Desarrollo de una librería en JavaScript para manipulación gráfica de datos

David Armero Descalzo
daardes@teleco.upv.es

Tutorizado por:

Valeriana Naranjo Ornedo
vnaranjo@dcom.upv.es
y
Félix José Fuentes Hurtado
ffuentes@upv.es

Trabajo Fin de Grado presentado en la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Graduado en Ingeniería de Tecnologías y Servicios de Telecomunicación
Curso 2017-2018
Valencia, septiembre de 2018.

Escuela Técnica Superior de Ingeniería de Telecomunicación
Universitat Politècnica de València
Edificio 4D. Camino de Vera, s/n, 46022 Valencia
Tel. +34 96 387 71 90, ext. 77190 • ext. 77199
www.etsit.upv.es

Agradecimientos

Me encuentro ante la redacción de lo que se convertirá en el último trabajo que realizo como alumno de la ETSIT de Valencia. Mi trayectoria en esta escuela la podría definir con un adjetivo por encima de todos, "ardua". Nunca me han sido fácil estos estudios, no sé si se debió a una mala elección, a una falta de constancia por mi parte o a la inherente dificultad que supone la realización de los mismos.

Pero no por ello me arrepiento, todo lo contrario, me ha aportado gran parte de los valores y un conocimiento en diferentes ámbitos científicos que dudo mucho que puedas encontrar en otras escuelas. Pero si tengo que decir qué he aprendido por encima de todo una vez acabado los estudios diría, que aquí nos enseñan a levantarnos, aunque te intenten tirar mil veces.

Pero la verdad, es que mucha veces para levantarnos, no solo puedes confiar en que tú solo vas a poder conseguirlo, ya que no es así. Uno se puede levantar tres, cuatro veces, pero sin lo que hay detrás de uno mismo sería imposible.

Es por ello, que quería agradecer ese apoyo, sin el cual hoy no podría estar redactando estas lineas. Ellos son mi **familia**, pero no sólo la de sangre, sino la que yo he querido, que hoy en día forme parte de mi vida. Por ello quería agradecer a:

Moisés, mi hermano y apoyo, gracias por ayudarme cuando te he necesitado.

Mis tíos, Santi y Yolanda, y mi prima Andrea, gracias por vuestro apoyo incondicional.

Mis abuelos, Raúl y Juan José, que aunque no puedan leer este trabajo sé que estarán orgullosos, **y mi abuela Elisa**, gracias por estar siempre a nuestro lado, querernos y cuidarnos, pese a los kilómetros que nos separan.

Mis amigos de toda la vida, (**Borja, Deivid, Negro, Sanchis**) sois un pilar fundamental en mi vida y os quiero como hermanos.

Rafa, mi único amigo que me llevo de esta carrera, gracias por hacérmela más llevadera.

Mis tutores, Valery y **Félix**, gracias por darme un proyecto muy interesante y ayudarme/aconsejarme en todo lo necesario para realizarlo.

Pero, por encima de todo, cuatro personas merecen que las mencione aquí:

Mi abuela Amalia, mujer incansable, buena, generosa, trabajadora, preciosa etc. no puedo ni podré agradecerle todo lo que ha hecho por mi, me ha cuidado y querido como a un hijo, y yo como a una madre, todo lo que diga es poco.

Mis padres, **Julia** y **Pedro**, luchadores y buenas personas, me han dado todo lo necesario para convertirme en lo que hoy soy y que aunque no suelo decírselo mucho, les quiero. Nunca podré agradecer todo lo que hacéis por mí.

Por último a **mi pareja, María**, aunque llevemos cinco años, mereces estar a la misma altura que mis padres y mi abuela, ya que en ese tiempo me has demostrado la clase de persona que eres y aquí la lista de adjetivos, como en los otros casos, se quedaría corta, gracias por todo, sin ti no me habría levantado tan rápidamente. Te quiero.

Resumen

El Big Data está en pleno auge. Millones de datos se recogen cada día permitiendo caracterizar desde nuestra actividad en internet hasta los sitios a los que vamos. No en vano, se considera que los datos son la nueva materia prima del siglo XXI. De este modo, se hace evidente la necesidad de herramientas que permitan su visualización y manipulación, asistiendo en procesos como la representación, la limpieza o el cribado de datos. Esto, sumado a la versatilidad de las aplicaciones web, pone de manifiesto la utilidad de una librería que permita el manejo y visualización de grandes cantidades de datos de una forma fácil y amena.

El objetivo principal del trabajo fin de grado que aquí se propone es el de desarrollar una librería en JavaScript que permita la manipulación gráfica de grandes volúmenes de datos. En concreto, se pretende desarrollar una librería para ser usada en los navegadores más comunes que permita cargar un conjunto de datos multidimensional en memoria y aplicar operaciones sobre este. Estas operaciones incluirán la selección gráfica de subconjuntos, la representación de dichos datos mediante distintos métodos, el cribado y limpieza de los mismos, y la aplicación de distintas técnicas de reducción de dimensionalidad que posibiliten su visualización en 2D sin descartar información importante.

David Armero Descalzo daardes@teleco.upv.es
Valeriana Naranjo Ornedo vnaranjo@dcom.upv.es
Félix José Fuentes Hurtado ffuentes@upv.es

Palabras Clave: Visualización de datos, Citometría de flujo, JavaScript, HTML5, Flow Cytometry Standard

Acrónimos

HTML HyperText Markup Language

CSS Cascading Style Sheets

API Application Programming Interface

FCS Flow Cytometry Standard

WWW World Wide Web

ADC Application Programming Interface

CERN Conseil Européen pour la Recherche Nucléaire

WINMDI Windows Multiple Document Interface for Flow Cytometry

HTTP Hypertext Transfer Protocol

NPM Node Package Manager

DOM Document Object Model

Índice general

Resumen III

Índice general IX

1 Introducción 1

2 Estado del arte 3

2.1 Aplicaciones similares . . . 3
2.1.1 WINMDI . . . 3
2.1.2 FCS Express . . . 4
2.1.3 Cytobank . . . 5

3 Fundamentos 7

3.1 Citometría de flujo . . . 7
3.1.1 Análisis de los datos de un citómetro de flujo . . . 8
3.1.2 Formato de fichero .fcs . . . 9
3.2 HTML5 . . . 10
3.2.1 HTML . . . 11
3.2.2 CSS . . . 12
3.2.3 JavaScript . . . 12
3.2.4 Node.js . . . 13
3.2.5 Gestor de paquetes NPM . . . 14
3.2.6 APIs . . . 14
3.2.7 Librerías externas . . . 14
3.2.8 Librería Socket.io . . . 14
3.2.9 FCS para JavaScript . . . 15
3.2.10 Chart.js . . . 15
3.2.11 HTTP . . . 16
3.2.12 API Express . . . 17
3.2.13 jQuery . . . 17

4 Diseño y desarrollo de la aplicación 19
4.1 Estructura de la aplicación 19
4.1.1 Diseño Web 19
4.1.2 El Servidor 24
4.2 Funciones del cliente 26

5 Conclusiones 37

Bibliografía 39

Capítulo 1

Introducción

En la actualidad, los laboratorios de biomedicina que se encargan del estudio citométrico de las células utilizan software de pago [7, 9], destinando una parte de sus presupuestos a este tipo de licencias. El estudio citrométrico de la sangre es un método para la detección de múltiples enfermedades, como el cáncer, entre otras. Los datos que se recogen del citómetro de flujo [4] se almacenan en ficheros `.fcs` [5]. Al igual que las licencias citadas anteriormente, el software de visualización de estos ficheros son de pago.

Es por ello que la principal motivación del presente proyecto es crear una herramienta gratuita para visualizar los datos de los ficheros `.fcs`, de modo que cualquier persona pueda tanto analizar como visualizar este tipo de ficheros de manera gratuita, lo que puede conllevar a que los laboratorios destinen gran parte de sus presupuestos a lo verdaderamente importante: investigar posibles avances a las enfermedades actuales, sin destinar grandes cantidades de dinero a adquirir este tipo de software. También existe una finalidad educativa en la intención de este proyecto, ya que mediante la presente plataforma pretendemos que los estudiantes también puedan tener un acceso libre y gratuito.

La plataforma que se desarrolla a lo largo de este trabajo final de grado consiste en diseñar un lector y visualizador de ficheros `.fcs` a través de una página web de libre acceso. Para ello, solo necesitamos un fichero de este tipo para poder abrirlo, visualizarlo y trabajar con él, gracias a las herramientas que hemos desarrollado, así de sencillo.

En concreto, para la creación de esta aplicación hemos empleado `HTML5` (HyperText Markup Language 5), lo que nos ha permitido valernos de infinidad de módulos ya desarrollados.

En este trabajo, describimos el estado del arte en el capitulo dos, donde se comenta cómo son las aplicaciones actuales y cuál es su tecnología, explicando las funciones de cada una de ellas.

En el capitulo tres “Fundamentos” se describe todo lo relacionado con los estudios relativos a la citrometría de flujo, HTML5 y demás tecnologías inherentes a la creación de esta plataforma.

Asimismo, en el capítulo cuarto se analiza cómo hemos diseñado y desarrollado nuestra aplicación.

Finalmente, en el capítulo cinco se presentan las conclusiones y el trabajo futuro.

Capítulo 2

Estado del arte

En este capítulo vamos a explicar la tecnología que utilizan actualmente los biólogos para la visualización de datos a través de aplicaciones similares con las que ellos trabajan. Así seremos conscientes de las herramientas disponibles en el mercado y en qué podemos nosotros contribuir con nuestro proyecto, para mejorar y ayudar a la comunidad científica [3].

2.1 Aplicaciones similares

Actualmente hay una gran variedad de software para el análisis de datos de archivos `.fcs`. Cada uno ofrece unas características diferentes y por ellas son elegidas por los científicos dependiendo de sus necesidades. Nosotros vamos a describir los programas más relevantes y utilizados por los laboratorios [4]:

- `WINMDI`
- `FCS Express`
- `Cytobank`

2.1.1 WINMDI

`WINMDI` [16] son las siglas en inglés de (Windows Multiple Document Interface for Flow Cytometry) lo que en castellano significa, interfaz de múltiples documentos de Windows para citometría de flujo, es un software gratuito de análisis de citometría de flujo basado en ordenador, que fue desarrollado por Joe Trotter[5]. `WINMDI` es una aplicación que lee la mayoría de los archivos compatibles con `FCS 2.0` que existían en el momento de la última compilación en el 2000. Esta herramienta no reconoce datos digitales `FCS 3.0 BD`.

`WINMDI` es el software más básico y no es muy popular en el ámbito profesional, debido a sus escasas herramientas ya que tan sólo es capaz de hacer gráficas muy básicas. Su principal utilidad está en el ámbito educativo ya que es un programa gratuito y aporta unas funciones básicas para el manejo de los datos.

En la Figura 2.1 podemos ver el entorno gráfico del programa a la hora de analizar las muestras.

