Universidad de las Ciencias Informáticas

Facultad 6

Título: “Servidor de Gráficos Chart Server versión 2.0”

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas.

Autores: Arelis Isada Molina

Jorge Luis Quintana Díaz

Tutores: Ing. Marleysi López Duque

Ing. Dioleisys Fontela González

La Habana, Cuba

Mayo de 2013

“Año 55 de la Revolución”

“Creemos en los jóvenes, creemos en los jóvenes, creemos en los jóvenes —y lo repito—porque creer en los jóvenes significa una actitud, creer en los jóvenes significa un pensamiento.”

"Si los jóvenes fallan, todo fallará. Es mi más profunda convicción que la juventud cubana luchará por impedirlo. Creo en ustedes.”

Fidel Castro Ruz

Declaración de autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

_____	_____
Arelis Isada Molina	Jorge Luis Quintana Díaz
Firma del Autor	**Firma del Autor**
_____	_____
Ing. Marleysi López Duque	Ing. Dioleisys Fontela González
Firma del Tutor	**Firma del Tutor**

Datos de contacto

Tutor: Ing. Marleysi López Duque
Ingeniero en Ciencias Informáticas
Correo electrónico: mduque@uci.cu

Tutor: Ing. Dioleisys Fontela González
Ingeniero en Ciencias Informáticas
Correo electrónico: dfontela@uci.cu

Autor: Arelis Isada Molina
Correo electrónico: aisada@estudiantes.uci.cu

Autor: Jorge Luis Quintana Díaz
Correo electrónico: jlquintana@estudiantes.uci.cu

Agradecimientos

Jorge
A mis familiares: Que siempre me apoyaron, me dieron consejos, y me demostraron confianza.
Mis padres: Lo son todo para mí, sigo sus ejemplos en todas las formas posibles. Yo trato siempre de ser una copia exacta de mi papá y muchas cosas de las que aprendí de él son las me llevaron a donde me encuentro hoy en día. Mi mamá es una madre especial, como ella no existe ni va existir para mí alguien que se parezca a ella, siempre me dio su apoyo, su amor, su confianza y dedicación. A ambos les digo "Muchas gracias por siempre estar al lado mío los quiero muchísimo".

Mi abuela: Mi abuela es un caso especial, ella es mi segunda madre, siempre está ahí para todo lo que necesito incluso lo que no necesito. Cuando estoy con ella siento lo mismo que al lado de mi mamá, me transmite el mismo sentimiento de amor y cariño.

Carlito: Agradecerte por estar siempre presente sin importar la distancia que nos separa. Por ser preocupado y por quererme a mí y Arelis como tus hijos.

Mi novia: Ella fue, es y seguirá siendo una persona muy importante en mi vida. En todo el curso de mi carrera siempre me ayudaba, me daba consejos y estaba presente siempre para mí. A ella le digo "Gracias por haber aparecido en mi vida".

La familia de mi novia: Agradecer muchísimo a estas personas tan importantes, que me demostraron que me aceptaron como un miembro más de su familia. Nunca mostraron ninguna indiferencia hacia mí y me demostraron su amor y cariño.

Mis amigos: Quiero agradecerle a los amigos que de una forma u otra me ayudaron en el transcurso de la universidad: Fidel, Arian, José Roberto, Edgar, Wilfredo y el Flaco.

Agradecimientos

Arelis
A mi mamita querida por ser la mejor madre del mundo. Te amo mucho.
A mi abuelita flaquita por ser la mejor abuela del mundo. Te Amo mucho Abu.

A mi hermanito Samuel Alejandro por ser la personita que me impulsa a lograr mis metas y sueños. Sami gracias a ti conocí el amor de hermanos, gracias por ser el hermanito que eres, por darme cada día esa sonrisa y esas travesuras que son las que me hacen feliz. Te quiero Sami.

A mi hermanito Cesar Antonio hermanito cómico te agradezco por darme la oportunidad de quererte, a pesar de las trabas de la vida. La vida nos ha demostrado que el querer de hermanos es sagrado. Gracias por tu cariño y ternura. Te quiero Cesar.

A mi Abuelo por demostrarme que no importa como seas, lo que importa es lo que sientes y las acciones que haces. Gracias abuelo por todo.

A mi familia por demostrarme ese amor lindo y sincero, por darme el apoyo incondicional en todo momento y por la confianza que depositan en mí día a día.

A mi novio Jorge: Papi lindo gracias por todo lo que has hecho por mí, gracias a ti cumpliré mi sueño. Te agradezco mucho por permitirme conocer al hombre más especial que existe en el mundo y ese hombre eres tú. Te Amo.

A la Familia de Jorge: por permitirme ser un miembro más en su familia, por quererme como si fuera Jorgito, sin ustedes me habría sido imposible llegar hasta aquí; me han dado el apoyo y el amor de una familia. Los quiero mucho y les agradezco por todo.

A mi suegrita gracias por quererme, te quiero mucho y no tengo como agradecerte todo lo que has hecho por mí, ojala algún día yo te pueda recompensar.

A Mirtha por ser mi mimi por ser una abuela más para mí, por darme todo el cariño y apoyo que necesitaba. Te quiero mucho mimi. Gracias por permitirme ser parte de tu familia.

A Yanet te agradezco por todo, por tu dedicación, preocupación y amor hacía mí. Por darme ese ánimo que solo tú sabes dar.

A Carlito a pesar del poco tiempo que nos conocemos me has brindado tu cariño y afecto. Gracias por querernos como si fuéramos tus hijos.

A Maura por darme consejos como si fuera mi abuela, por quererme y tratarme como si fuera miembro de su familia. Te agradezco mi gorda por todo lo que has hecho por mí.

Agradecimientos

A Minerva le agradezco por muchas cosas, por cuidar de mí familia en el momento que más necesitaba, por prestarnos el teléfono sin límites y por ser la vecina que es.

A mis amigos les agradezco por creer en mí y por darme ese apoyo que siempre necesité en especial a: Robert, Lázaro, Carménate, Gretell y Dadira.

A Cecilia y familia por ayudar a mi mamá en la operación, gracias a ustedes todo salió bien. Gracias.

A mis tutores Marleysi y Dioleisys por ser los guías de este trabajo, por su apoyo incondicional y su preocupación. Gracias por haber estado siempre en los momentos que los necesitaba.

Gracias al tribunal y al oponente por sus revisiones constructivas que hicieron posible este sueño.

A todas las personas que de una forma o de otra hicieron posible la realización de este trabajo de diploma.

A mi Dios lindo, mi San Lázaro y mi Virgencita gracias por cuidarme y por protegerme siempre al igual que a mi familia. Les agradezco por permitirme conocer la dicha de creer en ustedes y ser feliz.

Gracias a la Revolución y a nuestro Fidel Castro Ruz por permitirme llegar hasta aquí.

Dedicatoria

Jorge
Mi trabajo de diploma va dedicado sin duda alguna a toda mi familia, pero existe una persona muy importante para mí a la cual le quiero dedicar estas palabras.

"Purro ya cumplí el sueño que tenías y del que tanto conversábamos cuando estábamos acostados en tu cama, siempre fuiste la persona que me daba impulso para ir adelante sin parar. Como te prometí este diploma va dedicado a ti el cual va para tu cuarto como lo conversamos. Nunca tuve la oportunidad de despedirme de ti y no quería hacerlo hasta el día que me graduara así que ahora te digo Abuelo siempre te voy a querer muchísimo y nunca te voy a defraudar".

Dedicatoria

Arelis

Quiero dedicarles este trabajo a las personas que más amo en la vida y que sin ellos habría sido imposible cumplir este sueño.

A mi mamita linda por ser la persona más importante que hay en mi vida, por darme ese amor incondicional que solo una madre sabe dar. Por confiar en mí en todo momento y por darme la oportunidad de ser hija de una mujer tan especial y maravillosa. Me siento orgullosa de haber podido realizar un sueño de las dos. Te quiero dar las gracias por dedicarte tanto, a mí y a mi hermanito, eres una persona increíble y muy bella, eres lo más sagrado que tengo en la vida. Mamita muchas gracias por existir y ser mi mamá.

Te Adoro mamita.

A mi abuelita querida por ser mi segunda madre, por quererme tanto, por darme toda la confianza y apoyo en mi vida. Por ser la persona más fuerte y enérgica que conozco, por ser la persona más importante para mi mamita y para toda la familia. Abu eres una persona maravillosa a la que quisiera imitar pero nunca podré, porque tú eres una estrella, no mejor tu eres mi estrella, la que me guía y guía a la familia. Gracias por ser la persona que siempre ha estado a mi lado y por demostrarme que todo es posible.

Te quiero Abu.

Resumen

La Universidad de las Ciencias Informáticas cuenta con el componente Chart Server 1.0.3 el cual permite generar gráficos estadísticos mediante una petición realizada al servidor. Sin embargo resulta muy difícil para un usuario con pocos conocimientos en el área de la informática, elaborar la url necesaria para crear un gráfico estadístico. Para darle solución a la situación planteada anteriormente se desarrolló el Chart Server 2.0. La nueva versión del componente cuenta con varias características para facilitarle al usuario la construcción de gráficos estadísticos. Cuenta con una interfaz web para interactuar con el usuario. El conjunto de datos a utilizar puede ser introducido manualmente, importado desde hojas de cálculo o ser extraído de una Base de Datos. Permite exportar el conjunto de datos utilizado, la imagen que contiene el gráfico estadístico y la url que lo genera. Además el componente permite construir gráficos de pastel, barras, líneas, área bajo la línea, curvas, área bajo la curva, puntos, pasos, burbujas y mixtos.

Palabras Claves

Chart Server, gráfico estadístico, hojas de cálculos, servidor.

Tabla de Contenido

Introducción

La información es un elemento vital para cualquier empresa o institución, e influye de manera directa en la forma en que estas operan y en la correcta toma de decisiones. Constituye un conjunto de datos acerca de un suceso, hecho o fenómeno, que organizados en un contexto determinado tienen su significado. El propósito de la información puede ser el de reducir la incertidumbre o incrementar el conocimiento acerca de algo. Otorga significado o sentido a la realidad, ya que mediante códigos y conjuntos de datos, da origen a los modelos de pensamiento humano. (1)

Existen varias formas de representar cierta información deseada, pero independientemente de la necesidad o el enfoque que se le quiera dar, la presentación final debe comunicar los resultados esperados; las tablas y los informes son efectivos, pero sin duda ninguno es tan eficiente y preciso como los gráficos estadísticos. Los gráficos estadísticos se han convertido en un método efectivo para describir los valores de datos económicos, políticos, sociales, psicológicos, biológicos o físicos; sirviendo como herramienta para relacionar y comparar dichos datos.

Un gráfico estadístico es un instrumento que presenta datos numéricos por medio de figuras geométricas, líneas, pictogramas, o mapas estadísticos. Es la presentación artística de los resultados de un informe. Un gráfico básicamente compara cifras, datos y proporciones; por ello para que exista, al menos debe haber dos elementos de comparación. El gráfico es una fotografía de los datos existentes en el momento, pero estos cambian con el tiempo; y para actualizar las cifras debe hacerse un nuevo gráfico.

Existen varios tipos de gráficos entre los que se encuentran:

- Gráfico de Barras: es el más simple y utilizado, ya que las comparaciones se basan en el tamaño de las barras, se ordenan de mayor a menor para facilitar su lectura y el espacio entre las barras le da mayor claridad.
- Gráfico Circular y de Barras de 100%: presentan proporciones en porcentajes, permiten presentar la importancia relativa de un dato y no posee ejes.
- Gráficos Lineales Aritméticos y Logarítmicos: se usan especialmente para presentar series de tiempo y crecimiento, siempre tienen dos escalas, una horizontal y otra vertical, que al formar parejas representan puntos específicos de un gráfico y permiten presentar mayor cantidad de datos dentro del mismo gráfico.
- Pictogramas: es un gráfico construido con figuras o dibujos, no usa escalas, todos los dibujos son

iguales y se presentan horizontalmente. Las magnitudes se representan con la cantidad de dibujos empleados y cada símbolo representa un valor específico.

- Mapas Estadísticos: muestran la información sobre un mapa, muestran datos de áreas geográficas de un país. (2)

A raíz del desarrollo de los gráficos estadísticos ha surgido un nuevo paradigma que son los servidores de gráficos, su función fundamental es la generación dinámica de gráficos estadísticos a partir de datos enviados al servidor. Los servidores utilizan bibliotecas especializadas en generar gráficos. Existen gran variedad de bibliotecas para generar gráficos escritas en diferentes lenguajes de programación. Entre las bibliotecas más comunes se encuentran: JFreeChart, Open Flash Chart, FusionCharts, PHP/SWF Charts, Max'sCharts y amCharts.

La Universidad de las Ciencias Informáticas (UCI) tuvo la necesidad de crear su propio servidor de gráficos, ya que existen varias aplicaciones que necesitaban una forma rápida y sencilla de tener representada su información. La tarea fue realizada por el departamento de Integración de Soluciones del Centro de Tecnologías de Gestión de Datos (DATEC), actualmente el componente está en la versión 1.0.3. Su funcionamiento está basado en el lenguaje PHP y contiene un conjunto de servicios que pueden ser consumidos desde cualquier otra aplicación web, está ubicado en la siguiente dirección: http://graficos.prod.uci.cu. Soporta gran cantidad de gráficos tales como: barras, líneas, área bajo las líneas, curvas, área bajo las curvas, pastel, puntos y mixtos.

Actualmente en el centro DATEC la mayoría de las aplicaciones que se implementan usan la biblioteca Ext JS y el marco de trabajo Symfony 2, los cuales no se utilizaron en el desarrollo del Chart Server 1.0.3. Actualmente el Chart Server 1.0.3 cuenta con la biblioteca Pchart 1.27d; sin embargo dicha versión ha evolucionado hasta la versión 2.1.3 con varias características nuevas como: soporte para nuevos tipos de gráficos, nuevas características de personalización y optimización de código. El Chart Server 1.0.3 está diseñado para desarrolladores avanzados por lo que resulta engorroso para usuarios con pocos conocimientos en el área de la informática elaborar la url necesaria para generar un gráfico.

Debido a la necesidad de dar solución a la situación planteada, se identifica como problema de la investigación: ¿Cómo contribuir a la construcción de gráficos estadísticos del Chart Server? Este problema determinó que el objeto de estudio de esta investigación sea: Procesos de construcción de gráficos y se precisa como campo de acción: Procesos de construcción de gráficos estadísticos.

Para dar solución al problema planteado se define como objetivo general: Desarrollar el componente Chart Server versión 2.0 para la construcción de gráficos estadísticos.
Este objetivo general se desglosa en los siguientes objetivos específicos:

- Identificar las nuevas funcionalidades del componente.
- Realizar el diseño del componente a partir de los requisitos identificados.
- Implementar las funcionalidades definidas.
- Realizar pruebas que demuestren el correcto funcionamiento del Chart Server 2.0.

Para dar cumplimiento a los objetivos se realizaron esencialmente las siguientes tareas:

- Investigación de los servidores de gráficos estadísticos que existen actualmente.
- Revisión bibliográfica sobre los elementos a utilizar en el desarrollo del componente.
- Selección de las herramientas, lenguaje y metodología a utilizar para desarrollar el componente.
- Identificación de los requisitos funcionales y no funcionales para el correcto funcionamiento del componente.
- Confección del modelo de casos de uso del componente.
- Elaboración del modelo de diseño para facilitar la implementación del componente.
- Elaboración del modelo de implementación para dar cumplimiento a los requisitos identificados.
- Diseño de los Casos de Prueba basados en los casos de uso previamente definidos.
- Ejecución de pruebas funcionales para comprobar el correcto funcionamiento del componente.

El documento está estructurado de la siguiente manera: resumen, introducción, tres capítulos, conclusiones, recomendaciones, referencias, bibliografía y anexos.

Capítulo 1: Fundamentación Teórica

En este capítulo se realiza un estudio sobre algunos servidores de gráficos teniendo en cuenta sus principales características. Se manifiesta el estado del arte de las tecnologías a usar en el desarrollo del sistema: herramientas, metodología, lenguajes de programación y bibliotecas que se utilizarán en el desarrollo del componente.

Capítulo 2: Características del Sistema

En este capítulo se precisan todas las funcionalidades que debe brindar el componente. Se diseña el modelo de dominio; se elabora una lista donde se reflejan los requisitos funcionales y no funcionales del sistema. Además se identifican los actores, casos de uso del sistema y la relación que existe entre ellos. Se describen el modelo de paquetes del sistema y el modelo de clases del diseño.

Capítulo 3: Implementación y Pruebas

En este capítulo se refleja todo lo asociado a la etapa de implementación y pruebas, donde se representa el diagrama de componentes del sistema. Se presentan los estándares de codificación a utilizar y se realizan las pruebas funcionales de caja negra para comprobar el correcto funcionamiento del Chart Server 2.0.

