Universidad Central De Venezuela

Facultad de Ciencias

Aplicación Web Para Gráficar Con GnuPlot

Trabajo Especial de Grado presentado por López Gonzalez Carlos Alfonso

2019

Centro de Computación Gráfica

Universidad Central de Venezuela

Facultad de Ciencias

Escuela de Computación

Centro de Computación Gráfica

ACTA DEL VEREDICTO

Quienes suscriben miembros del Jurado, designado por el Consejo de Escuela de Computación, para examinar el Trabajo Especial de Grado presentado por el **Br. Carlos Lopez, C.I. 21096536**, titulado: "**Aplicación web para graficar con GNUPLOT**" a los fines de optar al título de Licenciado en Computación, dejan constancia lo siguiente:

Leído como fue, dicho trabajo por cada uno de los miembros del Jurado, se fijó el día 4 de octubre del 2019 a las 2:00 p.m., para que su autor lo defendiera en forma pública, en el Centro de Computación Gráfica, Facultad de Ciencias, mediante una presentación oral de su contenido. Finalizada la defensa pública del Trabajo Especial de Grado, el Jurado decidió aprobarlo.

En fe de lo cual se levanta la presente Acta, en Caracas al cuarto día del mes de octubre del año dos mil diecinueve, dejándose también constancia de que actuó como Coordinador del Jurado el Prof. Héctor Navarro, que estuvo presente por videoconferencia. Firma en su lugar el Director de la Escuela de Computación Prof. Robinson Rivas.

REPUBLICA DE VENEZUELA
Facultad de Ciencias
Escuela de Computación
Universidad Central - Caracas

Prof. Héctor Navarro, Tutor

Prof. , Walter Hernandez

Prof. , Jaime Parada

Resumen

La tecnología y la computación, así como el estudio de las ciencias de la computación son áreas que han estado en constante crecimiento en los últimos años, y con ello, se han creado diversas herramientas relacionadas a estas, entre ellas, Gnuplot.

Gnuplot es una herramienta de utilidad gráfica de linea de comandos para la mayoría de los sistemas operativos que existen actualmente y que muchos programas están utilizando como motor base de graficación. Esta herramienta presenta algunas dificultades para un grupo elevado de usuarios con respecto al uso y a la instalación, por estas razones se implemento una aplicación web que facilite la utilización de dicha herramienta.

La solución fue sometida a una serie de pruebas cualitativas con el objetivo de determinar la experiencia de los usuarios en términos de utilidad, complejidad y cumplimiento de los objetivos de la investigación. Los resultados obtenidos indican que la aplicación cumple satisfactoriamente con lo esperado.

Palabras clave: Gnuplot, Aplicación Web, Graficación de funciones, Gráficos 2D, Gráficos 3D.

Índice general

Introducción

La tecnología y la computación, así como el estudio de las ciencias de la computación son áreas que han estado en constante crecimiento en los últimos años, y con ello, se han creado diversas herramientas relacionadas a estas, entre ellas, Gnuplot.

Gnuplot es una herramienta de utilidad gráfica de linea de comandos para la mayoría de los sistemas operativos que existen actualmente y que muchos programas están utilizando como motor base de graficación. Esta herramienta presenta algunas dificultades para un grupo elevado de usuarios con respecto al uso y a la instalación, por estas razones se implemento una aplicación web que facilite la utilización de dicha herramienta.

Capítulo 1. En este capítulo se plantea el contexto del trabajo de grado y la descripción del planteamiento del problema. También se describe la justificación de la investigación, objetivo general, objetivos específicos, principales requerimientos, arquitectura de la solución y propuesta de la solución.

Capítulo 3. En este capítulo se despliega la información que llevan de la mano las metodologías para el desarrollo. En el cual se definirá esta metodología como propia de esta investigación y su implementación en el proyecto final.

Capítulo 4. En este capítulo de despliega principalmente el marco aplicativo para el trabajo especial de grado, así como las herramientas utilizadas y procesos realizados para la culminación del mismo.

Finalmente se plantearan las conclusiones y recomendaciones, anexos y referencias bibliográficas.

Capítulo 1

Generalidades

Al momento de trabajar con Gnuplot es necesario el uso de una terminal y el conocimiento de una serie de comandos, lo cual actualmente dificulta el uso de dicha herramienta y hace que la curva de aprendizaje sea más larga para muchos usuarios que no tengan experiencia trabajando con una terminal, además de esto un usuario ingenuo podría tener problemas con la instalación de la herramienta, el manejo de las variables de entorno y el sistema de ventanas.

Es por esto que se propone el desarrollo de una aplicación web bajo la arquitectura cliente-servidor, la cual consistirá de dos módulos. El primero tendrá lugar en el lado del cliente donde se recibirán los datos de entrada de parte del usuario para generar el plot deseado. En consecuencia el segundo tendrá lugar en el lado del servidor, donde se procesará la entrada suministrada previamente, enviando de vuelta el plot generado para mostrarlo ante el usuario.

Para llevar a cabo el desarrollo de la aplicación antes mencionada se han seleccionado un conjunto de lenguajes, tecnologías y bases de datos, este conjunto está conformado por Python, HTML5, CSS3, PostgreSQL, jQuery, entre otros.

1.1. Objetivo General

Desarrollo de un sitio web que permita la creación de gráficos y el uso de todas las funcionalidades de Gnuplot.

1.2. Objetivos Específicos del TEG

- Permitir a los usuarios importar archivos de datos y mostrar la gráfica correspondiente que genera Gnuplot como salida.
- Proporcionarle al usuario un conjunto de diversas opciones para generar el gráfico deseado tales como definir la lista de etiquetas correspondientes, definir los rangos en los ejes, las dimensiones, una lista de funciones explícitas, la representación de los datos (vectores, líneas, puntos, cajas, etc).
- Permitir a los usuarios acceder a un historial de sus últimos trabajos.
- Realizar pruebas de usabilidad con diferentes usuarios.
- Analizar los datos que se manejarán en el sitio y establecer un modelo de datos.
- Permitir a los usuarios guardar los trabajos realizados con la aplicación.
- Permitir a los usuarios crear sus propios scripts desde la aplicación

1.3. Herramientas

Para lograr los objetivos descritos, es necesario hacer uso de las siguientes herramientas:

- Django: (y por extensión, Python) como framework web para el desarrollo del lado del servidor.
- HTML5: para la estructura del documento generado. Es la versión ms reciente y ampliamente soportada en los navegadores actuales.

- CSS3: para utilizar las últimas características de estilos que incluyen la utilización de gradientes, sombras y transiciones que permiten tener un diseo más atractivo.

- JavaScript: para modificar dinámicamente la aplicación de acuerdo a la interacción del usuario. Esto incluye la biblioteca jQuery por las facilidades que ofrece en la manipulación del documento y por su capacidad de envío y recepción de información con AJAX. Este, a su vez, permitirá el envío y recepción de información con el servidor sin recargar la página actual.

- MySQL: al tener el mejor soporte de base de datos por parte del framework Django.

1.4. Justificación

Como se vio en los capítulos anteriores, Gnuplot es una herramienta de gran utilidad ya que permite realizar graáficos en 2D y 3D facilitando así el trabajo para aquellas personas que posean esta necesidad.

Anteriormente, se plantearon las dificultades que conllevan a utilizar esta herramienta a pesar de su utilidad. Por lo que es necesario un sistema que facilite el proceso. Esto traería consecuencias favorables para los usuarios de Gnuplot, ya que podrían concentrarse más en lo que necesitan hacer que en como lo deben hacer.

1.5. Alcance y delimitación de la investigación

La investigación está dedicada al desarrollo de un sistema para la creación de gráficos utilizando la herramienta Gnuplot, ademas de la administración de estos gráficos bajo cuentas de usuarios.

El sistema dispondrá de una interfaz gráfica de usuario donde se crearán dichos gráficos. Éstos serán mostrados al usuario como salida con el formato que él mismo elija al momento de su creación.

El sistema deberá proveer al usuario la creación de una cuenta, en la que podrá almacenar y administrar (ver, editar y eliminar) los gráficos generados. La creación de cuenta será una herramienta opcional para el usuario.

Para acceder al sistema se deberá solicitar la petición al servidor correspondiente mediante un navegador web. Los navegadores que abarcarán las pruebas de desarrollo serán Firefox versión 36.0.1, Safari versión 7.1.5, Chrome versión 41.0 e Internet Explorer versión 11. Se recomienda utilizar los navegadores a partir de las versiones mencionadas o posteriores.

Capítulo 2

Marco Teórico

En este capítulo se expone las bases teóricas que permiten respaldar el desarrollo del trabajo. Además de exponer la teoría, se presenta una revisión de trabajos anteriores relacionados. La investigación teórica previa a la experimentación ubica al investigador en un correcto encuadre dentro del proceso, y le sugiere cuales son los problemas sin soluciones y que son objetos de estudio.

2.1. Gnuplot

Gnuplot es una utilidad gráfica de línea de comandos portátil para Linux, OS / 2, MS Windows, OSX, VMS y muchas otras plataformas. El código fuente está protegido por derechos de autor pero distribuido libremente (es decir, no tiene que pagar por ello). Originalmente fue creado para permitir a los científicos y estudiantes visualizar funciones matemáticas y datos de forma interactiva, pero ha crecido para soportar muchos usos no interactivos, como scripts web. También se utiliza como un motor de trazado de aplicaciones de terceros como Octave. Gnuplot ha sido apoyado y en desarrollo activo desde 1986 [1].

Gnuplot soporta muchos tipos de gráficos en 2D y 3D. Puede dibujar usando líneas, puntos, cajas, contornos, Campos de vectores, superficies y varios textos asociados. También soporta varios tipos de trama especializada.

Gnuplot soporta muchos tipos diferentes de salida: terminales de pantalla interactiva (con mouse y hotkey Entrada), salida directa a plotters de pluma o impresoras modernas, y salida a muchos formatos de archivo (eps, emf, fig, jpeg, LaTeX, pdf, png, postscript, ...). Gnuplot es fácilmente extensible para incluir nuevos modos de salida. Adiciones recientes Incluyen terminales interactivos basados en wxWidgets (utilizables en múltiples plataformas) y Qt. Gráficos con ratón Incrustados en páginas web pueden generarse utilizando los controladores de terminal cavas svg o HTML5.

2.1.1. Operación por lotes / interactiva

Gnuplot puede ser ejecutado en modo discontinuo o interactivo, y los dos pueden incluso ser mezclados juntos en muchos sistemas.

Se supone que cualquier argumento de línea de comandos es o bien opciones de programa (el primer carácter es -) o nombres de archivos que contienen comandos de gnuplot. La opción -e "comando"se puede utilizar para forzar la ejecución de un comando gnuplot. Cada archivo o cadena de comandos se ejecutará en el orden especificado. El nombre de archivo especial "-ïndica que los comandos deben leerse desde stdin. Gnuplot sale cuando se procesa el último archivo. Si no se especifican archivos de carga y no se especifican cadenas de comandos, gnuplot acepta la entrada interactiva de stdin.

Ejemplos:

- Para iniciar una sesión interactiva:

gnuplot

Para iniciar una sesión por lotes utilizando dos archivos de comandos entrada:

gnuplot entrada1 entrada2

Para iniciar una sesión interactiva después de un archivo de inicialización "hea-

der" y seguido por otro archivo de comandos "trailer":

gnuplot header - trailer

Para dar comandos de gnuplot directamente en la línea de comandos, usando la opción "-persist"para que la gráfica permanezca en la pantalla después:

gnuplot -persist -e "set title 'Sine curve'; plot sin(x)"

2.1.2. Comentarios

El carácter de comentario # puede aparecer casi en cualquier parte de una línea de comandos y gnuplot ignorará el resto de esa línea. Un # no tiene este efecto dentro de una cadena entre comillas. Tenga en cuenta que si una línea comentada termina en '/', la línea siguiente se trata como parte de ese comentario.

