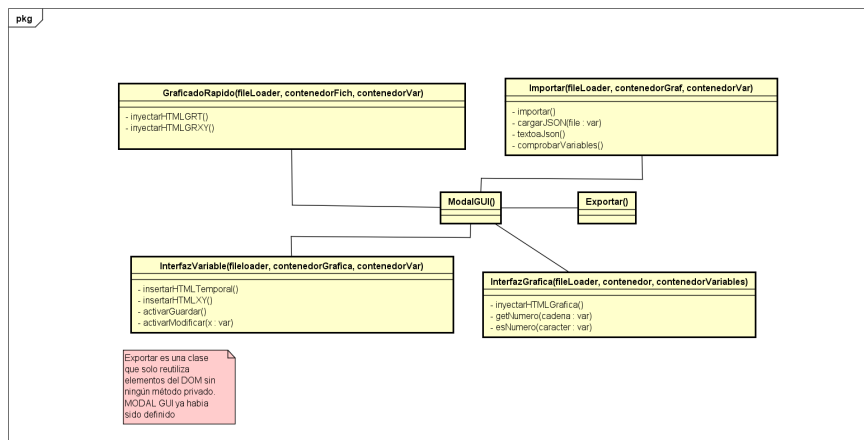


Figura C.5: Diagrama de clases vistas gráficas

## Búsqueda y filtrados.

Diagrama de clases correspondiente al funcionamiento de la búsqueda y filtrados de la gráfica. En primer lugar se calculan los intervalos que cumplen las sentencias, y luego se calculan los puntos que cumplen dichos intervalos tanto para la búsqueda como para los filtrados.

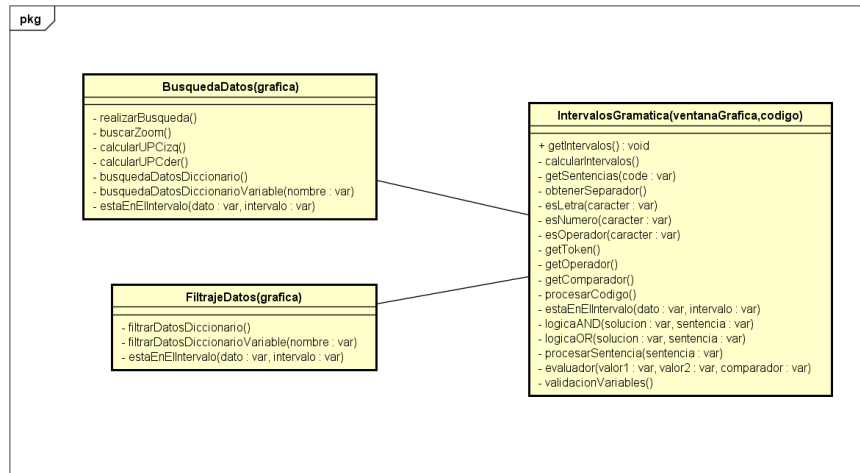


Figura C.6: Diagrama clases Búsqueda y filtrados

## C.4. Diseño arquitectónico

En este apartado, tendremos un diagrama más general del conjunto de los elementos de la aplicación de visualización de variables.

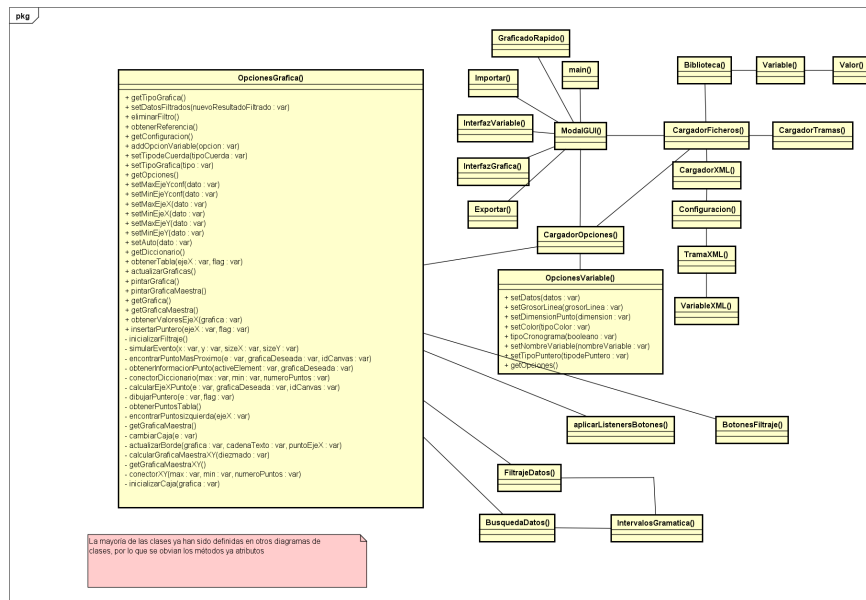


Figura C.7: Diagrama clases general

## *Apéndice D*

---

# Documentación técnica de programación

---

### D.1. Introducción

En este apartado, vamos a indicar los pasos de la instalación y configuración necesarios para ejecutar este proyecto. La página del proyecto es : <https://github.com/rmb0033/analizadortramas>

### D.2. Estructura de directorios

Dentro de la estructura de nuestro proyecto, la cual se encuentra alojada en:

<https://github.com/rmb0033/analizadortramas>

Podremos ver distintos directorios y archivos, donde se detallará su uso a continuación.

Estructura de directorios de la aplicación:

#### **Carpeta raiz.**

Se encuentran:

- La imagen la universidad de Burgos.
- La clase PHP utilizada para mapear los ficheros.
- El archivo de lectura para el repositorio.
- El archivo XML utilizado para la lectura de variables.

- El archivo XML utilizado como pruebas para un único valor.
- La página HTML principal.

## Datalogger

Es la carpeta donde se encuentran los ficheros que tienen que ser analizados. Este directorio cuelga un subdirectorio, REPOS. El cual también contiene esos datos.

Estos ficheros y directorios fueron proporcionados por el tutor como enunciado del proyecto.

## CSS

- Es el directorio donde se encuentran las hojas de estilo utilizadas en el proyecto. En la carpeta principal estarán los 2 spinners utilizados para la gráfica, los cuales han sido modificados y la hoja de estilos principal.
- Dentro de los directorios que contiene css, estarán “librerías”, la cual contiene hoja de estilos de terceros (bootstrap) y fonts que también contiene fuentes creadas por terceros.

## JS

Es el directorio donde se encuentran los archivos Javascript de la aplicación. En la carpeta principal se encuentra el fichero main.js, el cual es el fichero principal que llama a los demás archivos. Dentro de JS tenemos dos directorios:

- Vendor: En el cual se encuentran las librerías de terceros utilizadas en el proyecto.
- Modelos. La cual alberga todas las librerías creadas por el programador del proyecto.

Dentro de esas librerías se encuentran:

- Behaviors: Se encuentran las clases encargadas de asignar comportamientos a los elementos de la página web y de la gráfica.
- Biblioteca: Se encuentran las clases encargadas del almacenamiento de los datos de cada variable.

- **CargadorFicheros:** Se encuentran las clases encargadas tanto de cargar las líneas de los ficheros, como de cargar los XML [6]. El cargador de ficheros se encarga también de unir tanto las clases que se encuentran en biblioteca como en configuración.
- **Configuración:** Se encuentran las clases encargadas de almacenar la información de la disposición de las variables, y para ello, de analizar el XML para obtener dicha disposición.
- **Interfaz:** Es donde se encuentra los archivos javascript encargados de gestionar la interfaz gráfica de la creación y gestión de vistas.
- **Gráfica:** En el directorio raíz se encuentran las clases de variables y gráficas, las cuales son las encargadas de configurar y crear tanto gráficas como variables. A su vez tiene otro directorio llamado procesamiento de datos: Se encuentran las clases encargadas de calcular los intervalos que cumplen una consulta y aplicarlos. ya sea a través de una búsqueda o a través de un filtrado.

### D.3. Manual del programador

En el siguiente anexo, se explicará como dejar a punto el sistema para que se pueda ejecutar el servidor y a su vez nos permita trabajar en nuestro ordenador como herramienta de trabajo. Este manual de programador está adaptado para Windows, pero tiene semejanzas con los demás sistemas operativos.

#### PhpStorm

Se trata de nuestro IDE y por tanto nuestra herramienta principal de trabajo. PhpStorm [1] ofrece a profesores y alumnos licencias gratuitas, por lo que no tendrá ningún coste para nosotros. Para ello entramos en el siguiente link y descargamos la aplicación:

<https://www.jetbrains.com/phpstorm/download/#section=windows>

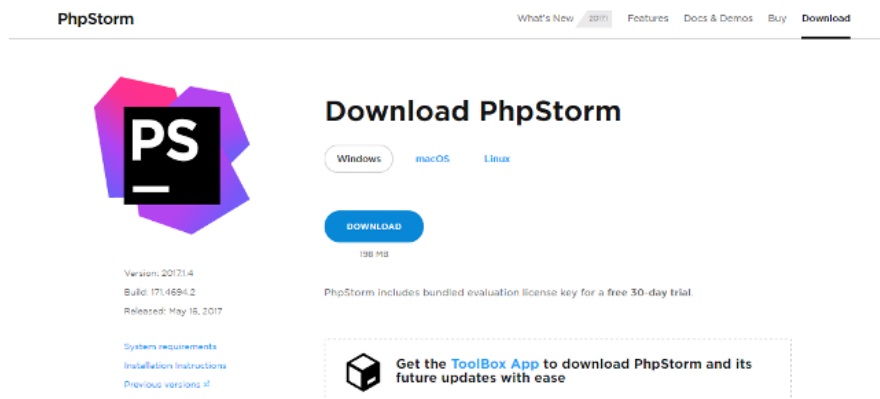


Figura D.1: Pagina oficial de PHPStorm

Una vez descargado, seguimos los pasos de la instalación:

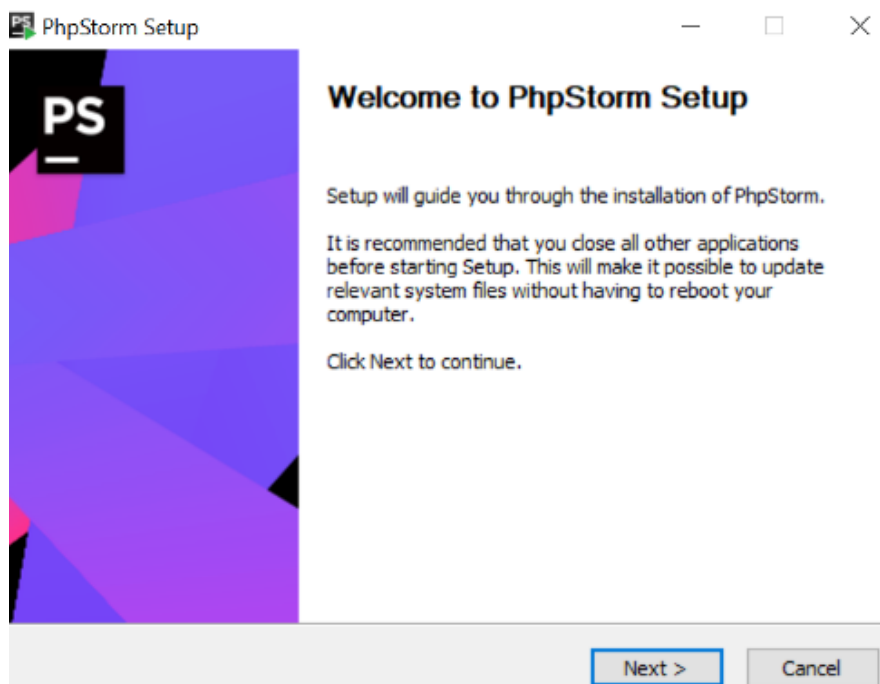


Figura D.2: Paso 1 de la instalación

Para evitar incompatibilidades seleccionamos estas opciones:

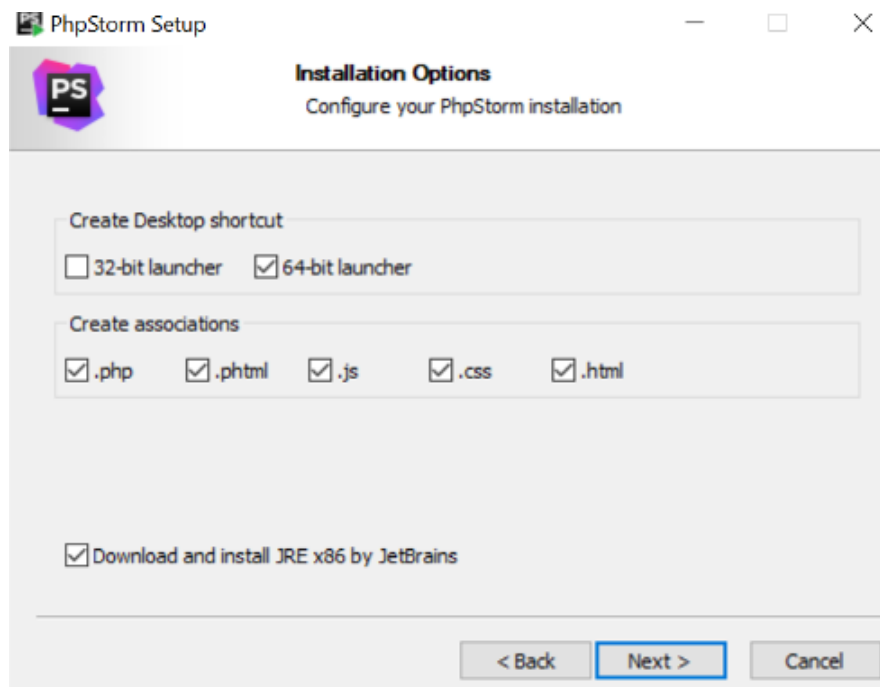


Figura D.3: Paso 2 de la instalación

Por último instalamos:

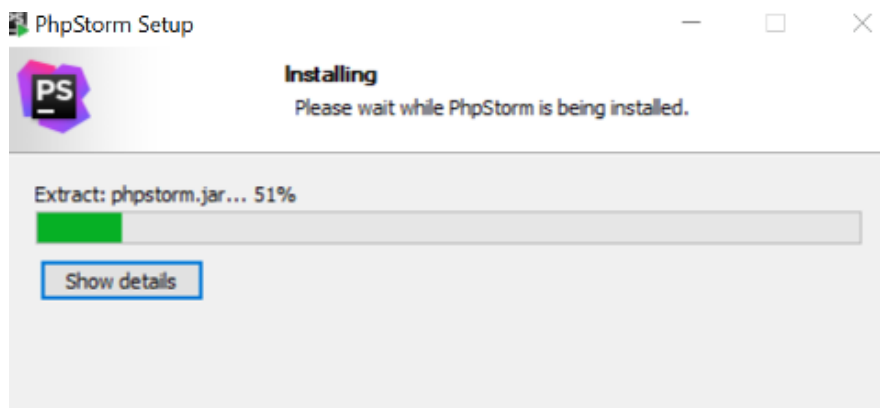


Figura D.4: Paso 3 de la instalación

Antes de realizar ningún cambio vamos a proceder a instalar git.



## Git

Git es un sistema de control de versiones, que es necesario para trabajar con nuestro repositorio [3]. Para ello, como Windows no trae git por defecto tenemos que instalarlo.

### Downloading Git

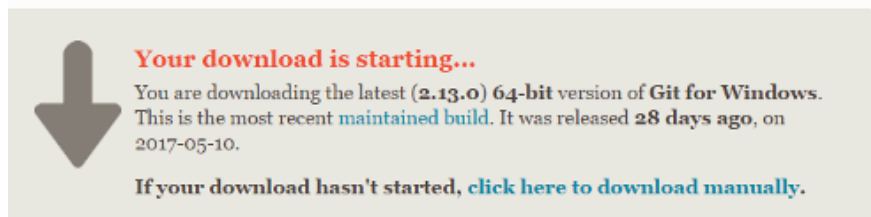


Figura D.5: Página de descarga de git

Procedemos a instalar git:

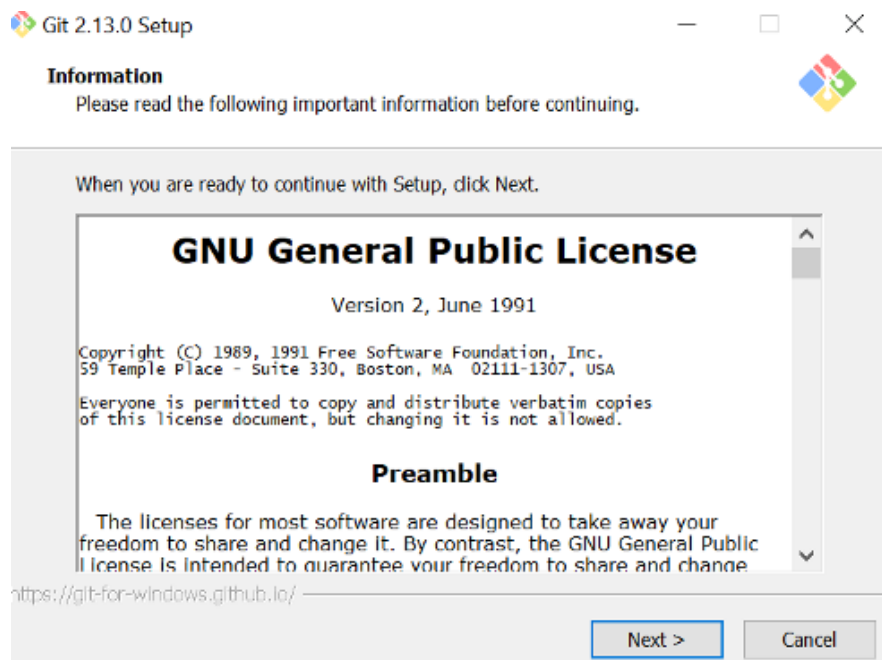


Figura D.6: Paso 1 de instalación de git

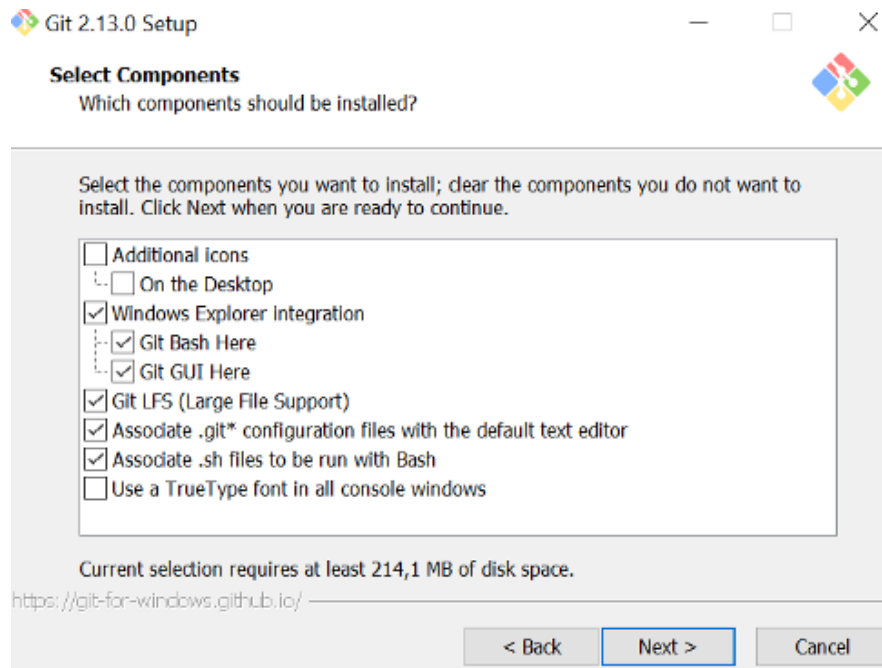


Figura D.7: Paso 2 de instalación de git

Marcamos estas opciones:

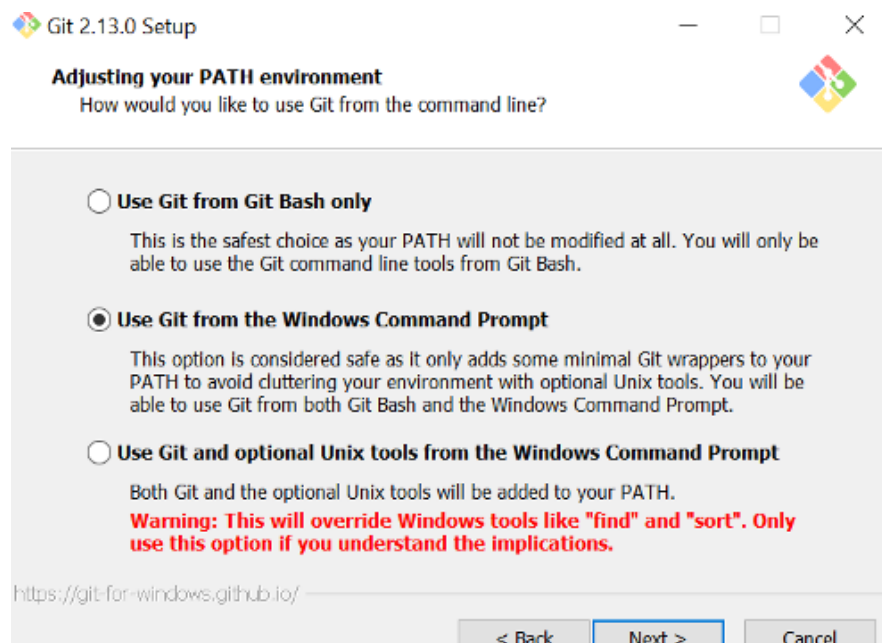


Figura D.8: Paso 3 de instalación de git

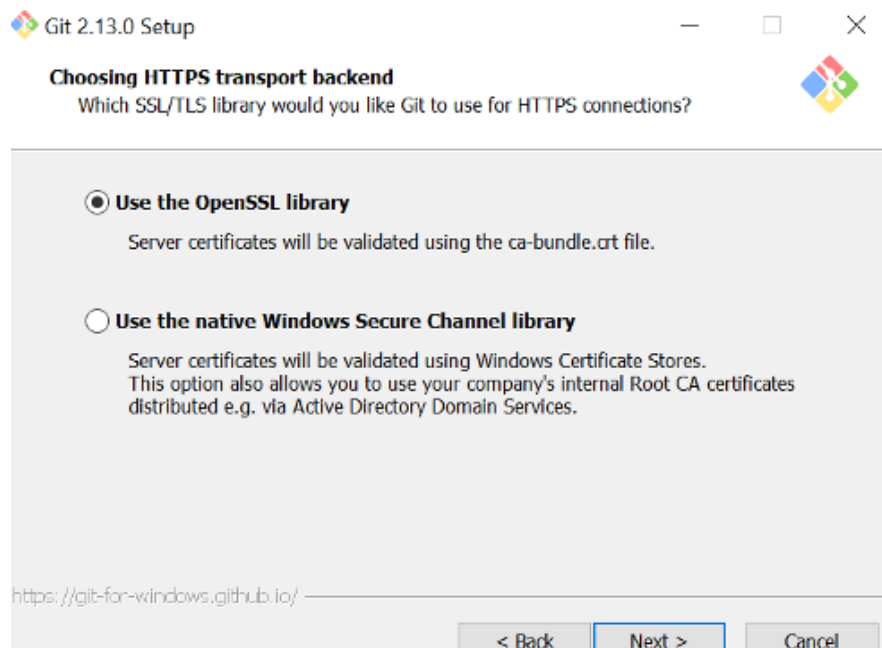


Figura D.9: Paso 4 de instalación de git

Seguimos con la instalación: los demás pasos de la instalación deberán ser los establecidos por git por defecto.

## D.4. Compilación, instalación y ejecución del proyecto

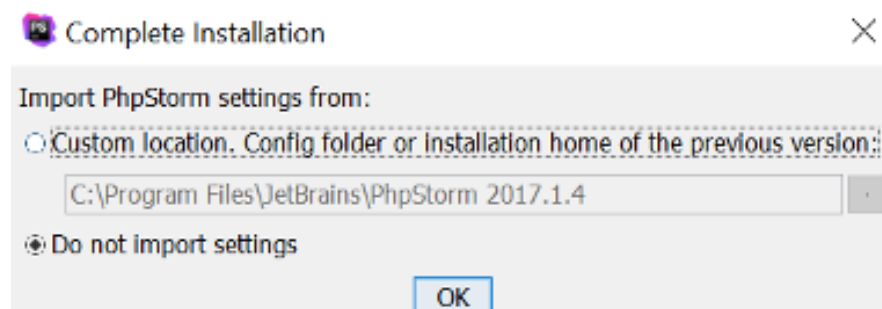


Figura D.10: Primera imagen al abrir la aplicación

Abrimos PHPStorm, por defecto no importaremos opciones. Aceptaremos las condiciones del contrato y entraremos en nuestra cuenta gratuita.



Figura D.11: Carga de PhpStorm

Una vez dentro, seleccionaremos la opción de:

“Check out from Version Control”

Aquí podremos clonar nuestro repositorio [4] : <https://github.com/rmb0033/analizadortramas.git>



Figura D.12: Carga de repositorio en PHPStorm.

Seleccionaremos la opción de clone. Y se nos bajará el repositorio.

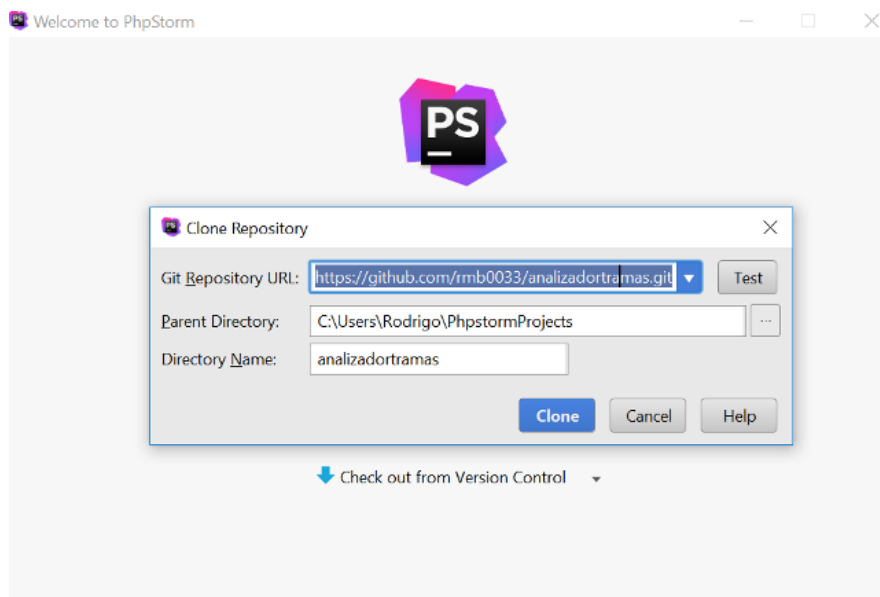


Figura D.13: Carga de repositorio en PHPStorm.

Nos saldrá un cartel advirtiéndolo como este:

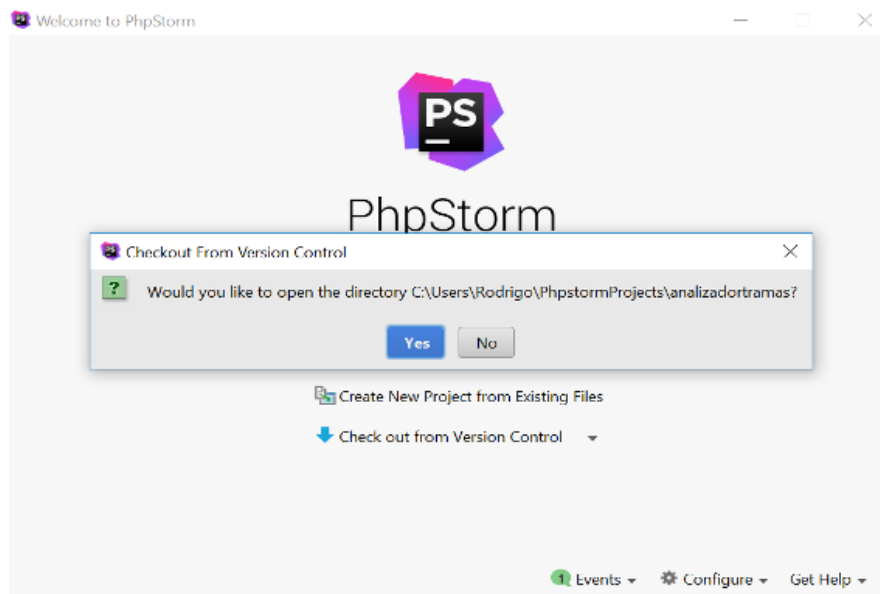


Figura D.14: Advertencia de carga repositorio.

Si no queremos/podemos realizar el clone. Siempre podremos descargar el repositorio y copiar el contenido de este dentro de un nuevo proyecto.

Tendremos que instalar el intérprete PHP. Para ello vamos al apartado de PhpStorm donde nos mandan instalar el intérprete:

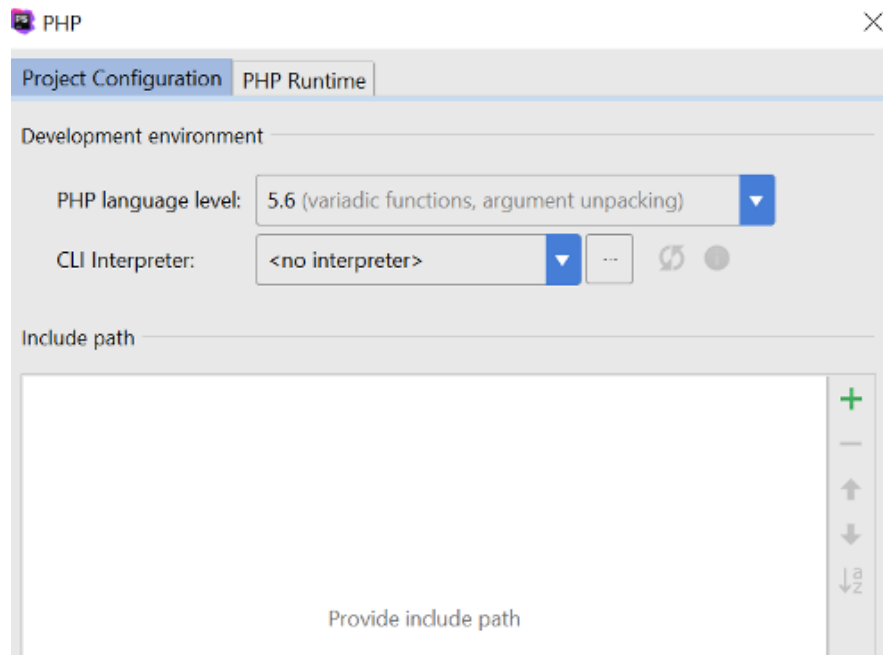


Figura D.15: Selección interprete PHP

Para ello necesitamos la versión 5.5 o 5.6 de PHP.

Accedemos a la siguiente página: <http://windows.php.net/download/> y descargamos una release de PHP 5.6. Para este proyecto hemos usado la siguiente: <http://windows.php.net/downloads/releases/php-5.6.30-nts-Win32-VC11-x86.zip>

Una vez descargado el zip, procedemos a descomprimirlo en una carpeta conocida.