Capítulo 1: Fundamentación teórica

En este capítulo se realiza una caracterización sobre algunos servidores de gráficos que existen actualmente, para así simplificar la búsqueda de las nuevas características del componente que se desea construir. Se describen detalladamente las tecnologías, herramientas, metodología de desarrollo, lenguaje de modelado y las bibliotecas a utilizar, donde se selecciona lo apropiado de acuerdo a las características existentes durante el ciclo de vida del software. Además se proponen los patrones de software para el diseño de la solución.

Servidores de gráficos estadísticos

Un servidor de gráficos estadísticos es una aplicación que brinda un servicio de petición-respuesta. El usuario envía una petición que contiene el conjunto de variables necesarias para que el servidor pueda generar el gráfico. El servidor captura la petición y le devuelve al usuario el resultado en una imagen que contiene el gráfico.

A continuación se muestra el resultado de la investigación realizada a algunos de los servidores de gráficos estadísticos del mundo. El propósito de la investigación es buscar que características de los servidores actuales se pudieran incluir en la nueva versión del Chart Server.

1.1 Servidores de gráficos estadísticos

ChartGo

- Los datos se ingresan manualmente.
- Las gráficas son estáticas.
- En caso de tener los datos en una hoja de cálculo, se pueden copiar y pegar.
- Interfaz gráfica para crear los gráficos.
- Las gráficas creadas se pueden grabar, compartir mediante redes sociales y correo electrónico o embeber en páginas web o blogs. (3)

Rich Chart Live

- Los datos se ingresan manualmente.
- Se importan datos desde Excel u Open Office.
- Interfaz gráfica para crear los gráficos.
- Las gráficas son animadas con tecnología Flash.

- Las gráficas creadas se pueden embeber en páginas web o blogs. (4)

Lucid Chart

- Los datos se ingresan manualmente.
- Además de crear gráficas permite crear diagramas de redes, procesos, mapas mentales, diagramas de flujo y mapas de sitio.
- Las gráficas son estáticas.
- Interfaz gráfica enriquecida con las últimas tecnologías del lenguaje HTML5. (5)

Hohli

- Los datos se ingresan manualmente o desde una url.
- Interfaz gráfica para crear los gráficos.
- Las gráficas son estáticas.
- Utiliza la biblioteca de Google API Charts. (6)

1.2 Servidor de gráficos de la Universidad de las Ciencias Informáticas

Chart Server 1.0.3

- Los datos se ingresan por la url.
- Utiliza la biblioteca Pchart 1.27d.
- Su implementación está basada en el lenguaje PHP.
- Cuenta con una ayuda web muy completa. (7)

Luego del estudio realizado se propone incluir las siguientes características a la nueva versión del componente:

- Interfaz web para interactuar con el usuario.
- Generar automáticamente la url que permite crear el gráfico.
- Insertar el conjunto de datos manualmente.
- Importar el conjunto de datos desde una Base de Datos.
- Importar el conjunto de datos desde una hoja de cálculo.

1.3 Tecnologías

Para la realización de un software se deben tener en cuenta las tendencias actuales en torno a las herramientas a utilizar para el desarrollo de la misma, de forma que el trabajo se haga más sencillo y a la vez con mayor calidad. Se deben utilizar tecnologías avanzadas y en continuo progreso, para que de esta forma la aplicación cumpla con las expectativas de los usuarios finales y además se pueda actualizar la misma con gran facilidad. (8)

JavaScript 1.5

Es un lenguaje de programación interpretado desarrollado por Netscape[1] que se utiliza en la mayoría de páginas web y aplicaciones web en todo el mundo. Los programas escritos en este lenguaje se pueden ejecutar directamente en cualquier navegador sin necesidad de procesos intermedios. Se utiliza principalmente para crear páginas web dinámicas permitiendo mejoras en la interfaz de usuario. Permite realizar validaciones y cambios en las páginas web en tiempo real. (9)

Se decide utilizar JavaScript como lenguaje en el lado del cliente por las características antes mencionadas, es un lenguaje muy sencillo, tiene gran documentación en la web, y es totalmente gratuito. Además es el lenguaje en el cual está desarrollada la biblioteca Ext JS.

PHP 5.3.10

PHP (Hypertext Preprocessor por sus siglas en inglés) es un lenguaje de código abierto especialmente adecuado para desarrollo web. PHP puede hacer cualquier cosa que se pueda hacer con un script CGI[2], como procesar la información de formularios, generar páginas con contenidos dinámicos, o enviar y recibir cookies. PHP no se encuentra limitado solo a resultados en HTML. Entre las habilidades de PHP se incluyen: creación de imágenes, archivos PDF y películas Flash, procesamiento de texto y soporte para acceso a bases de datos. (10) También se puede usar para crear y manipular ficheros de imágenes en diferentes formatos. Aún más práctico es que PHP puede transferir flujos de imagen directamente al navegador. (11)

Se decide utilizar el lenguaje de programación PHP en el lado del servidor debido a la amplia

[1] Netscape fue un navegador web y el primer producto comercial de la compañía Netscape Communications.

[2] Es un mecanismo de comunicación entre el servidor web y una aplicación externa que permite a un cliente (navegador web) solicitar datos de un programa ejecutado en un servidor web.

documentación que existe, su facilidad de uso y el soporte para crear imágenes. Además es el lenguaje base del marco de trabajo Symfony 2.

Ajax

AJAX, acrónimo de Asynchronous JavaScript And XML (eXtensible Markup Language por sus siglas en inglés, es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications por sus siglas en inglés). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.

JSON

JSON (JavaScript Object Notation por sus siglas en inglés) es un formato sencillo para el intercambio de información. El formato JSON permite representar estructuras de datos (arreglos) y objetos (arreglos asociativos) en forma de texto. La notación de objetos mediante JSON es una de las características de JavaScript y es un mecanismo definido en los fundamentos básicos del lenguaje. En los últimos años, JSON se ha convertido en una alternativa al formato XML, ya que es más fácil de leer y escribir, además de ser mucho más conciso. (12)

Se decide utilizar AJAX y el formato de intercambio de datos JSON, debido a que permiten que el tiempo de espera de una petición realizada al servidor sea menor. No es necesario refrescar la página web al realizar una petición al servidor y se reduce el consumo de ancho de banda. Además AJAX y JSON se pueden utilizar en cualquier plataforma o navegador.

1.4 Marco de trabajo y entorno de desarrollo integrado

Marco de trabajo

Un marco de trabajo es una colección de patrones de diseño, interfaces, clases y código de fuente, que conforman un sistema que proporciona apoyo al programador. En lugar de crear algo desde cero y propenso a introducir errores, se recomienda el uso y explotación de los marcos de trabajo ya existentes.

Symfony 2.1.3

Symfony 2 es un marco de trabajo diseñado para optimizar el desarrollo de las aplicaciones web. Separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además automatiza las tareas que son comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación.

Algunas características del marco de trabajo Symfony 2:

- Fácil de instalar y configurar en la mayoría de las plataformas.
- Independiente del sistema gestor de bases de datos.
- Sigue la mayoría de mejores prácticas y patrones de diseño para la web.
- Preparado para aplicaciones empresariales y adaptables a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
- Código fácil de leer que incluye comentarios y que permite un mantenimiento muy sencillo.
- Fácil de extender, lo que permite su integración con bibliotecas desarrolladas. (13)

En Symfony 2 todo es un bundle. Un bundle es un directorio que tiene una estructura definida, puede alojar cualquier elemento, desde clases para los controladores hasta recursos web. Facilita escoger o seleccionar qué características habilitar en la aplicación y optimizarlas de la manera que se desee. Se decide utilizar el marco de trabajo Symfony 2 por las características mencionadas anteriormente. Además si la aplicación se quisiera integrar con otras aplicaciones del centro solamente haría falta añadir los bundles del componente y sus dependencias.

Entorno de desarrollo integrado

Un entorno de desarrollo integrado, llamado también IDE (integrated development environment por sus siglas en inglés), es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien puede utilizarse para varios. En el mundo existen diferentes entornos de desarrollo integrados por ejemplo: Zend Studio, Aptana, Lazarus y NetBeans. Estos comparten peculiaridades como son:

- Soporte para múltiples lenguajes.
- Corren en varias plataformas como Windows y Linux.

- Completamiento de código.

NetBeans 7.2

Es un entorno de desarrollo integrado que permite escribir, compilar, depurar y ejecutar programas. Está escrito en el lenguaje Java pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el IDE NetBeans. Es un producto libre y gratuito sin restricciones de uso. Entre sus funcionalidades se encuentran:

- Editor de código sensible al contenido.
- Soporte para autocompletar el código, coloreado de etiquetas y auto tabulación.
- Soporte para varios lenguajes de programación, entre los que se encuentran: PHP, JavaScript, Java y HTML.
- Incluye CVS (control de versiones) y Ant (compilación avanzada).
- Posibilidad de utilizar otras versiones de compiladores y depuradores.
- Creación visual de componentes gráficos.
- Herramientas con asistentes para facilitar la escritura de código. (14)

Se decide utilizar NetBeans 7.2 por ser un entorno de desarrollo muy completo y profesional. Contiene muchas funcionalidades, para distintos tipos de aplicaciones y para facilitar al máximo la programación, la prueba y la depuración de las aplicaciones que se desarrollan.

1.5 Patrones de arquitectura

Expresa un esquema fundamental de organización estructural para un sistema de software. Donde provee una serie de subsistemas predefinidos, especificando sus responsabilidades, e incluye reglas y guías para organizar las relaciones entre ellos. Básicamente, un patrón de arquitectura es una plantilla para una arquitectura de aplicaciones. Estos especifican las propiedades generales a la estructura del sistema, y repercuten la arquitectura de sus subsistemas. La selección de un patrón de arquitectura es por lo tanto una decisión fundamental al desarrollar un sistema de software. (15)

Cliente – Servidor: es un modelo de aplicación distribuida en el que las tareas se reparten entre los servidores y los clientes. Un cliente realiza peticiones al software y el servidor da la respuesta.

Modelo – Vista – Controlador: el patrón proporciona grandes ventajas como la organización de código, reutilización y flexibilidad. La programación es organizada pues separa los datos de la aplicación, la

interfaz de usuario y la lógica de los datos en tres componentes diferentes, en el cual cada capa se especializa en una función específica.

- Modelo: representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos.
- Vista: muestra el modelo en un formato adecuado para interactuar. Maneja la presentación visual de los datos representados por el modelo.
- Controlador: responde a eventos e invoca cambios en el modelo y probablemente en la vista.

Ventajas del Modelo – Vista – Controlador:

- Soporte para múltiples vistas pues no existe dependencia directa entre la vista y el modelo.
- Proporciona un mecanismo de configuración a componentes complejos mucho más tratable que el puramente basado en eventos.
- Mayor soporte a los cambios, pues los requisitos de interfaz tienden a cambiar más rápidamente que las reglas de negocio. Puesto que el modelo no depende de las vistas, la adición de nuevos tipos de vistas al sistema generalmente no afecta al modelo. (16)

1.6 Patrones de diseño

En la terminología de objetos, el patrón es una descripción de un problema y su solución que recibe un nombre y que puede emplearse en otros contextos; en teoría, indica la manera de utilizarlo en circunstancias diversas. Muchos patrones ofrecen orientación sobre cómo asignar las responsabilidades a los objetos ante determinada categoría de problemas. Expresado lo anterior con palabras más simples, el patrón es una pareja de problema/solución con un nombre y que es aplicable a otros contextos, con una sugerencia sobre la manera de usarlo en situaciones nuevas. (17)

Patrones de diseño GRASP

GRASP es el acrónimo para General Responsibility Assignment Software Patterns (Patrones Generales de Software para Asignar Responsabilidades). Los patrones GRASP representan los principios básicos de la asignación de responsabilidades a objetos, expresados en forma de patrones.

Patrones básicos de asignación de responsabilidades:

- Experto: se encarga de asignar una responsabilidad al experto en información, o sea, aquella clase que cuenta con la información necesaria para cumplir la responsabilidad.
- Creador: este patrón es el responsable de asignarle a la clase B la responsabilidad de crear una

instancia de clase A. B es un creador de los objetos A.

- Alta Cohesión: asigna las responsabilidades de modo que se mantenga una alta cohesión.
- Bajo Acoplamiento: asigna las responsabilidades de modo que se mantenga bajo acoplamiento.
- Controlador: asigna la responsabilidad de administrar un mensaje de eventos del sistema a una clase. (17)

Patrones de diseño GoF

Los patrones GoF se clasifican en 3 categorías basadas en su propósito: creacionales, estructurales y de comportamiento:

- Creacionales: los patrones creacionales abstraen el proceso de creación de instancias y ocultan los detalles de cómo los objetos son creados o inicializados.
- Estructurales: los patrones estructurales se ocupan de cómo las clases y objetos se combinan para formar grandes estructuras y proporcionar nuevas funcionalidades.
- Comportamiento: los patrones de comportamiento están relacionados con los algoritmos y la asignación de responsabilidades entre los objetos. Son utilizados para organizar, manejar y combinar comportamientos.

Patrones Creacionales

- Factoría abstracta: proporciona una interfaz para crear familias de objetos que dependen entre sí, sin especificar sus clases concretas.
- Prototipo: especifica los tipos de objetos a crear por medio de una instancia prototípica y crea nuevos objetos copiando este prototipo.
- Única instancia: garantiza que una clase sólo tenga una instancia y proporciona un punto de acceso global a ella.

Patrones estructurales

- Adaptador: convierte la interfaz de una clase en otra distinta que es la que esperan los clientes. Permite que cooperen clases que de otra manera no podrían por tener interfaces incompatibles.
- Puente: desvincula una abstracción de su implementación, de manera que ambas clases puedan variar de forma independiente.
- Decorador: añade dinámicamente nuevas responsabilidades a un objeto, proporcionando una alternativa flexible a la herencia para extender la funcionalidad.

Patrones de comportamiento

- Cadena de Responsabilidades: evita acoplar el emisor de una petición a su receptor, al dar a más de un objeto la posibilidad de responder a la petición. Crea una cadena con los objetos receptores y pasa la petición a través de la cadena hasta que esta sea tratada por algún objeto.
- Intérprete: define una representación de su gramática junto con un intérprete que usa dicha representación para interpretar las sentencias del lenguaje.
- Visitante: representa una operación sobre los elementos de una estructura de objetos. Permite definir una nueva operación sin cambiar las clases de los elementos sobre los que opera. (16)

1.7 Herramienta CASE

Herramienta CASE (Computer Aided Software Engineering por sus siglas en inglés) es un conjunto de métodos, utilidades y técnicas que facilitan la automatización del ciclo de vida del desarrollo de software, completamente o en alguna de sus fases. Existen varias herramientas CASE como son: Rational Rose, Visual Paradigm for UML, EasyCase, Oracle Designer y Power Designer. Estas herramientas pueden proveer muchos beneficios en todas las etapas del proceso de desarrollo de software, algunos de ellos son:

- Verificar el uso de todos los elementos en el sistema diseñado.
- Automatizar el dibujo de diagramas.
- Ayudar en la creación de relaciones en la Base de Datos.
- Generar estructuras de código.

La principal ventaja de la utilización de una herramienta CASE, es la mejora de la calidad de los desarrollos realizados y el aumento de la productividad. Para conseguir estos dos objetivos es conveniente contar con una organización y una metodología de trabajo eficiente, además de la propia herramienta. (18)

Visual Paradigm for UML 6.4

Visual Paradigm for UML es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor costo. Permite dibujar todos los tipos de diagramas de clases, generar diagramas desde códigos, generar códigos desde diagramas y generar documentación. (19)

Presenta un diseño centrado en casos de uso y proporciona a los desarrolladores de software una interfaz simple y amigable, con muchas opciones tales como: diversidad de idiomas, generación de código para varios lenguajes de programación, posee facilidad para la instalación y actualización, así como compatibilidad entre sus ediciones. También facilita la interoperabilidad con otras herramientas CASE y con la mayoría de los principales entornos de desarrollo integrados. (20)

Para el desarrollo del ciclo de vida del componente se propone utilizar la herramienta Visual Paradigm for UML 6.4 teniendo en cuenta que esta herramienta permite aumentar la calidad del software, a través de la mejora de la productividad en el desarrollo y mantenimiento del software. También permite la reutilización del software, portabilidad y estandarización de la documentación, además del uso de las distintas metodologías propias de la Ingeniería del Software.

1.8 Lenguaje de modelado

Lenguaje Unificado de Modelado (UML)

El Lenguaje Unificado de Modelado establece un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos, y describe la semántica esencial de lo que estos diagramas y símbolos significan. Incluye aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables. (21) Los objetivos de UML son muchos, pero se pueden sintetizar sus funciones en: visualizar, especificar, construir y documentar. (22)

Se decidió utilizar UML en el desarrollo del componente ya que se puede aplicar en gran variedad de formas para dar soporte a una metodología de desarrollo de software. Además tiene una notación gráfica muy expresiva que permite representar en mayor o menor medida todas las fases de un proyecto informático: desde el análisis con los casos de uso, el diseño con los diagramas de clases, objetos, hasta la implementación y configuración con los diagramas de despliegue.