2.1.3. Coordenadas

Los comandos **set arrow**, **set label** y **set object** permiten dibujar algo en una posición arbitraria en el gráfico. Esta posición se especifica mediante la sintaxis:

{<system>} <x>, {<system>} <y>{,{<system>} <z>}

Cada <system>puede ser **first**, **second**, **graph**, **screen** o **character**.

First coloca la coordenada x, y o z en el sistema definido por los ejes izquierdo e inferior; **Second** lo coloca en el sistema definido por los ejes x2, y2 (arriba y derecha); **Graph** especifica el área dentro de los ejes - 0,0 abajo izquierda y 1,1 es arriba derecha; **Screen** especifica el área de la pantalla (el área entera - no sólo la porción seleccionada por el tamańo del conjunto), con 0,0 en la parte inferior izquierda y 1,1 en la parte superior derecha; y **Character** da la posición en anchos y alturas de caracteres desde la parte inferior izquierda del área de pantalla (pantalla 0,0), las coordenadas de caracteres depende del tamańo de fuente elegido.

Si no se especifica el sistema de coordenadas para x, se utiliza **first**. Si no se especifica el sistema para y, se adopta el utilizado para x.

2.1.4. Cadena de datos

Los archivos de datos pueden contener datos de cadena consistentes en una cadena arbitraria de caracteres imprimibles que no contengan espacios en blanco o una cadena arbitraria de caracteres, posiblemente incluyendo espacios en blanco, delimitados por comillas dobles. La siguiente línea de un archivo de datos se interpreta que contiene cuatro columnas, con un campo de texto en la columna 3:

1.000 2.0000 “Tercera columna es todo este texto”4.00

Los campos de texto pueden colocarse dentro de un gráfico 2D o 3D utilizando los comandos:

plot ’datafile’ using 1:2:4 with labels

splot ’datafile’ using 1:2:3:4 with labels

También se puede usar una columna de datos de texto para etiquetar las marcas tic (ticmarks) a lo largo de uno o más de los ejes de trazado. El ejemplo siguiente traza una línea a través de una serie de puntos con coordenadas (X,Y) tomadas de las columnas 3 y 4 del archivo de datos de entrada. Sin embargo, en lugar de generar tics regularmente espaciados a lo largo del eje x etiquetados numéricamente, gnuplot colocará una marca tic a lo largo del eje x en la coordenada X de cada punto y rotulará la marca tic con el texto tomado de la columna 1 del archivo de datos de entrada.

set xtics

plot ’datafile’ using 3:4:xticlabels(1) with linespoints

También hay una opción que interpretará la primera entrada en una columna de datos de entrada(es decir, el encabezado de columna) como un campo de texto y utilizará como título de la clave para los datos trazados desde esa columna. El ejemplo dado a continuación utilizará la primera entrada en la columna 2 para generar un título en el cuadro de claves mientras se procesa el resto de las columnas 2 y 4 para dibujar la línea requerida:

plot 'datafile' using 1:(f(2)/4) with lines title columnhead(2)

2.1.5. Operadores

Los operadores en gnuplot son los mismos que los operadores correspondientes en el lenguaje de programción C, excepto que todos los operadores aceptan argumentos enteros, reales y complejos, a menos que se indique lo contrario. El operador ** (exponenciación) es compatible, como en FORTRAN.

Los paréntesis se pueden utilizar para cambiar el orden de la evaluación.

2.1.6. Variables definidas por gnuplot

Gnuplot mantiene una serie de variables de sólo lectura que reflejan el estado interno actual del programa y el gráfico más reciente. Estas variables comienzan con el prefijo "GPVAL". Los ejemplos incluyen GPVAL TERM, GPVAL X MIN, GPVAL X MAX, GPVAL Y MIN. Para ver la lista completa de variables y sus valores actuales existe el comando show variables all.

2.1.7. Variables definidas por el usuario

Las nuevas variables y funciones definidas por el usuario de una a doce variables pueden ser declaradas y utilizadas en cualquier lugar, incluso en el propio comando para gráficar.

Sintaxis de la función definida por el usuario:

<func-name>(<dummy1>{,<dummy2>} ... {,<dummy12>}) = <expression>

Donde <expression>se define en términos de <dummy1>a traves de <dummy12>.

Sintaxis de variables definidas por el usuario:

<variable-name>= <constant-expression>

Ejemplos:

w=2

q = floor(tan(pi/2 - 0.1))

*f(x) = sin(w*x)*

*sinc(x) = sin(pi*x)/(pi*x)*

delta(t) = (t == 0)

ramp(t) = (t >0) ? t : 0

min(a,b) = (a <b) ? a : b

comb(n,k) = n!/(k!(n-k)!)*

*len3d(x,y,z) = sqrt(x*x+y*y+z*z)*

*plot f(x) = sin(x*a), a = 0.2, f(x), a = 0.4, f(x)*

file = "mydata.inp"

Los nombres válidos son los mismos que en la mayoría de los lenguajes de programación: deben comenzar con una letra, pero los caracteres subsiguientes pueden ser letras o dígitos.

Cada definición de función se pone a disposición como una variable especial con valor de cadena con el prefijo 'GPFUN'.

2.1.8. Fuentes

Gnuplot no proporciona ninguna fuente propia. Se basa en el manejo de fuentes externas, cuyos detalles desafortunadamente varían de un tipo de terminal a otro.

2.1.9. Capas

Un gráfico de gnuplot se construye dibujando sus varios componentes en un orden fijo. Este orden se puede modificar asignando algunos componentes a una capa específica utilizando las palabras clave **behind**, **back** o **front**. Por ejemplo, para reemplazar el color de fondo del área de trazado puede definir un rectángulo de color con el atributo **back**.

set object 1 rectangle from graph 0,0 to graph 1,1 fc rgb "gray"behind

El orden del dibujo es

behind

back

the plot itself

the plot legend ('key')

front

Dentro de cada capa se dibujan los elementos en el siguiente orden:

Objetos (rectángulos, círculos, elipses y polígonos) en orden númerico

Etiquetas en orden númerico

Flechas en orden númerico

En el caso de gráficos múltiples en una sola página (modo de multiplicacion), este pedido se aplica por separado a cada componente del gráfico.

2.1.10. Entrada del ratón

Muchos terminales permiten la interacción con el gráfico actual utilizando el ratón. Algunos también admiten la definición de teclas de acceso rápido para activar funciones predefinidas pulsando una sola tecla mientras el foco del ratón está en la ventana de trazado activo. Incluso es posible combinar la entrada del ratón con secuencias de comandos de comandos por lotes, invocando el comando pausar el ratón y luego utilizando las variables del mouse devueltas por el clic del ratón como parámetros para las siguientes acciones con guión.

2.1.11. Enlace (Bind)

El enlace permite definir o redefinir una tecla de acceso directo, es decir, una secuencia de comandos de gnuplot que se ejecutará cuando se presione una cierta clave o secuencia de teclas mientras la ventana tiene el foco de entrada. Tenga en cuenta que **Bind** sólo está disponible si gnuplot ha sido compilado con soporte de ratón y es utilizado por todos los terminales con capacidad de ratón.

2.1.12. Trazado (Plotting)

Hay cuatro comandos de gnuplot que realmente crean un gráfico: **plot**, **splot**, **replot** y **refresh**. Otros comandos controlan el diseño, el estilo y el contenido de la trama que eventualmente se creará. **Plot** genera gráficos 2D. **Splot** genera tramas 3D (en realidad proyecciones 2D, por supuesto). **Replot** vuelve a ejecutar el comando **plot** o **splot** anterior. **Refresh** es similar a **replot**, pero reutiliza los datos previamente almacenados en lugar de volver a leer los datos de un archivo o flujo de entrada.

Cada vez que emita uno de estos cuatro comandos, volverá a dibujar la pantalla o generará una nueva página de salida que contenga todos los ejes, etiquetas, títulos y todas las funciones o fuentes de datos definidos actualmente en el comando de trazado original. Si, en cambio, necesita colocar varios gráficos completos uno junto al otro en la misma página.

2.2. Contexto Tecnológico

2.2.1. Arquitectura Cliente/Servidor

2.2.1.1. Definición

Cliente/servidor es un arquitectura de red que sirve como modelo de aplicaciones distribuidas, en el que cada ordenador o proceso de la red es cliente o servidor. Los servidores están dedicados a proveer recursos o servicios, mientras que los clientes demandan los servicios que provee los servidores.

Esta arquitectura establece la relación entre procesos que solicitan servicios (clientes) y los que responden a estos (servidores). Un mismo ordenador puede cumplir las tareas del cliente y del servidor simultáneamente, desde luego, manteniendo la separación logica de sus funciones [2].

2.2.1.2. Ventajas

* La ventaja importante es que permite modular las funciones, cada función se situá en la plataforma más adecuada según su ejecución. El objetivo de separar

una gran pieza de software en módulos es facilitar su proceso de desarrollo, depuración y mantenimiento.

* Los múltiples procesadores en una red permite que se ejecute en partes distribuidas la misma aplicación, logrando la concurrencia de procesos.

* Existe la posibilidad de migrar aplicaciones de un procesador a otro con modificaciones mínimas en los programas.

* Las aplicaciones son escalables. Permite una escalabilidad horizontal o vertical. Escalabilidad horizontal al permitir la capacidad de añadir o suprimir estaciones de trabajo que hacen uso de la aplicación (clientes), sin afectar sustancialmente el rendimiento del sistema global. Escalabilidad vertical al permitir mejorar las capacidades del servidor, añadir múltiples o migrar a más potentes o de distinta arquitectura sin afectar a los clientes.

* Posibilidad de acceder a los datos independientemente de la ubicación del usuario.

* Ambiente heterogéneo. Las especificaciones de hardware y de sistema operativo entre cliente y servidor no transparentes, se conectan independientemente de sus plataformas.

2.2.1.3. Desventajas

* Cuando una gran cantidad de clientes realizan peticiones simultáneas al mismo servidor, puede congestionarse el tráfico de red.

* Si el servidor se cae, todas las peticiones clientes quedan desatendidas.

* El software y el hardware de un servidor son generalmente muy determinantes. Para atender cierta cantidad de clientes, sobre todo numerosa, un hardware regular de un ordenador personal puede no ser suficiente para suplir el trabajo, normalmente se necesita hardware de mayor potencia, lo que implica mayor coste.

2.2.1.4. Arquitectura multicapas y multiniveles

La arquitectura multicapa es una técnica o estilo de programación con el objetivo de separar las funciones lógicas del desarrollo. Cada capa se le confía una misión, que solamente interactuán con sus capas adyacentes lo que le permite abstraerse de las funcionalidades del resto de las capas. De este modo, en caso de que sobrevenga algún cambio, solo se revisa y modifica la capa afectada [3].

La lógica de presentación se encarga de la entrada y salida entre el usuario y la aplicación. Sus principales tareas son: obtener información del usuario, enviarla a la lógica de negocio, recibirla después de su procesamiento y prepararla para retornarla al usuario. Esta capa se comunica únicamente con la lógica de negocio.

La lógica de negocios se encarga del procesamiento de los datos. Sus principales tareas son: comunicarse con la lógica de presentación para recibir las solicitudes y enviar resultados, interactuar con la lógica de datos para ejecutar las reglas de negocio que tiene que cumplir la aplicación, y comunicarse con la lógica de datos para solicitar al gestor de base de datos almacenar o recuperar datos de él.

La lógica de datos se encarga de acceder y gestionar los datos. Sus principales tareas son: comunicarse con la lógica de negocios para recibir solicitudes de almacenamiento o recuperación de datos, también de mantener y asegurar la integridad de los datos.

La arquitectura por multinivel se refiere a la división de las capas de software en piezas físicas de diferentes hardware. Programas de múltiples capas se pueden construirse en un mismo nivel o distribuirse en varios.

Es bastante común confundir los términos de capas y niveles. Las capas se ocupan de la división lógica, por el contrario, los niveles se ocupan de la distribución de las capas lógicas de forma física.