## APÉNDICE D. DOCUMENTACIÓN TÉCNICA DE PROGRAMACIÓN41

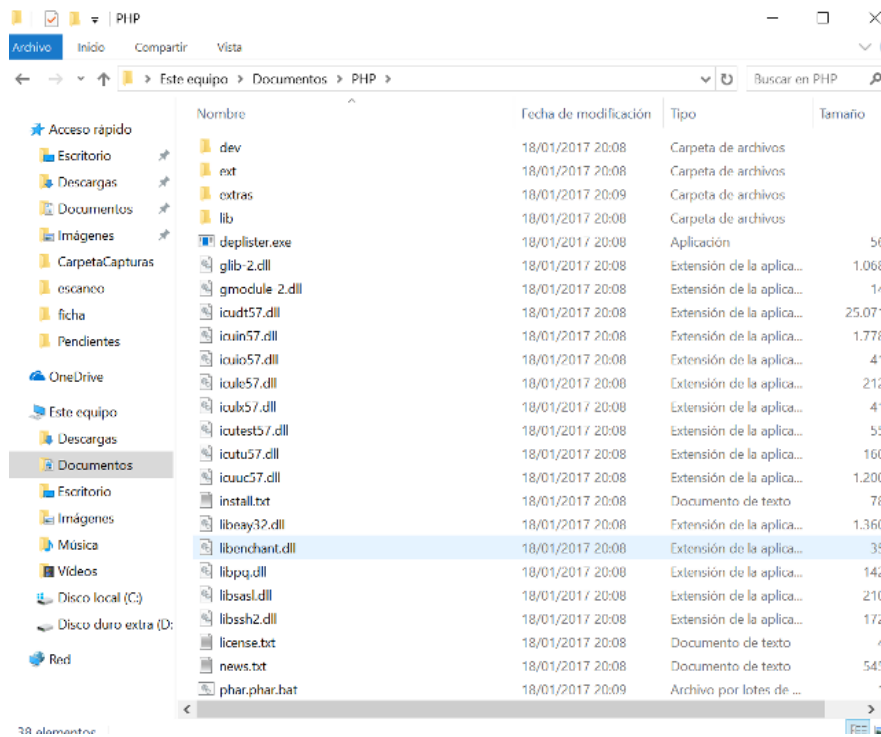


Figura D.16: Carpeta con contenido de PHP 5.6

En PHPStorm, siguiendo con la última pantalla. Presionamos en las opciones de CLI Interpreter:



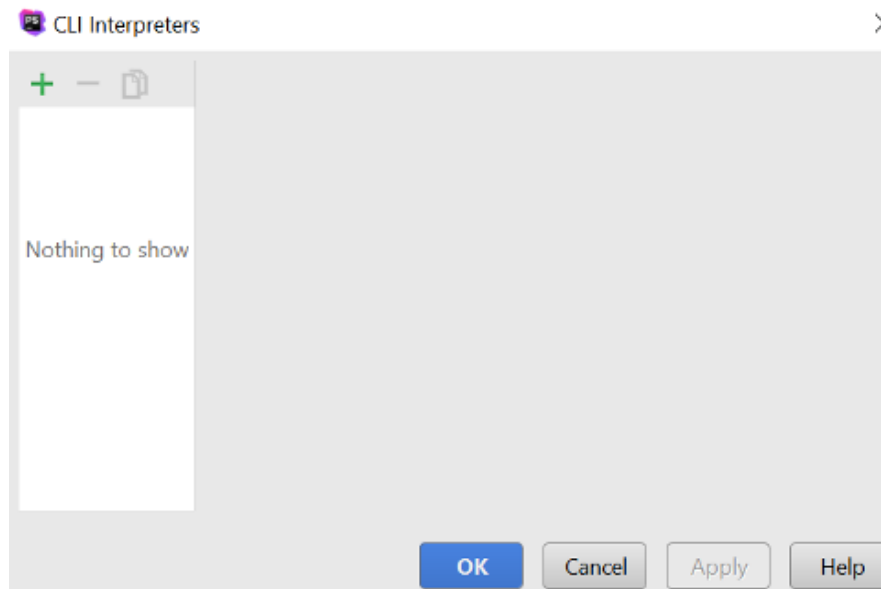


Figura D.17: Selección del interprete CLI. Paso 1

Pulsamos en el icono “+” y seleccionamos “Local Path”.

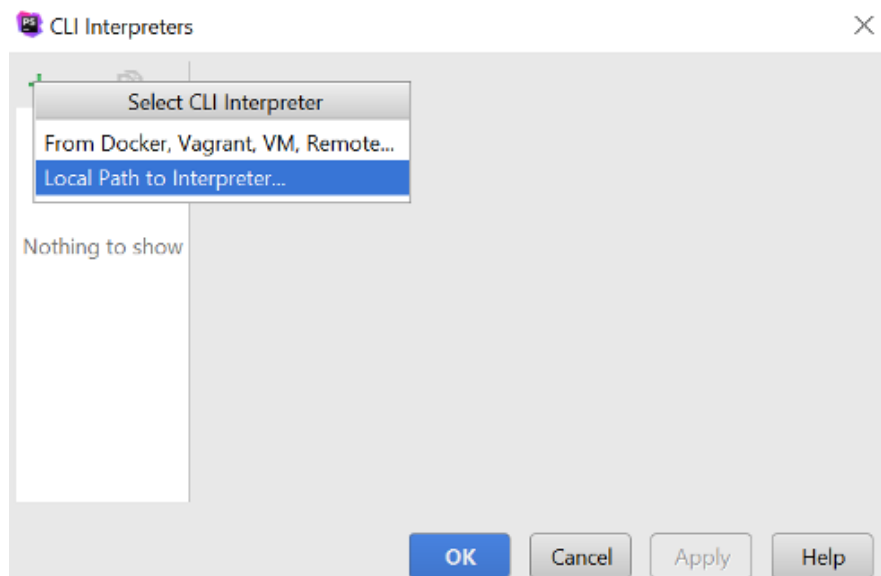


Figura D.18: Selección del interprete CLI. Paso 2

Recorremos la carpeta que hemos descomprimido de PHP hace unos instantes y seleccionamos el interprete (php-cgi-exe) y aceptamos.

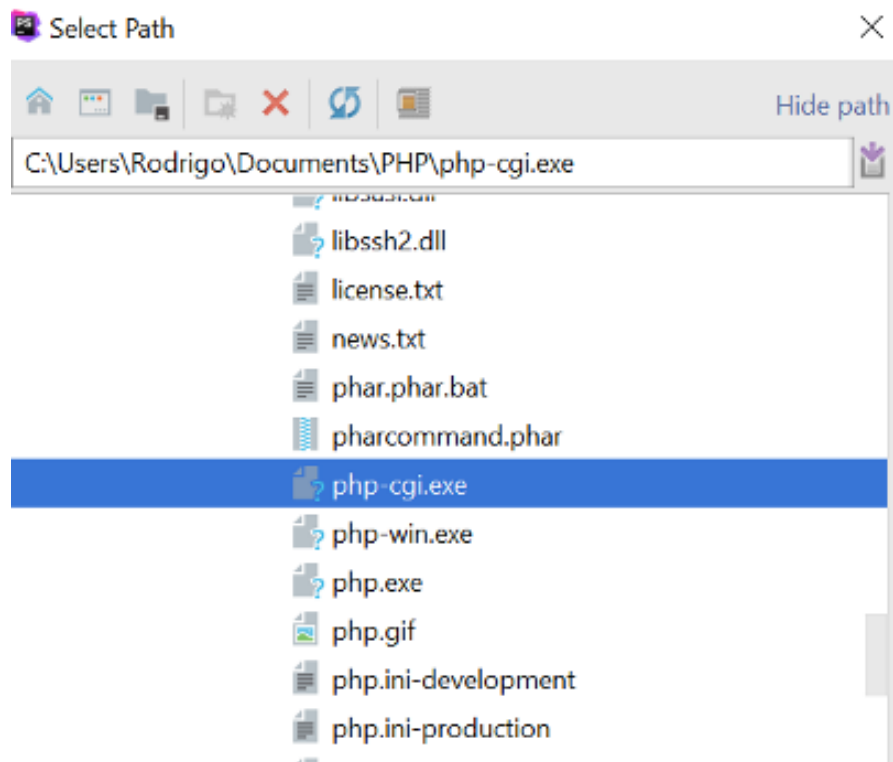


Figura D.19: Selección del interprete CLI. Paso 3

Ahora el “project configuration” debería tener este aspecto:

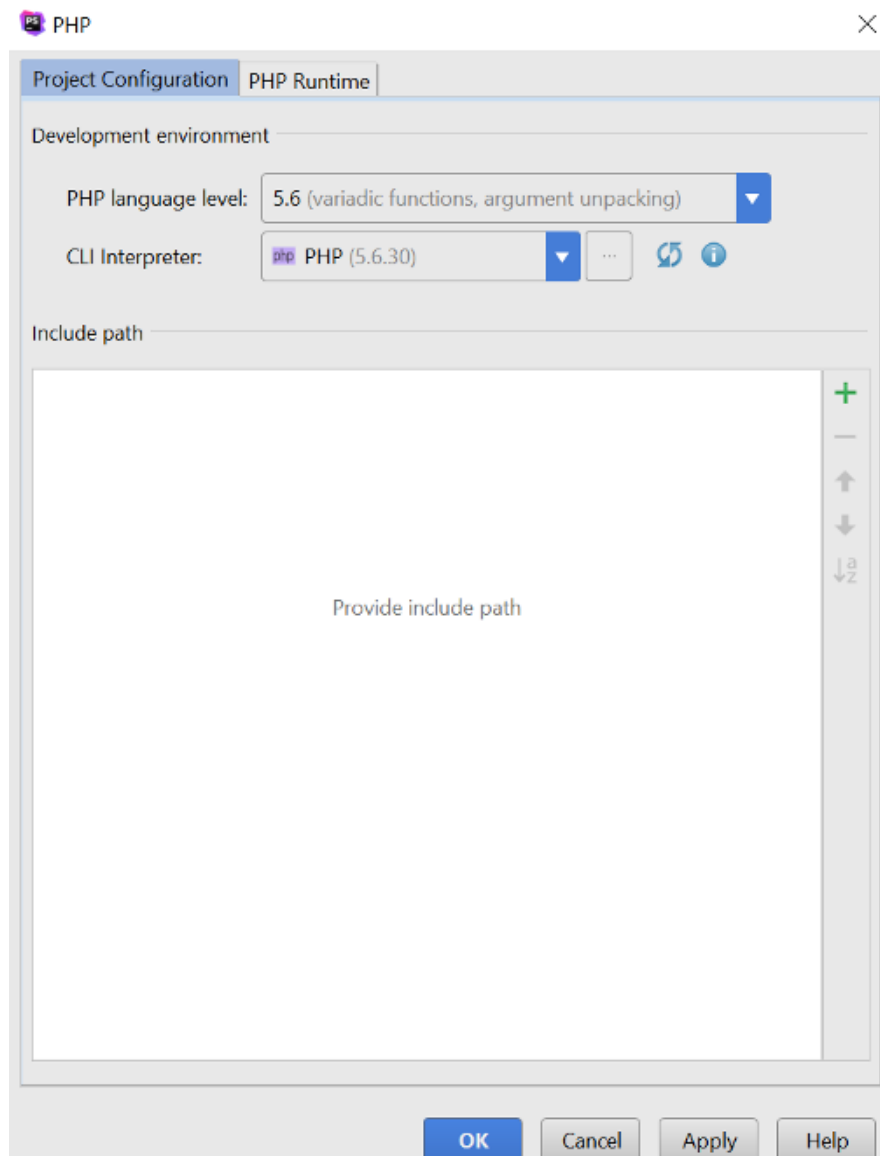


Figura D.20: Aspecto final configuración del proyecto PHPStorm.