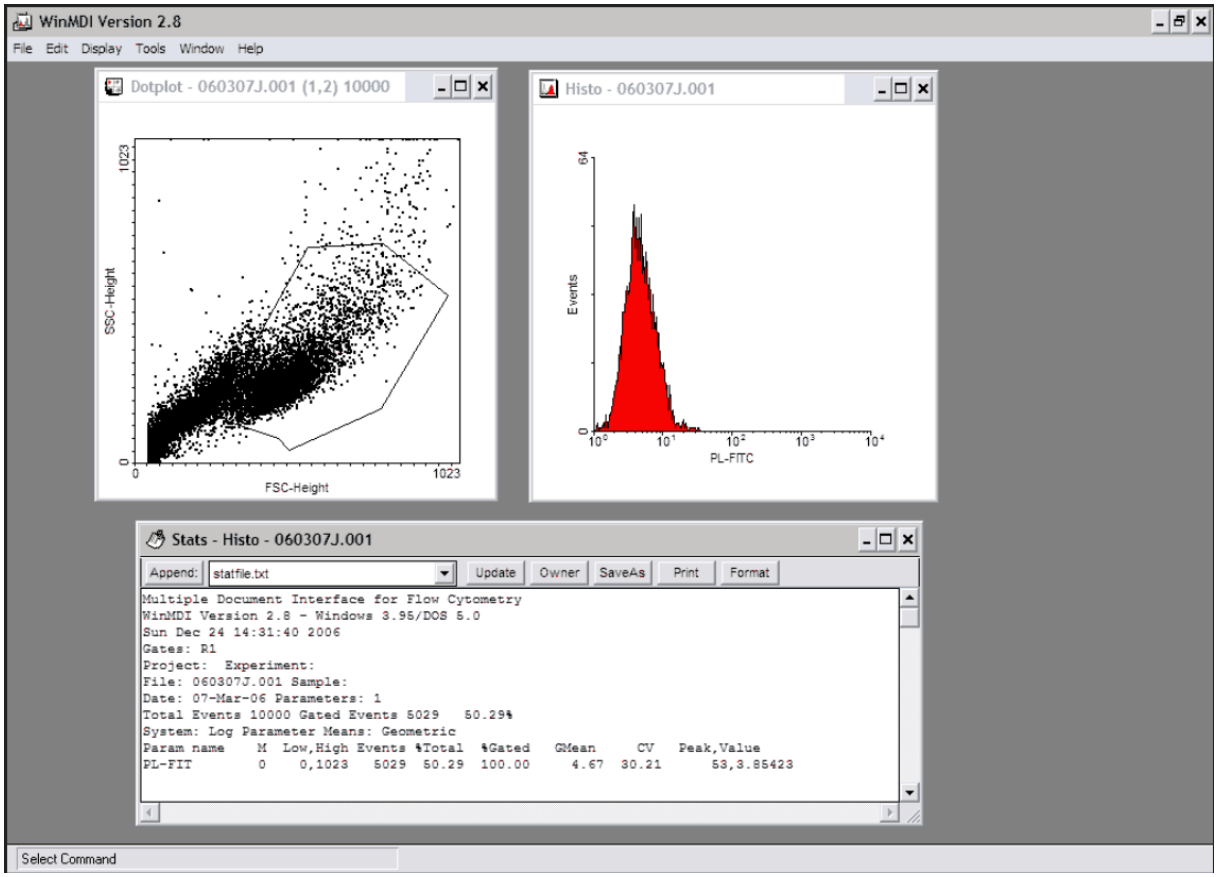

Figura 2.1: Captura del software `WINMDI` [16].

2.1.2 FCS Express

`FCS Express` [9] es una herramienta muy similar a la anteriormente descrita `WINMDI`, pero con funciones mucho más avanzadas, como la exportación directa a `PowerPoint`, procesamiento por lotes, análisis de proliferación, gráficos de barra, 3D etc. `FCS Express` también se desarrolla activamente, se mejora continuamente y es compatible con archivos FCS de cualquier citómetro de flujo. Aunque `FCS Express` no es un software gratuito, De Novo Software ofrece una versión menos potente por un precio mucho más inferior al que tiene el profesional y también dispone de una versión de prueba por un periodo de 30 días.

Como observamos en la Figura 2.2, `FCS Express` es un software muy profesional con infinidad de herramientas, gráficos etc. Todo ello hace de este software un producto mucho más apto que `WINMDI` para laboratorios.

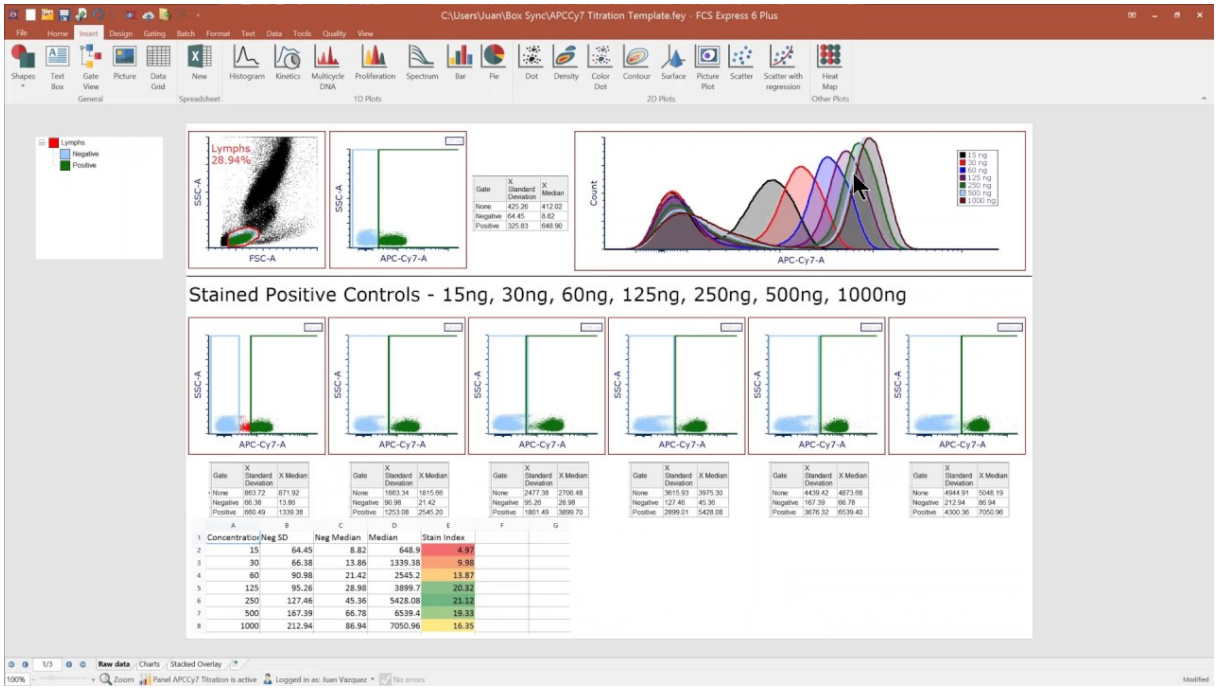

Figura 2.2: Captura del software FCS Express [5].

2.1.3 Cytobank

`Cytobank` [7] es, en la actualidad, el software más utilizado por la comunidad científica y ello porque es una plataforma en la nube que acelera la productividad de la investigación permitiendo analizar y visualizar simultáneamente múltiples conjuntos de datos de células individuales. En concreto, los citómetros analizan más de treinta parámetros por célula y millones de células por muestra.

En particular, `Cytobank` se basa en el concepto de análisis de células individuales con todas las visualizaciones de datos conectadas a los eventos celulares individuales, lo que permite una comunicación ideal entre investigadores clínicos, científicos e informáticos.

Entre las herramientas que más destacan en `Cytobank` son las siguientes:

1. Sunburst: Visualiza jerárquicamente los parámetros celulares y los datos estadísticos, siendo de gran ayuda a la hora de interpretar.
2. Dose Response (Respuesta a la Dosis): Analiza la respuesta de una muestra sometida a una dosis, llegando a tener una resolución de una única célula, permitiendo un análisis en detalle, que podría no verse en un estudio de una muestra completa.
3. Mapas de Calor:Permite visualizar las muestras en diferentes condiciones.
4. SPADE: Es una herramienta que agrupa células fenotípicamente similares en un árbol jerárquico de interpretación simple para un análisis intuitivo.
5. viSNE: Es una herramienta de visualización de datos, con gráficas 2D con la información biológica de los parámetros, facilitando a los científicos no solo la visualización de los subconjuntos biológicos interesantes y raros, sino también el control de eventos de una sola célula en muestras distintas.

6. Superposiciones de puntos e histogramas: Es una herramienta que superpone los datos de células individuales para poder comparar tendencias a lo largo del tiempo, diferencias entre pacientes, relaciones de jerarquía de poblaciones, etc.
7. CITRUS: Esta herramienta subdivide automáticamente datos de células individuales e identifica biomarcadores predictivos con poder estadístico entre grupos de muestras.

Como se puede observar se trata del software más potente actualmente, y por eso lo hemos analizado con más detalle en las Figuras 2.3 y 2.4.

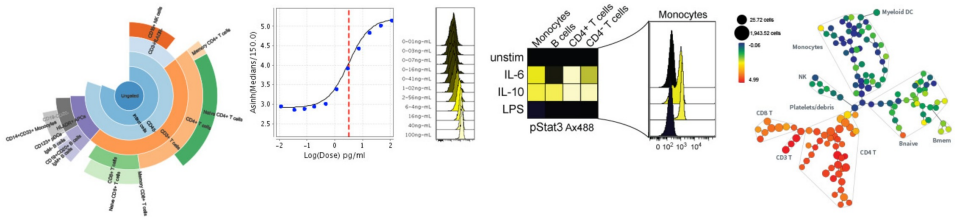

Figura 2.3: Muestra de las herramientas de Cytobank de izquierda a derecha: Szunburst, Dose, Mapas de Calor, SPADE [7].

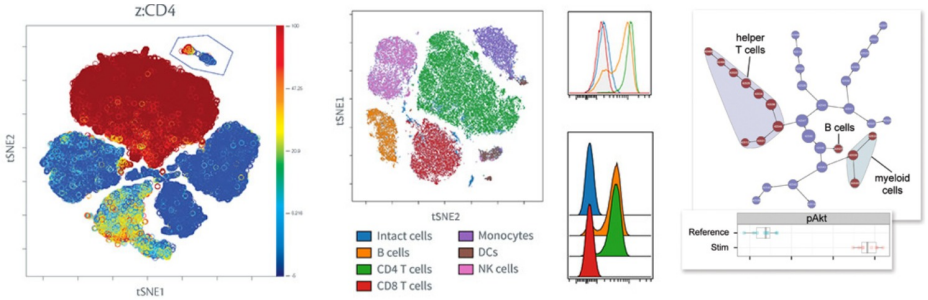

Figura 2.4: Muestra de las herramientas de `Cytobank` de izquierda a derecha: viSNE, Superposiciones de puntos e histogramas, CITRUS [7].

Capítulo 3

Fundamentos

En este capítulo se describen todos los conceptos teóricos utilizados para la realización de este trabajo. Empezamos describiendo lo que es la citometría de flujo y como se analizan los datos recogidos por esta tecnología, en los ficheros `.fcs`*. Posteriormente hablamos de la tecnología del diseño web* `HTML5` *donde englobamos el* `HTML`*,* `JavaScript` *y* `CSS`*. Por último dentro de* `JavaScript` *nos adentramos en el entorno de ejecución del servidor* `Node.js` *y el uso de librerías.*

3.1 Citometría de flujo

La gran evolución que ha sufrido el diagnóstico clínico de enfermedades en los últimos tiempos viene propiciada por las nuevas tecnologías o métodos, llevados a cabo en el análisis celular. De entre todos estos métodos destaca, por encima de todos, la citometría de flujo.

Pese a ser una técnica de análisis celular bastante reciente, sus especiales características han llevado a ésta a una implantación muy rápida en los laboratorios clínicos.

El uso de esta tecnología ha ayudado al estudio de diversas ciencias tales como: oncología, hematología, inmunología, biología celular o la anatomía patológica, entre otras [3, 6].

Si comparamos la citometría de flujo con otros métodos bioquímicos para el análisis de las células, observamos que esta tecnología puede proporcionar información cuantitativa de cada una de las células, así como, identificar subpoblaciones de células diferentes en una muestra. Por el contrario, el resto de métodos proporciona el resultado promedio para una muestra, ahí es donde estriba la principal diferencia y, por tanto, la gran aceptación que ha tenido en el ámbito científico.

Esta técnica es usada de forma habitual para el diagnóstico y seguimiento de múltiples enfermedades, como la leucemia y el VIH, entre otras. Además, se utiliza en ensayos clínicos y en investigación básica y práctica [4].

El precedente de lo que hoy en día conocemos como citómetro de flujo basado en el principio separador de células fue desarrollado por Mack Fulwyler, en 1965. Un citómetro de flujo actual tiene los siguiente componentes:

- Celda de flujo laminar.

- Sistema de medición, que suele ser de conductividad (Principio Coulter)[1]
- Sistema de iluminación.
- Detector y conversor de señales ADC.
- Sistema de amplificación, lineal o logarítmico.
- Ordenador para analizar las señales.

La citometría de flujo es un método analítico que proporciona de forma ágil una medición de las características físicas y químicas de células suspendidas en líquido que al interferir con una fuente de luz emiten una señal individualizada.

En particular, el método consiste en pasar por delante de un láser células u otras partículas en suspensión alineadas de una en una, distinguiéndose la información suministrada entre la que se genera por la dispersión de la luz y la relacionada con la emisión de luz. Esta información es procesada por un ordenador.

A modo de ejemplo, podría decirse que un citómetro de flujo es análogo a un microscopio, puesto que permite cuantificar diversos parámetros celulares con alta precisión; si bien, la principal diferencia radica en que no produce imagen de la misma [3].

La citometría de flujo se caracteriza, pues, por medir diversos parámetros celulares así como cualquier componente celular. En concreto, esta tecnología mide los parámetros nucleares, citoplásmicos, de superficie y extracelulares [6].