1.9 Metodología de desarrollo

Una metodología es un conjunto de procedimientos, técnicas, herramientas que ayuda a los desarrolladores a realizar un nuevo software. Cada metodología tiene su propio enfoque para el desarrollo de software e indican paso a paso todas las actividades a realizar en un producto informático. Toda metodología de desarrollo de software debe incluir la forma en que se va a realizar la captura de requisitos, el diseño, la implementación y prueba. Actualmente existen varias metodologías como son:

Rational Unified Process (RUP), eXtreme Programming (XP), Scrum, Crystal Clear y OpenUP.

OpenUP

Open Unified Process (OpenUP) es un proceso de desarrollo unificado que está basado en Rational Unified Process (RUP). Mantiene las mismas características de RUP, pues está dirigido por casos de uso, centrado en la arquitectura y además es iterativo e incremental. El ciclo de vida de un proyecto según la metodología OpenUP se divide en 4 fases fundamentales:

- Inicio: se determina la visión del proyecto.
- Elaboración: el objetivo es determinar la arquitectura óptima.
- Construcción: el objetivo es obtener la capacidad operacional inicial.
- Transición: se obtiene la integración del proyecto.

OpenUP propone 6 flujos de trabajo:

- Requisitos: en este flujo de trabajo se realizan entrevistas con el cliente para comprender el problema a resolver y se definen los requisitos.
- Análisis y Diseño: se realiza el diseño de los requisitos que serán después implementados.
- Implementación: en esta disciplina se realiza la implementación del sistema basándose en el diseño realizado.
- Prueba: busca los defectos a lo largo del ciclo de vida.
- Gestión del Proyecto: involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.
- Gestión de Configuración y Cambios: describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto. (23)

Después de haber realizado un estudio sobre la metodología OpenUP, se escoge como metodología a utilizar ya que es un proceso de desarrollo de software de código abierto, extensible, dirigido a gestión y desarrollo de proyectos de software basados en desarrollo iterativo, ágil e incremental. Aplicable a un conjunto amplio de plataformas y aplicaciones de desarrollo.

1.10 Bibliotecas

En ciencias de la computación, una biblioteca (del inglés library) es un conjunto de subprogramas utilizados para desarrollar software. Las bibliotecas contienen código y datos, que proporcionan servicios a programas independientes, es decir, pasan a formar parte de estos. Esto permite que el código y los datos

se compartan y puedan modificarse de forma modular.

Ext JS 4.1.1

Es una biblioteca de JavaScript que permite construir aplicaciones complejas con interfaces enriquecidas para internet. Además de flexibilizar el manejo de componentes de la página como el DOM (Document Object Model por sus siglas en inglés), peticiones AJAX y DHTML (Dynamic HTML por sus siglas en inglés), tiene la gran funcionalidad de crear interfaces de usuario bastante funcionales. (24)

Ext JS tiene varias ventajas que facilitan su uso:

- Permite crear aplicaciones complejas utilizando componentes predefinidos.
- Evita el problema de tener que validar el código para que funcione bien en cada uno de los navegadores.
- Relación entre Cliente-Servidor balanceado: se distribuye la carga de procesamiento entre estos, permitiendo que el servidor pueda atender más clientes al mismo tiempo.

Pchart 2.1.3

Pchart es una biblioteca de PHP que permite crear gráficos estadísticos o imágenes directamente en el servidor web. Provee gran cantidad de gráficos entre los que se encuentran: barras curvas, áreas, pastel y burbuja. El resultado se puede mostrar en el navegador del cliente ya que usa extensiones compatibles con los navegadores. Pchart provee sintaxis de código orientada a objetos y está completamente preparada para los nuevos estándares de la web 2.0.

PHPExcel 1.7.6

PHPExcel es una biblioteca escrita en puro PHP y provee una serie de clases que permiten crear y leer de diferentes extensiones para Hojas de Cálculo, como Excel (BIFF).xls, Excel 2007 (OfficeOpenXML).xlsx, CSV, Libre/OpenOfficeCalc.ods, Gnumeric, PDF, HTML. Se rige por los estándares de OpenXML y PHP.

El estudio permitió proponer las bibliotecas a utilizar para facilitar el desarrollo del componente, de una manera más sencilla y eficiente. Ext JS 4.1.1 por la facilidad que brinda para crear aplicaciones enriquecidas para la web. La biblioteca Pchart 2.1.3 ya que permite generar gran cantidad de gráficos estadísticos. Y la biblioteca PHPExcel 1.7.6 ya que facilita la obtención de los datos que se encuentran en una hoja de cálculo y además permite crear nuevas hojas de cálculo.

Conclusiones del capítulo

En este capítulo se realizó un estudio acerca de los servidores de gráficos que existen en el mundo, lo cual contribuyó a definir los diseños preliminares de las interfaces del componente. Como resultado de la investigación se llega a la conclusión de utilizar las siguientes tecnologías: Symfony 2.1.3 como marco de trabajo del lado del servidor. Como lenguajes de programación se seleccionaron PHP 5.3.10 y JavaScript 1.5, utilizando las bibliotecas Ext JS 4.1.1, Pchart 2.1.3 y PHPExcel 1.7.6; el entorno de desarrollo integrado NetBeans 7.2, el cual es compatible con el marco de trabajo Symfony 2 y los lenguajes de programación seleccionados. OpenUP como metodología de desarrollo ya que es aplicable a un conjunto amplio de plataformas y aplicaciones de desarrollo. Para modelar los diagramas necesarios en la elaboración del componente, se utilizará como herramienta CASE Visual Paradigm for UML 6.4 que es capaz de soportar el ciclo de vida completo del desarrollo de software, y tiene una gran disponibilidad en múltiples plataformas (Windows, Linux) utilizando el lenguaje de modelado UML. Toda la selección también está regida por la arquitectura definida por el Departamento de Integración de Soluciones.

Capítulo 2: Características del sistema

En este capítulo se describen los conceptos que influyen en el diseño y elaboración del sistema. Se identifica la estructura física, donde se diseña el modelo de dominio. Se definen los requisitos funcionales y no funcionales, lo que contribuiría al buen funcionamiento del componente. Se elabora el diagrama de casos de uso del sistema. Se representan el modelo de paquetes del sistema y el modelo de clases del diseño, así como el modelo de despliegue donde se refleja el hardware utilizado en las implementaciones del sistema y las relaciones entre sus componentes.

2.1 Modelo de dominio

El modelo de dominio muestra las clases conceptuales significativas en un dominio del problema. Es una representación visual de las clases conceptuales u objetos del mundo real en un dominio de interés. También se les denomina modelos conceptuales, modelo de objetos del dominio y modelos de objetos de análisis.

Utilizando el lenguaje de modelado UML, un modelo del dominio se representa con un conjunto de diagramas de clases en los que no se define ninguna operación. Se pueden mostrar de las siguientes formas:

- Objetos del dominio o clases conceptuales.
- Asociaciones entre las clases conceptuales. (25)

Descripción del modelo de dominio

El Chart Server 2.0 es un servidor encargado de construir un gráfico estadístico mediante los parámetros que le precise el usuario. Un gráfico estadístico contiene leyenda, escala y un conjunto de datos. Además puede ser personalizado o sea, cambiarle la combinación de colores y el tipo de fuente utilizada en los textos. El servidor muestra el resultado final en una imagen de formato PNG (Portable Network Graphics por sus siglas en inglés).

Usuario: personas que utilizan el Chart Server 2.0 para construir gráficos estadísticos.

Servidor de gráficos estadísticos: componente de software que se utiliza para la construcción de los gráficos estadísticos.

Gráfico estadístico: es una representación gráfica que presenta datos numéricos por medio de figuras geométricas; básicamente es la comparación de cifras, datos y proporciones.

Leyenda: forma parte del gráfico estadístico, representa las descripciones para cada una de las categorías de un gráfico.

Escala: forma parte del gráfico estadístico, representa la relación entre el valor de la representación y el valor de la realidad.

Conjunto de datos: son los valores numéricos que utiliza el servidor para generar el gráfico estadístico que pueden ser importados desde una Base de Datos, desde un Excel e introducidos manualmente.

Imagen: es el resultado final que muestra el servidor, es la representación visual del gráfico en una imagen de formato PNG.

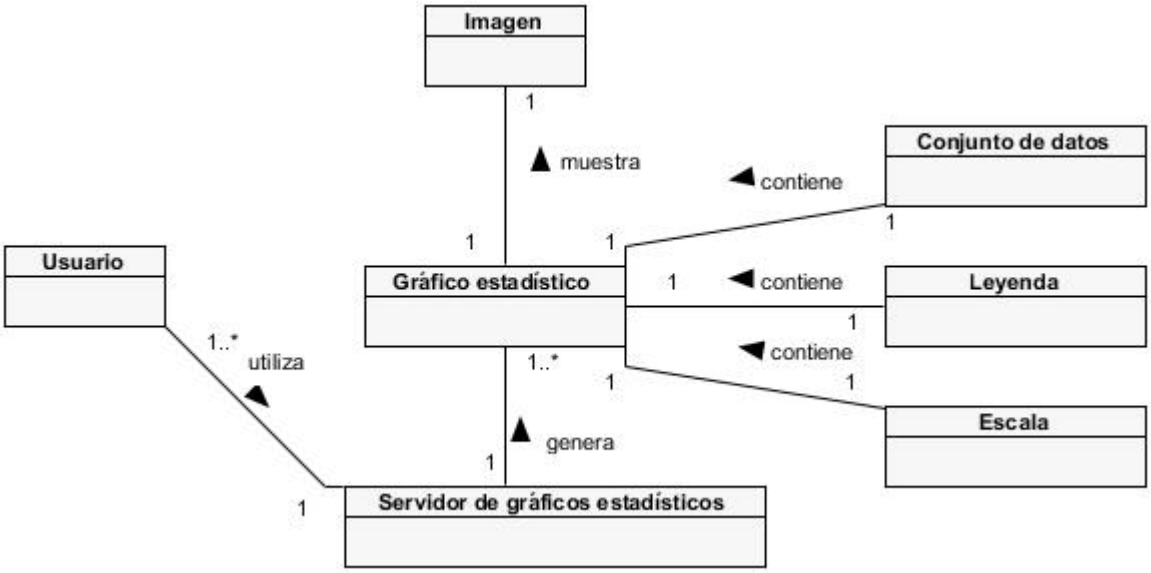


Figura 1 Modelo de dominio

2.2 Requisitos de software

Los requisitos son una descripción de las necesidades de un producto. La meta primaria de la fase de requisitos es identificar y documentar lo que en realidad se necesita, en una forma que claramente se lo comunique al cliente y a los miembros del equipo de desarrollo. Condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo.

Requisitos funcionales

Los requisitos funcionales (RF) son capacidades o condiciones que el sistema debe cumplir. Definen las funciones que el sistema será capaz de realizar (26) y especifican el comportamiento de entrada y salida del componente. Además se mantienen invariables sin importar con qué propiedades o cualidades se relacionen.

El componente Chart Server 2.0 debe efectuar los requisitos funcionales que a continuación se describen:

RF1: Construir gráfico de pastel

- Descripción: el componente debe permitir la construcción del gráfico de pastel.
- Entrada:
    - Tipo de gráfico: en este campo se selecciona el tipo de gráfico pastel en dos dimensiones o en tres dimensiones.
    - Título del gráfico: en este campo se introduce el título del gráfico.
    - Tamaño del gráfico: en este campo se introduce las dimensiones del gráfico.
- Salida: una imagen de formato PNG y la url que genera el gráfico.

RF2: Construir gráfico de barras

- Descripción: el componente debe permitir la construcción del gráfico de barras.
- Entrada:
    - Tipo de gráfico: en este campo se selecciona el tipo de gráfico de barras.
    - Título del gráfico: en este campo se introduce el título del gráfico.
    - Tamaño del gráfico: en este campo se introduce las dimensiones del gráfico.
- Salida: una imagen de formato PNG y la url que genera el gráfico.

RF3: Construir gráfico de líneas

- Descripción: el componente debe permitir la construcción del gráfico de líneas.
- Entrada:
    - Tipo de gráfico: en este campo se selecciona el tipo de gráfico líneas.
    - Título del gráfico: en este campo se introduce el título del gráfico.
    - Tamaño del gráfico: en este campo se introduce las dimensiones del gráfico.
- Salida: una imagen de formato PNG y la url que genera el gráfico.

RF4: Construir gráfico de área bajo la línea

- Descripción: el componente debe permitir la construcción del gráfico de área bajo la línea.
- Entrada:
    - Tipo de gráfico: en este campo se selecciona el tipo de gráfico de área bajo la línea.
    - Título del gráfico: en este campo se introduce el título del gráfico.
    - Tamaño del gráfico: en este campo se introduce las dimensiones del gráfico.
- Salida: una imagen de formato PNG y la url que genera el gráfico.

RF5: Construir gráfico de curvas

- Descripción: el componente debe permitir la construcción del gráfico de curvas.
- Entrada:
    - Tipo de gráfico: en este campo se selecciona el tipo de gráfico de curvas.
    - Título del gráfico: en este campo se introduce el título del gráfico.
    - Tamaño del gráfico: en este campo se introduce las dimensiones del gráfico.
- Salida: una imagen de formato PNG y la url que genera el gráfico.

RF6: Construir gráfico de área bajo la curva

- Descripción: el componente debe permitir la construcción del gráfico de área bajo la curva.
- Entrada:
    - Tipo de gráfico: en este campo se selecciona el tipo de gráfico de área bajo la curva.
    - Título del gráfico: en este campo se introduce el título del gráfico.
    - Tamaño del gráfico: en este campo se introduce las dimensiones del gráfico.
- Salida: una imagen de formato PNG y la url que genera el gráfico.

RF7: Construir gráfico de puntos

- Descripción: el componente debe permitir la construcción del gráfico de puntos.
- Entrada:
    - Tipo de gráfico: en este campo se selecciona el tipo de gráfico de puntos.
    - Título del gráfico: en este campo se introduce el título del gráfico.
    - Tamaño del gráfico: en este campo se introduce las dimensiones del gráfico.

- Salida: una imagen de formato PNG y la url que genera el gráfico.

RF8: Construir gráfico de pasos

- Descripción: el componente debe permitir la construcción del gráfico de pasos.
- Entrada:
    - Tipo de gráfico: en este campo se selecciona el gráfico de pasos.
    - Título del gráfico: en este campo se introduce el título del gráfico.
    - Tamaño del gráfico: en este campo se introduce las dimensiones del gráfico.
- Salida: una imagen de formato PNG y la url que genera el gráfico.

RF9: Construir gráfico de burbujas

- Descripción: el componente debe permitir la construcción del gráfico de burbujas.
- Entrada:
    - Tipo de gráfico: en este campo se selecciona el tipo de gráfico burbujas.
    - Título del gráfico: en este campo se introduce el título del gráfico.
    - Tamaño del gráfico: en este campo se introduce las dimensiones del gráfico.
- Salida: una imagen de formato PNG y la url que genera el gráfico.

RF10: Construir gráfico mixto

- Descripción: el componente debe permitir la construcción del gráfico mixto. Este gráfico soporta la combinación de un par de gráficos (uno principal y uno secundario) que puede ser barras, líneas y curvas; pero solo dos de estos. Se debe especificar cuál de las series formales corresponden al gráfico secundario.
- Entrada:
    - Tipo de gráfico: en este campo se selecciona la combinación de gráficos a utilizar.
    - Título del gráfico: en este campo se introduce el título del gráfico.
    - Tamaño del gráfico: en este campo se introduce las dimensiones del gráfico.
    - Series secundarias: en este campo se debe especificar cuáles son las series de datos que van a ser utilizadas por el gráfico secundario.

- Salida: una imagen de formato PNG y la url que genera el gráfico.

RF11: Crear un origen de datos desde un Excel

- Descripción: el componente debe permitir obtener el conjunto de datos que se va a utilizar desde un Excel. El fichero debe cumplir con las siguientes características:
  - El fichero tiene que tener extensión xlsx o xls.
  - El Excel debe tener como mínimo dos columnas y la primera celda debe estar en blanco.
  - La primera columna deberá ser las etiquetas de las series formales (no deben estar vacías) y la primera fila deberá ser las etiquetas de las series reales (no deben estar vacías), los datos restantes son los valores de las series formales (deben ser valores numéricos).
- Entrada:
  - Ubicación: dirección donde se encuentra el fichero.
- Salida: conjunto de datos a utilizar para generar el gráfico.

RF12: Crear un origen de datos manualmente

- Descripción: el componente debe permitir introducir manualmente el conjunto de datos que se va utilizar.
- Entrada:
  - Cantidad de series reales[3]: en este campo se debe insertar el total de series reales.
  - Cantidad de series formales[4]: en este campo se debe insertar el total de series formales que existen para las series reales.
- Salida: conjunto de datos a utilizar para generar el gráfico.