La arquitectura cliente/servidor se refieren a veces como arquitecturas de dos capas y dos niveles. Dos capas primordialmente para separar la lógica de presentación de la lógica de negocios y dos niveles para cada lado, cliente y servidor. Pero pueden existir más capas y niveles dependiendo de los requerimientos del proyecto.

2.2.2. Aplicación Web

2.2.2.1. Definición

Se denomina aplicación web a aquellas herramientas que los usuarios pueden utilizar mediante un navegador para acceder a un servidor web vía Internet o de una intranet. Una aplicación web es un tipo especial de aplicación cliente/servidor, donde tanto el cliente (el navegador) como el servidor (el servidor web) y el protocolo mediante el que se comunican (por lo general http) están estandarizados [4].

2.2.2.2. El cliente web

El cliente web es un programa con el que interacciona el usuario para solicitar un servicio remoto a un servidor web.
Los lenguajes del lado del cliente son ejecutados por el navegador web con el fin de interactuar con el usuario. Entre los lenguajes se tiene HTML, como el formato en que se suele presentar la información al navegador (que conforma la página web), CSS y código script (por lo general Javascript). Además, puede contener pequeños programas o plug-ins que permite enriquecer la aplicación, por ejemplo, en contenidos multimedia (como Flash de Adobe o Silverlight de Microsoft).

2.2.2.3. El servidor web

El servidor web es un programa que realiza las conexiones bidireccionales y/o uni-direccionales y síncronas o asíncronas con el cliente. Generalmente se utiliza el protocolo HTTP para estas comunicaciones.
Los scripts o lenguajes de programación del lado del servidor son ejecutados por servidor. La respuesta del programa suele ser una página HTML estándar que se envía al navegador del cliente. Tradicionalmente este programa o script que es ejecutado por el servidor web se encarga de alguna tarea, como componer una página, acceder a la base de datos o conexiones en red. Los lenguajes de lado servidor más ampliamente utilizados para el desarrollo de páginas son PHP, ASP, Java, Ruby, PERL, Python o Javascript.

2.2.2.4. Arquitecturas de las aplicaciones web

Las aplicaciones web se basan en la arquitectura cliente/servidor: por un lado está él cliente (el agente de usuario) y por otro lado el servidor (el servidor web). Existen diversas variantes de la arquitectura baśica según cómo se implementen las diferentes funcionalidades de la parte servidor [5].

2.2.3. HTML5

2.2.3.1. Definición HTML

El lenguaje de marcas de hipertexto o HTML (de las siglas en inglés Hypertext Markup Language) es el lenguaje para describir la estructura de las páginas web. Es un estándar a cargo del World Wide Web Consortium (W3C) o Consorcio WWW.

2.2.3.2. Definición HTML5

HTML5 es la más reciente y quinta versión del estándar del lenguaje de marcado HTML de la World Wide Web.

Esta versión representa una nueva evolución del estándar que define HTML, con nuevos elementos, atributos y un extenso conjunto de tecnologías que permite a sitios y aplicaciones web soportar un contenido más potente y diverso sin la necesidad de instalar software adicionales como plugins de navegador. La quinta versión ofrece desde una mejora semántica hasta reproducción multimedia y animación de gráficos 2D y 3D [6].

2.2.3.3. Soporte de HTML5 en exploradores web

HTML5 se encuentra aún en proceso de desarrollo. El continuo trabajo de HTML5 suscita probables cambios necesarios en sus especificaciones. Debido a estos cambios, no todas las funcionalidades que provee HTML5 son soportadas por todos los navegadores.

Sin embargo, los navegadores más populares como Chrome, Firefox, Internet Explorer, Safari u Opera, en sus últimas versiones estables hasta la fecha de este

documento, tienen un buen soporte y continúan el apoyo hacia más de sus características.

2.2.4. Hojas de Estilo en Cascada

2.2.4.1. Definición

Hojas de Estilo en Cascada o CSS (por sus siglas en inglés Cascading Style Sheets), es un lenguaje utilizado para describir el aspecto de los documentos HTML y XML. Este mecanismo permite el control sobre estilo y formato de los documentos separandolo de la presentación [7].

CSS permite el control de estilo y formato de múltiples elementos y páginas web al mismo tiempo. Cualquier cambio en el estilo marcado para un elemento en la CSS afectará a todas las páginas vinculadas a esa CSS en las que aparezca ese elemento

Algunos de los aspectos que se puede controlar con CSS sobre los elementos del documento son: color, tamaño y tipo de letra del texto, separación horizontal/vertical entre elementos o posición dentro de la página.

2.2.5. Javascript

2.2.5.1. Definición

JavaScript (abreviado comúnmente JS) es un lenguaje ligero e interpretado, dialecto del estándar ECMAScript, orientado a objetos con funciones de primera clase, más conocido como el lenguaje de script para páginas web, pero también usado en muchos entornos sin navegador [8].

Surgió con el objetivo de programar ciertos comportamientos sobre las páginas web, respondiendo a las interacciones del usuario, realizando acciones automáticas sencillas o animaciones. Sin embargo, las necesidades de las aplicaciones web modernas y el HTML5 ha provocado que el uso de Javascript haya llegado al nivel de complejidad y prestaciones tan alto como otros lenguajes de primer nivel.
Javascript ya no solo es un lenguaje que se encuentra en Internet y la Web del lado

del cliente, también existe del lado del servidor, en programas de videojuegos y en base de datos. JavaScript ya no es uso exlusivo para el ámbito de páginas web.

2.2.5.2. Programación basada en prototipos

JavaScript dispone de fuertes capacidades de programación orientada a objetos, a pesar de que han tenido lugar algunos debates respecto a sus diferencias de su capacidades en comparación con otros lenguajes.

JavaScript es un lenguaje basado en prototipos, un estilo de programación orientado a objetos en la que las clases no están presentes3 (como en C++ o Java). Un prototipo es un objeto abstracto, capaz de contener otros objetos dentro, los cuales pueden ser distintos tipos de datos: variables, vectores, funciones e inclusive otros grupos de objetos.

Entonces, en lugar de declarar clases, JavaScript utiliza funciones como clases (define prototipos). Las variables dentro de este serán las propiedades, y las funciones serán los métodos.

Las clases de javascript esta introducidas en ECMAScript 6. La sintaxis de las clases no introduce un nuevo modelo de herencia orientada a objetos a JavaScript. Las clases de JavaScript proveen una sintaxis mucho más clara y simple para crear objetos y lidiar con la herencia. Sin embargo esta tecnología parte del estándar de ECMAScript 2015 (ES6) no es compatible para la fecha con todos los navegadores más usados.

2.2.6. MySql

2.2.6.1. Definición

MySql es un sistema de gestión de base de datos de código abierto es desarrollado, distribuido, y soportado por Oracle Corporation. Utiliza el lenguaje de consultas SQL.

Una base de datos es una colección estructurada de datos. Podría ser una colección de imágenes o una gran cantidad información en una red corporativa. Para acceder, agregar, eliminar o modificar los datos almacenados en una base de datos se necesita de un sistema de gestión de base de datos como MySQL Server [9].

2.2.6.2. Base de datos relacional

Las bases de datos de MySql son relacionales. Una base de datos relacional almacena datos en tablas separadas en lugar de un solo gran almacén. La base de datos organiza los archivos para optimizar la velocidad de acceso.

Ofrece un modelo logíco con objetos tales como bases de datos, tablas, filas y columnas para un entorno de programación flexible. Se configura reglas que rigen las relaciones entre los diferentes campos de datos, como por ejemplo uno-a-uno, uno-a-muchos, único, obligatorio u opcional, y punteros entre las diferentes tablas.

2.2.6.3. Ventajas

* Escalable, MySQL Server puede funcionar perfectamente en un escritorio o portátil, junto con sus otras aplicaciones, servidores web, y así sucesivamente. Si se dedica una máquina completa con MySQL, se puede ajustar la configuración para aprovechar toda la memoria, potencia de la CPU, y la capacidad de E/S disponibles. MySQL también puede escalar hasta grupos de máquinas, conectados en red.

* Muy rápida, MySQL Server se desarrolló originalmente para manejar grandes bases de datos mucho más rápido que las soluciones existentes. Su conectividad, velocidad y seguridad hacen de MySQL Server altamente apropiado para acceder bases de datos en Internet.

* Soporte multi-hilos, múltiples clientes tienen acceso concurrente.

2.2.7. Django

2.2.7.1. Definición

Django es un framework para desarrollo web basado en Python que fue desarrollado originalmente para gestionar varias páginas orientadas a noticias de la World Company de Lawrence, Kansas, y fue liberada al público bajo una licencia BSD en julio de 2005, desde entonces este framework ha estado en crecimiento. Actualmente, la Fundación de Software Django, lo define como:

Un framework para aplicaciones Web que estimula el desarrollo rápido y un diseno limpio y pragmático. Elaborado por desarrolladores experimentados, se encarga de gran parte de las dificultades del desarrollo Web, mientras el programador puede enfocarse en escribir su aplicación sin reinventar la rueda. Es gratuito y de código abierto [10].

Django se describe así mismo como: "El framework web para perfeccionistas con fechas topes", ya que provee una serie de funcionalidades robustas que acortan el tiempo de desarrollo, como: mapeo objeto relacional (ORM), vistas genéricas, sistema de plantillas, un despachador de URLs basado en expresiones regulares, cacheo, compresión de la salida, normalización de URLs, protección CSRF, soporte de sesiones, soporte de internacionalización, entre otras.

Uno de los principios que definen a Django, es el "No Te Repitas"(DRY, del inglés Don't Repeat Yourself), donde las funcionalidades o aplicaciones desarrolladas con Django son altamente modulares y fáciles de integrar en cualquier otro sistema.

2.2.7.2. Características

Los orígenes de Django en la administración de páginas de noticias son evidentes en su diseño, ya que proporciona una serie de características que facilitan el desarrolo rápido de páginas orientadas a contenidos. Por ejemplo, en lugar de requerir que

los desarrolladores escriban controladores y vistas para las áreas de administración de la página, Django proporciona una aplicación incorporada para administrar los contenidos, que puede incluirse como parte de cualquier página hecha con Django y que puede administrar varias páginas a partir de una misma instalación; la aplicación administrativa permite la creación, actualización y eliminación de objetos de contenido, llevando un registro de todas las acciones realizadas sobre cada uno, y proporciona una interfaz para administrar los usuarios y los grupos de usuarios (incluyendo una asignación detallada de permisos).

La distribución principal de Django también aglutina aplicaciones que proporcionan un sistema de comentarios, "páginas planas"que permiten gestionar páginas de contenido sin necesidad de escribir controladores o vistas para esas páginas, y un sistema de redirección de URLs.

Otras caractersticas de Django son:

* Un mapeador objeto-relacional.
* Una Api de base de datos robusta.
* Un sistema incorporado de "vistas genéricas"que ahorra tener que escribir la lógica de ciertas tareas comunes.
* Un sistema extensible de plantillas basado en etiquetas, con herencia de plantillas.
* Un despachador de URLs basado en expresiones regulares.
* Un sistema "middleware"para desarrollar características adicionales; por ejemplo, la distribución principal de Django incluye componentes middleware que proporcionan cacheo, compresión de la salida, normalización de URLs, protección CSRF y soporte de sesiones.
* Soporte de internacionalización, incluyendo traducciones incorporadas de la interfaz de administración.

* Documentación incorporada accesible a través de la aplicación administrativa (incluyendo documentación generada automáticamente de los modelos y las bibliotecas de plantillas añadidas por las aplicaciones).

2.2.7.3. Arquitectura

Aunque Django está fuertemente inspirado en la filosofía de desarrollo Modelo Vista Controlador, sus desarrolladores declaran públicamente que no se sienten especialmente atados a observar estrictamente ningún paradigma particular, y en cambio prefieren hacer "lo que les parece correcto". Como resultado, por ejemplo, lo que se llamaría "controlador.[en] un "verdadero"framework MVC se llama en Django "vista", y lo que se llamaría "vista"se llama "plantilla".