La principal ventaja de la citada técnica consiste en la facultad de cuantificar tantos parámetros como anticuerpos se dispongan para ello, identificados con diversos fluorocromos, que son productos químicos que al ser estimulados con fotones con longitud de onda comprendida entre determinados rangos (de excitación) emiten a su vez otro fotón de longitud de onda mayor (de emisión). Esto posibilita un análisis celular multiparamétrico, ágil y preciso [5].

3.1.1 Análisis de los datos de un citómetro de flujo

El muestreo de los datos a partir de un citómetro de flujo se denomina adquisición. La adquisición se realiza mediante el software de un ordenador que está conectado a dicho citómetro.

El software calibra varios de los parámetros para cada una de las muestras analizadas, asegurándose que éstos se han ajustado debidamente [4].

Una vez hemos llevado a cabo la adquisición de estos datos, hay tres formas de representarlos gráficamente, mediante:

- Histogramas
- Gráficos de puntos (Dot-Plot) o bidimensionales.
- Gráficos tridimensionales.

[1] El Principio de Coulter hace referencia al uso de un campo eléctrico para contar y dimensionar partículas que se encuentran en suspensión en un líquido conductor.

En estos gráficos podemos separar de una manera sencilla diferentes regiones delimitadas por la intensidad de la fluorescencia, creando con ello subgrupos denominados "gates".

Frecuentemente, como consecuencia del solapamiento de los espectros de emisión, la representación de tales gráficos se lleva a cabo mediante escalas logarítmicas [6].

El análisis de los datos se desarrolla mediante softwares de pago, tales como `Cytobank`,`FCS Express`, `WINMDI`, etc., (descritos en el capítulo 2) y se guardan en ficheros `.fcs` [7, 16, 9].

Tras la recopilación de datos, no hay ninguna razón por la que tengamos que tener conectado el citómetro al ordenador, pudiendo desconectarse éste, ya que tales ficheros pueden abrirse en cualquier ordenador del mundo que disponga de dichos softwares [3].

En la Figura 3.1 aparece un esquema del proceso de adquisición de datos y análisis citométrico descrito anteriormente 3.2.9.

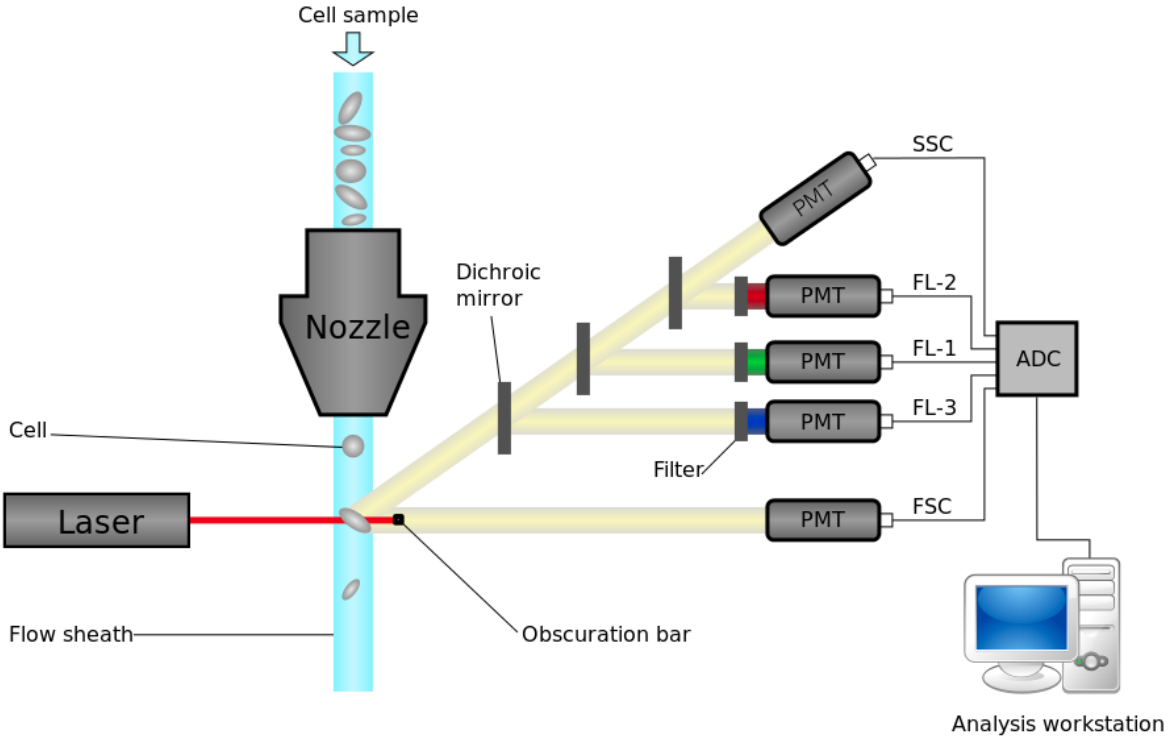

Figura 3.1: Ejemplo de adquisición de datos por un citómetro de flujo [5].

3.1.2 Formato de fichero .fcs

Los datos generados por la mayoría de los citómetros de flujo comercial se almacenan en el formato .fcs (Flow Cytometry Standard). Dentro de este formato, el estándar de almacenamiento más común es el modo lista. Los ficheros de datos en modo lista tienen una cabecera de texto seguida de los valores experimentales y que finaliza con un bloque de texto en donde se incluyen los protocolos de análisis de datos [6, 5]. En la figura 3.2 podemos apreciar un fichero de este tipo, abierto en el bloc de notas.

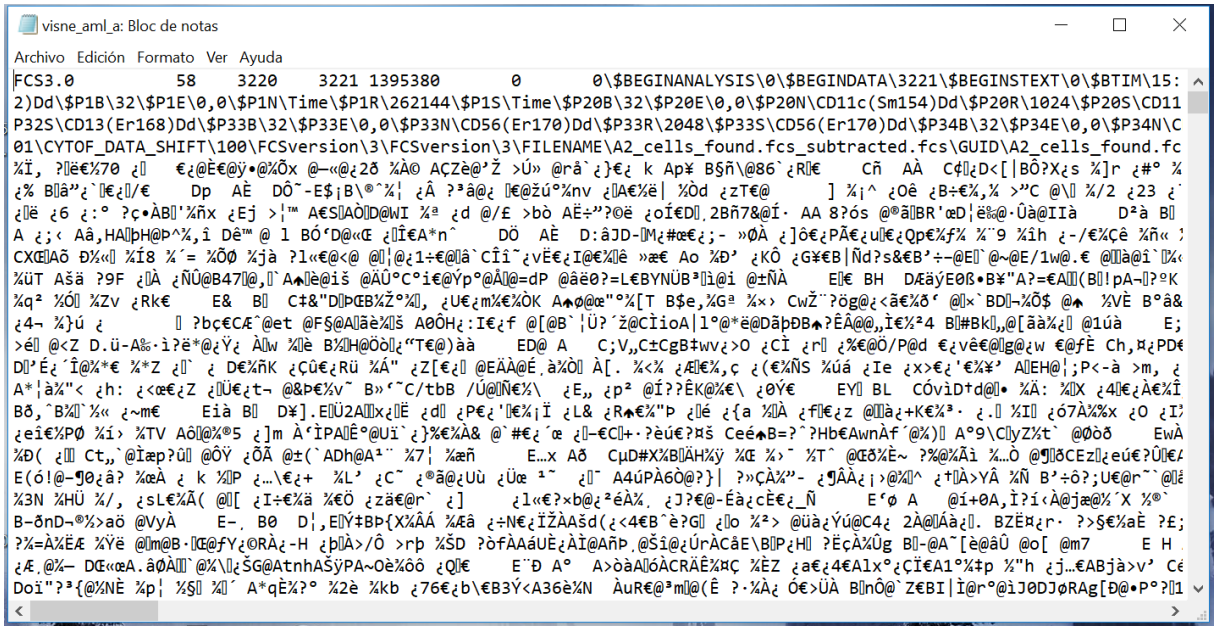

Figura 3.2: Captura de un fichero .fcs abierto en un bloc de notas. Podemos observar como es el fichero al abrirlo en este programa.

3.2 HTML5

HTML5 es un concepto para la construcción de diseño web y aplicaciones que supone una auténtica revolución en el ámbito de los programadores web. Ciertamente no se trata de una nueva versión, ni de una mejora del conocido lenguaje de etiquetas HTML (del inglés HyperText Markup Language), sino que estamos ante una combinación de tres lenguajes: HTML, CSS (del inglés Cascading Style Sheets) y JavaScript, eliminando las barreras entre sitios web y aplicaciones.

El origen de este lenguaje viene constituido por la falta integración entre aplicaciones y documentos que permitían los lenguajes predecesores [15].

Así mientras que HTML surgía con la finalidad de comunicar información por medio de texto, creando la estructura básica de páginas web, no interactuaba con los lenguajes de programación web coetáneos Java y Flash.

En un principio Java y Flash tuvieron una gran acogida, pero a medida que internet fue avanzando como un medio empresarial y social, las limitaciones y, sobretodo, la falta de interacción entre ambas tecnologías se hicieron mas evidentes.

Ante esta situación surgió JavaScript como un lenguaje que permite interactuar en el documento web con HTML, facilitando la programación y la utilización por parte de los usuarios. Ahora bien, no ha sido un camino fácil, ya que no tuvo la acogida esperada por parte de la comunidad. Es más, en un primer momento, no fue capaz de reemplazar muchas de las funciones propias de Flash o Java como, por ejemplo, la mera reproducción de un vídeo.

No obstante, conforme los usuarios iban requiriendo aplicaciones más complejas, se hizo patente que ni Java, ni Flash podían facilitar las herramientas necesarias para ello, siendo la mejor solución en ese contexto JavaScript.

JavaScript se convirtió, por tanto, en un lenguaje vanguardista que revolucionó la concepción que de él tenían los desarrolladores, permitiéndoles innovar hasta cotas impensables [2].

La combinación entre HTML, CSS y JavaScript da como resultado HTML5, permitiendo, de este modo, la evolución de la web. Es más, HTML5 supone una mejora para cada una de las citadas tecnologías:

- HTML facilita los elementos estructurales de la web,
- CSS se encarga del diseño gráfico y visual de la web realizada en HTML, y
- JavaScript dota de toda funcionalidad necesaria a las páginas web.

Como hemos visto, HTML5 engloba HTML, CSS y JavaScript y, por tanto, suministra estructura (HTML), estilo (CSS) y funcionalidad (JavaScript) del diseño web, como si de una sola tecnología se tratase, es decir, como un todo, debido a la perfecta simbiosis existente entre estas tres tecnologías.

En cuanto a la estructura, cabe señalar que se configura como una parte importante del documento, dotando de contenido estático y dinámico, y siendo una plataforma esencial para aplicaciones, que permite la interacción en la web [10].

La estructura sirve, pues, para dotar de forma, organización y flexibilidad a la web y aplicaciones.

HTML5 nace con la intención de simplificar, concretar y organizar el código, para lo que incluye nuevas etiquetas y atributos.

3.2.1 HTML

Las siglas HTML provienen del inglés y se corresponde a (HyperText Markup Language), es decir, el equivalente en castellano a Lenguaje de Marcas de Hipertexto [2].

Este lenguaje fue creado por CERN (Organización Europea de Investigación Nuclear) en 1945, con la intención de desarrollar un sistema de almacenamiento que conectan mediante hipervínculos. Término que fue definido en 1965 creando una estructura que estaba conectada electrónicamente, que, posteriormente, facilitaría la WWW (World Wide Web) un sistema de hipertexto que permitía compartir distinta información a través de internet.

El primero en realizar un código HTML fue el estadounidense Tim Berners Lee en 1991.

HTML es un lenguaje que permite ordenar diferentes documentos, utilizándose para determinar los nombres de las etiquetas que se emplearán al ordenar. No existen reglas para esta organización, siendo un formato abierto y simple [15].

Para la programación en este lenguaje, se utilizan etiquetas(tags), que permiten enlazar distintos conceptos y formatos.

Asimismo, este lenguaje permite códigos denominados scripts, que conforman instrucciones a los navegadores que procesan este lenguaje. Entre todos los scripts el más usado para el diseño web es JavaScript y PHP.

Si deseamos ver el código HTML usado en cualquier página web, tan solo es necesario pulsar "ver código fuente" en nuestro navegador web y nos proporcionará el código HTML en nuestro editor de texto[10, 2].