RF13: Crear un origen de datos desde una Base de Datos

- Descripción: el componente debe permitir obtener el conjunto de datos que se va a utilizar desde una Base de Datos mediante una consulta sql. El resultado de la consulta debe tener las siguientes características:
  - La primera columna deberá ser las etiquetas de las series formales.
  - Los datos restantes deben ser valores numéricos.
- Entrada:

3 Se denomina serie real a una unidad completa de datos que se puede representar en un gráfico determinado.
4 Le denomina serie formal a cada una de las partes del parámetro en la especificación de datos.

- Servidor: servidor de Base de Datos (BD).
- Usuario: usuario para acceder a la BD.
- Contraseña: contraseña para acceder a la BD.
- Puerto: puerto disponible para el servidor de BD.
- Base de Datos: nombre de la BD.
- Sql: consulta sql para obtener los datos de la BD.

➢ Salida: conjunto de datos a utilizar para generar el gráfico.

RF14: Exportar series de datos

➢ Descripción: el componente debe permitir exportar en un Excel el conjunto de datos que ha sido insertado al sistema.

➢ Entrada:

- Conjunto de datos: conjunto de datos que se desea exportar.

➢ Salida: un fichero Excel.

RF15: Personalizar el gráfico

➢ Descripción: el componente debe permitir personalizar el gráfico para así mejorar la experiencia del cliente en el momento de visualizar el gráfico. Los valores de personalización deben tener valores por defecto para facilitarle el trabajo al usuario en la aplicación.

➢ Entrada:

- Paleta de colores: conjunto de colores que se utilizan para dibujar el gráfico.
- Icono para la leyenda: figura geométrica para representar la leyenda.
- Fuente de la leyenda: fuente que se utiliza para dibujar la leyenda.
- Color de la leyenda: color que se utiliza para dibujar la leyenda.
- Fuente del título: fuente que se utiliza para dibujar el título.
- Color del título: color que se utiliza para dibujar el título.
- Modo de la escala: modo que utiliza el componente para calcular la escala del gráfico.

➢ Salida: variables necesarias para personalizar el gráfico.

RF16: Exportar el gráfico

- Descripción: el componente debe permitir una vez realizados todos los pasos para generar un gráfico, exportar el gráfico generado y la url que permite construirlo.
- Entrada: imagen que contiene el gráfico y la url que lo genera.
- Salida: fichero que contiene la imagen y la url que genera el gráfico.

Requisitos no funcionales

Los requisitos no funcionales (RNF) son propiedades o cualidades que el producto debe tener. Estas propiedades o cualidades se refieren a las características que hacen al producto atractivo, usable, rápido o confiable. Por lo general los requisitos no funcionales son fundamentales en el éxito del producto; normalmente están vinculados a los requisitos funcionales, es decir, una vez que se conoce lo que el sistema debe hacer, se puede determinar cómo ha de comportarse, qué cualidades o propiedades debe tener. (27)

Software

Se especifica el software del que se debe disponer, después de implementado el sistema.

Servidor:

- **RNF1**: un servidor web que soporte y tenga instalado php5, php5-gd, php5-mysql, php5-pgsql.

Cliente:

- **RNF2**: disponer el navegador web Mozilla Firefox 18.

Hardware

Se deben enunciar los elementos de hardware que se necesitan para que el software cumpla sus funcionalidades.

Cliente:

- **RNF3**:
    - CPU Intel Pentium D 3 GHz (o superior).
    - 1 GB RAM (o superior).
    - 80 GB disco duro (o superior).
    - Tarjeta de red.

Servidor:

- **RNF4**:
    - CPU Intel Pentium D 3 GHz (o superior).
    - 1 GB RAM (o superior).

- 80 GB disco duro (o superior).
- Tarjeta de red.

Usabilidad

La usabilidad es el grado en el cual un producto puede ser utilizado por sus usuarios para lograr metas con efectividad, eficiencia y satisfacción en un determinado contexto de uso.

- **RNF5:** Debe ser una herramienta sencilla y fácil de usar para el usuario.

Apariencia o Interfaz Externa

Describe la apariencia del producto. Especifica cómo se pretende que sea la interfaz externa del componente.

- **RNF6**: El componente permitirá claridad con todo lo relacionado a la construcción de gráficos estadísticos para propiciar así que el usuario tenga un buen entendimiento de las utilidades que brinda.

Soporte

Le brinda al cliente la facilidad de instalación, facilidad de mantenimiento, lo que requiere código y diseño documentado y facilidad de actualización hacia versiones modernas.

- **RNF7**: El componente deberá tener un manual de usuario con una descripción completa de todas las funcionalidades del componente.

2.3 Modelo de casos de uso del sistema

El diagrama de casos de uso representa la forma en cómo un Cliente (Actor) opera con el sistema en desarrollo, además de la forma, tipo y orden en cómo los elementos interactúan (operaciones o casos de uso). (28) Por lo tanto los casos de uso determinan los requisitos funcionales del sistema, es decir, representan las funciones que un sistema puede ejecutar. Su ventaja principal es la facilidad para interpretarlos, lo que hace que sean especialmente útiles en la comunicación con el usuario.

Un diagrama de casos de uso consta de los siguientes elementos:

- Actor.
- Casos de uso.
- Relaciones de Uso, Herencia y Comunicación.

Luego de realizar el análisis se definieron 16 requisitos funcionales agrupados en el caso de uso Construir_Gráfico, los cuales el Chart Server 2.0 debe cumplir.

Actor	Descripción
Usuario	Cualquier persona que utiliza el Chart Server 2.0 para construir gráficos estadísticos.

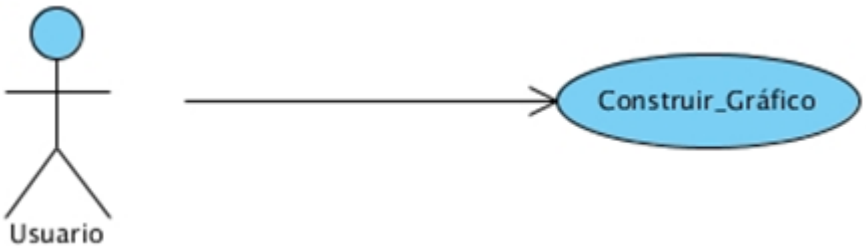


Figura 2 Diagrama del caso de uso Construir_Gráfico

Descripción del Caso de uso: Construir_Gráfico		
Caso de Uso:	Construir_Gráfico	
Actores:	Usuario	
Resumen:	El caso de uso comienza cuando el usuario necesita generar un gráfico estadístico y accede al sistema. El sistema le brinda una serie de formularios en los cuales se introducen los parámetros que le permite la construcción del gráfico. El caso de uso finaliza cuando se exporta el gráfico en una imagen de formato PNG y la url que lo genera.	
Precondiciones:	Al menos una solicitud para crear un gráfico.	
Referencias	RF1…RF16	
Prioridad	Crítico	
Flujo Normal de Eventos		
Sección "Construir_Gráfico"		
Acción del Actor		**Respuesta del Sistema**
1.	Accede al sistema	
2.		Muestra en la interfaz un formulario con los siguientes campos: • Título de gráfico

		• Tipo de gráfico • Dimensiones (largo y ancho)
3.	Introduce los datos en la interfaz y da clic en el botón “Siguiente”.	
4.		Valida los datos, en caso que existan campos incorrectos, ver paso 1 del Flujo Alterno.
5.		Muestra en la interfaz tres opciones para seleccionar el origen de datos que desea utilizar: • Manual • Excel • Base de Datos (BD) Permite: Seleccionar el botón “Anterior”, ver paso 2 del Flujo Alterno.
6.	Selecciona una de las opciones y da clic en el botón “Siguiente”.	
7.		En caso que el usuario seleccione: • Manual: ver sección 1 • Excel: ver sección 2 • BD: ver sección 3
8.		Muestra en la interfaz un formulario con los siguientes campos: • Tipo de fuente • Tamaño de la fuente • Paleta de colores • Modo de la escala • Icono para la leyenda Permite: Seleccionar el botón “Anterior”, ver paso 4 del

		Flujo Alterno.
9.	Modifica las opciones de configuración y da clic en el botón “Finalizar”.	
10.		Valida los datos, en caso que existan campos incorrectos, ver paso 3 del Flujo Alterno.
11.		Muestra en la interfaz una imagen con el gráfico y la url que lo genera.
12.	Da clic en el botón “Exportar”.	
13.		Muestra una interfaz para elegir la ubicación donde se va a guardar el fichero que contiene el gráfico y la url. Permite: Seleccionar el botón “Cancelar”, ver paso 5 del Flujo Alterno.
14.	Selecciona la ubicación y da clic en el botón “Aceptar”.	
15.		Finaliza el caso de uso.

Prototipo de Interfaz

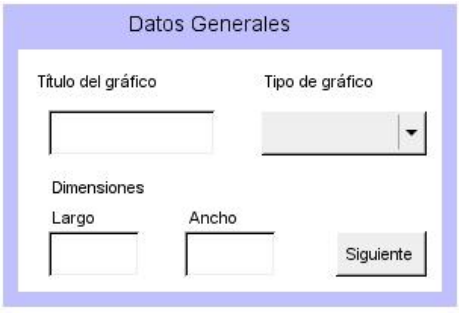


Figura 3: Datos generales

Figura 4: Origen de datos

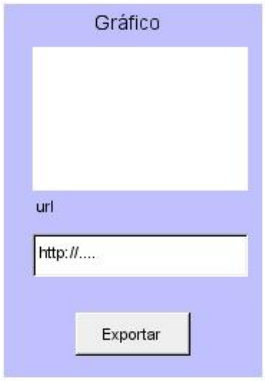


Figura 5: Gráfico

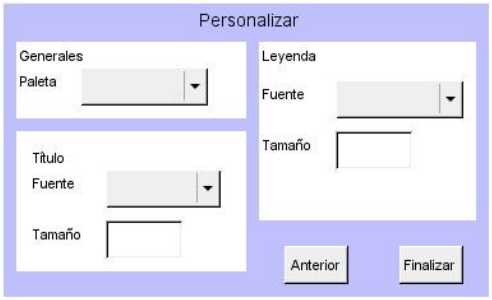


Figura 6: Personalizar

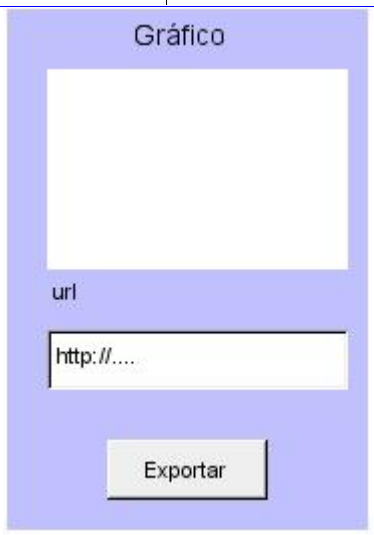


Figura 7 Exportar gráfico

Flujos Alternos		
Acción del Actor		**Respuesta del Sistema**
1.		Muestra un mensaje de error al actor informando que existen campos con errores.
2.		Muestra la interfaz anterior (paso 2 Flujo Normal de Eventos, sección Construir_Gráfico).
3.		Muestra un mensaje de error al actor

		informando que existen campos con errores.
4.		Muestra la interfaz anterior (paso 6 Flujo Normal de Eventos, sección Construir_Gráfico).
5.		Cierra la interfaz.
Sección 1: "Manual"		
Flujo Básico		
Acción del Actor		**Respuesta del Sistema**
1.		Muestra en la interfaz un formulario con los siguientes campos: • Cantidad de series reales • Cantidad de series formales
2.	Introduce los datos en la interfaz y da clic en el botón "Aceptar".	
3.		Valida los datos, en caso que existan campos incorrectos, ver paso 1 del Flujo Alterno.
4.		Muestra los campos con un conjunto de datos aleatorio para que el actor lo complete con sus datos. Permite: Seleccionar el botón "Anterior", ver paso 2 del Flujo Alterno. Seleccionar el botón "Exportar", ver sección 4.
5.	Introduce los datos en la interfaz y da clic en el botón "Siguiente".	
Prototipo de Interfaz		

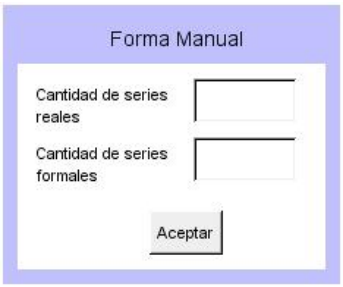


Figura 8: Manual

Figura 9 Conjunto de datos

Flujos Alternos		
Acción del Actor		**Respuesta del Sistema**
1.		Muestra un mensaje de error al actor informando que existen campos con errores.
2.		Muestra la interfaz anterior (paso 2 Flujo Normal de Eventos, sección Construir_Gráfico).
Sección 2: "Excel"		
Flujo Básico		
Acción del Actor		**Respuesta del Sistema**
1.		Muestra en la interfaz un formulario con el siguiente campo: • Ubicación
2.	Introduce la ubicación del Excel y da clic en el botón "Aceptar".	
3.		Comprueba que el Excel es válido. En caso de ser inválido ver el paso 1 del Flujo Alterno.
4.		Muestra los campos con el conjunto de datos obtenido desde el Excel. Permite: Seleccionar el botón "Anterior", ver paso 2 del

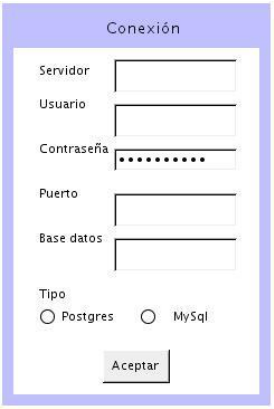
		Flujo Alterno. Seleccionar el botón “Exportar”, ver sección 4.
5.	Da clic en el botón “Siguiente”.	

Prototipo de Interfaz

Figura 10: Subir fichero

Flujos Alternos		
Acción del Actor		**Respuesta del Sistema**
1.		Muestra un mensaje de error y va al paso 1 del Flujo Básico sección 2.
2.		Muestra la interfaz anterior (paso 2 Flujo Normal de Eventos, sección Construir_Gráfico).

Sección 3: “Base de Datos”		
Flujo Básico		
Acción del Actor		**Respuesta del Sistema**
1.		Muestra en la interfaz un formulario con los siguientes campos: • Servidor • Usuario • Contraseña • Puerto. • Base de Datos (BD). • Tipo (Postgres o MySQL).
2.	Introduce los datos en la interfaz y da	

	clic en el botón “Siguiente”.	
3.		Valida los datos, en caso que existan campos incorrectos, ver paso 1 del Flujo Alterno.
4.		Crea la conexión a la BD. En caso de que no sea posible la conexión, ver paso 2 del Flujo Alterno.
5.		Solicita la consulta que permite generar el conjunto de datos.
6.	Introduce la consulta y da clic en el botón “Finalizar”.	
7.		El sistema ejecuta la consulta y crea el conjunto de datos en caso de no existir errores. En caso de haber errores ver paso 3 del Flujo Alterno.
7.		Muestra los campos con el conjunto de datos obtenido desde la BD. Permite: Seleccionar el botón “Exportar”, ver sección 4.
8.	Da clic en el botón “Siguiente”.	
Prototipo de Interfaz		
 ***Figura 11: Conexión***		 ***Figura 12 Insertar consulta***
Flujos Alternos		

Acción del Actor		Respuesta del Sistema
1.		Muestra un mensaje de error al actor informando que existen campos con errores.
2.		Muestra un mensaje de error informando que no es posible realizar la conexión.
3.		Muestra un mensaje de error al actor informando que la consulta no es correcta.
Sección 4: "Exportar conjunto de datos"		
Flujo Básico		
Acción del Actor		**Respuesta del Sistema**
1.		Muestra una interfaz para elegir la ubicación donde va a guardar el archivo. Permite: Seleccionar el botón "Cancelar". Ver paso 1 del Flujo Alterno.
2.	Elige la ubicación y da clic en el botón "Aceptar".	
3.		El sistema guarda el fichero.
Flujos Alternos		
Acción del Actor		**Respuesta del Sistema**
1.		Muestra la interfaz anterior.

2.4 Patrones

Patrones de diseño GRASP utilizados

Controlador: este patrón se evidencia por el hecho de usar Symfony 2 ya que este dirige cada petición realizada por el cliente a una acción en un controlador específico. Ejemplo cuando un usuario accede a la ruta /api, Symfony 2 dirige esta petición a la clase mainController que es la encargada de atender esta petición con el método apiAction().

Experto: este patrón se evidencia cuando el sistema recibe la petición de crear un gráfico en específico y

la clase mainController crea una instancia de la clase encargada de generarlo. Ejemplo si un usuario solicita un gráfico de pastel se crea una instancia de la clase pastelChart. Este patrón trae grandes beneficios ya que se conserva el encapsulamiento y el bajo acoplamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide.