### Realizar cambios en el repositorio.

Utilizaremos esta opción:

## APÉNDICE D. DOCUMENTACIÓN TÉCNICA DE PROGRAMACIÓN 45

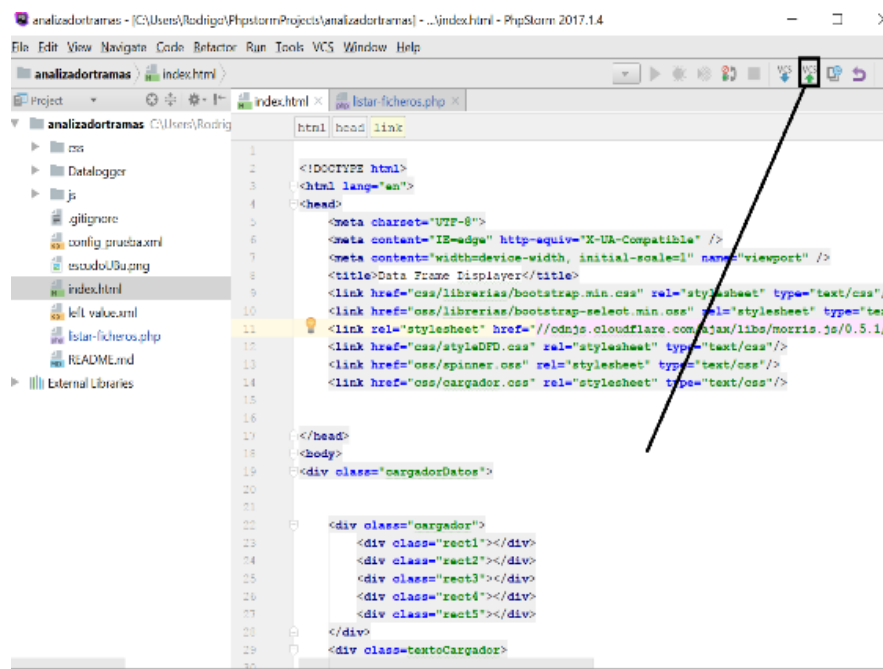


Figura D.21: Vista general desde PHPStorm. VCS.

Por último, si queremos subir los cambios a github utilizaremos un commit and push, siempre y cuando tengamos acceso a ese repositorio.

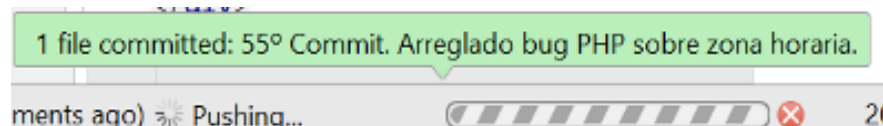
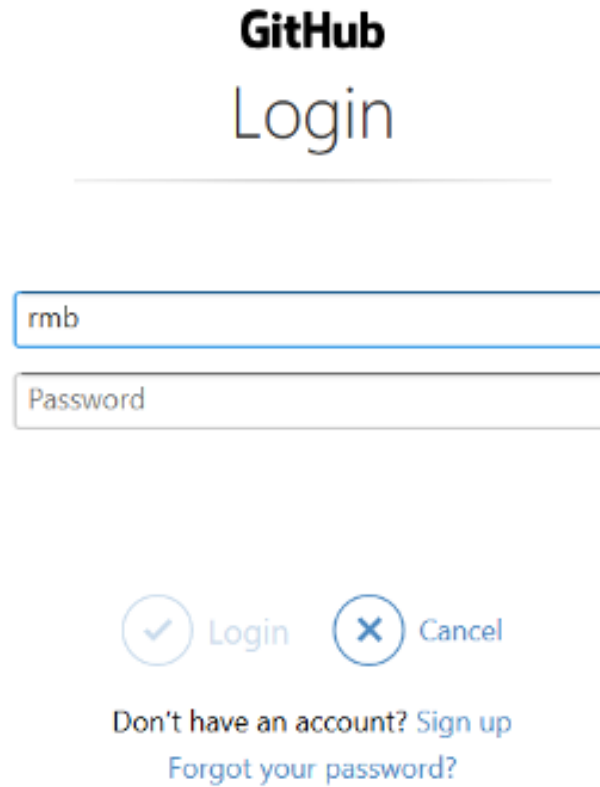


Figura D.22: Progreso barra control de versiones

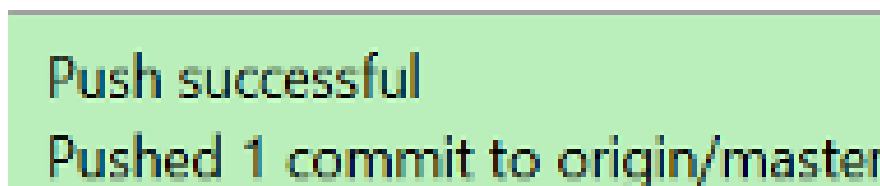
Nos mandará loguearnos antes de hacer el push por primera vez en esta pantalla:



The image shows the GitHub login interface. At the top, the GitHub logo is displayed above the word "Login". Below this, there are two input fields: the first contains the username "rmb" and the second is labeled "Password". At the bottom, there are two buttons: "Login" with a checkmark icon and "Cancel" with an 'X' icon. Below the buttons, there are two links: "Don't have an account? Sign up" and "Forgot your password?".

Figura D.23: Pantalla de login de gitHub

Por último, la aplicación nos devolverá un mensaje de éxito si se ha realizado correctamente el commit:



The image shows a green rectangular box with the text "Push successful" and "Pushed 1 commit to origin/master" in a bold, black, monospaced font.

Figura D.24: Mensaje de éxito en VCS

Observamos los cambios si se han realizado con éxito:

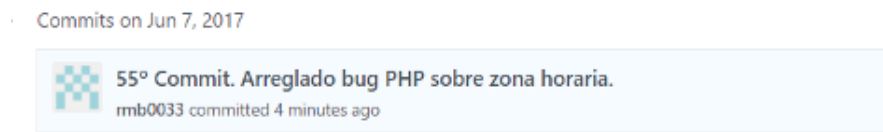


Figura D.25: Commit en github del nuevo commit realizado desde github

## Para ejecutar el proyecto

Simplemente vamos al index.html. Seleccionamos de la barra superior el menú contextual “RUN” y seleccionamos el navegador con el que queremos abrir nuestro proyecto.

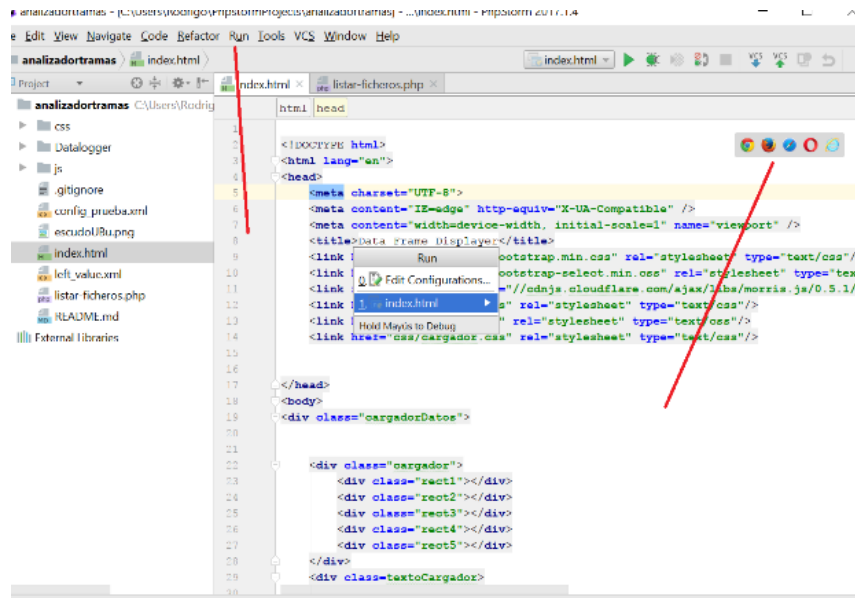


Figura D.26: Commit en github del nuevo commit realizado desde github

Por defecto utilizamos Chrome, que tiene que estar instalado en nuestro equipo para poder ser ejecutado nuestro proyecto.

De esta manera ya tendríamos nuestro equipo funcionando, y de la misma manera que tenemos preparado nuestro entorno de desarrollo podríamos ejecutar nuestro proyecto de forma completa.

## D.5. Pruebas del sistema

Al tratarse una aplicación que básicamente está centrada en un eso en interfaz gráfica, hemos descartado las pruebas unitarias y nos hemos centrado

en realizar bugtesting en la aplicación en ejecución.

Para ello, la aplicación ha sido probada en entornos reales de producción, para verificar el correcto funcionamiento de forma manual, de principio a fin, donde en la última versión, la aplicación no ha producido errores, asegurándonos completamente del buen funcionamiento del proyecto dentro del dispositivo.

Se ha probado la aplicación tanto en navegadores con versiones actuales (1-7-2017) de Google Chrome, como Mozilla Firefox.

## **Documentación de usuario**

---

### **E.1. Introducción**

Para este manual de usuario mostraremos el funcionamiento de la página omitiendo la instalación del servidor. Recomendamos usar como complemento los videotutoriales que se adjuntan en el DVD del proyecto.

### **E.2. Requisitos de usuarios**

Los requisitos para que funcione el servidor son:

- PHP 5.5 [2] o PHP 5.6
- Servidor web, por ejemplo, apache.

Esta aplicación es compatible con los principales navegadores del mercado, entre ellos se ha verificado su uso en:

- Google Chrome
- Mozilla Firefox

### **E.3. Instalación**

Para emular el servidor, utilizaremos el servidor que nos genera PHPStorm, cuyos pasos ya han sido marcados en el manual del programador.

### **E.4. Manual del usuario**

**Pantalla de inicio**



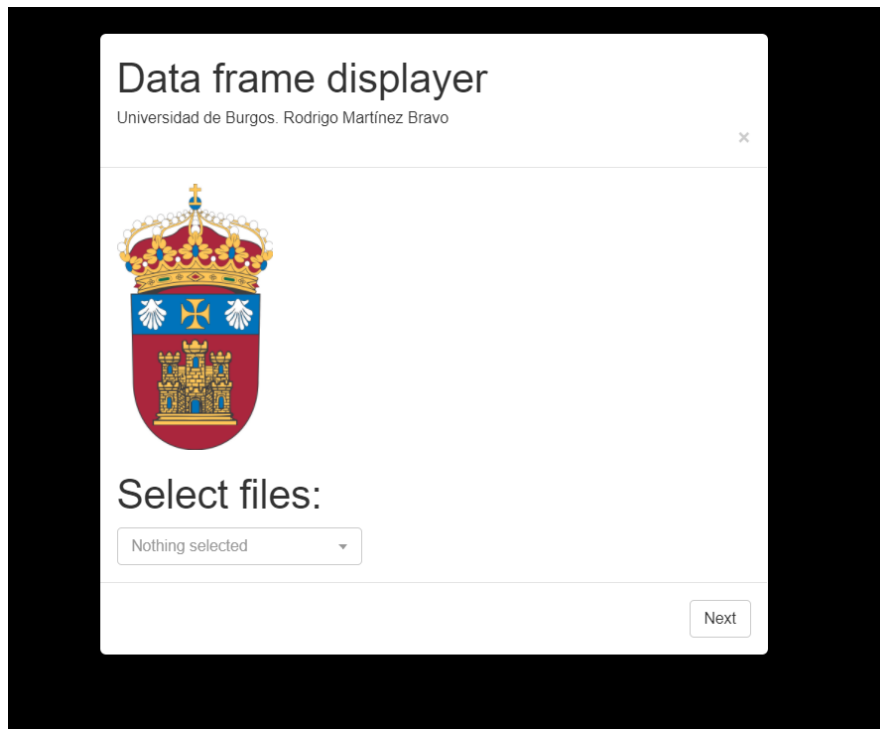


Figura E.1: Pantalla de inicio del proyecto

En esta pantalla el usuario tendrá que seleccionar los ficheros con los que quiere trabajar, en este caso trabajaremos con 2 ficheros distintos. Para seleccionar simplemente hacer click izquierdo, y saldrá un tick verificando que hemos seleccionado dichos ficheros:

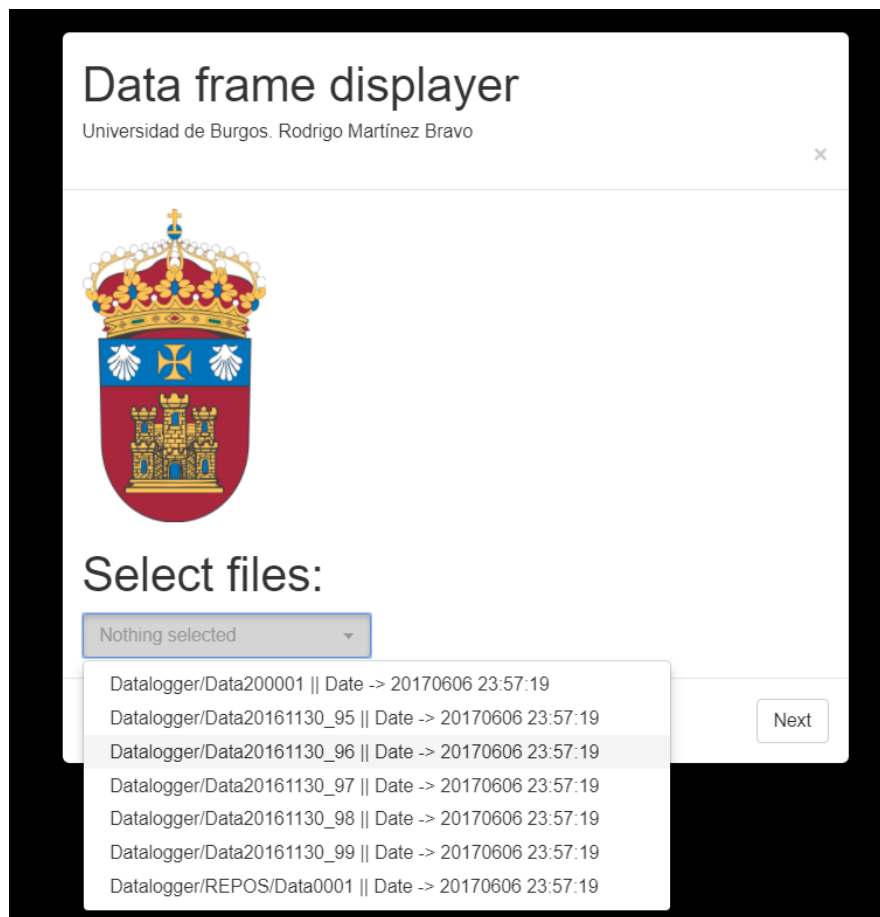


Figura E.2: Selección de archivos

Ahora presionamos siguiente:

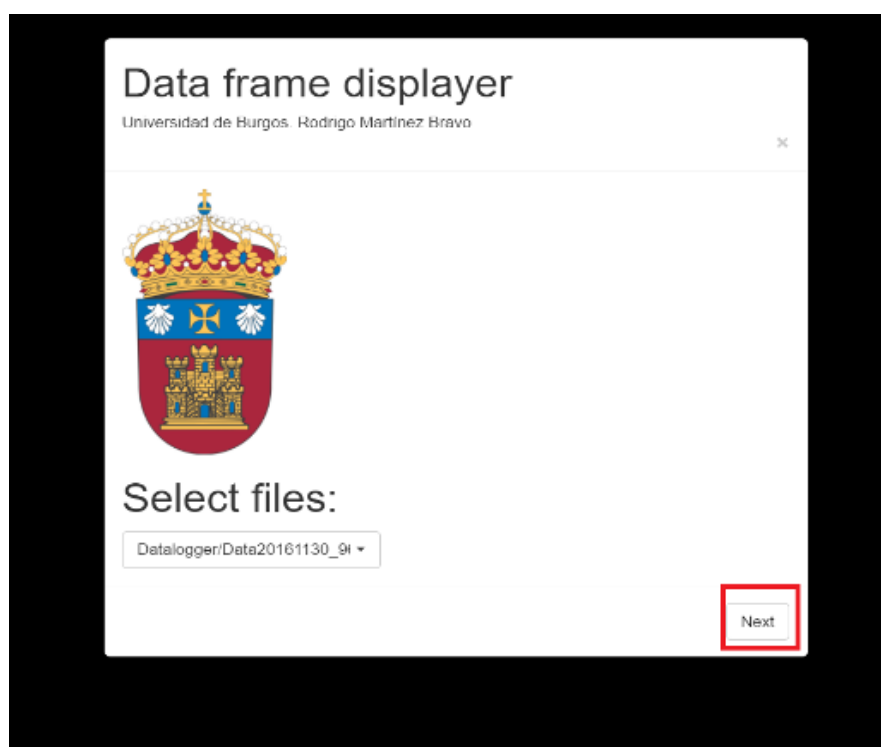


Figura E.3: Salir pantalla de inicio

Ahora nuestro sistema está calculando las bibliotecas (ventana de carga), el navegador no responderá hasta que se haya calculado completamente el diccionario:



Figura E.4: Pantalla de carga

Una vez cargadas las tramas pasaremos a tener una pantalla como esta:

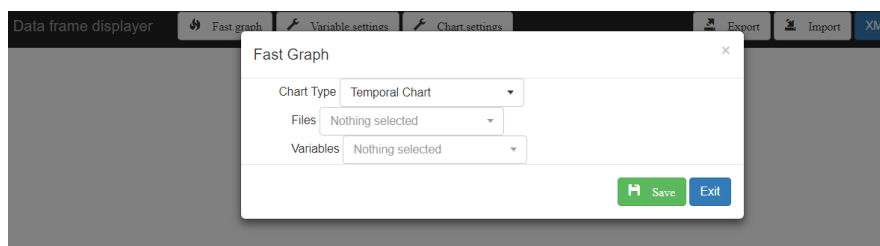


Figura E.5: Pantalla Fast Graph

Por defecto, la aplicación nos abre la opción Fast Graph.

### Fast Graph

Vamos a dibujar nuestra primera gráfica, para ello, seleccionamos los ficheros que queremos que nos muestre de las variables y las variables que queremos graficar (ver Pantalla Fast Graph del apartado anterior). Seleccionaremos 1

fichero y 1 variable. Aunque podemos graficar muchas más variables y seleccionar varios ficheros.

Damos al botón de save, y nos saldrá una alerta en el navegador diciendo que se han guardado los cambios.

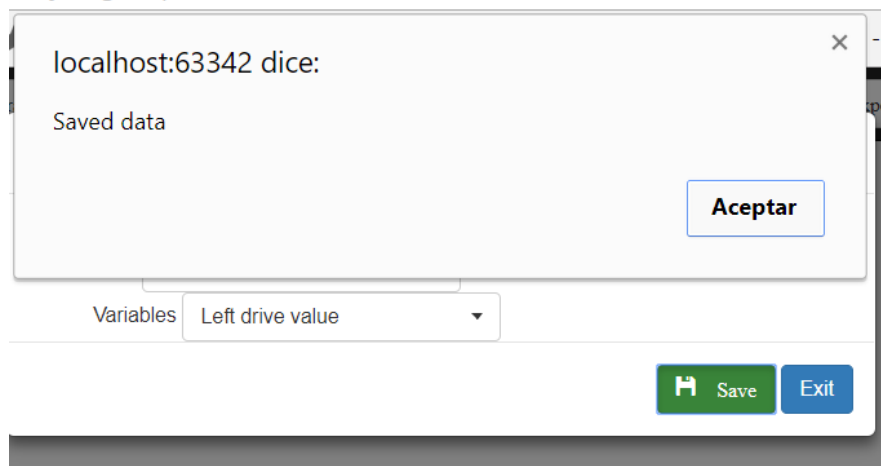


Figura E.6: Pantalla de confirmación

Salimos. Y se nos graficará la variable:



Figura E.7: Pantalla de confirmación

## Añadir y modificar vistas

Respecto al punto anterior, queremos modificar el nombre de “Left drive value FG” por otro más corto, y queremos visualizar en la misma gráfica la misma variable con un escalado de \*20.

Para ello vamos a Variable Settings:

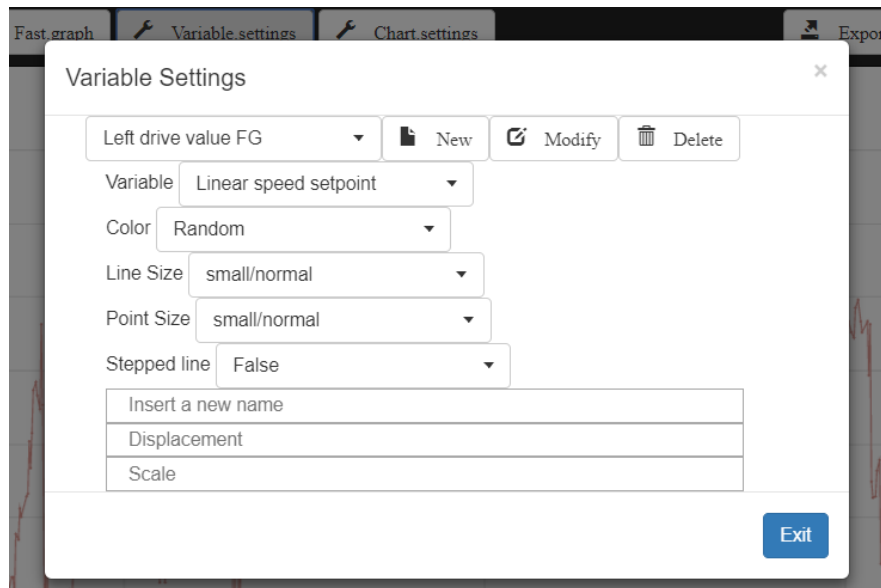


Figura E.8: Opciones de variable

En la parte superior nos está indicando que por defecto está seleccionada esa vista de la variable.

Para ello escribimos un nombre más corto, cambiamos el color a rojo y presionamos “MODIFY”.

Variable Settings ×

Left drive value FG ▼

New ✎ Modify ✎ Delete 🗑

Variable

Linear speed setpoint ▼

Color

Red ▼

Line Size

small/normal ▼

Point Size

small/normal ▼

Stepped line

False ▼

LFG
Displacement
Scale

Exit

Figura E.9: Cambio de nombre de variable.

Una vez dado “Modify”, sabremos que lo hemos realizado con éxito si no tenemos más vistas realizadas dando al desplegable.

Ahora desde Variable Settings añadiremos la misma variable con un escalado de \* 2 y color azul. Y presionaremos el botón NEW.