3.2.2 CSS

Las siglas CSS provienen del inglés (Cascading Stylesheets), es decir, el equivalente en castellano a hojas de estilo en cascada.

El lenguaje CSS fue propuesto por primera vez en 1994 por Håkon Wium Lie a la vez que trabajaba con Tim Berners Lee (primera persona en realizar un código HTML). A finales de 1996 el organismo WWW publicó la primera recomendación oficial llamada "CSS nivel 1", publicándose en 1998 su segunda recomendación oficial "CSS nivel 2". Actualmente se encuentra vigente el "CSS nivel 3" [10].

El CSS es un lenguaje creado como complemento al lenguaje HTML con la finalidad de superar las limitaciones y la complejidad de éste, separando la estructura de la presentación. Esto es CSS se encarga del diseño visual, siendo de especial relevancia para el diseño web [2].

Por lo tanto, CSS es un lenguaje que, en paralelo a HTML, dota al documento de carácter visual mediante tamaño, color, fondo, etc [15].

3.2.3 JavaScript

JavaScript fue creado por Brendan Eich de Netscape Communications Corporation, una prestigiosa empresa de software, con el nombre de Mocha, que más tarde se denominó Livescript. Se cambió al nombre de JavaScript cuando el navegador de la empresa Netscape tuvo compatibilidad con la tecnología Java en 1995 [15].

Con el fin de evitar confrontaciones, Netscape propuso estandarizar el lenguaje JavaScript. Así pues, en 1997 se envió la especificación JavaScript 1.1 al organismo ECMA (European Computer Manufacturers Association).

Bastantes programadores suelen usar la denominación ECMAScript para hacer referencia al lenguaje JavaScript. No obstante, JavaScript es la implementación que realizó la compañía Netscape del estándar ECMAScript.

JavaScript es un lenguaje de programación interpretado por el navegador línea a línea, al cargarse la página, y que únicamente puede realizar las instrucciones programadas en el entorno de esa página HTML en la cual reside. Solamente se puede utilizar junto a otro programa capaz de interpretarlo, siendo éstos los navegadores web.

Este lenguaje es orientado a objetos y basado en instancias, lo cual significa, que la mayor parte de las instrucciones que se emplean en los programas, suelen ser métodos y llamadas a objetos y propiedades del navegador, y en otros casos del propio lenguaje [10].

La sintaxis de este lenguaje es similar a la utilizada en lenguaje C, pero también usa nombres y convenciones de Java. Pese a esto, tanto JavaScript como Java tienen propósitos totalmente diferentes.

Este lenguaje suele ser usado del lado del cliente (client-side), implementado en un navegador web, consiguiendo mejorar la interfaz del usuario y haciendo las páginas web más dinámicas.

También podemos usar `JavaScript` para el lado del servidor(Server-side `JavaScript` o `SSJS`), dónde nos centraríamos en aplicaciones externas a la web, o en aplicaciones de escritorio (comúnmente widgets) [2].

En 2012, la mayor parte de los navegadores soportan completamente ECMAScript 5.1, una versión de `JavaScript`.

Primeramente se utilizaba `JavaScript` en el desarrollo web para la realización de operaciones del lado del cliente, y no del servidor. A día, de hoy se hace de forma indistinta.

3.2.4 Node.js

`Node.js` es un entorno de ejecución de `JavaScript` basado en eventos.

Esta librería actúa como un intérprete de `JavaScript` del lado del servidor, cambiando los roles con los que trabajan los programadores.

Normalmente, `JavaScript` se desarrolla del lado del cliente, obligando a éstos a trabajar con un lenguaje diferente para el lado del servidor.

Sin embargo, `Node.js` posibilita la utilización del mismo lenguaje de programación para ambos lados, al actuar como interprete de `JavaScript`.

La citada librería está basada en eventos utilizando el motor V8 desarrollado por `Google` para uso de su navegador `Chrome`, logrando ejecutar `JavaScript` a increíbles velocidades.

`Node.js` es un entorno de código abierto y multiplataforma, pudiendo ejecutarlo sin ningún tipo de problema en Windows, Mac OS X y Linux.

Otra de las principales ventajas de Node.js es la gestión de todas las operaciones mediante una programación asíncrona denominada “Bucle de Eventos o Event Loop".

El diseño de `Node.js` tiene la capacidad de soportar gran cantidad de conexiones a un servidor simultáneamente, para lo que emplea un único hilo y un bucle de eventos asíncronos, siendo capaz de gestionar eficazmente múltiples peticiones.

Otro de sus grandes atractivos consiste en su gestor de paquetes `NPM` (Node Package Manager), que nos permite acceder a muchas librerías desarrolladas por programadores [1].

Todo ello hace que se haya convertido en un componente indispensable en el desarrollo de aplicaciones Web en la actualidad.

3.2.5 Gestor de paquetes NPM

NPM (Node Package Manager) es un gestor de paquetes, que nos ayuda a trabajar con Node.js. Gracias a él podemos tener cualquier librería disponible ejecutando una sola línea de código, también nos ayudará a administrar nuestros módulos, distribuir paquetes y agregar dependencias muy fácilmente. A la hora de instalar nuevos paquetes, lo que hace NPM es instalarlo de manera local en nuestro proyecto dentro de la carpeta node-modules, aunque también es posible decirle que lo instale de una manera global. A la hora de instalar Node.js por defecto vienen instalados muchos módulos que no habrá que instalar usando NPM, se les conoce como "módulos nativos". Para poder instalar NPM, es necesario tener Node.js en nuestro ordenador [13].

3.2.6 APIs

Las siglas APIs provienen del inglés (Application Programming Interface), es decir, el equivalente en castellano a Interfaces de Programación de Aplicaciones [15, 10].

Las APIs son un conjunto de comandos, funciones y protocolos informáticos que ayudan a los programadores a crear programas específicos. Facilitan el trabajo de los desarrolladores ya que no parten de un código en blanco, sino que hacen uso de funciones predefinidas, lo cual aligera mucho el trabajo [2].

3.2.7 Librerías externas

En gran medida la aceptación de HTML5 es debida a que se creó con el fin de proporcionar al desarrollador web todas las herramientas que necesita, he ahí la ayuda extra que ofrecen las librerías externas [2].

Anteriormente a la aparición de HTML5, se habían creado algunas librerías JavaScript con el fin de competir contra las otras tecnologías. Éstas tenían diferentes objetivos, desde manipulación de gráficas hasta procesamiento de datos, entre otros [15].

No obstante, aun cuando haya funciones programadas en las librerías, la constante evolución de los programadores desarrollan mejoras constantemente para las mismas, evolucionando y otorgando soluciones frente a los futuros retos [10].

3.2.8 Librería Socket.io

Socket.io es una librería de JavaScript que tiene su nacimiento en la necesidad del intercambio de información a tiempo real en la comunicación web. Anteriormente a esta librería, existía el protocolo HTTP, que estaba pensado para realizar comunicaciones en un único sentido, desde el servidor hacia el cliente. Pero, actualmente, las necesidades web que han ido surgiendo han hecho que este protocolo no sea el más idóneo, ya que para que podamos sacarle el máximo potencial a éstas, es necesario que el flujo de información ocurra en ambos sentidos, en el mismo instante que ocurren los eventos.

Debido a esto surgieron diferentes estrategias, pero sólo dos fueron las más importantes: "Long Polling y Websocket". En la primera, el cliente se mantiene haciendo preguntas al servidor sobre un evento concreto. Sin embargo, websocket es un protocolo que permite la interacción entre el

servidor y el cliente, consiguiendo trasmitir datos en ambas direcciones a tiempo real. De este ultimo surge `Socket.io`.

`Socket.io` es una librería en `JavaScript` para `Node.js`, la cual permite una comunicación bidireccional a tiempo real entre servidor y cliente. Se basa principalmente en websocket, aunque también usa otras como `JSONP`, polling o sockets de Adobe Flash, usando siempre la alternativa que mejor le vaya al cliente justo en tiempo de ejecución.

Para la instalación de esta librería sólo es necesario ejecutar un comando de `npm`: "npm install socket.io".

No obstante, tenemos que dejar claro que las aplicaciones hechas en `Socket.io` tienen la desventaja de no soportar interacciones con otros clientes que usen websocket estándar, debido a que `Socket.io` es una librería de comunicación web a tiempo real que utiliza varios protocolos, no siendo implementación del protocolo websocket. Pese a esto, es una herramienta muy útil y de fácil manejo para proyectos web en los que tanto servidor, como cliente puedan usar la misma librería [12].

3.2.9 FCS para JavaScript

La librería FCS[2] es una librería implementada utilizando JavaScript/`Node.js` para leer ficheros de citometría de flujo `.fcs`. Lee todos los segmentos HEADER, TEXT y ANALYSIS en pares clave/valor. También lee datos sin procesar (probablemente no compensados), ya sea en matrices numéricas para un análisis posterior, o como cadenas para escanear rápidamente los datos.

3.2.10 Chart.js

`Chart.js`[3] es una de las librerías más populares para la creación de gráficos sobre Canvas `HTML5`.

Esta librería permite usar todo tipo de gráficos: histogramas, nube de puntos 2D, gráficos 3D, secciones de círculos etc. en diferentes escalas (lineal, logarítmica etc.). También aguanta animaciones para el cambio de datos o actualización de colores.

En definitiva, permite representar los datos mediante gráficas y todo lo que conllevan éstos, tales como: tipos, colores, escalas, títulos, leyendas etc.

Chartjs-plugin-zoom

Para implementar la opción de acercar, alejar y mover el gráfico, se ha utilizado un plugin para la librería `Chart.js`. Este plug-in permite manejar el gráfico a través del ratón haciendo click o moviendo la rueda para hacer zoom.

[2] https://github.com/MorganConrad/fcs
[3] https://www.chartjs.org/

Chartjs-plugin-annotation

Hemos utilizado este plugin para la librería `Chart.js` con el fin de poder dibujar un área de un rectángulo arrastrando el ratón pulsado.

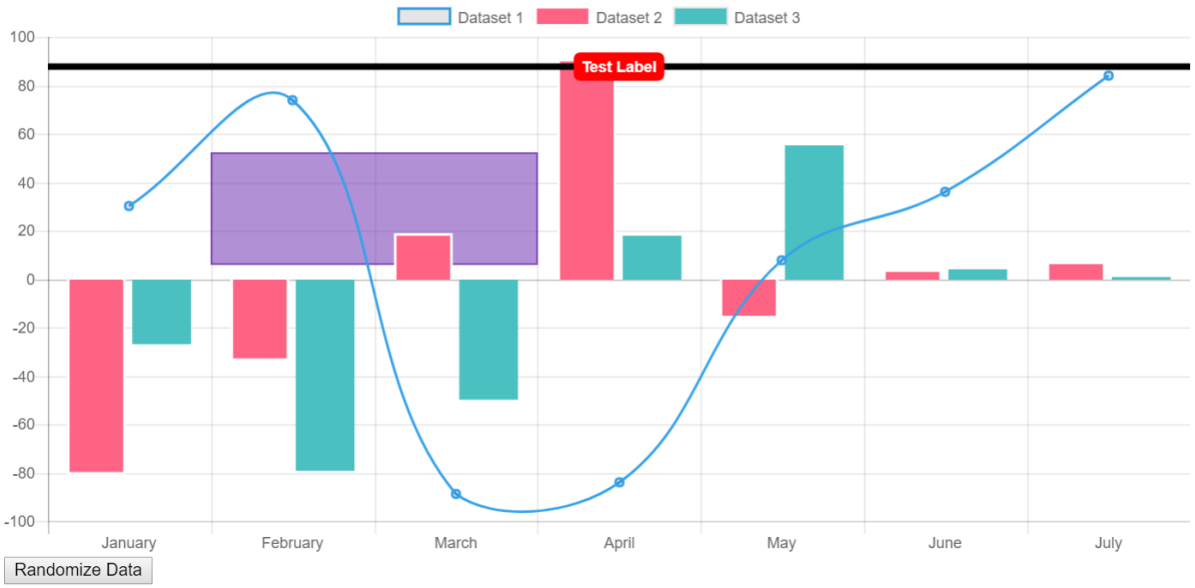

Figura 3.3: Captura del plugin annotation. En la figura 3.2.10, observamos que el área representada por el plugin es la del rectángulo en morado.

3.2.11 HTTP

Protocolo de transferencia de hipertexto (`HTTP`) nos permite enviar documentos de un lado a otro en la web. Un protocolo es un conjunto de reglas que determina qué mensajes se pueden intercambiar y qué mensajes son respuestas apropiadas a otros.