Alta cohesión: el concepto de este patrón dicta que una clase tiene responsabilidades moderadas en un área funcional, colabora con las otras para llevar a cabo las tareas. Symfony organiza el trabajo en cuanto a la estructura del proyecto, lo cual permite crear y trabajar con clases que tienen una alta cohesión. Por ejemplo la clase mainController colabora con varias clases (auxiliarCalc, imagenDatos, traductor) para realizar todas las acciones previas antes de proceder a crear un gráfico.

Patrón de diseño GOF utilizado

Decorador: este patrón se evidencia en la plantilla index.html.twig la cual hereda de la plantilla base layout.html.twig, esta plantilla contiene todo el código HTML y JavaScript común para todas las vistas que se diseñan en la aplicación.

Patrones de arquitectura utilizados

Cliente – **Servidor**: este patrón se evidencia en los bundle principales de la aplicación ya que se separa el cliente del servidor. El bundle mainBundle es el encargado de atender las solicitudes del servidor y el chartServerAppBundle encargado de atender las solicitudes del lado del cliente.

Modelo – **Vista** – **Controlador**: este patrón se evidencia por el simple hecho de utilizar Symfony 2 en el desarrollo del componente ya que este se rige por dicho patrón arquitectónico. Este patrón cuenta con tres capas, las cuales se evidencian en el componente de la siguiente manera: en la capa de la vista se encuentra el bundle chartServerAppBundle, en la capa controlador el bundle mainBundle y la capa del modelo no se utiliza ya que el componente no cuenta con una Base de Datos.

2.5 Diagrama de paquetes del componente Chart Server 2.0

La organización física de los elementos que conforman el sistema se realizó teniendo en cuenta los principios establecidos por Symfony 2 que estructuran el proyecto en paquetes o bundles. Un bundle es un directorio que tiene una estructura definida, puede alojar cualquier elemento, desde clases para los controladores hasta recursos web. La diferencia clave es que todo es un bundle en Symfony 2, incluyendo tanto la funcionalidad del núcleo del marco de trabajo como el código escrito para su aplicación. Esto le da la flexibilidad de usar características pre-construidas empaquetadas en bundles de terceros o distribuir sus

propios bundles. Facilita escoger o seleccionar qué características habilitar en la aplicación y optimizarlas de la manera que se desee.

A continuación se muestra la figura 13 con la estructura del componente Chart Server 2.0 a través del diagrama de paquetes del componente Chart Server 2.0.

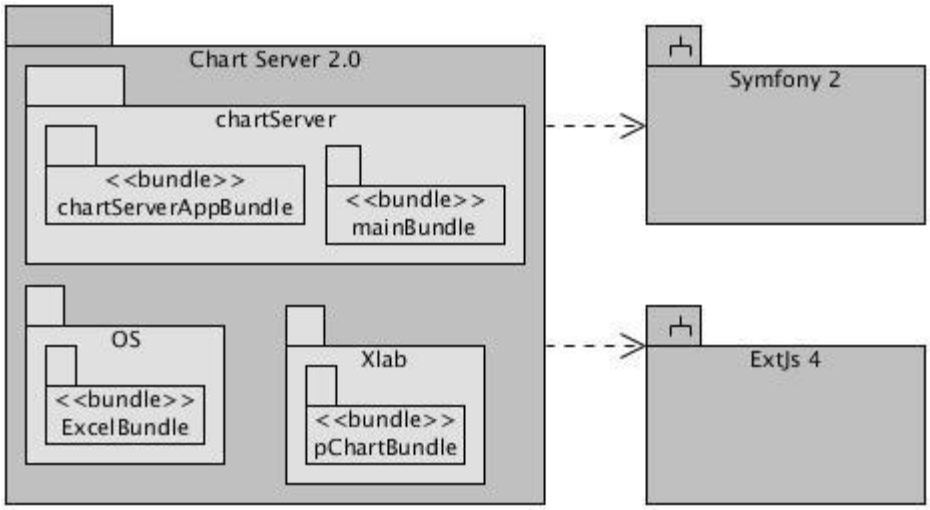


Figura 13 Diagrama de paquetes del componente Chart Server 2.0

Básicamente el componente está compuesto por 4 bundles:

- **mainBundle**: bundle encargado de manejar todas las acciones del lado del servidor tales como: generar el gráfico, obtener información desde un Excel y conectarse a la Base de Datos.

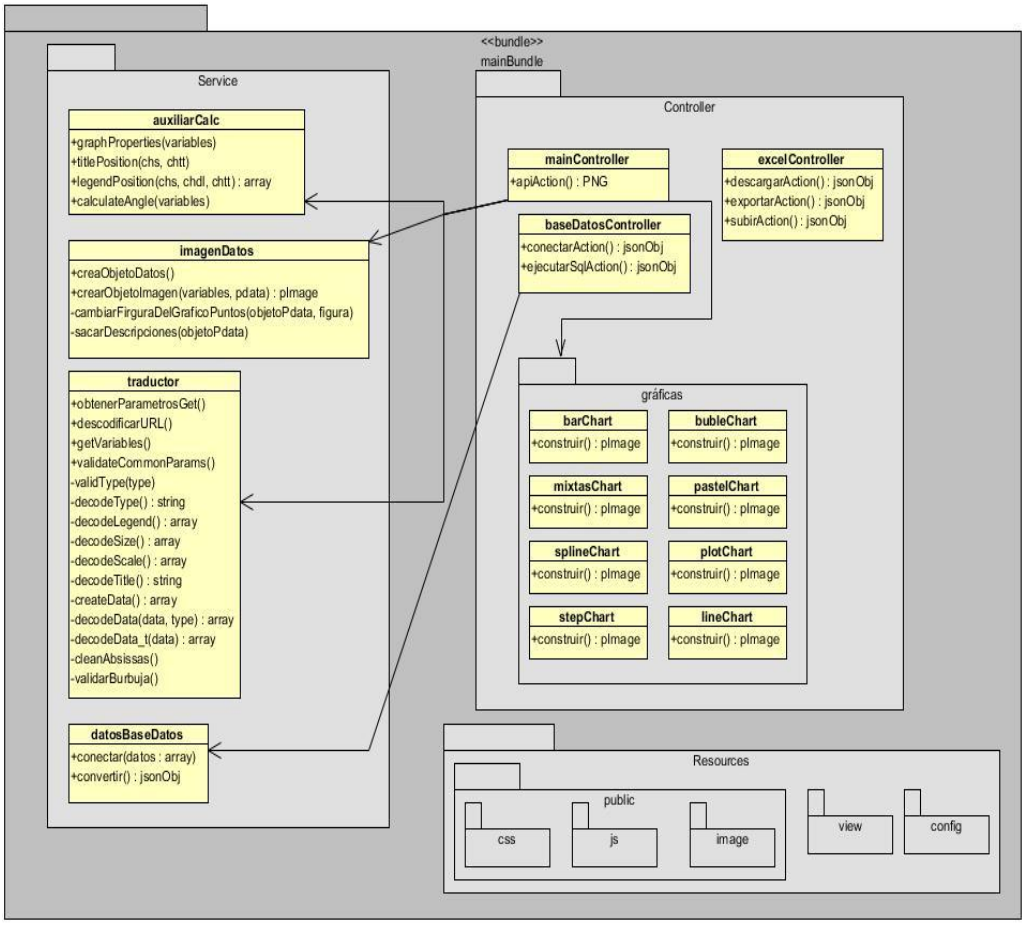


Figura 14 Estructura del bundle en el lado del servidor

- **chartServerAppBundle:** bundle encargado de manejar todas las acciones del lado del cliente, o sea, es el que contiene la interfaz con la cual interactúa el usuario.

Ext JS propone una arquitectura para el desarrollo de aplicaciones, para así facilitar la escalabilidad, mantenibilidad y flexibilidad de la aplicación. Consiste en una estructura de carpetas para separar todos los componentes que se utilicen. En el desarrollo de la interfaz se utilizó la arquitectura propuesta por Ext JS y se encuentra en el paquete ChartServerApp (figura 15) dentro del bundle chartServerAppBundle.

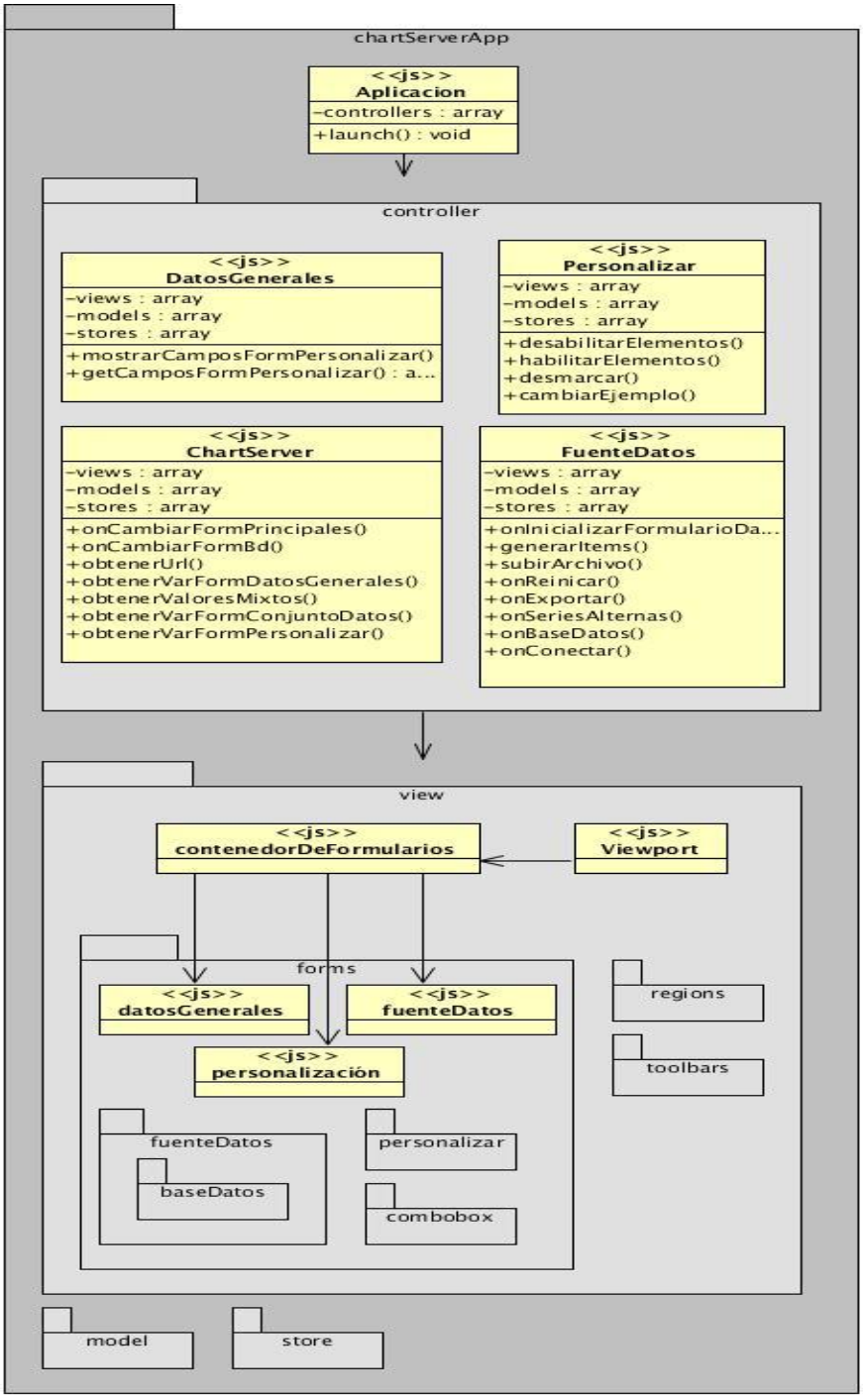


Figura 15 Estructura del paquete ChartServerApp

- **pChartBundle** y **ExcelBundle**: estos bundles contienen las bibliotecas (Pchart 2.1.3 y PHPExcel 1.7.6) necesarias para el funcionamiento del componente.

2.6 Diagramas de clases del diseño

Un diagrama de clases del diseño describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Los cuales son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema. Además de los componentes que se encargarán del funcionamiento y la relación entre uno y otro.

En la figura 16 se muestra el diagrama de clases del diseño del Chart Server 2.0.

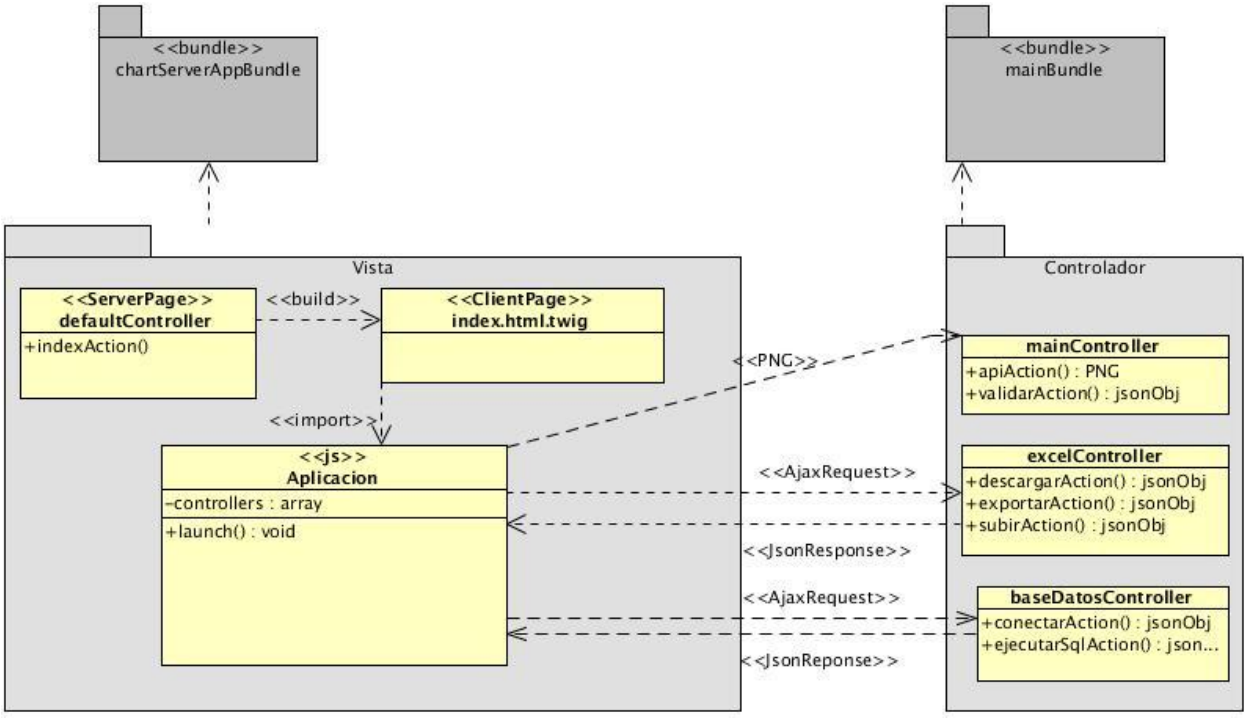


Figura 16 Diagrama de clases del diseño del Chart Server 2.0

Descripción de las clases

Bundle: "mainBundle"	
Clase	**Descripción**
Clases para generar gráficos	
mainController	Controlador para atender las solicitudes para crear los gráficos.
traductor	Clase auxiliar para obtener y validar los parámetros de la petición.

auxiliarCalc	Clase auxiliar que contiene métodos para calcular las posiciones de los elementos del gráfico.
imagenDatos	Clase auxiliar para inicializar el objeto pImage necesario para generar un gráfico.
barChart	Clase para generar gráficos de barras.
bubleChart	Clase para generar gráficos de burbuja.
mixtasChart	Clase para generar gráficos mixtos.
pastelChart	Clase para generar gráficos de pastel.
splineChart	Clase para generar gráficos de curvas.
plotChart	Clase para generar gráficos de puntos.
stepChart	Clase para generar gráficos de pasos.
lineChart	Clase para generar gráficos de líneas.
Clases para manipulación de datos	
excelController	Controlador para atender las solicitudes relacionadas con la lectura y escritura de un Excel.
baseDatosController	Controlador para atender las solicitudes relacionadas con una Base de Datos.
datosBaseDatos	Clase auxiliar para el trabajo con una Base de Datos.

Bundle: "chartServerAppBundle"	
Clase	**Descripción**
defaultController	Controlador encargado de generar la vista.
Principales clases de JavaScript	
Aplicación	Clase que extiende de Ext.application, es la encargada de inicializar los controladores de la vista.
ChartServer	Clase que extiende de Ext.app.Controller, es la encargada de manejar los eventos principales de la aplicación.
DatosGenerales	Clase que extiende de Ext.app.Controller, es la encargada de manejar los eventos del formulario Datos Generales.
datosGenerales	Formulario para insertar los datos generales.