2.2.7.4. Presentación

Aquí se maneja la interacción entre el usuario y el computador. En Django, esta tarea la realizan el motor de plantillas y el cargador de plantillas que toman la información y la presentan al usuario (vía HTML, por ejemplo). El sistema de configuración de URLs es también parte de la capa de presentación.

2.2.7.5. Control

En esta capa reside el programa o la lógica de aplicación en sí. En Django son representados por las vistas y los manipuladores. La capa de presentación depende de ésta y a su vez ésta depende de la capa de dominio.

2.2.7.6. Mediator

Es el encargado de manejar la interacción entre el subsistema Entity y foundation. Aquí se realiza el mapeo objeto-relacional a cargo del motor de Django.

2.2.7.7. Entity

El subsistema entity maneja los objetos de negocio. El mapeo objeto-relacional de Django permite escribir objetos de tipo entity de una forma fácil y estándar.

2.2.7.8. Foundation

La principal tarea del subsistema foundation es la de manejar a bajo nivel el trabajo con la base de datos. Se provee soporte a nivel de foundation para varias bases de datos y otras están en etapa de prueba.

2.2.7.9. Soporte de Bases de datos

Respecto a la base de datos, son soportadas MySQL, PostgreSQL y SQLite 3. Se encuentra en desarrollo un adaptador para Microsoft SQL Server. Una vez creados los modelos de datos, Django proporciona una abstracción de la base de datos a través de su API que permite crear, recuperar, actualizar y borrar objetos. También es posible que el usuario ejecute sus propias consultas SQL directamente. En el modelo de datos de Django, una clase representa un registro de una tabla en la base de datos y las instancias de ésta serán las tuplas en la tabla.

2.2.7.10. Soporte de servidores Web

Django incluye un servidor web liviano para realizar pruebas y trabajar en la etapa de desarrollo. En la etapa de producción, sin embargo, se recomienda Apache 2. Aunque Django soporta la especificación WSGI, por lo que puede correr sobre una gran variedad de servidores como FastCGI o SCGI en Apache u otros servidores (particularmente Lighttpd).

2.2.7.11. Requerimientos

Django requiere Python 2.5 o superior. No se necesitan otras bibliotecas de Python para poder obtener una funcionalidad básica. En un entorno de desarrollo especialmente si queremos experimentar con Django no necesitamos un web server instalado, ya que Django trae su propio servidor liviano para éste propósito, con la restricción de solo permitir un usuario a la vez.

Base de datos: PostgreSQL, MySQL, Oracle o SQLite.

2.2.7.12. Patrón Modelo - Plantilla - Vista

Django tiene una interpretación diferente para el patrón MVC. En su esta, las Vistas describen qué datos serán presentados, y no necesariamente cómo los datos lucen.
Entonces, en su caso, una Vista representa la función que es llamada por una URL en específico, esta función a través del Modelo selecciona qué datos serán presentados. Luego, la Vista despacha los datos a una Plantilla (template) que describirá cómo los datos son presentados. Por estas razones, Django se define así mismo como un framework Modelo-Plantilla-Vista (MTV del inglés, Model - Template - View).

2.2.8. Mapeo Objeto - Relacional

El Mapeo Objeto Relacional o (ORM) consiste en una abstracción de alto nivel, es una técnica que permite crear una capa de comunicación en la que el acceso a una base de datos está basada en un ambiente orientado a objetos. Esto permite al programador escribir código Python (en el caso de Django) y no SQL (del inglés, Structured Query Language, Lenguaje de Consultas Estructuradas), permitiendo una abstracción referente al manejador de base de datos a usar.
Una consulta en lenguaje SQL como:
SELECT * FROM USUARIOS WHERE codigo=14894;
Es equivalente a esta porción en código Python:
usuarios = USUARIOS.objects.filter(codigo=94107)
Donde una clase es equivalente a una tabla (usuarios) y un atributo es equivalente a una columna de dicha tabla (codigo).
Entre las características y funcionalidades más relevantes, tenemos:

* Todos los modelos son representados en el archivo models.py.

* La cantidad de tipos de datos disponibles para los atributos es extensa, desde los tipos de datos básicos hasta tipos de datos dedicados a archivos.

* Es posible hacer todas las operaciones CRUD (crear, leer, actualizar y borrar).

* Soporta las relaciones uno-a-uno con el método OneToOneField(), uno-a-muchos con el método ForeignKey() y las relaciones muchos-a-muchos con el método ManyToManyField().

* Provee la posibilidad de escribir métodos propios por cada clase, permitiendo representar la mayoría de la lógica de negocios en el modelo.

* Provee un modúlo migratorio de SQL al esquema orientado a objetos. Es decir, que si la definición de la estructura de la base de datos solo la tenemos en lenguaje SQL este módulo migra ese código a clases escritas en Python y al revés.

Capítulo 3

Análisis

En este capítulo se determinan los requisitos funcionales como los no funcionales del sistema. Estos requisitos determinan de forma precisa las propiedades o restricciones que debe satisfacer el sistema. Los requerimientos que se dictan son el resultado de la investigación de las especificaciones de soluciones actuales y de las necesidades sin cubrir.

Una vez identificados los requerimientos se procede a realizar el modelo del sistema a través de la construcción de los diagramas de casos de uso.

Por último se presenta las entidades relevantes del sistema, sus relaciones y propiedades a través del modelo entidad-relación.

3.1. Modelo de entidad-relación

En el modelo entidad-relación de la figura se especifica la estructura de base de datos que debe implementar sistema. Cada entidad corresponde a una tabla de la base de datos:

users entidad que almacena las propiedades de los usuarios relevantes para identificar sus cuentas.

plots entidad utilizada para almacenar los gráficos creados por los usuarios registrados.

Un usuario puede o no tener gráficos creados, entonces un registro de la entidad users puede tener de ninguno a varios registros de la entidad plots, pero un registro de plots solo puede pertenecer a un autor o registro de users.

Figura 1: **Diagrama de clases**

3.2. Identificación de los requerimientos

3.2.1. Requerimientos funcionales

* 1. Definir los rangos en los ejes.
* 2. Seleccionar el estilo de dibujado.
* 3. Definir el titulo del gráfico.
* 4. Seleccionar las dimensiones del gráfico.
* 5. Seleccionar en caso de requerirlas, las funciones a gráficar.
* 6. Definir las etiquetas de los ejes.
* 7. Cargar en caso de necesitarlo, el archivo con los datos a procesar.
* 8. Definir en caso de ser necesario, las columnas.

* 9. Definir el alto y ancho de la imagen del resultado.
* 10. Definir el formato de la imagen del resultado.

3.2.2. Requerimientos no funcionales

* Confiabilidad: el sistema debe comportarse de acuerdo a los que los usuarios esperan de él, ejecutando las acciones deseadas en el tiempo preciso. Sin embargo, cuando la renderización es muy compleja este tiempo podría estar comprometido.
* Usabilidad: proporcionar una interfaz gráfica de usuario intuitiva, sencilla, poco propenso a errores y con metáforas establecidas y estandarizadas para un más fácil aprendizaje y uso del mismo.
* Seguridad: garantizar que toda la información contenida en el sistema debe estar protegida contra accesos no autorizados, mediante mecanismos de autenticación y sesiones que no permitan la fuga de datos.
* Consistencia: no deben existir contradicciones entre las funcionalidades del sistema.
* Accesibilidad: es accesible en los navegadores de escritorio: Chrome, Firefox, Opera y Safari, mediante los eventos de usuario accionados por los dispositivos de E/S estándar: ratón y teclado.

3.2.3. Diagrama de casos de uso

Los actores que se comunican con el sistema son los siguientes:

Usuario autenticado: es un tipo de usuario que tiene los siguientes permisos:

* Gestionar lista de gráficos creados desde su cuenta.
* Acceder a la sección para crear gráficos.

Usuario no autenticado: es un tipo de usuario que tiene los siguientes permisos:

∗ Acceder a la sección para crear gráficos.

En la figura se ilustra el diagrama de casos de uso en el nivel 0. En este nivel se describe únicamente los actores que intervienen en el sistema.

En la Figura 3.2 se muestra el diagrama de casos de uso en el nivel 1. En este nivel se desglosa las funcionalidades del sistema y se indica la asociación de estos con los actores.

En la figura 3.6 se muestra el diagrama de casos de uso en el nivel 2. En este nivel se muestra a mayor detalle las relaciones de extensión, inclusión y generalización entre los casos de uso.