En `HTTP` se distinguen dos partes: servidor y cliente. A priori, el cliente siempre inicia la conversación, a la que el servidor da respuesta. `HTTP` se basa en texto, los mensajes son casi en su totalidad bits de texto, aunque el cuerpo del mensaje puede contener otros.

Los mensajes `HTTP` constan de un encabezado y un cuerpo. El cuerpo puede estar vacío, éste contiene los datos que desea transmitir a través de la red, y utilizarlos según se nos especifica en el encabezado. El encabezado contiene metadatos, como la información de codificación; pero, en el caso de una solicitud, también contiene los métodos `HTTP` importantes. En el estilo REST, encontrará que los datos de cabecera suelen ser más significativos que el cuerpo.

`HTTP` define un conjunto de métodos de petición para indicar la acción que se desea realizar para un recurso determinado. A estos métodos de solicitud se les suele llamar `HTTP` verbs, realizando cada uno de ellos una función diferente [11].

- GET: El método GET indica al servidor que transmita los datos identificados por la URL al cliente. Los datos nunca deben ser modificados en el lado del servidor como resultado de una solicitud GET. En este sentido, una petición GET es de sólo lectura, pero, una vez que el cliente recibe los datos, es libre de hacer cualquier operación con ella.

- PUT: El método PUT reemplaza todas las representaciones actuales del recurso de destino con la carga útil de la petición. Una petición PUT se utiliza cuando se desea crear o actualizar el recurso identificado por la URL.
- DELETE: El método DELETE borra un recurso en específico.
- CONNECT: El método CONNECT establece un túnel hacia el servidor identificado por el recurso.
- POST: El método POST solicita que un servidor web acepte los datos incluidos en el cuerpo del mensaje de solicitud para almacenarlos. A menudo se usa al cargar un archivo o al enviar un formulario web completo.
- OPTIONS: El método OPTIONS es utilizado para describir las opciones de comunicación para el recurso de destino.

3.2.12 API Express

`Express.js`, según sus creadores, es un framework (entorno de trabajo) de desarrollo de aplicaciones web minimalista y flexible para `Node.js`. Está inspirado en `Sinatra` [14], además es robusto, rápido, flexible y muy simple. Entre otras características, ofrece Router de `URL` (Get, Post, Put ...), que son las que nosotros vamos a necesitar en nuestro proyecto [8].

3.2.13 jQuery

La librería más utilizada y conocida actualmente es `jQuery`.

El éxito de ésta estriba tanto en su acceso libre y gratuito como en su simplicidad. De hecho, `jQuery` se desarrolló para simplificar la creación de páginas web.

Esta librería nos ayuda en la selección de elementos `HTML`, en la creación de efectos y animaciones, así como en el control de eventos, e incluso implementa `Ajax` en las aplicaciones.

`jQuery` presenta un fácil manejo para el programador, de modo que una vez incluimos esta librería en nuestro documento resulta sencillo utilizar los métodos que incorpora la misma.

Esta librería puede ser utilizada en cualquier navegador web, bien sea mediante `HTML5` o, si éste no está preparado, puede reemplazar las funciones de dicha tecnología en tales navegadores.

Capítulo 4

Diseño y desarrollo de la aplicación

En esta sección se va a describir el desarrollo del proyecto, el diseño realizado contenido en los ficheros `HTML` y `CSS`, la implementación del servidor, así como las funciones del lado del cliente utilizando `JavaScript`.

La idea principal de este proyecto es utilizar la librería `FCS` en el servidor de `JavaScript`, para leer ficheros `.fcs`. Para ello he creado un servidor responsable de procesar el fichero. El cliente envía el fichero que contiene los datos y el servidor los procesa y representa en una gráfica utilizando la librería `chart.js`. Para solicitar los datos del servidor se utiliza `socket.io`.

4.1 Estructura de la aplicación

Este proyecto contiene varios plugins, ficheros con extensiones `.html`, `.js`, `.css`, módulos `NPM` y todos los elementos necesarios para desarrollar el mismo. Para visualizar mejor la estructura que se ha creado, la presentamos en la Figura 4.1.

4.1.1 Diseño Web

En esta sección se va a describir el diseño de nuestra página web. Vamos a comentar el fichero `.html` que contiene todas las etiquetas utilizadas y posteriormente vamos a proceder a describir nuestra hoja de estilos `.css` que hemos utilizado en el proyecto.

HTML

El fichero `.html` corresponde al esqueleto de nuestra aplicación, que contiene todos los elementos del diseño web. En la Figura 4.2 puede verse el diseño de nuestra página web, donde se aprecia directamente un botón para seleccionar archivos, con el cual al pulsarlo, podemos seleccionar un archivo `.fcs` desde un disco local. Una vez seleccionado el archivo aparecen dos desplegables con los cuales podemos seleccionar las columnas que queremos visualizar y que son las cabeceras de los datos cargados del fichero `.fcs`. Al aparecer los desplegables, aparecerá la gráfica en el centro de nuestra web dibujando dos variables por defecto. Adicionalmente se han implementado botones que permiten interactuar con la gráfica. Uno de ellos permite hacer el zoom y movernos

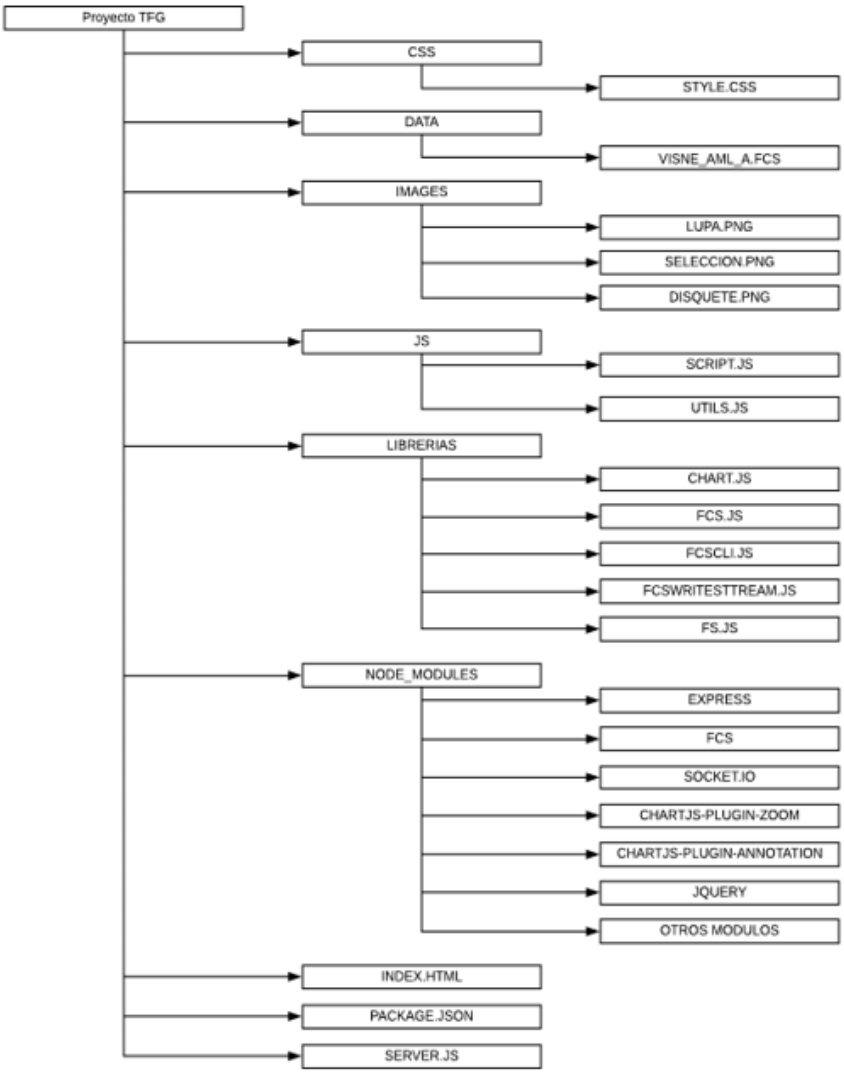

Figura 4.1: Distribución de carpetas y ficheros del visor de archivos `.fcs`.

a una parte concreta de la gráfica y el otro sirve para marcar un subconjunto de los datos y posteriormente guardarlos en un fichero `.csv`. Una vez hemos explicado la parte visual de nuestra página web damos paso a explicar los elementos `HTML` utilizados.

Los elementos principales son los siguientes:

- Botón seleccionar archivo: Con este botón seleccionamos un archivo `.fcs` que se encuentra en nuestro disco local. Podemos observar este botón en la esquina superior izquierda de la Figura 4.2 con el nombre de "Seleccionar archivo". Listing 4.1 contiene el código `HTML` del botón.

Listing 4.1: Botón seleccionar archivo.

```
<input id="seleccionarArchivo" type="file"/><br/>
```

- Una vez pulsado el botón seleccionar archivo, aparecen dos desplegables con todos los nombres de las columnas de los datos del fichero `.fcs`. Estos desplegables se encuentran

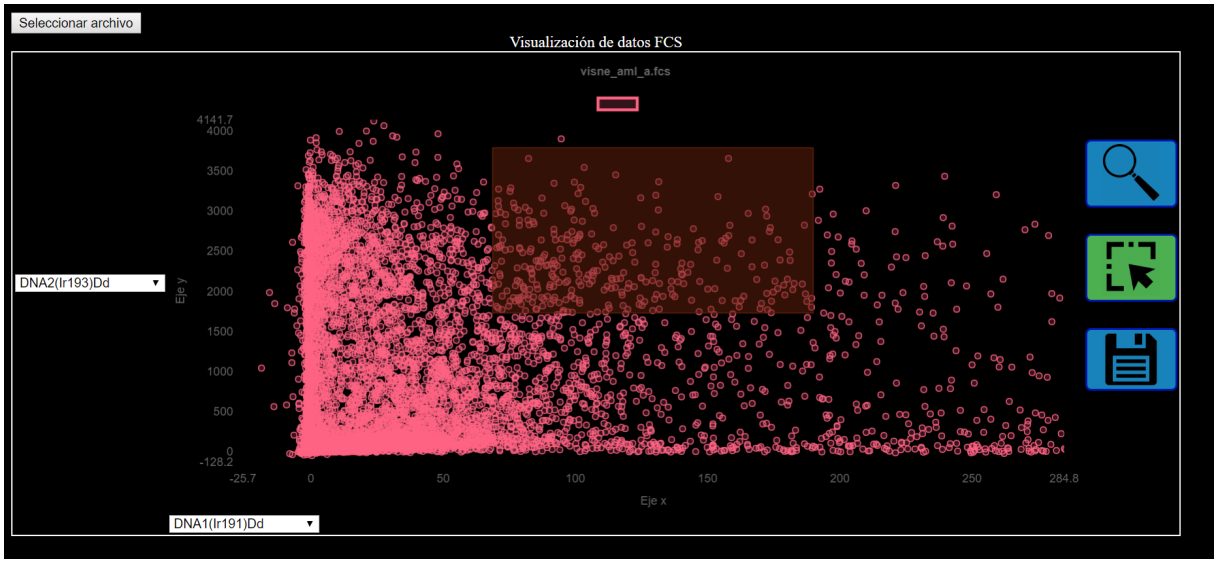

Figura 4.2: Captura del diseño web.

dentro del elemento tabla que describimos más adelante. Podemos observar desplegables en el lado izquierdo e inferior de la gráfica en la Figura 4.2, pueden adoptar diferentes nombres dependiendo de los parámetros que elijamos. Listing 4.2 contiene el código `HTML` del botón.

Listing 4.2: Desplegables con los nombres de columnas.

```
<input id="desplegables" type="file"/><br/>
```

- Elemento tipo `<table>`: Hemos creado una tabla de dos filas y tres columnas. Dentro de la tabla se encuentran todas las opciones para visualizar los datos. Hemos escogido el elemento tabla con el fin de organizar nuestro entorno web. Ahora vamos a describir el contenido de cada celda de nuestra tabla. Vamos a referirnos a las filas con la variable "x", las columnas con la variable "y", siendo la notación "(x,y)". En la celda (0,0) hemos situado el desplegable descrito anteriormente que se corresponde con el parámetro del eje de ordenadas de nuestra gráfica. La gráfica como tal se encuentra en la celda (0,1), situada dentro de un elemento `<canvas>`. En la celda (0,2) se encuentran nuestros botones para trabajar con nuestra gráfica. Las celdas (1,0) y (1,2) se encuentran vacías. La celda (1,1) contiene a nuestro segundo desplegable, que se corresponde con el parámetro del eje de abscisas de nuestra gráfica. Además, la tabla contiene un título encerrado dentro de las etiquetas `<caption>`. Listing 4.3 contiene el código `HTML` del elemento `<table>`.