Variable Settings ×

LDV ▼

New ✎ Modify ✎ Delete 🗑

Variable

Left drive value ▼

Color

Blue ▼

Line Size

small/normal ▼

Point Size

small/normal ▼

Stepped line

False ▼

LDV*2
0
2

Exit

Figura E.10: Creación de nueva vista de variable.

Ahora podremos comprobar que tenemos 2 vistas creadas, pinchando en el desplegable.

Para visualizar los cambios en la gráfica pulsamos el botón exit o pinchamos fuera.

Como vemos se nos visualizarán las 2 variables:

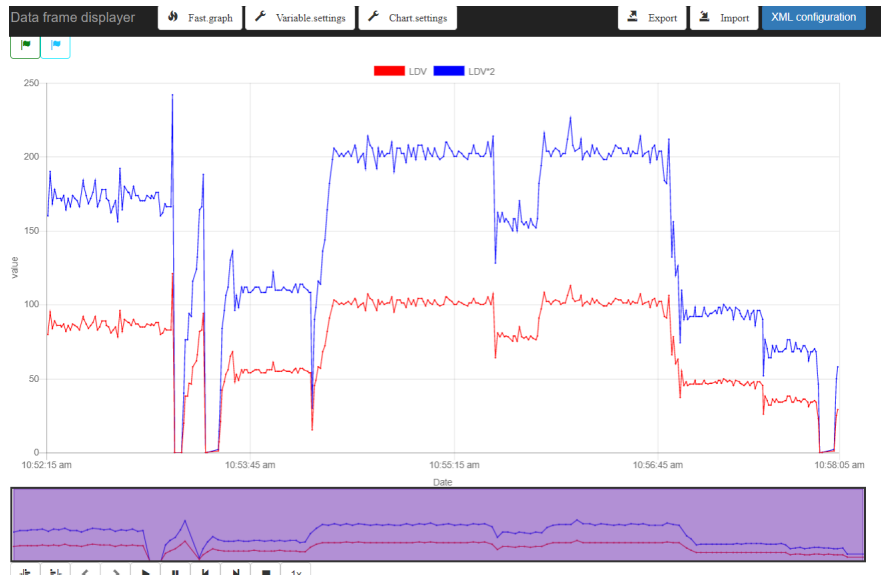


Figura E.11: Resultado creación variables.

En la parte superior de la gráfica, en la leyenda del gráfico, podremos quitar o poner a nuestro gusto las variables pulsando con el botón izquierdo encima de la leyenda.

Como vemos LFG, se ha quedado muy pequeño aunque estemos sólo visualizando gracias a la función de la leyenda. Accedemos a Chart Settings y ponemos la opción de auto-axes adjustment a true.



Chart Settings ×

---

Files 20161130\_95 23:57:19 ▼

Chart Type Temporal Chart ▼

Line Type Line ▼

Auto-axes adjustment True ▼

Axis Y (MAX)
Axis Y (MIN)

Save
Exit

Figura E.12: Cambio de autoajuste de ejes.

Ahora cuando quitamos la vista “LFG\*2”, se nos autoajustarán los ejes y podremos ver si problema a LFG.



Figura E.13: Resultado quitando una variable.

### Importar y exportar vistas.

Si queremos utilizar las mismas vistas tendremos la opción de exportar e importar dichas vistas. Para ello seleccionamos la opción de exportar en el menú superior y nos saldrá una captura como esta:

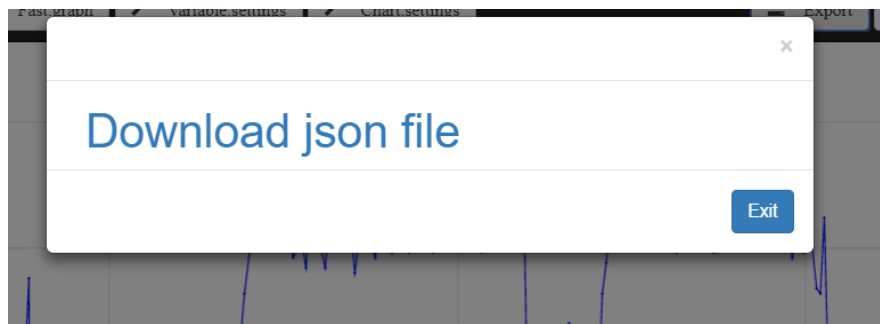


Figura E.14: Opción de exportar vista.

Pinchamos en el enlace y nos descargamos el fichero.

Ahora vamos a iniciar de nuevo la página pero vamos a cargar ficheros diferentes. Cuando nos salga la opción Fast Graph damos exit, y seleccionamos en el menu superior la opción “Import”. Seleccionamos los ficheros en los que queremos aplicar la vista que vamos a importar y seleccionamos el fichero que deseamos:

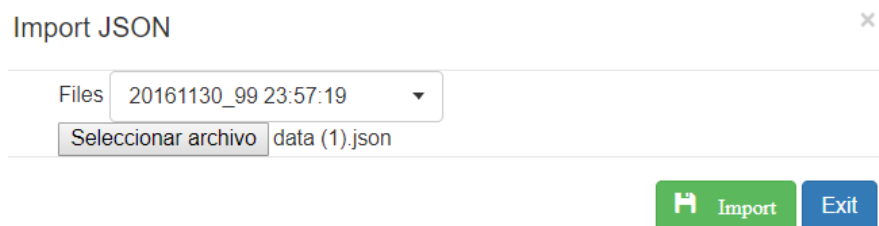


Figura E.15: Opción de importar vista.

Damos al botón de import, y nos saldrá una alerta diciendo que se han guardado los cambios.

Pulsamos exit y observamos los cambios:

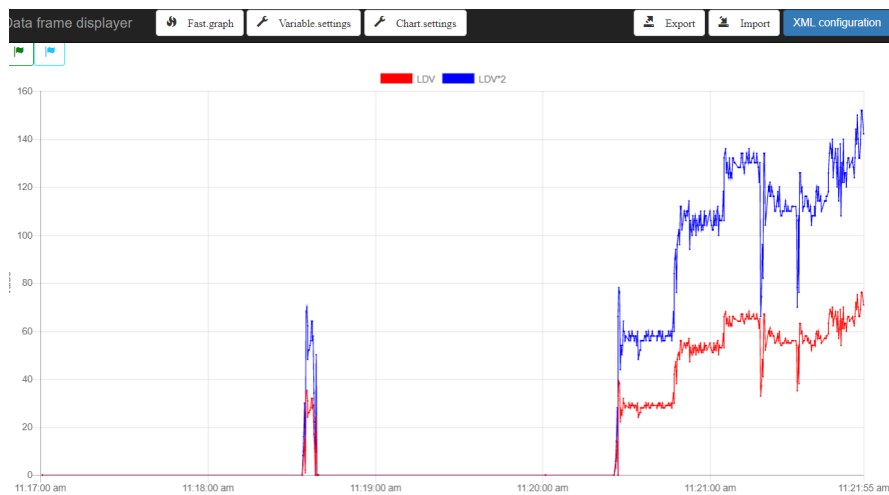


Figura E.16: Solución al importar una vista.

### Creación de gráficas sin Fast Graph.

De nuevo iniciamos la página y seleccionamos un fichero. Descartamos la opción de Fast Graph. Y presionamos en chart settings. Esta vez vamos a representar una gráfica XY, para ello seleccionamos los ficheros que queremos trabajar y seleccionamos en el Chart Type “X Y Chart”. Damos el botón guardar (nos saldrá una alerta de que se han guardado cambios).

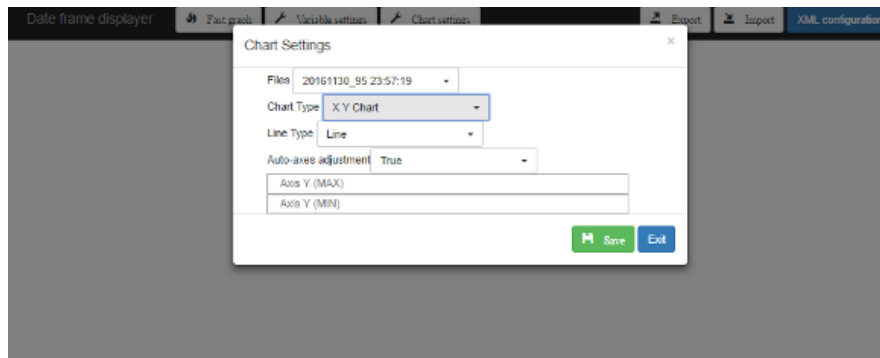


Figura E.17: Opciones gráficos.

Ahora vamos a variable settings e introducimos los 2 valores que queremos visualizar en el eje x y, con su respectivo nombre y sus respectivas opciones. Damos botón guardar (nos saldrá una alerta). Después daremos el botón exit o pulsaremos fuera.

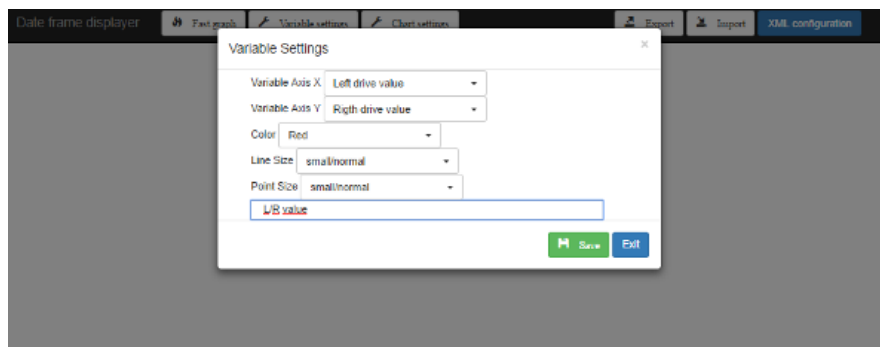


Figura E.18: Creación variable XY.

Se nos graficará la gráfica XY.

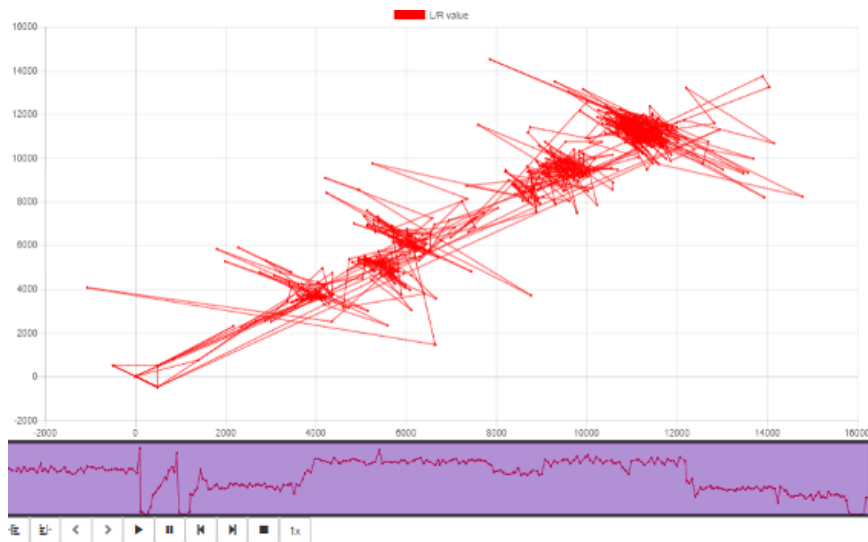


Figura E.19: Solución gráfica XY.