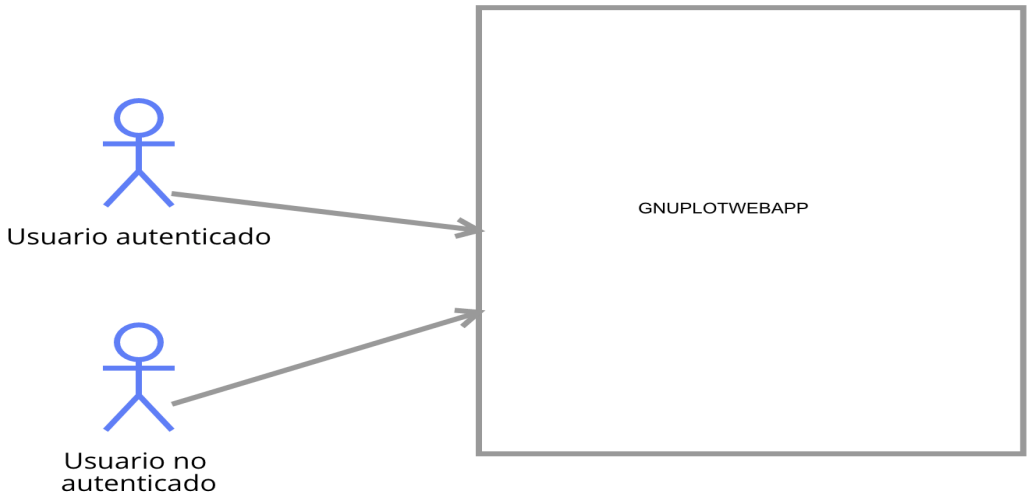


Figura 2: **Casos de uso nivel 0**

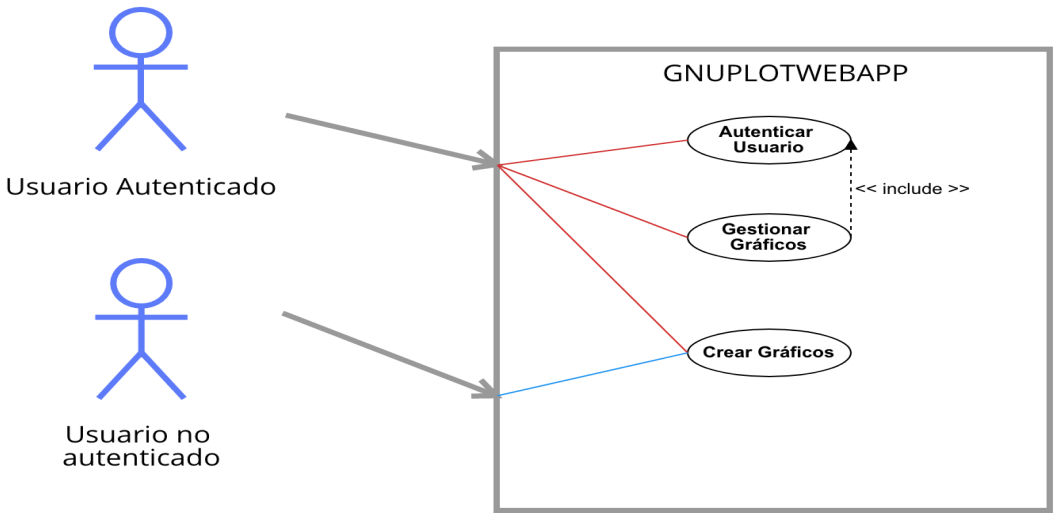


Figura 3: **Casos de uso nivel 1**

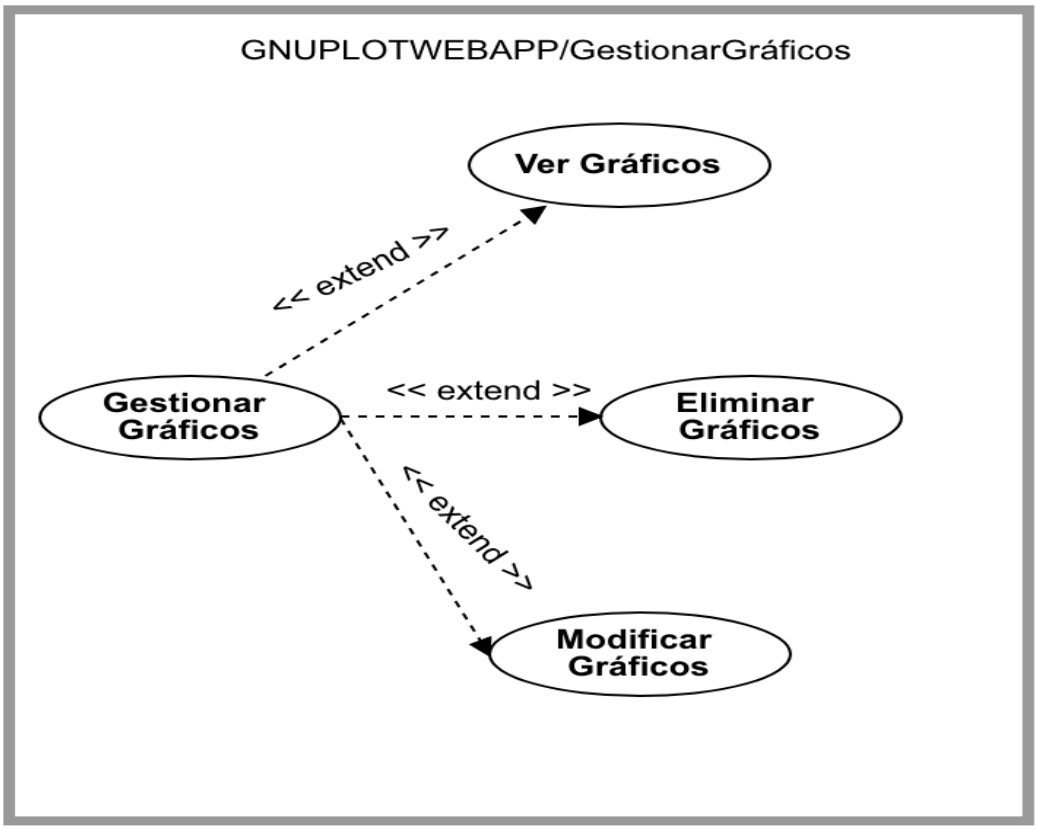


Figura 4: **Casos de uso nivel 2**

Capítulo 4

Diseño e Implementación

Este capítulo trata sobre el proceso de traducir los requisitos del capítulo anterior en una representación de software, que se asemeja al código fuente. Es decir, implementa todos los requisitos explícitos en la etapa de análisis.

Se muestra en detalle la arquitectura del sistema con la finalidad de ser una guía para que los desarrolladores puedan entender, probar y mantener el software. La arquitectura es la estructura jerarquizada de los módulos del programa, la interacción entre ellos y las interfaces usadas por estos módulos.

Para apoyar el desarrollo, aliviando el exceso de carga asociado con actividades comunes usadas en desarrollos web, se utiliza un framework para aplicaciones web, en este caso Django versión 2.2. El framework se basa en el patrón de **arquitectura MPV**.

La solución basada en MPV, está formada por tres módulos principales: Modelo, Plantilla y Vista. El objetivo de dividir la solución en módulos es hacerla más legible, manejable y fácil de mantener. Cada uno de los módulos del sistema representa una capa lógica con tareas específicas. Dichos módulos se pueden comunicar entre ellos a través de interfaces bien definidas, tales como clases, métodos y atributos.

La arquitectura general del sistema se define como arquitectura MPV. Sin embargo

para estudiar con mayor detalle la solución, es conveniente separar el diseño de la creación de gráficos del diseño de manejo de almacenamiento de documentos y cuentas de usuarios ya que para estas ultimas el framework se encargo de hacer el trabajo permitiendo concentrarnos únicamente en la solución al problema de graficación.

4.1. Arquitectura

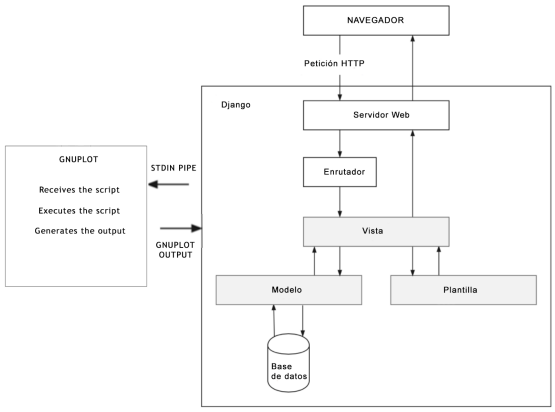


Figura 5: **Arquitectura de la solución**

De esta manera como se muestra en la figura 5, cuando llega una nueva solicitud al servidor la vista se encarga de procesar los datos que fueron ingresados por el usuario en el formulario para la creación de un nuevo gráfico y realizar la ejecución de Gnuplot con el script correspondiente. Una vez ha finalizado la ejecución de Gnuplot, el nuevo gráfico generado se almacena en la base de datos esta tarea se lleva a cabo gracias a la representación en el modelo de datos. Por ultimo se muestra el gŕafico generado al usuario dentro de la plantilla asociada.

4.1.1. Modelo de datos

A continuación se presenta el modelo de datos que se realizó para los gráficos y el de los usuarios del sistema respectivamente.

```
from django.db import models
class Plot(models.Model):

  style = (
      ('line', 'linesPoint'),
      ('vector', 'vectors'),
      ('point', 'points'),
      ('bar', 'bars'),
      ('candlesticks', 'candlesticks')
  )

  type = (
      ('2d','2d'),
      ('3d','3d'),
  )

  out = (
    ('svg','svg'),
    ('png','png'),
  )


  title = models.CharField(max_length=200, blank=True)

  script = models.TextField()

  draw_style = models.CharField(max_length=200, choices= style, default
   ='line')

  x_range = models.CharField(max_length=300, blank=True)

  y_range = models.CharField(max_length=300, blank=True)

  z_range = models.CharField(max_length=300, blank=True)

```

```
35    plot_type = models.CharField(choices = type, max_length=100, default=
        '2d')
36
37    output = models.CharField(choices = out, max_length=100, default='png
        ')
38
39    function = models.TextField()
40
41    width = models.CharField(max_length = 50, default='600')
42
43    height = models.CharField(max_length = 50, default='400')
44
45    x_label = models.CharField(max_length=200, blank=True)
46
47    y_label = models.CharField(max_length=200, blank=True)
48
49    z_label = models.CharField(max_length=200, blank=True)
50
51    data = models.FileField(upload_to='data/',blank=True)
52
53    columns = models.CharField(max_length=200, blank=True)
54
55    pub_date = models.DateTimeField('date published')
56
57    user = models.ForeignKey(CustomUser, on_delete=models.CASCADE)
58
59    def __str__(self):
60      return self.script
```

```
1  from django.db import models
2  from django.contrib.auth.models import AbstractUser
3
4  class CustomUser(AbstractUser):
5    name = models.CharField(max_length=50)
6
7    def __str__(self):
8      return self.name
```

4.1.2. Plantillas

A continuación se muestra el código referente a las plantillas donde se muestran los gráficos creados por el usuario y el formulario para la creación de los mismos.

```
{% extends "base_generic.html" %}

{% block content %}
    <h1>Plot By {{plot2.user.name}}</h1>

        <pre> {{plot2.script}} </pre>

        <img src="/media/plots/{{plot2.user.id}}/{{plot2.id}}.{{plot2.output}}" height={{plot2.height}} width={{plot2.width}}>

{% endblock %}
```

```

{% extends "base_generic.html" %}

{% block content %}
  <h1>New Plot</h1>
        <form method="POST" enctype="multipart/form-data" class="post-form">{% csrf_token %}
          {{ form.as_p }}
          <button type="submit" class="save btn btn-default">Save</button>
        </form>

{% endblock %}
```

4.1.3. Vistas

A continuación se presentan la vista que corresponde a la creación de un nuevo gráfico seguido de las vistas que se utilizan para desplegar informacion dentro del sistema.

```
def PlotNew2(request):
    if request.method == "POST":
        form = FormPlot2(request.POST, request.FILES)
        if form.is_valid():

            plot = form.save(commit=False)

            plot.user = request.user
            plot.pub_date = timezone.now()

            if plot.user:
                plot_dir = "media/plots/"+str(plot.user.id)

            if not os.path.exists(plot_dir):
                os.makedirs(plot_dir)

            plot.script = "set terminal "+plot.output+ " size "+plot.
    width+","+plot.height+"\n"

            if plot.title:
                plot.script+="set title "+'"'+plot.title+'"'+'\n'

            if plot.x_label:
                plot.script+="set xlabel "+'"'+plot.x_label+'"'+'\n'

            if plot.y_label:
                plot.script+="set ylabel "+'"'+plot.y_label+'"'+'\n'

            if plot.z_label:
                if plot.plot_type=="3d":
                    plot.script+="set zlabel "+'"'+plot.z_label+'"'+'\n
    '

            if plot.x_range:
```

```
33              plot.script += "set xrange [ "+plot.x_range+"]
    noreverse nowriteback\n"
34          if plot.y_range:
35              plot.script += "set yrange [ "+plot.y_range+"]
    noreverse nowriteback\n"
36          if plot.z_range:
37              if plot.plot_type=="3d":
38                  plot.script += "set zrange [ "+plot.z_range+"]
    noreverse nowriteback\n"
39
40
41          plot.save()
42
43          plot_file = plot_dir+"/"+str(plot.id)+"."+plot.output
44          out = "set output '"+plot_file+"'\n"
45          plot.script += out
46
47          if(len(form.cleaned_data['functions']) > 0):
48              fx = ""
49              l = 1
50              for k in (form.cleaned_data['functions']):
51                  if(l == len(form.cleaned_data['functions'])):
52                      fx+=k
53                  else:
54                      fx+=k+", "
55                  l+=1
56              plot.script += "plot "+fx+" \n"
57
58          if plot.plot_type=="2d":
59              if plot.data and plot.columns:
60                  plot.script += "plot '"+plot.data.url[1:]+"' using
    "+plot.columns+ " with "+plot.get_draw_style_display()+"\n"
61              elif plot.data:
62                  plot.script += "plot '"+plot.data.url[1:]+"' with "
    +plot.get_draw_style_display()+"\n"
63
64          plot.save()
65
```

```
66              proc = subprocess.Popen(['gnuplot','-p'],shell=True, stdin=
    subprocess.PIPE,)
67              proc.communicate(plot.script.encode())
68
69              return redirect('plotlive:plot_detail', pk=plot.pk)
70      else:
71          form = FormPlot2()
72      return render(request, 'plotlive/plot_edit2.html', {'form': form})
```

```
1  class PlotDetail(generic.DetailView):
2    template_name = 'plotlive/plot_detail.html'
3    model = Plot2
4
5  class UserPlotsView(generic.DetailView):
6    template_name = 'plotlive/user_plots.html'
7    model = CustomUser
8
9
10 class PlotListView(generic.ListView):
11   model = Plot2
12   def get_queryset(self):
13     return Plot.objects.order_by('-commands')[:2]
```

4.2. Funcionamiento del sistema

A continuación se presenta en la siguiente serie de imágenes un ejemplo de una corrida del sistema, los resultados, y lo que ocurre dentro del sistema para obtener los mismos.