Listing 4.3: Etiqueta `<table>`

```
<table id= "tabla" style="width:100%">
    <caption>FCS data visualization</caption>
    <tr>
      <td id='botones_izq' style='width:80px'> </td>
      <td><canvas id="canvas"></canvas></td>
      <td><button class="boton_lupa"><img style="margin-left:20px" src="
          images/loupe.png"></button>
      <button class="boton_figura_geometrica"><img style="margin-left:25
          px" class="imagen" src="images/hexagon.png"></button>
```

```
        <button class="boton_seleccion_puntos"><img style="margin-left:26
            px" class="imagen" src="images/pentagon.png"></button></td>
      </tr>
      <tr>
        <td></td>
        <td colspan="2" id='botones_abajo'></td>
      </tr>
    </table>
```

- Grupo de botones para el manejo de la gráfica: hemos dotado a nuestra aplicación de un grupo de herramientas para el manejo de nuestra gráfica, como podemos observar en la Figura 4.3. Hay tres botones: lupa, seleccionar datos y guardar datos.

Figura 4.3: Captura de las herramientas de nuestra aplicación.

  - Botón lupa: permite aumentar, disminuir y desplazar en la gráfica para visualizar mejor los datos mediante nuestro ratón o el touchpad. Una vez hemos seleccionado este botón se desactiva el botón de seleccionar datos, debido al conflicto a la hora de usarlos, ya que ambos se manejan con el ratón/touchpad y de no ser así tiene una mala manejabilidad. Este lo podemos ver en la Figura 4.3 con el icono de una lupa.
  - Botón seleccionar archivos: Este botón al igual que el anterior se deshabilita al pulsar botón lupa. Con este botón podemos seleccionar un subconjunto de elementos de la gráfica y guardarlos mediante el botón "guardar" representado en nuestra plataforma con la figura de una disquetera, una vez hemos seleccionado los datos.

CSS

La hoja de estilo `CSS` tiene un manejo muy fácil y es bastante intuitiva. Lo que se consigue aquí es darle el estilo que queremos a todos los elementos de la web.

Para ello utilizamos tres tipos de etiquetas:

- Etiquetas: Afectan a todos los elementos que estén en este grupo. En nuestro código hay dos elementos en los que lo utilizamos(`body` y `canvas`). En el primero sólo ponemos el color del fondo de nuestra web. En el `canvas` la propiedad `user-select` controla si el usuario puede seleccionar texto. En nuestro caso lo hemos desactivado para los navegadores *Mozilla*, *Chrome*, *Safari* y *Microsoft*.

- Clase: Hace referencia a un grupo al que queremos aplicarle los mismos estilos. Se diferencian porque tienen un punto antes del nombre. En nuestro caso en particular, hace referencia a los `.botones` de nuestra web a los que queremos dotarlos de las mismas propiedades(color del relleno, fuente, tamaño...).
- Id: Se distinguen en la web, ya que llevan una almohadilla delante del nombre. Los estilos solo afectan a las ID definidas en nuestro `.html`. En nuestro caso lo hemos aplicado a #tabla, #botones_variables entre otras. Listing 4.4 contiene el código `CSS` de la hoja de estilos utilizada en nuestro proyecto.

Listing 4.4: Hoja de estilos utilizada en el proyecto

```
body{
    background: #000;
}
canvas {
    -moz-user-select: none;
    -webkit-user-select: none;
    -ms-user-select: none;
}
.botones{
    margin-left: auto;
    margin-right: auto;
    padding-right: 15px;
    padding-left: 15px;
    /*font-family: arial;
    background-color: #1883ba;
    border-radius: 6px;
    border: 2px solid #0016b0;
    margin-bottom: 2em;
}
#div_global{
    border: 1px solid #000;
}
#div_canvas{
    border: 1px solid #000;

}
#tabla{
    border: 1px solid #ffffff;
    color: white;
    align-content: center;
}
#losBotones{
    visibility: hidden;
}
#Botones_variables{
    border: 1px solid #FFF;
    padding-right: 70px;
    padding-bottom: 500px;
    margin-top: 23px;
    margin-left: 70px;
}
```

```
#results{
    visibility: hidden;
}
```

4.1.2 El Servidor

A la hora de empezar a desarrollar nuestra página web, ha sido necesario investigar qué tipo de herramientas hay para procesar los ficheros `.fcs`. Desgraciadamente, no hemos podido encontrar ninguna librería implementada para el lado del cliente en `JavaScript`. Únicamente hemos encontrado la librería ya descrita anteriormente en la Sección 3.2.9. Para poder usar esta librería hacia falta implementar un servidor.

A continuación, vamos a describir las diferentes partes del servidor:

- Librerías utilizadas: Para crear un servidor `http`, es necesario el uso de las librerías `express` y `http` descritas en las secciones 3.2.12 y 3.2.11, respectivamente. También hemos utilizado una librería `socket.io` para poder establecer una comunicación entre nuestro servidor y la parte del cliente. También hemos importado la librería `fcs` para procesar este tipo de ficheros de datos. Listing 4.5 contiene el código `JavaScript` de las librerías utilizadas por parte del servidor.

Listing 4.5: Librerías utilizadas en el lado de servidor.

```
var express = require('express')
var http = require('http')
var FCS = require('fcs')
var io = require('socket.io')
```

- Creación del servidor: Con las librerías `express` y `http` hemos creado el servidor y además, hemos creado un `socket.io` cuya función es poner al servidor en alerta, por si llega algún mensaje por parte del cliente. Asignamos el puerto `8080` a nuestro servidor. Listing 4.6 contiene el código `JavaScript` para crear el servidor.

Listing 4.6: Creación del servidor.

```
var app = express()
var server = http.createServer(app)
var io = require('socket.io').listen(server)
server.listen(8080)
```

- Recursos utilizados: En esta parte del código le indicamos al servidor todas las rutas a los recursos que va a utilizar nuestro servidor, como, hojas de estilo, imágenes etc. Listing 4.1 contiene el código `JavaScript` donde se encuentran las carpetas con los recursos utilizados en nuestra plataforma.

Listing 4.7: Carpetas con los recursos utilizados.

```
app.use("/js",  express.static(__dirname + '/js'));
app.use("/lib",  express.static(__dirname + '/lib'));
app.use("/css", express.static(__dirname + '/css'));
app.use("/images",  express.static(__dirname + '/images'));
```

- Página por defecto: Cuando el cliente, introduce la dirección de nuestro servidor, configuramos nuestro servidor con el fin de que proporcione dicha página en el navegador de nuestro cliente. En este caso le enviamos una página llamada `index.html` que es la página que hemos descrito en la Sección 4.1.1. Para ello, hacemos uso del método `GET` de la librería `http`. Listing 4.8 contiene el código `JavaScript` de la página que se nos muestra por defecto.

Listing 4.8: Página por defecto.

```
app.get('/', function(req, res) {
    res.sendFile(path.join(__dirname + '/index.html'));
});
```

- Petición de procesamiento de datos: cuando el cliente nos envíe una solicitud, con un fichero `.fcs`, nuestro servidor lo procesará y devolverá al cliente la matriz con los datos para visualizar. Para ello hacemos uso del método `POST` de la librería `http`. Listing 4.9 contiene el código `JavaScript` para la solicitud del procesamiento de datos.

Listing 4.9: Solicitud de procesamiento de datos.

```
app.post('/', function(req, res) {
     req.on('data', function(chunk) {
      var result = procesar(res);
     });
});
```

- Configuración del `socket`: Necesitamos `socket.io` para la comunicación entre el servidor y el cliente. En concreto, configuramos nuestro `socket` para que el servidor espere un mensaje "file", cuyo contenido es el fichero a procesar. Listing 4.10 contiene el código `JavaScript` para configurar el socket en el lado del servidor.

Listing 4.10: Configuración del socket en el lado del servidor.

```
io.on('connection', function(client){
     console.log('new client connected ' + client);
     client.on('file', function(f){
          client.emit('data', procesar(f));
     });
});
```

- Función para procesar los datos: Se implementa una función cuyo propósito es procesar el fichero enviado por el cliente. En esta función hacemos uso de la librería `fcs`. En el Listing 4.11 se encuentra el código que implementa la función. La variable "option" define todas las opciones con las que se va a cargar el fichero. En nuestro caso solo especificamos que se va a cargar todas las filas del fichero ('eventsToRead': -1). El único argumento que recibe la función es el fichero "f" recibido del cliente. La función devuelve un vector de dos casillas, donde en la primera de ellas se encuentran los nombres de las columnas (`fcs.get$PnX('N')`) y en la segunda la matriz de datos extraídos del fichero (dataMatrix).

Listing 4.11: Función que procesa el fichero.

```
function procesar(f){
     var options = {'eventsToRead': -1}
```

```
    var dataMatrix=[];
    var fcs =new FCS(options, f);
    for(i of fcs.dataAsStrings){
        fila=i.slice(1,i.length-1)
        filaNumeros=fila.split(',').map(Number);
        dataMatrix.push(filaNumeros);
    }
    return [fcs.get$PnX('N'), dataMatrix]
}
```

4.2 Funciones del cliente

En esta sección vamos a describir todas las funciones implementadas para el lado del cliente. Además, aparte de las funciones hemos implementado varios escuchadores atentos al lanzamiento de algún evento.

- Configuración del `socket.io`. Antes de nada configuramos el `socket.io` para el cliente. Se requiere indicar la dirección *IP* del servidor y el puerto por el que se van a comunicar. Listing 4.12 contiene el código `JavaScript` para configurar el `socket.io` del lado del cliente.

  Listing 4.12: Configuración del `socket.io` en el lado del cliente.

  ```
  var socket = io.connect('http://localhost:8080');
  ```

- Variables globales: En `JavaScript` existen dos tipos de variables: globales y locales. Las variables globales se declaran fuera de cualquier función y se pueden usar en cualquier función del programa. Las variables locales se declaran dentro de una función y se pueden usar solo dentro de esa función.

  Las variables globales que hemos utilizado en este programa son las que representa Listing 4.13. Las variables "var1" y "var2" representarán las columnas de los datos que queramos representar. La variable "showButtons" sirve para detectar que los desplegables descritos en la sección 4.1.1 y que una vez se han creado no vuelvan a crearse. Las variables "selectIzq y selectBajo" son los desplegables los cuales hemos creado dinámicamente a través de `JavaScript` asignándoles un "id" que más tarde lo añadiremos al `DOM` (Document Object Model) al pulsar el botón "Dibujar gráfica". A continuación creamos un color de la librería `Chart.js` que utilizaremos en el objeto `scatterChartData`. Este color que hemos definido como rojo lo utilizaremos para representar el borde de los datos y aplicándole un factor de transparencia "alpha(0.2)" definimos el color del relleno de los mismos.

  Se denomina `scatterChartData` al dataset con sus correspondientes opciones que se representará dentro de la gráfica. Inicialmente definimos los datos como un conjunto vacío (data: []), ya que se añadirán posteriormente tras cargar y procesar el fichero. La variable "datos" se corresponde a los datos procesados por el servidor. Además, hemos añadido las variables "selección", "lupa" , "newx" y "newy", que nos servirán de ayuda para implementar las funciones para los botones lupa y selección.

  Listing 4.13: Variables globales.

  ```
  var var1 = 0
  ```

```
var var2 = 0
var datos
var showButtons = true
var selectIzq = document.createElement('select')
var selectBajo = document.createElement('select')
selectIzq.id = 'selectIzq'
selectBajo.id = 'selectBajo'
var color = Chart.helpers.color
var lupa = false
var seleccion = false
var newx =''
var newy = ''
var scatterChartData = {
    datasets: [{
        label: '',
        borderColor: window.chartColors.red,
        backgroundColor: color(window.chartColors.red).alpha
            (0.2).rgbString(),
        data: []
    }]
};
```

- Creación de la gráfica: Como el conjunto de herramientas de `HTML5` no son síncronas, antes de acceder a cualquier elemento de la página web, hay que asegurarse que los elementos de la página han sido cargados. Esto lo conseguimos con un escuchador del evento ‘‘`window.onload`’’. Al cargarse la página, se ejecutará la función para acceder a nuestra etiqueta `<canvas>` y crear un objeto `Chart.Scatter` con un dataset previamente definido que aún se encuentra sin datos para representar. Listing 4.14 contiene el código `JavaScript` para la creación del elemento `Chart.scatter`.