### Configuración del XML.

Tenemos la opción de aplicar un XML distinto siempre que nosotros queramos al de defecto. Para ello damos en el menú la opción de XML configuration y podemos cargar otro xml con distinta configuración de tramas CAN. Nosotros utilizaremos el xml que hay en nuestro proyecto llamado left drive value.

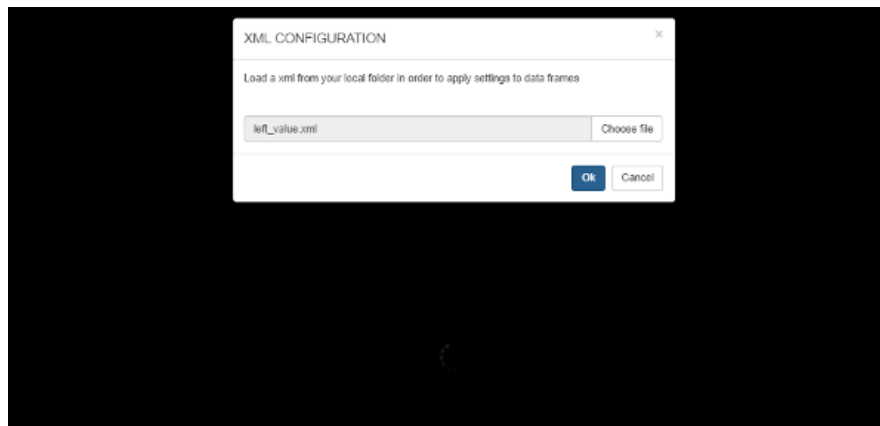


Figura E.20: Cambiar XML.

Hay que tener en cuenta que puede tardar un tiempo pues tiene que recalcular toda la biblioteca.

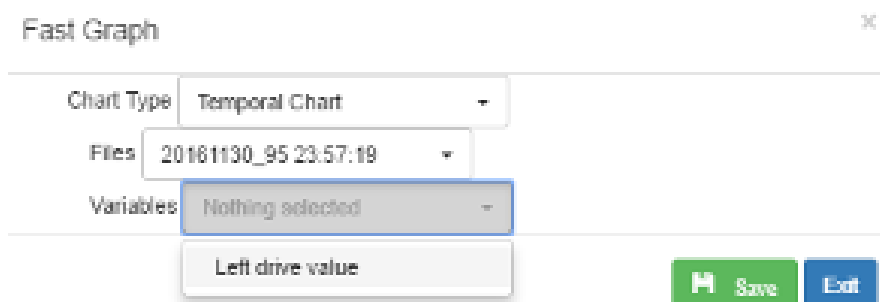


Figura E.21: Único resultado en variables.

Ahora en fast graph, en este ejemplo, la opción de graficar Left drive value será la única opción disponible dentro de las variables.



Figura E.22: Resultado con único resultado.

### Cursores.

Para utilizar los cursores, simplemente tenemos que pinchar en la parte superior izquierda escogiendo que cursor queremos utilizar (verde o azul). Una vez seleccionado el cursor, se nos quedará como fijado y podremos dibujarlo en la gráfica, presionando el click izquierdo las veces que nosotros consideremos oportuno hasta ajustar el punto exacto en el que queremos dicho punto.

Para seleccionar el segundo cursor, pinchamos de nuevo en el cursor seleccionado y pinchamos en el otro. Si dibujamos el segundo cursor (click izquierdo gráfica), nos creará automáticamente una tabla con la diferencia y suma de los 2 cursores.



Figura E.23: Resultado con único resultado.

### Utilizar la ventana temporal.

En la ventana temporal disponemos de un zoom horizontal. Para utilizar dicho zoom haremos lo siguiente: Limite izquierdo y derecho del zoom:



Figura E.24: Ventana temporal: en rojo zoom izquierdo, azul derecho.

Como vemos en la imagen en el cuadrado rojo tendremos el zoom izquierdo, y en el cuadrado derecho el zoom derecho. Para cambiar el zoom, pincharemos en uno de los 2 y podremos modificar el limite de ese lado pinchando en la gráfica inferior o utilizando los botones de desplazamiento. Hay que tener en cuenta que cuando utilizamos el click en la gráfica, se aproximará al punto más cercano. Así podremos modificar a nuestro gusto la parte que queremos visualizar de la gráfica.



Figura E.25: Zoom con ventana temporal.

Recordamos que cada vez que se hace zoom se hace un diezmado automático, lo que nos permite ver puntos que antes no podíamos ver sin realizar el zoom.

Dentro de las otras opciones de la ventana temporal tenemos un modo de reproducción, donde la gráfica se irá reproduciendo y podremos visualizar el punto actual gracias a ese zoom inferior. Además podremos incrementar la velocidad de reproducción pinchando en 1X, de mayor a menor entre las distintas opciones. Con el botón pause podremos pausar la reproducción de la gráfica.





Figura E.26: Modo reproducción.

Con el botón STOP volveremos al punto por defecto del zoom.

### Utilizar búsquedas y filtros

Esta herramienta de graficado permite realizar búsquedas y filtros aplicando una consulta (ver en la memoria la sintaxis de las consultas). Con el ejemplo anterior realizaremos un filtrado y una búsqueda.

Si presionamos el primer icono de consulta (filtro) se filtrarán dichos datos y solo veremos los que se cumplen. El resultado en este ejemplo es:

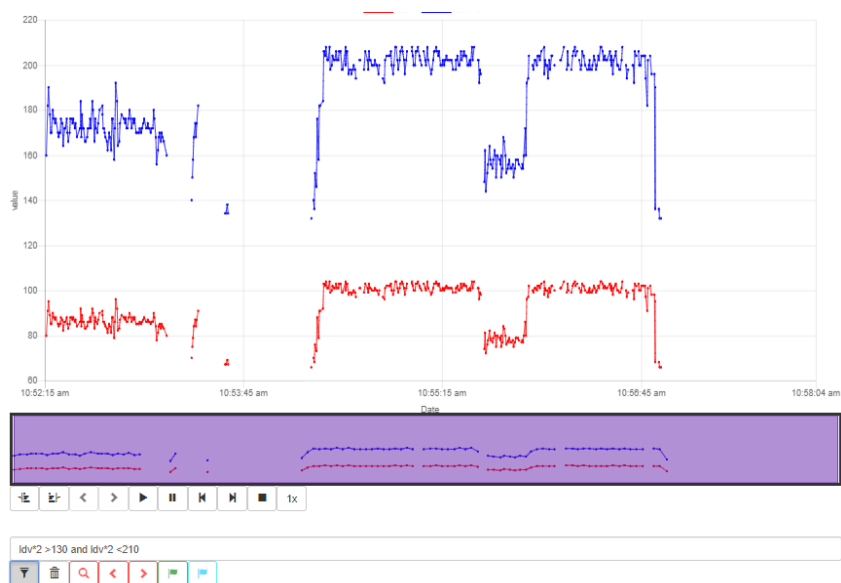


Figura E.27: Filtro en la parte inferior.

Si presionamos borrar filtro o de nuevo al botón de filtro, se eliminará el filtro.

La opción más interesante se presenta con el botón buscar. Con la misma consulta, nos situaría en el primer punto que se cumpliría con un cursor rojo. A continuación nosotros tenemos la opción de hacer nuevas consultas partiendo de ese cursor y seleccionando la flecha de izquierda o derecha para calcular el primer punto respecto a ese cursor.

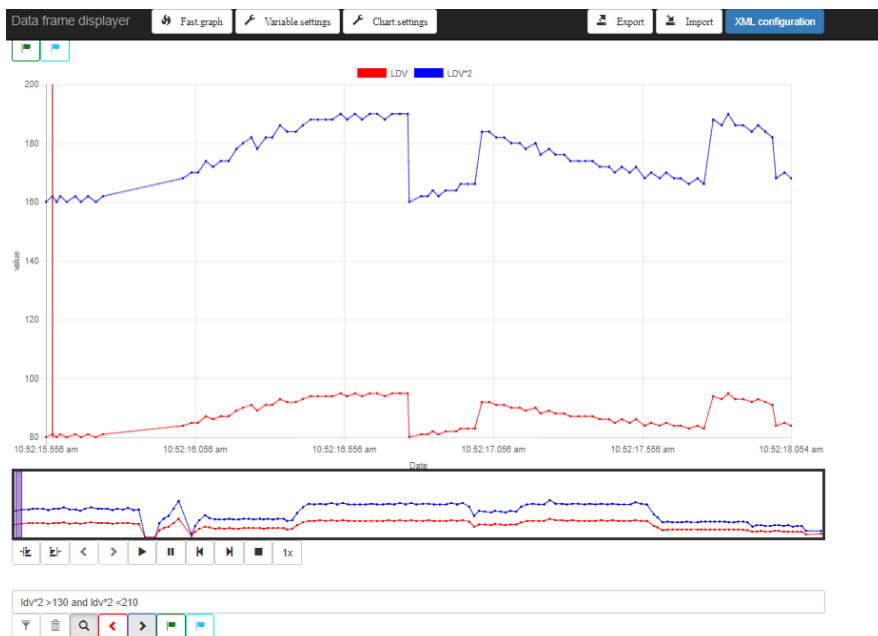


Figura E.28: Búsqueda con zoom automático.

Si no se trata del primer punto, la herramienta hará un zoom automático para poder ver mejor ese punto. Por ello, con el segundo punto realizaría un zoom.

También tenemos la opción de asignar los cursores anteriores al punto de la búsqueda.

Por ejemplo, asignamos el cursor verde a la primera consulta y el azul a la segunda (consulta  $ldv = 91$  y la otra  $ldv=94$ ). Y obteniendo el segundo punto, dando el botón stop para tener una visión global:



Figura E.29: Asignación cursores con el cursor de búsqueda.

---

## Bibliografía

---

- [1] JetBrains. PhpStorm, 2017. [Internet; consultado 13-mayo-2017] <https://www.jetbrains.com/phpstorm/>.
- [2] PHP. Php 5.5, 2017. [Internet; consultado 3-mayo-2017] <http://php.net/manual/es/migration55.changes.php>.
- [3] Wikipedia. Git, 2017. [Internet; consultado 3-marzo-2017] <https://es.wikipedia.org/wiki/Git>.
- [4] Wikipedia. Github, 2017. [Internet; consultado 14-abril-2017] <https://es.wikipedia.org/wiki/GitHub>.
- [5] Wikipedia. Metodología scrum, 2017. [Internet; consultado 4-junio-2017] [https://es.wikipedia.org/wiki/Scrum\\_\(desarrollo\\_de\\_software\)](https://es.wikipedia.org/wiki/Scrum_(desarrollo_de_software)).
- [6] Wikipedia. Xml, 2017. [Internet; consultado 3-marzo-2017] [https://es.wikipedia.org/wiki/Extensible\\_Markup\\_Language](https://es.wikipedia.org/wiki/Extensible_Markup_Language).