FuenteDatos	Clase que extiende de Ext.app.Controller, es la encargada de manejar los eventos del formulario Fuente de datos.
fuenteDatos	Formulario para manipular el conjunto de datos.
Personalizar	Clase que extiende de Ext.app.Controller, es la encargada de manejar los eventos del formulario Personalizar.
personalizar	Formulario para personalizar el gráfico.

2.7 Diagramas de interacción

Los diagramas de interacción modelan el comportamiento dinámico del sistema y el flujo central en una operación. Describen la interacción entre objetos, los cuales interactúan a través de mensajes para cumplir ciertas tareas. Existen dos tipos de diagramas de interacción: secuencia y colaboración. (16)

Diagrama de secuencia

Muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo. El diagrama de secuencia contiene detalles de la implementación del escenario, incluyendo los objetos y clases que se usan para implementar el escenario y mensajes intercambiados entre los objetos. (21)

Diagrama de secuencia del caso de uso Construir_Gráfico

En la figura 17 se muestra el diagrama de secuencia del caso de uso Construir_Gráfico del componente Chart Server 2.0.

El usuario accede al Chart Server 2.0, el sistema genera la interfaz con el formulario para introducir los datos generales. El usuario introduce los datos y si son correctos, el sistema le activa la opción “Siguiente”. Después de haber presionado el botón “Siguiente”, el sistema brinda los formularios necesarios para crear un conjunto de datos. Luego de haber cumplido con todos los pasos para crear un conjunto de datos, el componente le permite al usuario personalizar el gráfico. Finalmente el Chart Server 2.0 muestra el gráfico estadístico en una imagen de formato PNG y la url que lo genera.

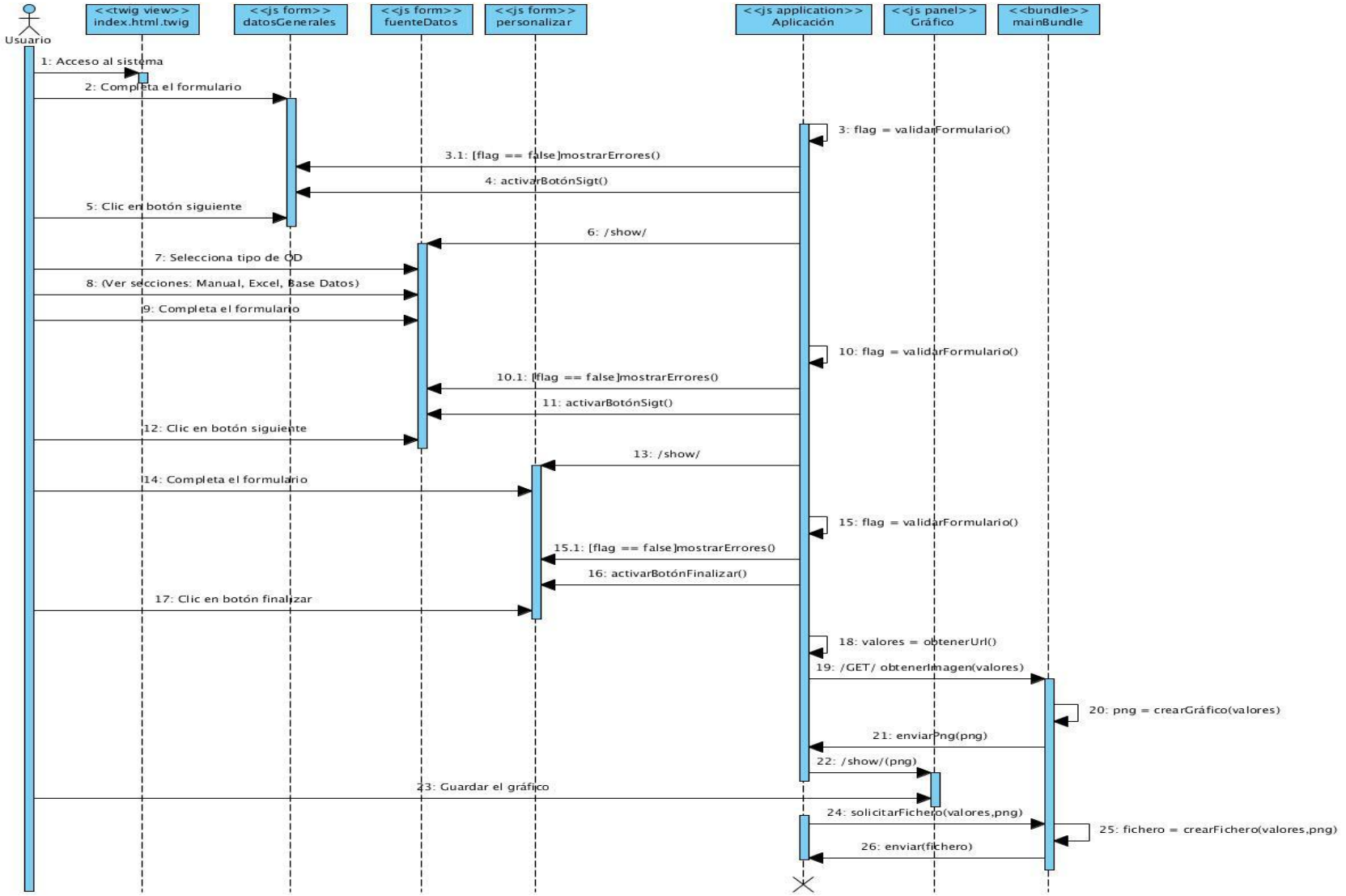


Figura 17 Diagrama de secuencia del caso de uso Construir_Gráfico

2.8 Modelo de despliegue

El modelo de despliegue se utiliza para modelar el hardware utilizado en las implementaciones del sistema y las relaciones entre sus componentes. Describe la topología del sistema, la estructura de los elementos de hardware y el software que ejecuta cada uno de ellos. Los diagramas de despliegue representan a los nodos y sus relaciones. Los nodos son conectados por asociaciones de comunicación tales como enlaces de red o conexiones TCP/IP. Son los complementos de los diagramas de componentes que, unidos, proveen la vista de implementación del sistema. (29)

Figura 18 Modelo de despliegue

- PC_Cliente: computadora que es usada por el usuario para acceder al Chart Server 2.0.
- PC_Servidor: computadora donde está publicado el servidor.

Conclusiones del capítulo

En el presente capítulo se diseñó el modelo de dominio en el cual se describen los principales conceptos para poder entender el sistema. Se identificaron 16 requisitos funcionales y 7 no funcionales, los cuales son las funcionalidades y características que el sistema debe cumplir. Los requisitos funcionales se agruparon en el caso de uso Construir_Gráfico el cual es representado en el diagrama de casos de uso. Se utilizaron los siguientes patrones de diseño: Controlador, Experto y Alta cohesión que son patrones GRAP y dentro de los GOF se utilizó el patrón Decorador. Los patrones de arquitectura utilizados fueron: Modelo-Vista-Controlador y Cliente-Servidor. Se representó la estructura del sistema, a través del diagrama de paquetes, clases y secuencia, además se modeló el diagrama de despliegue.

Capítulo 3: Implementación y pruebas

A raíz del resultado obtenido en la fase de diseño, este capítulo contiene todos los artefactos necesarios para realizar satisfactoriamente la fase de implementación y pruebas. Se representa el diagrama de componentes del sistema, el cual se utiliza para modelar la vista estática del componente, muestra y organiza las relaciones de dependencia entre los componentes y subsistemas que lo conforman. Se especifican los estándares de codificación a utilizar y se realizan pruebas funcionales de caja negra al componente.

3.1 Diagrama de componentes del sistema

Los diagramas de componentes se utilizan para modelar la vista estática de un sistema, muestran y organizan las relaciones de dependencia entre los componentes y subsistemas que lo conforman. Los componentes representan los elementos de un modelo dentro de un paquete, como son las clases en el modelo de diseño. Normalmente los diagramas de componentes contienen: componentes, interfaces, relaciones de dependencia, generalización, asociación y realización, paquetes y subsistemas. (16)

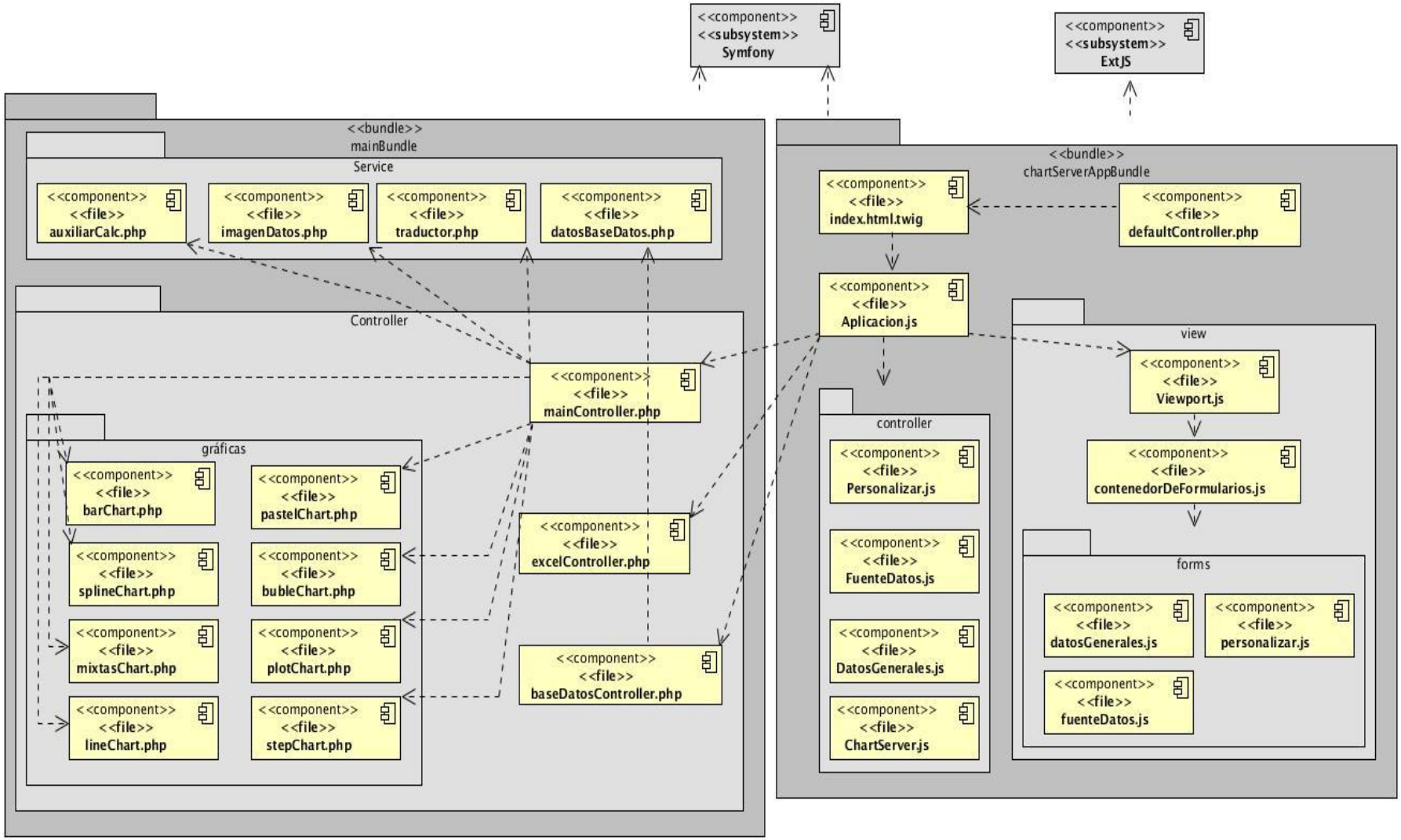


Figura 19 Diagrama de componentes del sistema

Descripción de los ficheros del diagrama de componentes del sistema

Bundle: "mainBundle"	
Fichero	**Descripción**
Clases para generar gráficos	
mainController.php	Fichero que contiene el controlador mainController que se encarga de atender las solicitudes para crear los gráficos.
traductor.php	Fichero que contiene la clase traductor que es la encargada de obtener y validar los parámetros de la petición.
auxiliarCalc.php	Fichero que contiene la clase auxiliarCalc, la cual contiene métodos para calcular las posiciones de los elementos del gráfico.
imagenDatos.php	Fichero que contiene la clase imagenDatos, la cual permite inicializar el objeto pImage necesario para generar un gráfico.
barChart.php	Fichero que contiene la clase barChart que es la encargada de generar gráficos de barras.
bubleChart.php	Fichero que contiene la clase bubleChart que es la encargada de generar gráficos de burbuja.
mixtasChart.php	Fichero que contiene la clase mixtasChart que es la encargada de generar gráficos mixtos.
pastelChart.php	Fichero que contiene la clase pastelChart que es la encargada de generar gráficos de pastel.
splineChart.php	Fichero que contiene la clase splineChart que es la encargada de generar gráficos de curvas.
plotChart.php	Fichero que contiene la clase plotChart que es la encargada de generar gráficos de puntos.
stepChart.php	Fichero que contiene la clase stepChart que es la encargada de generar gráficos de pasos.
lineChart.php	Fichero que contiene la clase lineChart que es la encargada de generar gráficos de líneas.
Clases para manipulación de datos	
excelController.php	Fichero que contiene el controlador excelController, el cual se encarga de atender las solicitudes relacionadas con la lectura y escritura de un Excel.
baseDatosController.php	Fichero que contiene el controlador baseDatosController, el cual se encarga de atender las solicitudes relacionadas con una Base de Datos.

datosBaseDatos.php	Fichero que contiene la clase datosBaseDatos que se encarga del trabajo con una Base de Datos.

Bundle: "chartServerAppBundle"	
Fichero	**Descripción**
defaultController.php	Fichero que contiene el controlador frontal defaultController, el cual es el encargado de generar la vista.
Aplicación.js	Fichero que contiene la clase Aplicación que extiende de Ext.application, es la encargada de inicializar los controladores de la vista.
ChartServer.js	Fichero que contiene la clase ChartServer que extiende de Ext.app.Controller, es la encargada de manejar los eventos principales de la aplicación.
DatosGenerales.js	Fichero que contiene la clase DatosGenerales que extiende de Ext.app.Controller, es la encargada de manejar los eventos del formulario Datos Generales.
datosGenerales.js	Fichero que contiene el formulario para insertar los datos generales.
FuenteDatos.js	Fichero que contiene la clase FuenteDatos que extiende de Ext.app.Controller, es la encargada de manejar los eventos del formulario Fuente de datos.
fuenteDatos.js	Fichero que contiene el formulario para manipular el conjunto de datos.
Personalizar.js	Fichero que contiene la clase Personalizar que extiende de Ext.app.Controller, es la encargada de manejar los eventos del formulario Personalizar.
personalizar.js	Fichero que contiene el formulario para personalizar el gráfico.
Viewport.js	Fichero que contiene la vista utilizada por la aplicación para interactuar con el cliente.
contenedorDeFormularios.js	Fichero que contiene el panel que contiene todos los formularios.

3.2 Estándares de codificación

Para el desarrollo del componente se utilizaron los estándares de codificación definidos por Symfony 2. La ventaja de utilizar estos estándares es que cada pieza de código se ve y se siente familiar, así facilita el

desarrollo y el mantenimiento futuro de la aplicación.