Listing 4.14: Creación del elemento `Chart.Scatter`.

```
window.onload = function() {
    var ctx = document.getElementById('canvas').getContext('2d');
    window.myScatter = Chart.Scatter(ctx, {
        data: scatterChartData,
        options: {
            title: {
                display: false,
                text: 'Dataset'
            }
        }
    });
};
```

- Carga del fichero y envío al servidor: Se ha creado un escuchador atento al evento ‘‘`onchange`’’ del botón ‘‘`myFile`’’. Al saltar el evento accedemos al fichero seleccionado (`this.files.item(0)`) y procedemos a cargarlo y enviarlo al servidor. Para ello se utiliza la librería `FileReader` que permite cargar el fichero en el lado del cliente. Posteriormente, se define el tamaño de los segmentos a enviar. En nuestro caso lo hemos definido a 100.000 Bytes. En cuanto se cargue el fichero procedemos a enviarlo al servidor haciendo uso del `socket.emit`. También, actualizamos el nombre del dataset en la gráfica y lo hacemos

visible. Cuando el servidor reciba el fichero va a procesarlo con la función anteriormente descrita 4.1.2. Listing 4.15 contiene el código `JavaScript` necesario para la carga del fichero y envío al servidor.

Listing 4.15: Carga del fichero y envío al servidor.

```
document.getElementById('myFile').onchange = function () {
    f=this.files.item(0)
    window.myScatter.options.title.display=true
    window.myScatter.options.title.text=f.name
    if (f) {
      var r = new FileReader();
      var slice = f.slice(0, 100000)
      r.readAsArrayBuffer(slice)
      r.onload = (evt) => {
          var arrayBuffer = r.result
          socket.emit('file', f);
      }
    }
};
```

- Recibir el fichero procesado del servidor: Con el `socket.io` el cliente se espera a que el servidor le envíe el fichero procesado. En el Listing 4.16 vemos el código que corresponde a esta parte. El servidor envía los datos a través de `socket.io` con un mensaje "datos". El primer elemento del array enviado por el servidor se corresponde con los nombres de las columnas de datos, y el segundo con los datos en sí. A continuación, se invoca la función `addButtons` que describiremos a continuación. Listing 4.1 contiene el código `HTML` del botón. Listing 4.1 contiene el código `HTML` del botón. Listing 4.1 contiene el código `JavaScript` del botón.

Listing 4.16: El cliente recibe el fichero procesado por el servidor.

```
socket.on('data', function(data) {
    headers=data[0]
    datos=data[1];
    addButtons(headers);
});
```

- Añadir desplegables al `DOM`: En cuanto el cliente reciba los datos del servidor, se crearán los desplegables par poder seleccionar las variables para representar en la gráfica. El código correspondiente a esta parte se encuentra en el Listing 4.17. En esta parte se utiliza el flag `showButtons` ya descrito anteriormente que nos indica que los desplegables ya han sido añadidos a la página. Una vez añadidos los desplegables, actualizamos los datos del dataset llamando la función `buildGraphicData()`, que vamos a describir a continuación.

Listing 4.17: Añadir desplegables.

```
function addButtons(variables){
    if (showButtons == true){
        for (var i of variables){
            selectIzq.options[selectIzq.options.length] = new Option
                (i, i)
            selectBajo.options[selectBajo.options.length] = new
                Option(i, i)
```

```
            }
            document.getElementById('botones_abajo').appendChild(
                selectIzq);
            document.getElementById('botones_izq').appendChild(
                selectBajo);
            scatterChartData.datasets.forEach(function(dataset) {
                dataset.data = buildGraphicData()
            });
            window.myScatter.update();
            showButtons=false;
        }
    }
```

- Actualizar los datos de dataset. Cada vez que se elija alguna de las variables a dibujar que contienen los desplegables, la gráfica se va a actualizar. Para ello se requiere actualizar los datos del dataset según las variables seleccionadas. La función responsable por ello es la del Listing 4.18.

Listing 4.18: Actualizar los datos del dataset.

```
function buildGraphicData(){
    col1= getColumn(var1)
    col2= getColumn(var2)
    var vars =[]
    for (var i=0; i<col1.length; i++){
        vars.push({
            x: col1[i],
            y: col2[i],
        })
    }
    return vars;
}
```

Para seleccionar los datos según la columna seleccionada, se utiliza la función `getColumn()` que vamos a describir a continuación.

- Selección de los datos según los nombre de las columnas. Para seleccionar los datos según los nombres de columnas se utiliza la función del Listing 4.19.

Listing 4.19: Selección de datos según los nombres de columnas.

```
function getColumn(index){
    var column=[]
    for (var i of datos){
        column.push(i[index])
    }
    return column
}
```

- Opción lupa: Para que se pueda hacer zoom en la gráfica y mover los ejes para ver una parte concreta de los datos representados, se han añadido las opciones "pan" y "zoom" activándolas con `''enabled: true''`. Estas opciones vienen del plugin descrito en la Subsección 3.2.10. En el Listing 4.20 se encuentra el código con el cuál añadimos las opciones anteriormente mencionadas.

Listing 4.20: Las opciones añadidas en el Chart para habilitar el zoom y desplazamiento de la gráfica.

```
pan: {
    enabled: true,
    mode: 'xy'
},
zoom: {
    enabled: true,
    mode: 'xy'
}
```

Para implementar el botón lupa hemos de tener en cuenta que, al cargar los datos y representarlos, esta opción esté deshabilitada, es decir, se encuentre a "false". Ahora bien, necesitamos de un escuchador para que en el caso de que hagamos click en el botón lupa ocurran dos sucesos:

Primer caso: Si nos encontramos en el supuesto en el que la lupa esté deshabilitada antes de pulsar con el ratón en su botón, hay que cambiar unas cosas, primeramente el botón lupa pasa de estar deshabilitado a habilitado pasando de "false" a "true", y las opciones que ella tiene como son "pan" y "zoom" también se habilitan. Por último, para saber que tenemos seleccionado el botón lupa cambiamos el color del fondo.

Segundo caso: Si nos encontramos en el supuesto en el que la lupa esté habilitada antes de pulsar con el ratón en su botón, hay que cambiar unas cosas, primeramente el botón lupa pasa de estar habilitado a deshabilitado pasando de "true" a "false", y las opciones que ella tiene como son "pan" y "zoom" también se deshabilitan. Por último, para saber que ya no tenemos seleccionado el botón lupa cambiamos el color del fondo al original.

En el Listing 4.21 se representan las funciones responsables de habilitar y deshabilitar el botón de lupa. Activación y desactivación de un botón implica que se cambia el color del mismo y cambia los valores de las opciones "pan" y "zoom". Además, se implementa un escuchador pendiente del evento "click" el cual se lanza cuando se produce un click en el botón de lupa.

Listing 4.21: Las funciones implementadas para activar y desactivar el botón de lupa.

```
function activarLupa(){
        lupa = true
        window.myScatter.options.pan.enabled= true
        window.myScatter.options.zoom.enabled= true
        document.getElementById('lupa').style.background = '#4CAF50'
}

function desactivarLupa(){
    console.log('dasctivando lupa')
        lupa= false
        window.myScatter.options.pan.enabled= false
        window.myScatter.options.zoom.enabled= false
        document.getElementById('lupa').style.background = '#1883ba'
}

document.getElementById('lupa').addEventListener('click', function(e
    ) {
```

```
        if (lupa == false){
            activarLupa()
            desactivarSeleccion()
        }else{
            desactivarLupa()
        }
    }, false);
```

- Opción Selección de datos: Para desarrollar esta opción hemos utilizado `chart.js-plugin-annotation`. Este plugin permite dibujar áreas dentro de un gráfico. En nuestro caso, seleccionamos el área que deseamos a través de un rectángulo. En el Listing 4.22 vemos la parte del código que nos proporciona la opción `annotation`. Para añadir esta opción se definen los vértices, el color de los bordes del rectángulo y el relleno, entre otros parámetros. Para implementar esta opción, nos hemos basado en un ejemplo encontrado[1].

Listing 4.22: Opción annotation para poder seleccionar datos.

```
annotation: {
    drawTime: "afterDraw",
    events: ['dblclick'],
    annotations: [{
        type: 'box',
        xScaleID: 'x-axis-1',
        yScaleID: 'y-axis-1',
        xMin: 0,
        xMax: 0,
        yMin: 0,
        yMax: 0,
        backgroundColor: 'rgba(101, 33, 11, 0.2)',
        borderColor: 'rgb(101, 33, 11)',
        borderWidth: 1,
        onDblclick: function(e) {
            console.log('Box', e.type, this);
        }
    }]
}
```

Al igual que en el botón lupa, se han implementado dos opciones para activar y desactivar el botón. También se ha implementado el escuchador atento al evento "click" del botón que se representa en el Listing 4.23.

Listing 4.23: Funciones implementadas para la activación y desactivación del botón selección de datos.

```
function activarSeleccion(){
    seleccion = true
    document.getElementById('results').style.visibility='visible'
    document.getElementById('seleccion').style.background = '#4CAF50
        '
    document.getElementById('boton_seleccion').style.background =
        '#4CAF50'
```

[1] https://github.com/chartjs/chartjs-plugin-annotation/blob/master/samples/box.html

```
}
function desactivarSeleccion(){
    seleccion = false
    document.getElementById('results').style.visibility='hidden'
    document.getElementById('seleccion').style.background = '#1883ba
        '
    document.getElementById('boton_seleccion').style.background =
        '#1883ba'
}

document.getElementById('seleccion').addEventListener('click',
    function(e) {
        if (seleccion == false){
            activarSeleccion()
            desactivarLupa()
        }else{
            desactivarSeleccion()
        }
        window.myScatter.update();
    }, false);
```

El código que identifica las coordenadas del ratón dentro del `canvas` respecto los valores de las variables elegidas está basado en un ejemplo encontrado[2]. Se utiliza la librería `jQuery` para implementar el escuchador del evento `mousemove`. El código adaptado para nuestro caso se encuentra en el Listing 4.24

Listing 4.24: Código para obtener las coordenadas del ratón dentro de la gráfica.

```
$('#canvas').mousemove(function(evt) {
  //console.log(evt.offsetX + "," + evt.offsetY);
  var ytop = window.myScatter.chartArea.top;
  var ybottom = window.myScatter.chartArea.bottom;
  var ymin = window.myScatter.scales['y-axis-1'].min;
  var ymax = window.myScatter.scales['y-axis-1'].max;
  //var newy = '';
  var showstuff = 0;
  if (evt.offsetY <= ybottom && evt.offsetY >= ytop) {
    newy = Math.abs((evt.offsetY - ytop) / (ybottom - ytop));
    newy = (newy - 1) * -1;
    newy = newy * (Math.abs(ymax - ymin)) + ymin;
    showstuff = 1;
  }
  var xtop = window.myScatter.chartArea.left;
  var xbottom = window.myScatter.chartArea.right;
  var xmin = window.myScatter.scales['x-axis-1'].min;
  var xmax = window.myScatter.scales['x-axis-1'].max;
  //var newx = '';
  if (evt.offsetX <= xbottom && evt.offsetX >= xtop && showstuff == 1)
      {
    newx = Math.abs((evt.offsetX - xtop) / (xbottom - xtop));
    newx = newx * (Math.abs(xmax - xmin)) + xmin;
```

[2]https://stackoverflow.com/questions/44959490/chart-js-2-0-current-mouse-coordinates-tooltip

```
    }
    if (newy != '' && newx != '') {
      document.getElementById("values").innerHTML ='Mouse Coordinates: '
          + newx.toFixed(2) + ',' + newy.toFixed(2);
    }
  });
```

Para marcar un área dentro de la gráfica utilizando el ratón, es necesario que ocurran dos sucesos, el primero tener el ratón pulsado y mantenido (lo llamaremos `mouseDown`) durante el desplazamiento cuyo resultado será el área que encierre a los datos que nosotros queramos estudiar o guardar y el segundo cuando soltemos el ratón (lo llamaremos `mouseUp`) que es el hecho que marca la finalización de nuestra selección de datos.

Al realizar `mouseDown` obtenemos las primeras coordenadas de nuestra área. Al realizar el `mouseUp` obtenemos la esquina opuesta de nuestro rectángulo, con estás dos coordenadas podemos dibujar el área y determinar qué muestras se encuentran dentro.

En el Listing 4.25 se muestra el código de las funciones `mouseUp y mouseDown` junto con el escuchador del evento `mousemove`.