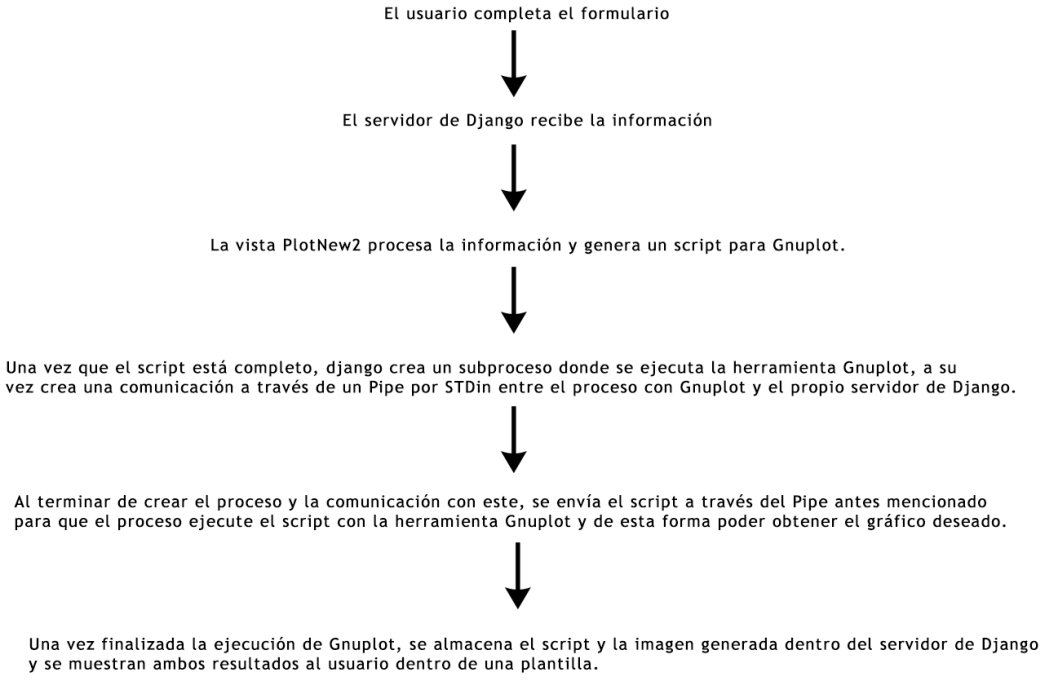


Figura 6: **Funcionamiento del sistema**

Figura 7: **Datos ingresados en el formulario por el usuario**

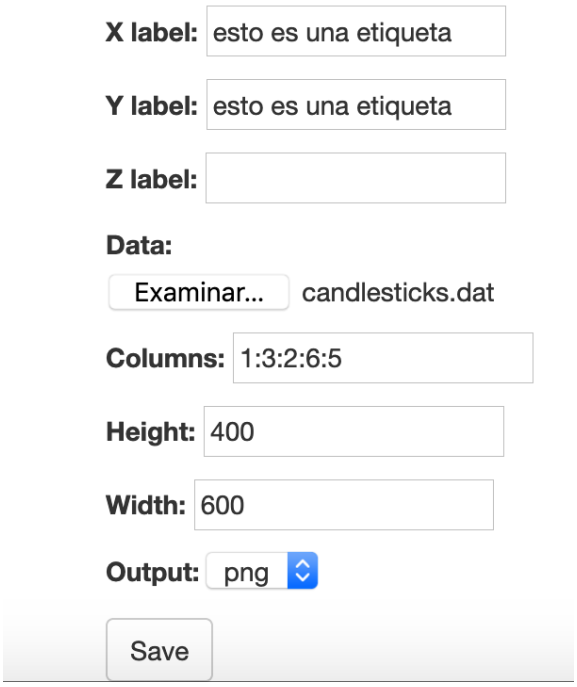


Figura 8: **Datos ingresados en el formulario por el usuario**

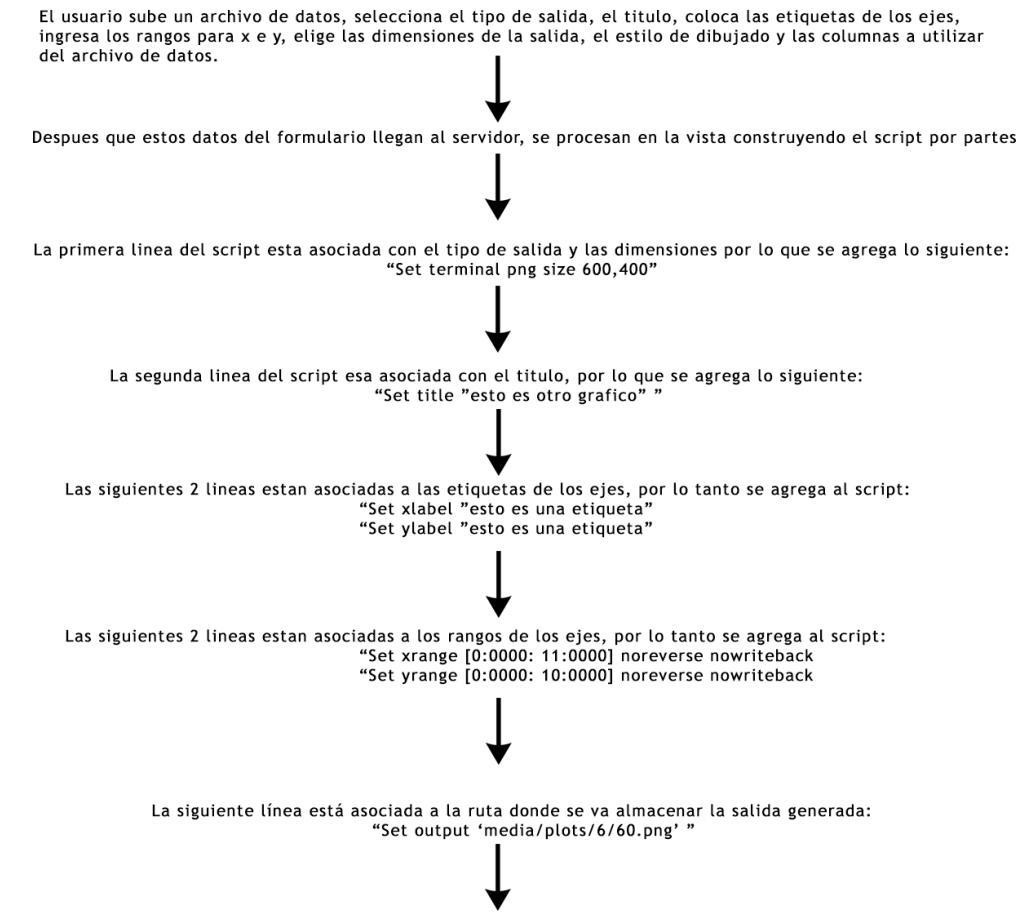


Figura 9: **Lo que ocurre dentro del sistema**

Plot By carlos

```
 set terminal png size 600
set title "esto es otro grafico"
set xlabel "esto es una etiqueta"
set ylabel "esto es una etiqueta"
set xrange [ 0.00000 : 11.0000] noreverse nowriteback
set yrange [ 0.00000 : 10.0000] noreverse nowriteback
set output 'media/plots/6/60.png'
plot 'media/data/candlesticks_D5VpjTv.dat' using 1:3:2:6:5 with candlesticks
```

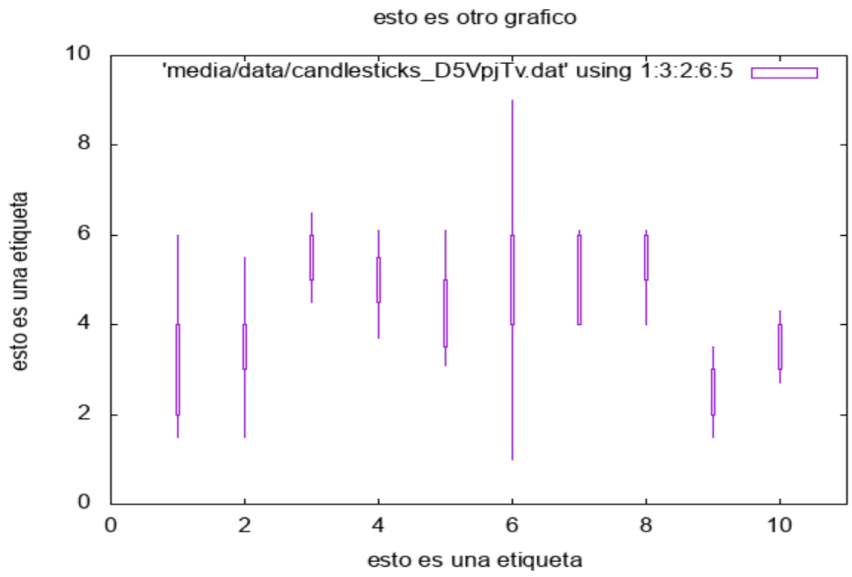


Figura 10: **script y gráfico generado**

Capítulo 5

Pruebas y Resultados

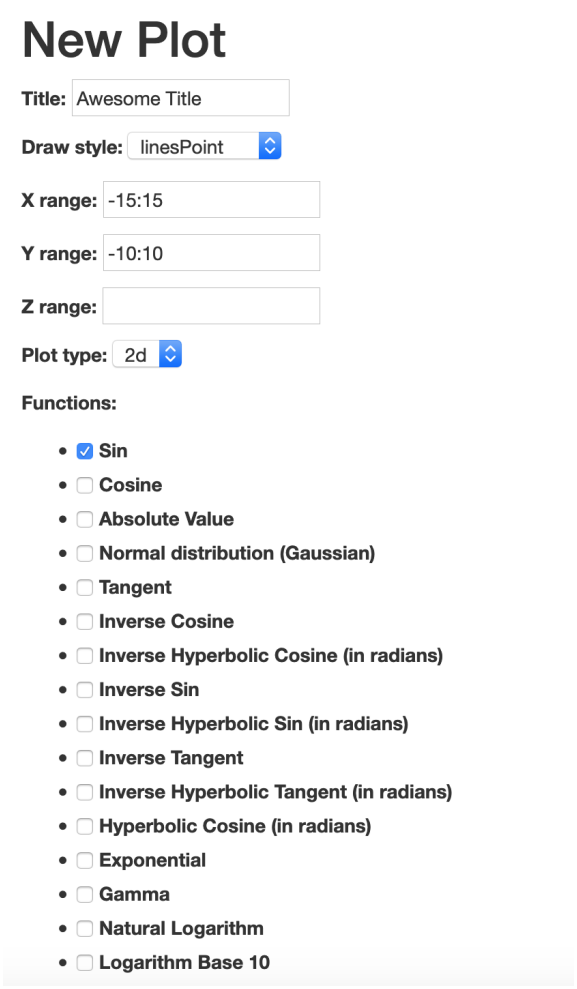
En este capítulo se expone la potencialidad de la aplicación mediante la descripción del proceso de creación de un gráfico empleando la colección de recursos que dispone la aplicación.

Para validar la efectividad de este proceso, la aplicación pasa por una serie de pruebas cualitativas obtenidas mediante la realización de encuestas a un grupo voluntario de profesores y estudiantes conformado por un total de 22 personas.

Por último se identifican las diferencias y se realiza un análisis comparativo entre la solución implementada con el resto de las aplicaciones disponibles, ésto con el objetivo de comprobar el aporte del trabajo realizado cubriendo sustancialmente las necesidades que no han sido atendidas, al menos por las aplicaciones encontradas durante la investigación.

5.1. Resultados en ejecución

A continuación se muestran en las siguientes figuras el formulario para crear gráficos y el resultado obtenido.