Estándares utilizados:

- Un solo espacio después de cada delimitador coma.

```
:.FuenteDatos', {
```

- Un solo espacio alrededor de los operadores.

```
value: i % 2 === 1 ? Ext.N
```

- Una coma después de cada elemento del arreglo en un arreglo multilínea.

```
$fuentes = array('verdana.ttf', 'Bedizen.ttf', 'calibri.ttf', 'Forgotte.ttf',
    'Bedizen.ttf', 'MankSans.ttf', 'pf_arma_five.ttf', 'Silkscreen.ttf', 'advent_light.ttf');
```

- Una línea en blanco antes de las declaraciones return, a menos que el valor devuelto solo sea dentro de un grupo de declaraciones.

```
getCamposPersonalizarGraficoEspecifico: function() {

    return Ext.ComponentQuery.query('fieldcontainer[
```

- Llaves para indicar la estructura del cuerpo de control, independientemente del número de declaraciones que contenga.

```
if (record[0].data.mixto){
    Ext.ComponentQuery.query('to
}
else{
    Ext.ComponentQuery.query('to
}
```

- Una clase por archivo — esto no se aplica a las clases ayudantes privadas, de las cuales no se tiene la intención de crear una instancia desde el exterior.
- Las propiedades de clase antes que los métodos.

```
private $xchxd = null;

/**
 * Constructor de la clase traductor
 */
public function __construct() {

}
```

- Usar paréntesis para instanciar clases. (30)

```
$imagenDatos = new imagenDatos();
```

3.3 Implementación significativa

En la figura 20 se muestra el método crearObjetoImagen() de la clase imagenDatos. En dicho método se crea el objeto de la clase pImage, este objeto se utiliza durante todo el proceso de generar el gráfico estadístico: personalización, crear la leyenda, crear la escala, definir el área del gráfico, el área de la leyenda y finalmente generar el gráfico.

```
function crearObjetoImagen($variables, $pdata) {
    $imagen = new pImage($variables['chs'][0], $variables['chs'][1], $pdata);
    if (isset($variables['con_estilo'])) {
        $Config = array("R" => 170, "G" => 183, "B" => 87, "Dash" => 1, "DashR" => 190, "DashG" => 203, "DashB" => 107);
        $imagen->drawFilledRectangle(0, 0, $variables['chs'][0], $variables['chs'][1], $Config);
        $Config = array("StartR" => 219, "StartG" => 231, "StartB" => 139, "EndR" => 1, "EndG" => 138, "EndB" => 68, "Alpha" => 50);
        $imagen->drawGradientArea(0, 0, $variables['chs'][0], $variables['chs'][1], DIRECTION_VERTICAL, $Config);
        $imagen->drawRectangle(0, 0, $variables['chs'][0]-1, $variables['chs'][1]-1, array("R" => 0, "G" => 0, "B" => 0));
    }
    $imagen->Antialias = isset($variables['antialias']) ? $variables['antialias'] : FALSE;
    $t_color = isset($variables['titulo_color']) ? $variables['titulo_color'] : "FFFFFF";
    $t_fuente = isset($variables['titulo_fuente']) ? $variables['titulo_fuente'] : "verdana.ttf";
    $t_size = isset($variables['titulo_size']) ? $variables['titulo_size'] : 10;
    list($r, $g, $b) = $this->extraerColores($t_color);
    $imagen->setFontProperties(array("FontName" => ROUTE_RESOURCE_FONTS . $t_fuente, "FontSize" => $t_size, "R" => $r, "G" => $g, "B" => $b)
    );
    isset($variables['chtt'][0]) ? $imagen->drawTitle2(array(0, 15, $variables['chs'][0], $variables['chs'][1]), $variables['chtt'], array($
    r, $g, $b)) : false;
    if (isset($variables['activar_leyenda']) && $variables['cht'] != 'p' && $variables['cht'] != 'p3') {
        list($r, $g, $b) = $this->extraerColores($variables['leyenda_color']);
        $imagen->setFontProperties(array("FontName" => ROUTE_RESOURCE_FONTS . $variables['leyenda_fuente'], "FontSize" => $variables['
        leyenda_size'], "R" => $r, "G" => $g, "B" => $b));
        $imagen->setShadow(true);
        $this->parametrosLeyenda($variables);
        $formato = array("R" => 255, "G" => 255, "B" => 255,"Family" => $variables['estilo_leyenda'],"Style" => $variables['formato_leyenda'
        ],"Alpha" => 50);
        isset($variables['chdl']) ? $imagen->drawLegend($variables['gp']['lp'][0], $variables['gp']['lp'][1], $formato) : true;
        $imagen->setShadow(false);
    }
    $imagen->setFontProperties(array("FontName" => ROUTE_RESOURCE_FONTS . "verdana.ttf", "FontSize" => 10, "R" => 1, "G" => 1, "B" => 1));
    if ($variables['cht'] != 'p' && $variables['cht'] != 'p3') {
        $imagen->setGraphArea($variables['gp']["x1"], $variables['gp']["y1"], $variables['gp']["x2"], $variables['gp']["y2"]);
        $Settings = array("Pos" => SCALE_POS_LEFTRIGHT
            , "Mode" => isset($variables['formato_escala']) ? $variables['formato_escala'] : SCALE_MODE_ADDALL
            , "LabelingMethod" => LABELING_ALL, "GridR" => 1, "GridG" => 1, "GridB" => 1, "GridAlpha" => 10, "TickR" => 255, "TickG" => 255
            , "TickB" => 255, "TickAlpha" => 10, "LabelRotation" => $variables['angle'], "CycleBackground" => 1, "DrawXLines" => 1, "
            DrawSubTicks" => 1, "SubTickR" => 255, "SubTickG" => 0, "SubTickB" => 0, "SubTickAlpha" => 50, "DrawYLines" => ALL);
        if ($variables['cht'] != 'lxy')
            $imagen->drawScale($Settings);
        if ($variables['cht'] != 'lxy')
            $imagen->drawThreshold(0, array("WriteCaption" => false, "R" => 1, "G" => 1, "B" => 1, "NoMargin" => TRUE));
    } return $imagen;
}
```

Figura 20 Implementación significativa

3.4 Pruebas

La prueba es un elemento crítico para la garantía de la calidad del software, es el proceso que permite verificar y revelar la calidad de un producto de software. Son utilizadas para identificar posibles fallos de implementación, calidad o usabilidad de un programa. Básicamente es una de las fases del desarrollo de

software consistente en probar las aplicaciones construidas. Las pruebas de software se integran dentro de las diferentes fases del ciclo del software en la Ingeniería de Software. (31)

Nivel de prueba

La prueba es aplicada para diferentes tipos de objetivos, en diferentes escenarios o niveles de trabajo. Se distinguen los siguientes niveles de pruebas: prueba de desarrollador, unidad, integración, sistema y aceptación (32).

En el Chart Server 2.0 se aplicarán las pruebas a nivel de desarrollador que incluye las pruebas funcionales.

Técnica de prueba

Existen varias técnicas de pruebas, una de estas técnicas son las funcionales. Esta técnica como su nombre indica, prueba una funcionalidad completa, donde pueden estar implicadas una o varias clases y la propia interfaz de usuario. (16)

Se seleccionó como técnica de prueba las funcionales pues son las que comprueban el correcto funcionamiento de los componentes del sistema, analizando las entradas y salidas verificando que el resultado es el esperado.

Método de prueba

Existen dos métodos de pruebas para realizar las pruebas funcionales: el método de caja negra y de caja blanca. La prueba de caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. Las cuales se pueden aplicar mediante los Casos de Pruebas basados en los casos de uso. La prueba de la caja blanca comprueba los caminos lógicos del programa, las condiciones y bucles. Se puede examinar el estado del programa en varios puntos para determinar si el estado real coincide con el esperado o mencionado. (32)

Algunas ventajas de estas pruebas:

- La prueba termina siendo imparcial porque la persona que diseñó el software y la que lo prueba son independientes.
- El usuario que lo prueba no necesita conocimientos de programación.
- Las pruebas se realizan desde un punto de vista de usuario y no como programador, tomando en cuenta todas las funciones de cómo luce y su usabilidad (16).

El método que se seleccionó para comprobar el funcionamiento del sistema es el de caja negra, el cual se centra en los requisitos funcionales del software. Revisa que la entrada se acepte de forma adecuada y la salida sea correcta. Se aplica mediante los casos de pruebas basados en los casos de uso.

Casos de pruebas

Es un conjunto de condiciones o variables bajo las cuales el analista determinará si el requisito de una aplicación es parcial o completamente satisfactorio. Además ayudan a validar que el sistema desarrollado realice las funciones para las que ha sido creado en base a los requisitos del usuario solicitante.

Para validar el correcto funcionamiento del componente se realizaron las pruebas funcionales a nivel de desarrollador, utilizando el método de caja negra a través de los casos de pruebas basados en casos de uso. Se realizaron 3 iteraciones, identificando 8 no conformidades. En la figura 21 se muestran las no conformidades encontradas a las cuales se le dieron solución, comprobando que el componente crea el gráfico estadístico solo si todos los campos están correctos.

Fecha	Iteración	Caso de Prueba	Tipo de Error	Descripción
8/4/2013	1	Construir_Gráfico	Validación	El campo Título del gráfico acepta caracteres no válidos.
8/4/2013	1	Construir_Gráfico	Validación	El campo Etiqueta de las series reales acepta caracteres no válidos.
8/4/2013	1	Construir_Gráfico	Validación	El campo Usuario acepta caracteres no válidos.
9/4/2013	1	Construir_Gráfico	Recomendación	Cambiar el nombre de los campos relacionados al conjunto de datos.
10/4/2013	1	Construir_Gráfico	Recomendación	Cambiar la interfaz relacionada con el origen de datos.
10/4/2013	2	Construir_Gráfico	Validación	El campo Servidor acepta campos no válidos.
10/4/2013	2	Construir_Gráfico	Validación	El campo Puerto acepta campos no válidos.

10/4/2013	2	Construir_Gráfico	Validación	El campo Base de datos acepta campos no válidos.
11/4/2013	3	Construir_Gráfico	-	-

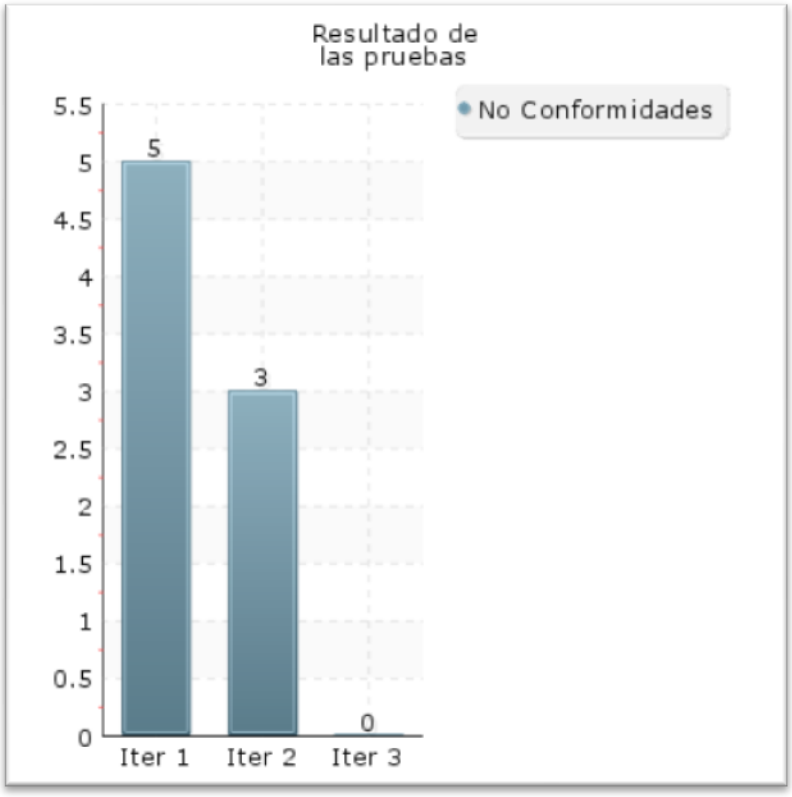


Figura 21 Gráfico de las no conformidades

En la primera iteración se identificaron 5 no conformidades, en la segunda iteración 3 y en la tercera iteración no se encontraron no conformidades.

Conclusiones del capítulo

En el presente capítulo se representó el diagrama de componentes asociado al caso de uso Construir_Gráfico, el cual representa la vista estática del sistema. Se mostró la implementación más significativa para un mejor entendimiento del componente. También se describen los estándares de codificación utilizados para garantizar que el código sea entendible y de fácil mantenimiento. Se realizaron pruebas funcionales para comprobar el funcionamiento del Chart Server 2.0 utilizando el método de prueba de caja negra, reflejándolo en los casos de pruebas obteniendo como resultado la solución a las no conformidades encontradas.

Conclusiones

- Se realizó la selección de las herramientas, metodología y tecnologías entre las que se encuentran: las herramientas Netbeans 7.2 y Visual Paradigm 6.4 for UML, la metodología OpenUP y el marco de trabajo Symfony 2.1.3.
- El levantamiento de los requisitos permitió identificar 16 requisitos funcionales y 7 requisitos no funcionales.
- Se realizó el análisis y diseño del componente Chart Server 2.0 para la construcción de gráficos estadísticos.
- Se implementaron todas las funcionalidades identificadas para dar solución a la problemática planteada.
- Con el objetivo de verificar el correcto funcionamiento del componente, se realizaron pruebas funcionales de caja negra, las cuales arrojaron 8 no conformidades a las que se les dieron una solución inmediata.

Recomendaciones

Se recomienda agregarle al componente Chart Server 2.0 los gráficos Radar, Polar y Scatter.

Referencias

1. **Thompson, Ivan.** *Definición de Información.* [En línea] [Citado el: 15 de Noviembre de 2012.] http://www.promonegocios.net/mercadotecnia/definicion-informacion.html.

2. **Rojas, Yesenia.** Scribd. [En línea] 2013. [Citado el: 15 de Noviembre de 2012.] http://www.scribd.com/doc/67106356/importancia-graficos-estadistica.

3. ChartGo. [En línea] 2009. [Citado el: 17 de Noviembre de 2012.] http://www.chartgo.com/index_es.jsp.

4. Rich Chart Live. [En línea] [Citado el: 17 de Noviembre de 2012.] http://www.richchartlive.com.

5. Lucid Chart. [En línea] 2012. [Citado el: 17 de Noviembre de 2012.] https://www.lucidchart.com.

6. Hohli. [En línea] 2011. [Citado el: 17 de Noviembre de 2012.] http://charts.hohli.com.

7. **Universidad de las Ciencias Informaticas.** Chart Server. [En línea] 2010. [Citado el: 17 de Noviembre de 2012.] http://graficos.prod.uci.cu..

8. **Mora, Adrián Santí.** *Componentes para la elaboración de diagramas de presentación en la herramienta de diagrmación del paquete ABAD.*

9. Acerca de JavaScript. [En línea] 2011. [Citado el: 23 de Noviembre de 2012.] https://developer.mozilla.org/es/docs/JavaScript/Acerca_de_JavaScript.

10. PHP. [En línea] [Citado el: 24 de Noviembre de 2012.] http://hu1.php.net/manual/es/intro-whatcando.php.

11. PHP. [En línea] [Citado el: 24 de Noviembre de 2012.] http://hu1.php.net/manual/es/intro.image.php.

12. **Pérez, Javier Eguíluz.** *Introducción a AJAX.* 2008.

13. librosweb. [En línea] 2013. [Citado el: 30 de Noviembre de 2012.] http://www.librosweb.es/symfony_1_1/capitulo_1/symfony_en_pocas_palabras.html.

14. NETBEANS. [En línea] 2013. [Citado el: 2 de Diciembre de 2012.] https://netbeans.org/features/.

15. **Marquina, Ernesto y Parra, Jose David.** *Guía de Patrones, Prácticas y Arquitectura .NET.* 2008. 2.

16. **Salazar, Lianet Labrada.** *Administrador de Reportes para la versión 2.0 del Generador Dinámico de Reportes.* 2012.

17. **Larman, Craig.** *UML y Patrones Introducción al análisis y diseño orientado a objetos.*

18. **Instituto Nacional de Estadística de Informática.** Inei. [En línea] [Citado el: 9 de Diciembre de 2012.] http://www.inei.gob.pe/biblioineipub/bancopub/Inf/Lib5103/Libro.pdf.

19. freedownloadmanager. [En línea] [Citado el: 15 de Diciembre de 2012.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/.

20. **Hernández, Isneysusana Herrera.** *Sistema de Información para la Gestión del Capital Humano en el Centro de Tecnologías de Gestión de Datos.* 2011.

21. **Popkin Software and Systems.** Modelado de Sistemas com UML. [En línea] [Citado el: 15 de Diciembre de 2012.] http://es.tldp.org/Tutoriales/doc-modelado-sistemas-UML/doc-modelado-sistemas-uml.pdf.

22. **Blanco, Yaneisy González.** *Perfil de UML para los proyectosde la línea Soluciones Integrales.* 2011.

23. Introduction to OpenUP. [En línea] [Citado el: 20 de Diciembre de 2012.] http://www.eclipse.org/epf/general/OpenUP.pdf.

24. **Rosas, Juan Eladio Sánchez.** Desarrollo en Web. [En línea] 22 de Octubre de 2008. [Citado el: 21 de Diciembre de 2012.] http://blogs.antartec.com/desarrolloweb/2008/10/extjs-lo-bueno-lo-malo-y-lo-feo/.

25. **Larman, Craig.** *UML y Patrones 2a Edición.* 2003.

26. **Marleysi López Duque, Raidel Oceguera Ravelo.** *Herramienta en Matlab para la obtención de información de la base de datos y ficheros electroencefalograma del proyecto Mapeo Cerebral Humano Cubano.* 2010.

27. **Mayelin Timoteo Guerra, Ayeris Lizandra Navarro González.** *Plugin de Visual Paradigm for UML para la aplicación del Método de Estimación UCI.* 2011.

28. Casos de Uso (Use Case). [En línea] [Citado el: 10 de Enero de 2013.] http://users.dcc.uchile.cl/~psalinas/uml/casosuso.html.

29. **Rubiano, Martha.** Scribd. [En línea] 2013. [Citado el: 15 de Enero de 2013.] http://es.scribd.com/doc/19808824/diagramas-de-despliegue-2222.

30. gitnacho. [En línea] 2011. [Citado el: 5 de Marzo de 2013.] http://gitnacho.github.io/symfony-docs-es/contributing/code/conventions.html.