Listing 4.25: Código para determinar las coordenadas del rectángulo.

```
xstart = undefined
ystart = undefined
var pressed = false

function mouseDown() {
    if (seleccion){
        pressed=true
        xstart=newx
        ystart=newy
        window.myScatter.options.annotation.annotations[0].xMin=xstart
        window.myScatter.options.annotation.annotations[0].yMin=ystart
        window.myScatter.options.annotation.annotations[0].xMax=xstart
        window.myScatter.options.annotation.annotations[0].yMax=ystart
    }
}

window.addEventListener('mousemove', function() {
    if (pressed){
        window.myScatter.options.annotation.annotations[0].xMax=newx
        window.myScatter.options.annotation.annotations[0].yMax=newy
        window.myScatter.update();
    }
})
var xMin, xMax, yMin, yMax = 0
function mouseUp() {
    if (seleccion){
        pressed = false
        window.myScatter.options.annotation.annotations[0].xMax=newx
        window.myScatter.options.annotation.annotations[0].yMax=newy
        window.myScatter.update();
```

```
        xMin= Math.min(window.myScatter.options.annotation.annotations[0].
            xMin, window.myScatter.options.annotation.annotations[0].xMax)
        xMax= Math.max(window.myScatter.options.annotation.annotations[0].
            xMin, window.myScatter.options.annotation.annotations[0].xMax)
        yMin= Math.min(window.myScatter.options.annotation.annotations[0].
            yMin, window.myScatter.options.annotation.annotations[0].yMax)
        yMax= Math.max(window.myScatter.options.annotation.annotations[0].
            yMin, window.myScatter.options.annotation.annotations[0].yMax)


        if (xMin != xMax && yMin != yMax) {
            document.getElementById('results').style.visibility='visible'
        }else{
            document.getElementById('results').style.visibility='hidden'
        }
        }
    }
```

En la Figura 4.4 se puede apreciar el proceso de selección de datos, podemos observar dónde dan lugar las funciones `mouseDown()` y `mouseUp()` que hemos descrito anteriormente.

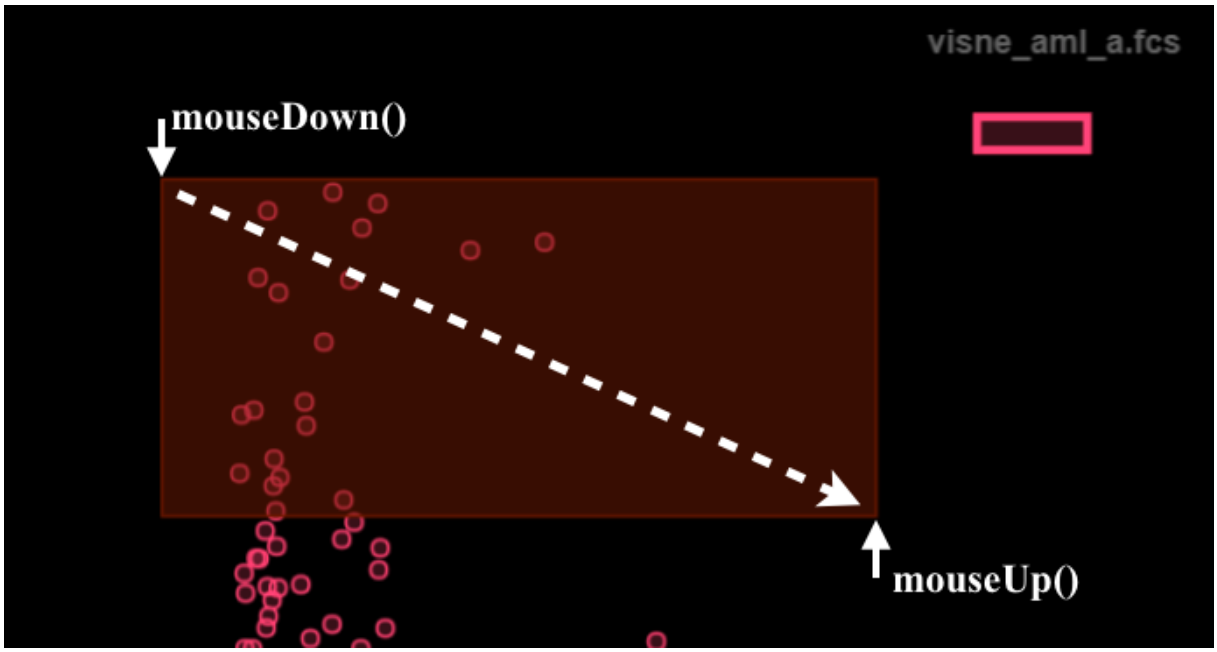

Figura 4.4: Captura de selección de datos.

- Exportación de datos a un fichero `.csv`: Para guardar los datos, hemos de tener claro los vértices que encierran a nuestro rectángulo. Para ello, determinamos el valor mínimo y máximo de las variables "x" e "y", ya que anteriormente en la selección de puntos nos daba igual cual era menor y cual era mayor, sin embargo aquí, es necesario tenerlo claro. También es necesario cerciorarse de que el área que hemos seleccionado no es un punto, ni una recta, para ello comprobamos que el valor de cada una de las variables es distinto, de no ser así, no nos aparecerá el icono de guardar. Esto lo comprobamos en la función `mouseUp`, si el área seleccionada es la correcta nos aparecerá el icono del botón guardar. Las muestras se

guardan en un fichero `.csv`, para ello hemos de ir comprobando cada una de las muestras que aparecen dibujadas en nuestra gráfica con los limites obtenidos de nuestro rectángulo.

Listing 4.26: Código para exportar datos a un csv.

```
function exportData() {
    var data='x\ty\n';
    if (xMin != xMax && yMin != yMax) {
        for (var i of scatterChartData.datasets[0].data){
            if ((i['x'] > xMin) & (i['x'] < xMax) && (i['y'] > yMin) & (
                i['y'] < yMax)){
            data +=i['x'] + '\t' + i['y'] + '\n'
            }
        }
        exportLink.setAttribute('href', 'data:text/csv;base64,' + window
            .btoa(data));
    }else{
        alert('No se ha seleccionado ningún grupo de muestras.')
    }
}
```

Con el código del Listing 4.26 solamente elegimos los puntos contenidos por el área. El siguiente paso es guardarlos en un `.csv`. Para ello, se ha realizado el código del Listing 4.27 donde se crea un hipervínculo asociado al elemento `div` que hace la función de un botón. Al hacer click, se abre una ventana para guardar el fichero en el disco.

Listing 4.27: Código para guardar datos en un `csv`

```
var exportLink = document.getElementById('a');
document.getElementById('results').appendChild(exportLink);
var img = document.createElement('img');
img.src = 'images/save.png'
img.style.width='60px'
img.classList.add("botones")
exportLink.appendChild(img);
```

A la hora de visualizar los datos guardados en el fichero `.csv`, podemos utilizar diferentes programas para abrirlos, nosotros en particular hemos usado el `Microsoft Excel`, como vemos en la Figura 4.5 los datos se agrupan en dos columnas las correspondientes con los ejes “x” e “y”.

	A	B	C	D	E
1	x	y			
2	-0.94	1555.08			
3	1.73	1272.72			
4	-0.86	2041.99			
5	3.67	1302.36			
6	-0.37	1382.58			
7	-0.64	1788.55			
8	-0.01	1666.99			
9	-0.2	1164.7			
10	-0.38	1423.95			

Figura 4.5: Captura de los datos seleccionados en el fichero .csv abiertos con `Microsoft Excel`.

Capítulo 5

Conclusiones

Los objetivos planteados en este trabajo se pueden considerar cumplidos. La idea inicial fue crear una herramienta libre y gratuita para visualización de archivos `.fcs`. La aplicación se ha desarrollado utilizando `HTML5` ya que es comúnmente utilizado para la implementación de sitios web. Por lo tanto, los usuarios no necesitan instalar ninguna herramienta adicional para poder disfrutar de las ventajas que ofrece nuestra plataforma.

Además, se han implementado varias opciones que ayudan a manejar los datos más fácilmente. Para ello, se ha habilitado la opción de lupa que permite hacer un enfoque a un grupo concreto de datos. También se ha implementado la posibilidad de seleccionar un subconjunto de datos y guardarlos en un fichero `.csv`. Todo ello ayuda al usuario a tener una experiencia de trabajo más satisfactoria.

A nivel personal, este trabajo me ha aportado el aprendizaje de nuevos lenguajes de programación, con los cuales no había trabajado en la carrera, como es todo lo relacionado con `HTML5`. Además, el poder realizar un trabajo, el cual puede ser utilizado en laboratorios, por biólogos, médicos u otros profesionales de este ámbito le aporta un sentido a todos mis estudios y también el conocimiento de todo lo relacionado con el estudio de la citometría de flujo, que por lo que he podido estudiar, es un área de la biomedicina en auge y donde personalmente creo que se pueden realizar grandes avances, sobretodo en enfermedades como el cáncer. Además ha sido mi primera experiencia en un proyecto de investigación.

Personalmente, creo que mi proyecto va a ser muy útil, ya que desarrollo un software muy completo de ámbito educativo y profesional, libre y gratuito, con el cual podrán trabajar e investigar en el área de la biomedicina.

Como trabajo futuro quedan los siguientes objetivos:

- Desarrollar la aplicación para diferentes plataformas, tales como móvil o tablet.
- Crear una nube en la cual los científicos puedan interactuar subiendo los archivos y así poder investigar en paralelo entre gran parte de la comunidad científica. Esto podría provocar un avance mucho más rápido en la investigación.

- Implementar más opciones que permitan al usuario interactuar con la gráficas de distintas formas, incluso incluyendo operaciones matemáticas para analizar los datos.

Bibliografía

[1] https://www.netconsulting.es/blog/nodejs/. Node js (vid. pág. 13).

[2] Ramón Tamarit Agusti. "Análisis de datos de citometría de flujo: aplicaciones de r-bioconductor". En: (2010) (vid. págs. 11-14).

[3] Lourdes Barrera Ramírez y col. "Citometría de flujo: vínculo entre la investigación básica y la aplicación clínica". En: () (vid. págs. 3, 7-9).

[4] Tiffany J Chen y Nikesh Kotecha. "Cytobank: providing an analytics platform for community cytometry data analysis and collaboration". En: () (vid. págs. 1, 3, 7, 8).

[5] *Citometría bioinformatica*. Brando, B.; Barnett, D.; Janossy, G.; Mandy, F.; Autran, B.; Rothe, G.; Scarpati, B.; d'Avanzo, G.; d'Hautcourt, J. L.; Lenkei, R.; Schmitz, G.; Kunkl, A.; Chianese, R.; Papa, S.; Gratama, J. W. (2000). Çytofluorometric methods for assessing absolute numbers of cell subsets in blood". (Vid. págs. 1, 3, 5, 8, 9).

[6] *Citometría de flujo*. Flow Cytometry in Clinical Diagnosis, v4, (Carey, McCoy, and Keren, eds), ASCP Press, 2007; Ormerod, M.G. (ed.) Flow Cytometry — A practical approach. 3rd edition. Oxford University Press, Oxford, UK. 2000. (Vid. págs. 7-9).

[7] *Cytobank*. https://https://www.cytobank.org/ (vid. págs. 1, 5, 6, 9).

[8] *Express*. https://www.solucionex.com/blog/expressjs-un-framework-para-nodejs (vid. pág. 17).

[9] $FCS_Express$. https://www.denovosoftware.com/ (vid. págs. 1, 4, 9).

[10] Juan Diego Gauchat. "El gran libro de HTML5, CSS3 y Javascript". En: (2012) (vid. págs. 11, 12, 14).

[11] *HTTP*. https://code.tutsplus.com/es/tutorials/a-beginners-guide-to-http-and-rest--net-16340 (vid. pág. 16).

[12] *Librería Socket.io.* https://hipertextual.com/archivo/2014/08/socketio-javascript/ (vid. pág. 15).

[13] *NPM.* https://devcode.la/blog/que-es-npm/ (vid. pág. 14).

[14] *Sinatra.* https://www.npmjs.com/package/sinatra (vid. pág. 17).

[15] Paul Stothard. "The sequence manipulation suite: JavaScript programs for analyzing and formatting protein and DNA sequences". En: (2000) (vid. págs. 10-12, 14).

[16] *WINMDI.* https://answers.microsoft.com/en-us/windows/forum/windows_8-winapps/winmdi/900bf129-cd65-4340-9314-ba68483b978d (vid. págs. 3, 4, 9).