New Plot
Title: Awesome Title
Draw style: linesPoint
X range: -15:15
Y range: -10:10
Z range:
Plot type: 2d
Functions:
Sin
Cosine
Absolute Value
Normal distribution (Gaussian)
Tangent
Inverse Cosine
Inverse Hyperbolic Cosine (in radians)
Inverse Sin
Inverse Hyperbolic Sin (in radians)
Inverse Tangent
Inverse Hyperbolic Tangent (in radians)
Hyperbolic Cosine (in radians)
Exponential
Gamma
Natural Logarithm
Logarithm Base 10

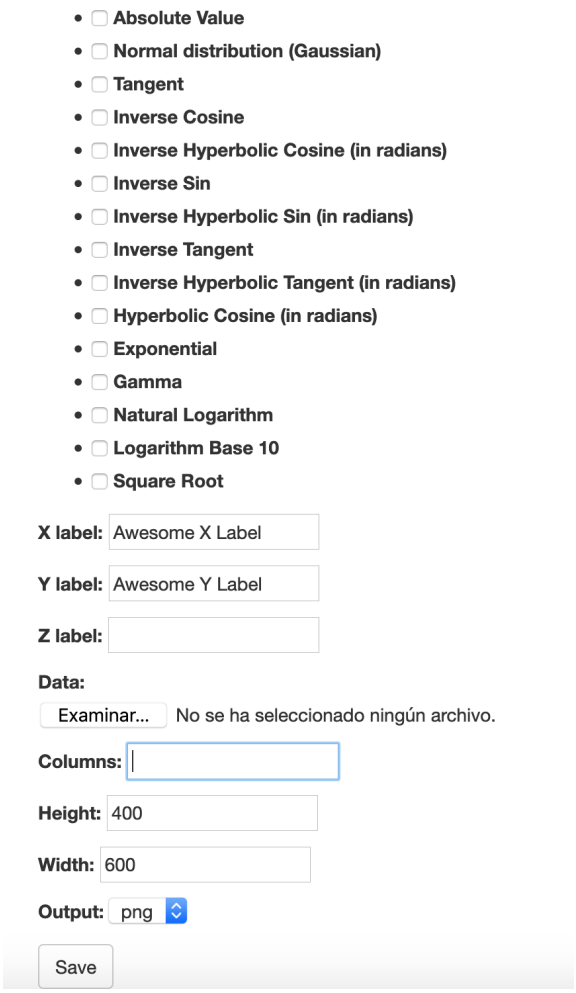


Figura 11: **Formulario para la creación de gráficos**

Plot By super

```
 set terminal png size 600,400
set title "Awesome Title"
set xlabel "Awesome X Label"
set ylabel "Awesome Y Label"
set xrange [ -15:15] noreverse nowriteback
set yrange [ -10:10] noreverse nowriteback
set output 'media/plots/7/81.png'
plot sin(x)
```

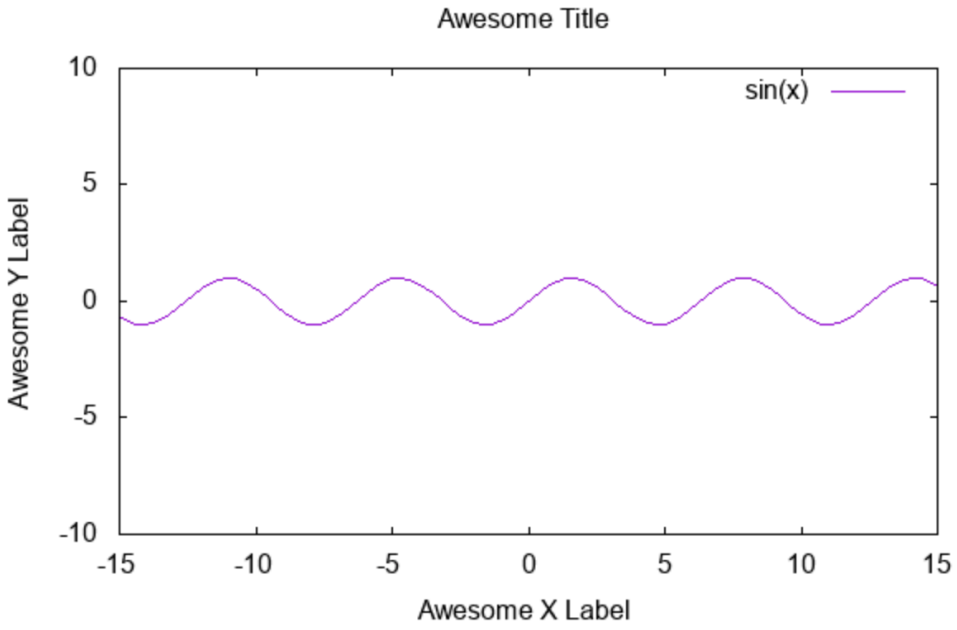
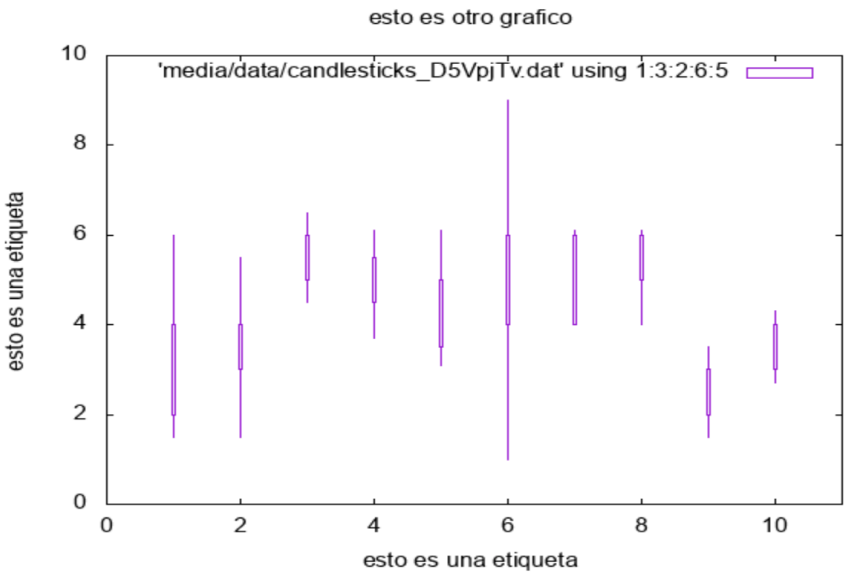


Figura 12: **Resultado del gráfico generado en la ejecución**

En las siguientes figuras se muestran gráficos obtenidos con la aplicación en otras ejecuciones.

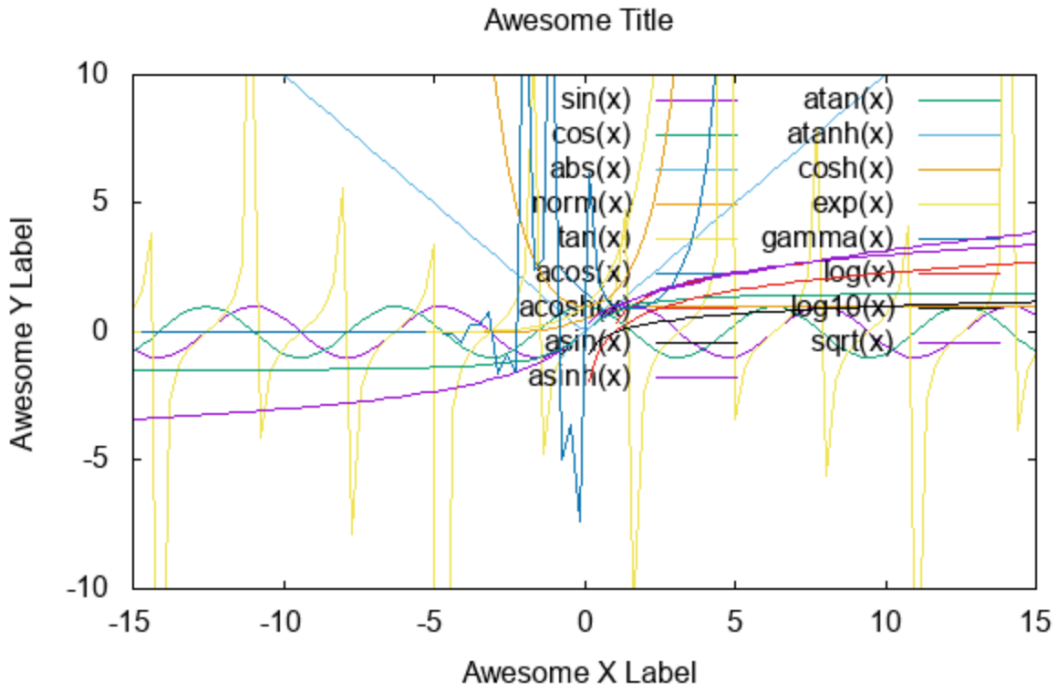
Plot By carlos

```
 set terminal png size 600
set title "esto es otro grafico"
set xlabel "esto es una etiqueta"
set ylabel "esto es una etiqueta"
set xrange [ 0.00000 : 11.0000] noreverse nowriteback
set yrange [ 0.00000 : 10.0000] noreverse nowriteback
set output 'media/plots/6/60.png'
plot 'media/data/candlesticks_D5VpjTv.dat' using 1:3:2:6:5 with candlesticks
```



Plot By super

```
 set terminal png size 600,400
set title "Awesome Title"
set xlabel "Awesome X Label"
set ylabel "Awesome Y Label"
set xrange [ -15:15] noreverse nowriteback
set yrange [ -10:10] noreverse nowriteback
set output 'media/plots/7/82.png'
plot sin(x), cos(x), abs(x), norm(x), tan(x), acos(x), acosh(x), asin(x), asi
```



5.2. Pruebas y resultados cualitativos

Las pruebas cualitativas se basan en medir la apreciación que tienen los usuarios acerca del sistema. Para estas pruebas se realizaron encuestas a un grupo de profesores y estudiantes de ciencias de la computación sumando 22 personas en total qué

debieron responder basado en la experiencia obtenida durante la interacción con el sistema.

Este grupo de usuarios tenían la tarea de dibujar el gráfico de la Figura 6.

A continuación se lista los criterios evaluados con sus respectivos resultados:

1. Definición de estilos.

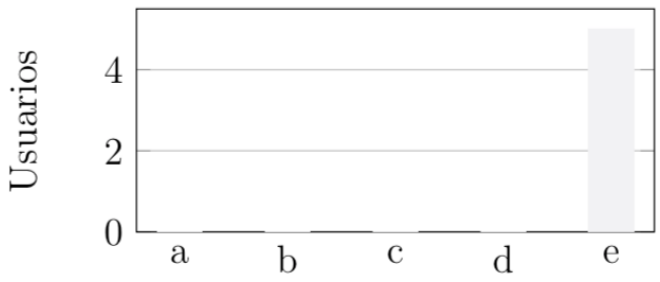
(a) pésimo

(b) mal

(c) indiferente

(d) bien

(e) excelente



2. Distribución de los elementos

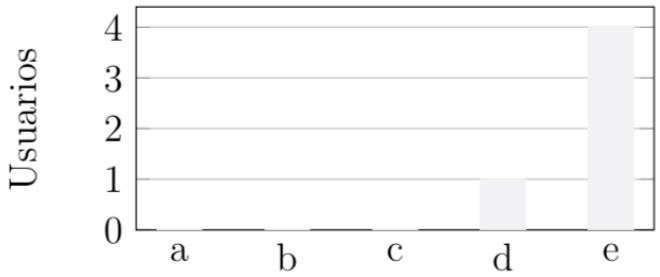
(a) desordenada

(b) un poco desordenada

(c) regular

(d) casi ordenada

(e) ordenada



3. Formas gráficas y lenguaje

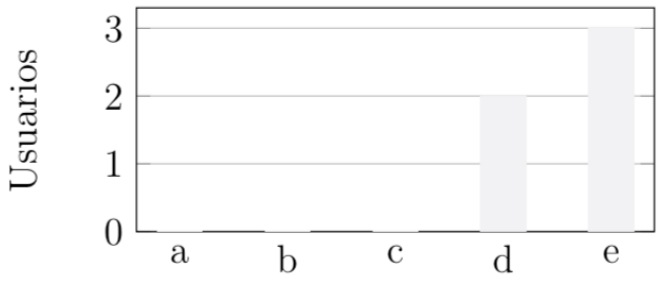
(a) pésimo

(b) mal

(c) indiferente

(d) bien

(e) excelente



4. Disponibilidad de recursos para definir la estructura del gráfico

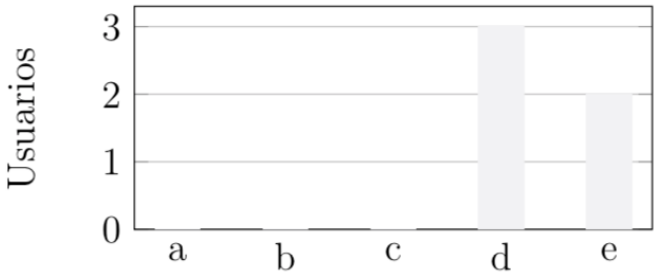
(a) insuficiente

(b) casi insuficiente

(c) básico

(d) casi suficiente

(e) completo



5. Disponibilidad de recursos para definir el estilo del gráfico

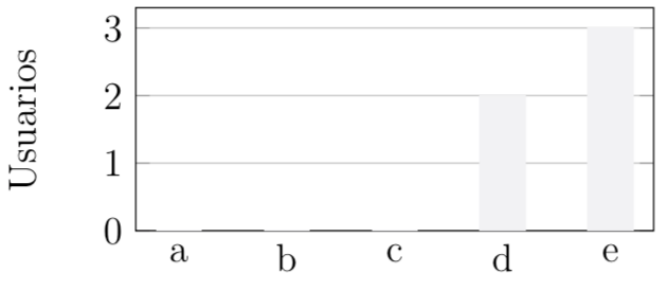
(a) insuficiente

(b) casi insuficiente

(c) básico

(d) casi suficiente

(e) completo



6. Complejidad de las herramientas para definir la estructura del gráfico

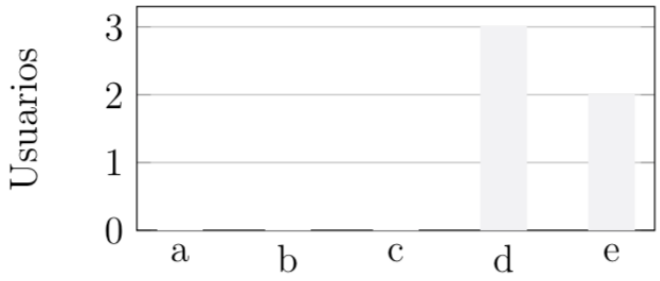
(a) frustrante

(b) difícil

(c) normal

(d) fácil

(e) muy fácil



7. Complejidad de las herramientas para definir el estilo de los gráficos

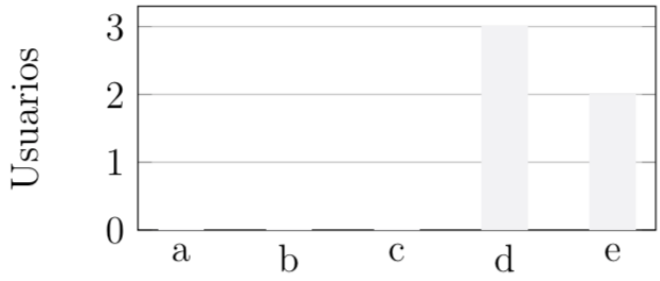
(a) frustrante

(b) difícil

(c) normal

(d) fácil

(e) muy fácil



8. Está de acuerdo con los valores de estilo sugeridos por el sistema?

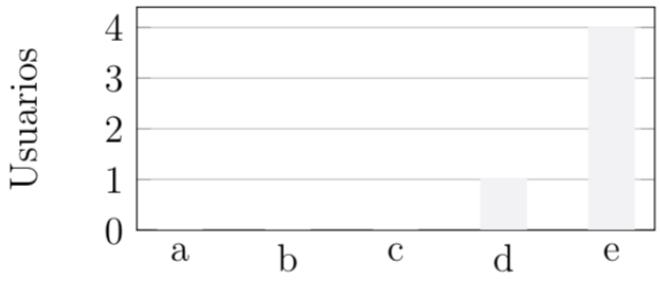
(a) nunca

(b) pocas veces

(c) a veces

(d) casi siempre

(e) siempre



9. Respuestas inesperadas en la ejecución de acciones

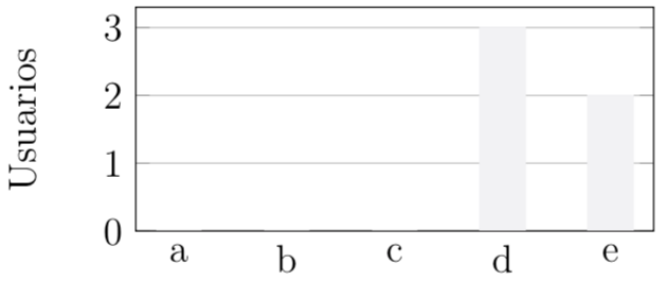
(a) nunca

(b) pocas veces

(c) a veces

(d) casi siempre

(e) siempre



10. Velocidad de respuesta en el tiempo preciso

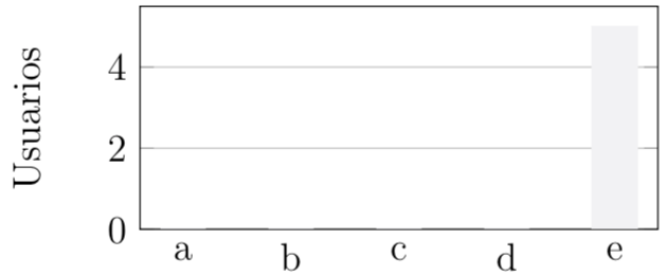
(a) nunca

(b) pocas veces

(c) a veces

(d) casi siempre

(e) siempre



5.2.1. Análisis de los resultados del estudio cualitativo

Se analizan los resultados de los criterios evaluados en cuatro tópicos:

Presentación en esta categoría se evalúa la apariencia de la interfaz gráfica de usuario (GUI por su nombre en inglés, Graphical User Interface). Se considera la definición de estilos, como la paleta de colores y los formatos de texto, la distribución de los elementos, las formas gráficas o imágenes, como botones e íconos y el lenguaje. La GUI es la carta de presentación del producto, que en muchas ocasiones, es la que determina si la aplicación será o no utilizada para resolver los problemas para los cuales fue diseñada.

Los resultados de las encuestas indican que la GUI cumple satisfactoriamente con los estándares de diseño para ofrecer un entorno visual sencillo y amigable, lo que contribuye a la realización de las tareas de manera rápida y eficaz.

Disponibilidad de recursos se refiere al conjunto de herramientas que dispone el editor para la construcción de ǵráficos que cumplan con las expectativas de los usuarios.

Según el resultado de las encuestas, tanto para la construcción de los gráficos como la definición de estilos, el editor dispone de una serie suficiente funcionalidades que permiten cubrir las exigencias de los usuarios, no está totalmente completa pero tampoco excede los recursos al punto de abarcar funcionalidades innecesarias.

Complejidad del manejo de las herramientas se evalúa la interfaz desde el punto de vista de usabilidad del editor, esto se refiere a la complejidad del proceso con la que el usuario puede manejar las herramientas para la construcción de los gráficos. Los criterios de presentación y disponibilidad de recursos, anteriormente mencionados, colaboran con el grado de complejidad del manejo de las herramientas; una buena presentación ayuda a recrear un entorno de trabajo sencillo y amigable, y una gama completa de recursos permite cubrir las peticiones de los usuarios en menor cantidad de pasos.

La apreciación obtenida por los usuarios señala que el proceso de edición tanto para la construcción de los gráficos, pueden ser operados con facilidad, no la óptima, pero suficiente para deducir que la solución implementada cumple satisfactoria con los criterios de usabilidad, aquella característica que hace que la aplicación sea fácil de utilizar y fácil de aprender.

Confiabilidad en la respuesta se refiere al comportamiento del sistema de acuerdo a lo que se espera de él, es decir, la ejecución de las acciones deseadas sin errores o respuestas inesperados en el tiempo preciso.

Los resultados de las encuestas sugieren que la solución es efectiva, hace lo que se espera que haga, y es eficiente, realiza las tareas en el tiempo necesario y sin errores.

5.3. Comparaciones

Durante la investigación se realizo una busqueda de aplicaciones web que permitieran la creación de gráficos utilizando la librería Gnuplot sin tener la necesidad de realizar un Script, esta busqueda no arrojo resultado alguno y esto ha sido una de las principales motivaciones de realizar este proyecto.

Capítulo 6

Conclusiones, Recomendaciones y Trabajos Futuros

Las conclusiones tras la investigación teórica, el desarrollo de la solución y los resultados obtenidos se centran en este capítulo. Adicionalmente se sugiere las recomendaciones y posibles trabajos a futuro para la continuación de la investigación y desarrollo del área.

6.1. Conclusiones

HTML se mantiene evolucionando para proporcionar mejoras y más ricos gráficos estandarizados, de este modo, los desarrolladores tienen la oportunidad de crear aplicaciones web ricas en gráficos usando tecnologías basadas en estándares evitando la instalación de complementos o códigos específicos del explorador.

La aplicación web desarrollada permite gráficar utilizando la herramienta Gnuplot sin tener ningun conocimiento acerca de la misma y guardar los gráficos generados.

Esta aplicación a diferencia de las aplicaciones web disponibles para utilizar Gnuplot posee una interfaz que abstrae al usuario de crear Scripts para generar sus gráficos y unicamente se enfoque en el gráfico que desee generar, permitiendo de esta manera que usuarios que no estén familiarizados con algún tipo de scripting.

Los resultados basados en la apreciación de un grupo de usuarios de ciencias de la computación han probado que el conjunto de herramientas que brinda la interfaz gráfica de usuario facilita la construcción de Gráficos complejos en simples pasos minimizando el costo, el tiempo y el riesgo a errores.

La aplicación genera aportes en la comunidad docente y estudiantil en las materias que están orientadas al calculo científico.

6.1.1. Recomendaciones

En las notas de docencia de las materias dictadas en la Universidad Central de Venezuela:

Calculo Científico Probabilidad y Estadísticas

La herramienta desarrollada puede ser sumamente útil para las notas de docencia de las materias.

Los estudiantes que cursan las materias mencionadas frecuentemente tienen que generar gráficos para el estudio de varios temas que estudian en dichas materias.

6.2. Trabajos Futuros

Algunos de los posibles objetivos que se plantean para continuar con la línea de desarrollo del sistema a partir de la investigación realizada son los siguientes:

- Opción de ayuda que muestre una guía breve del uso de la aplicación.
- Ampliar las funciones de Gnuplot que se pueden utilizar dentro de la aplicación
- Compartir los gráficos generados en redes sociales.
- Envió de contraseña y nombre de usuario por correo electrónico.

- Creación de gráficos mediante drag and drop

Bibliografía

[1] Gnuplot Official Homepage, [Fecha de consulta: 20 de agosto de 2018]. disponible en: *http://www.gnuplot.info/*

[2] Arquitectura Cliente-Servidor,[Fecha de consulta: 20 de agosto 2018]. disponible en: *https://es.wikipedia.org/wiki/Cliente-servidor*

[3] Benjamin Aumaille (2012). Desarrollo de aplicaciones Web.

[4] Lujan Mora, Sergio (2001). Programación en internet: Clientes Web.

[5] The Clean Architecture, [Fecha de consulta: 22 de agosto de 2018]. disponible en: *https://8thlight.com/blog/uncle-bob/2012/08/13/the-clean-architecture.html*

[6] W3C, Ian Hickson y David Hyatt, [Fecha de consulta: 22 de agosto de 2018]. disponible en: *www.w3.org*

[7] Cederholm, Dan; Ethan Marcotte (2009). Handcrafted CSS: More Bulletproof Web Design.

[8] var - JavaScript - MDN. The Mozilla Developer Network. [Fecha de consulta: 22 de agosto de 2018].

[9] MySQL 5.7 Release Notes. Oracle. [Fecha de consulta: 22 de agosto de 2018].

[10] Django Project Homepage, [Fecha de consulta: 22 agosto de 2018]. disponible en: *https://www.djangoproject.com/*