31. **Arrieche, Elizabeth.** *Scribd.* [En línea] Enero de 2011. [Citado el: 25 de Marzo de 2013.] http://es.scribd.com/doc/50741997/Pruebas-de-Software..

32. **PRESSMAN, ROGER S.** *Ingenieria de Software Un Enfoque Práctico.*

Bibliografía

1. Acerca de JavaScript. [En línea] 2011. https://developer.mozilla.org/es/docs/JavaScript/Acerca_de_JavaScript.
2. **Alvarez, Miguel Angel.** desarrolloweb. [En línea] http://www.desarrolloweb.com/articulos/392.php.
3. **Arrieche, Elizabeth.** *Scribd.* [En línea] Enero de 2011. http://es.scribd.com/doc/50741997/Pruebas-de-Software.
4. BuenasTareas.com. [En línea] . http://www.buenastareas.com/ensayos/Pruebas-De-Caja-Negra/1477804.html.
5. Carvajal, Juan José Hernández. Desarrollo del módulo Administrador de Reportes del Generador Dinámico de Reportes.2011.
6. Casos de Uso (Use Case). [En línea] http://users.dcc.uchile.cl/~psalinas/uml/casosuso.html.
7. Delgado, Cecilia Milián. Sistema Integrado de Gestión Estadística (SIGE): Componente para la captación de encuestas económicas.2011.
8. Duque, Marleysi López, Ravelo, Raidel Oceguera. Herramienta en Matlab para la obtención de información de la base de datos y ficheros electroencefalograma del proyecto Mapeo Cerebral Humano Cubano.2010.
9. EcuRed. [En línea] EcuRed, http://www.ecured.cu/index.php/UML.
10. EcuRed . [En línea] http://www.ecured.cu/index.php/Javascript.
11. Ecured. [En línea] http://www.ecured.cu/index.php/Patrones_de_dise%C3%B1o_y_arquitectura.
12. EcuRed. [En línea] .http://www.ecured.cu/index.php/Pruebas_de_caja_blanca.
13. freedownloadmanager. [En línea] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/.
14. github [En línea] https://github.com/php-fig/fig-standards/blob/master/accepted/PSR-0.md.
15. gitnacho. [En línea] 2011. http://gitnacho.github.io/symfony-docs-es/contributing/code/conventions.html.
16. Garnache, Alberto Mendoza. Definición de la arquitectura de software del Grupo de Desarrollo para la Gestión de Equipos Médicos. 2009.
17. Hernández Hernández, Yasmany. Pautas de arquitectura para el desarrollo de la versión 2 del GDR.
18. **Instituto Nacional de Estadística de Informática.** Inei. [En línea]

http://www.inei.gob.pe/biblioineipub/bancopub/Inf/Lib5103/Libro.pdf.

19. Introduction to OpenUP. [En línea] http://www.eclipse.org/epf/general/OpenUP.pdf.
20. Laborde, Mabel Valle. Generador Dinámico de Reportes V 2.0: Análisis de los Módulos Diseñador de Modelos y Diseñador de Reportes. 2011.
21. **Larman, Craig.** *UML y Patrones 2a Edición.* 2003.
22. **Larman, Craig.** *UML y Patrones Introducción al análisis y diseño orientado a objetos.*
23. León, Vania Elena Yanes. *Definición de la arquitectura de software para el Generador Dinámico de Reportes en su versión 2.0. 2011.*
24. Lobo, Armando Robert. Arquitectura de Software para el Sistema Integrado de Gestión Estadística 2.0 Nuragas.2009. *Dinámico de Reportes en su versión 2.0.2011.*
25. **Mayelin Timoteo Guerra, Ayeris Lizandra Navarro González.** *Plugin de Visual Paradigm for UML para la aplicación del Método de Estimación UCI.* 2011.
26. Morales, Yanet Rosales. Componente para la captura de datos del Sistema de Información de Gobierno: Módulo Encuestas.2011.
27. Mora, Adrián Santí. *Componentes para la elaboración de diagramas de presentación en la herramienta de diagrmación del paquete ABAD.*
28. NETBEANS. [En línea] 2013. https://netbeans.org/features/.
29. Oré, Ing. Alexander. CalidadySotware.2009. [En línea] http://www.calidadysoftware.com/testing/pruebas_funcionales.php.
30. **Popkin Software and Systems.** Modelado de Sistemas com UML. [En línea] http://es.tldp.org/Tutoriales/doc-modelado-sistemas-UML/doc-modelado-sistemas-uml.pdf.
31. **PRESSMAN, ROGER S.** *Ingenieria de Software Un Enfoque Práctico.*
32. Rumbaugh, Jim, Jacobson, Ivar y Booch, Grady.UML el lenguaje unificado de modelado. http://www.uml.org/.
33. Rodríguez, Ana Niuska Navarro. Asistente de Reportes para el modulo Diseñador de Reportes del Generador Dinámico de Reportes versión 2.0. 2011.
34. **Rubiano, Martha.** Scribd. . [En línea] 2013. http://es.scribd.com/doc/19808824/diagramas-de-despliegue-2222.
35. **Rodríguez, Msc. Dayra Iris Hechavarría.** Gestión de Requisitos. [En línea] 2012. http://www.monografias.com/trabajos92/gestion-requisitos/gestion-requisitos.shtml.
36. **Rosas, Juan Eladio Sánchez.** Desarrollo en Web. [En línea] 22 de Octubre de 2008.

http://blogs.antartec.com/desarrolloweb/2008/10/extjs-lo-bueno-lo-malo-y-lo-feo/.

37. **Salazar, Lianet Labrada.** *Administrador de Reportes para la versión 2.0del Generador Dinámico de Reportes.* 2012.
38. Scribd. [En línea] . http://es.scribd.com/doc/19808824/diagramas-de-despliegue-2222.
39. **Thompson, Ivan.** . *Definición de Información.* [En línea] http://www.promonegocios.net/mercadotecnia/definicion-informacion.html.
40. **Universidad de las Ciencias Informaticas.** Chart Server. [En línea] 2010. http://graficos.prod.uci.cu.

Anexo 1. Diagrama de secuencia del caso de uso Construir_Gráfico sección Manual

En la figura 22 se muestra el diagrama de secuencia del caso de uso Construir_ Gráfico sección Manual.

El sistema muestra un formulario para crear un conjunto de datos manualmente. El usuario introduce los datos, la aplicación los analiza y le envía un mensaje de error al usuario en caso de no ser correctos; en caso contrario el sistema le habilita al usuario la opción “Aceptar”. El usuario da clic en el botón “Aceptar”. La aplicación genera un conjunto de datos aleatorio y muestra al usuario el formulario con el conjunto de datos. Luego de esta secuencia continúan los pasos descritos en el diagrama Construir_Gráfico a partir del mensaje 9.

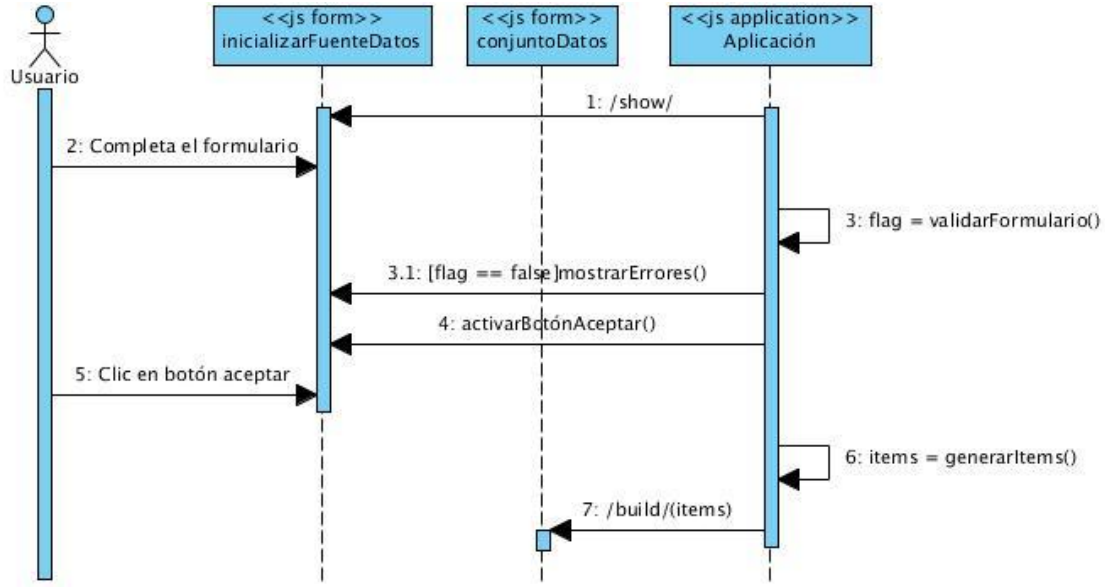


Figura 22 Diagrama de secuencia del caso de uso Construir_Gráfico sección Manual

Anexo 2. Diagrama de secuencia del caso de uso Construir_Gráfico sección Excel

En la figura 23 se muestra el diagrama de secuencia caso de uso Construir_Gráfico sección Excel.

El sistema muestra un formulario para escoger el fichero. El usuario escoge el fichero, la aplicación valida la extensión y en caso de ser correcta activa el botón “Aceptar”. El usuario da clic en el botón “Aceptar”. El servidor recoge los datos del fichero y devuelve el conjunto de datos obtenido o un mensaje de error. En caso de no existir errores muestra al usuario el formulario con el conjunto de datos enviado por el servidor. Luego de esta secuencia continúan los pasos descritos en el diagrama Construir_Gráfico a partir del mensaje 9.

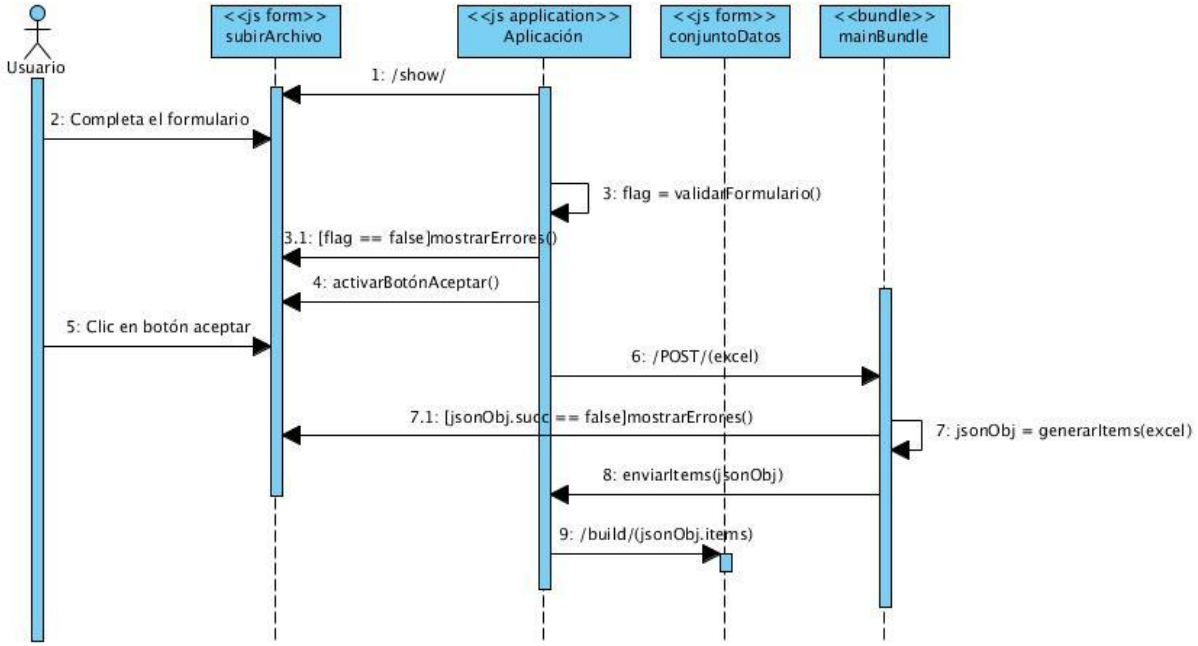


Figura 23 Diagrama de secuencia del caso de uso Construir_ Gráfico sección Excel

Anexo 3. Diagrama de secuencia del caso de uso Construir_Gráfico sección Base de Datos

En la figura 24 se muestra el diagrama de secuencia del caso de uso Construir_Gráfico sección Base de Datos.

El sistema muestra un formulario para realizar una conexión. El usuario introduce los datos necesarios para establecer una conexión al servidor de Base de Datos, la aplicación los analiza y le envía un mensaje de error al usuario en caso de no ser correctos; en caso contrario el sistema le habilita al usuario la opción “Siguiente”. El sistema realiza la conexión y en caso de ser satisfactoria muestra un formulario para insertar la consulta sql. El usuario introduce la consulta y da clic en el botón “Finalizar”. El sistema realiza la consulta y devuelve el conjunto de datos o un mensaje de error. En caso de no existir errores muestra al usuario el formulario con el conjunto de datos enviado por el servidor. Luego de esta secuencia continúan los pasos descritos en el diagrama Construir_Gráfico a partir del mensaje 9.

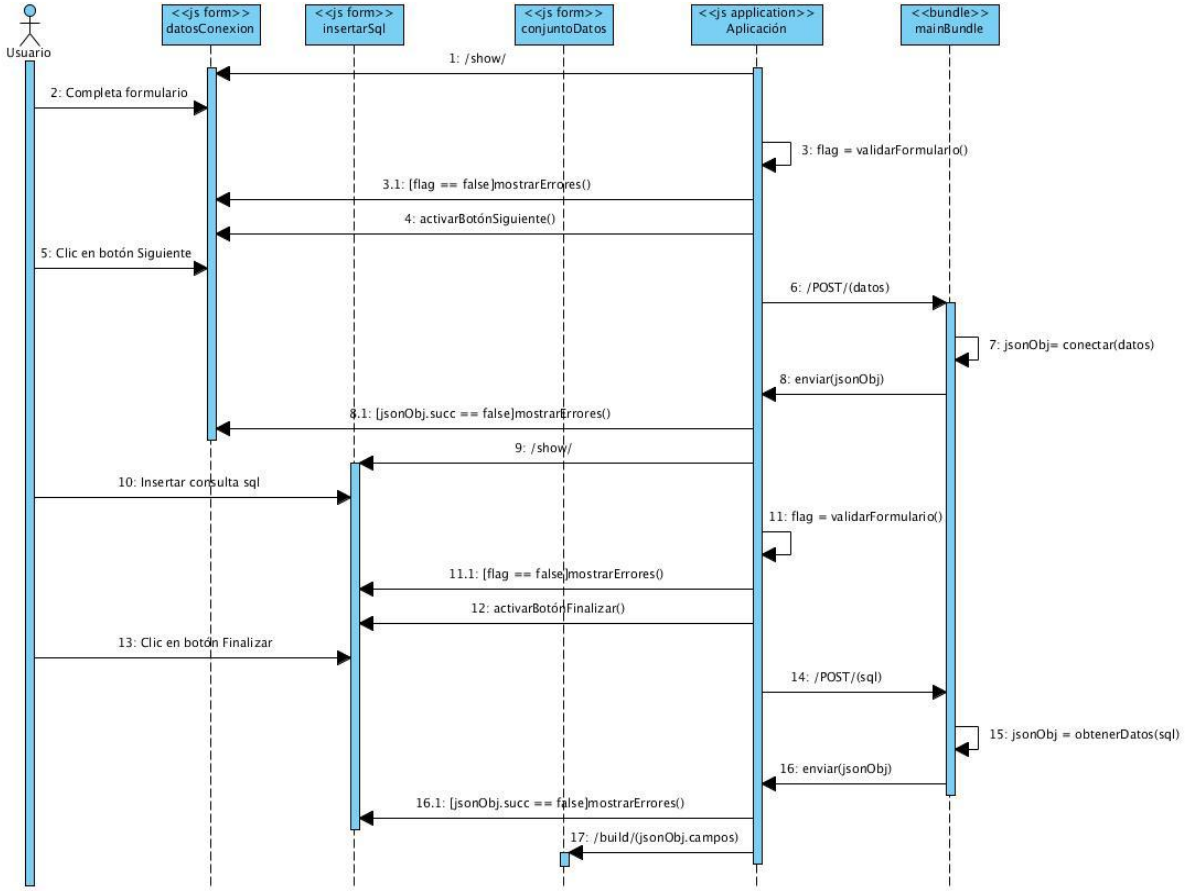


Figura 24 Diagrama de secuencia del caso de uso Construir_Gráfico sección BD

Anexo 4. Diagrama de secuencia del caso de uso Construir_Gráfico sección Exportar conjunto de datos

En la figura 25 se muestra el diagrama de secuencia del caso de uso Construir_ Gráfico sección Exportar conjunto de datos.

El sistema muestra un formulario para exportar el conjunto de datos. La aplicación envía al servidor el conjunto de datos y genera un fichero con dicho conjunto de datos. El servidor devuelve el fichero creado y el usuario guarda el fichero.

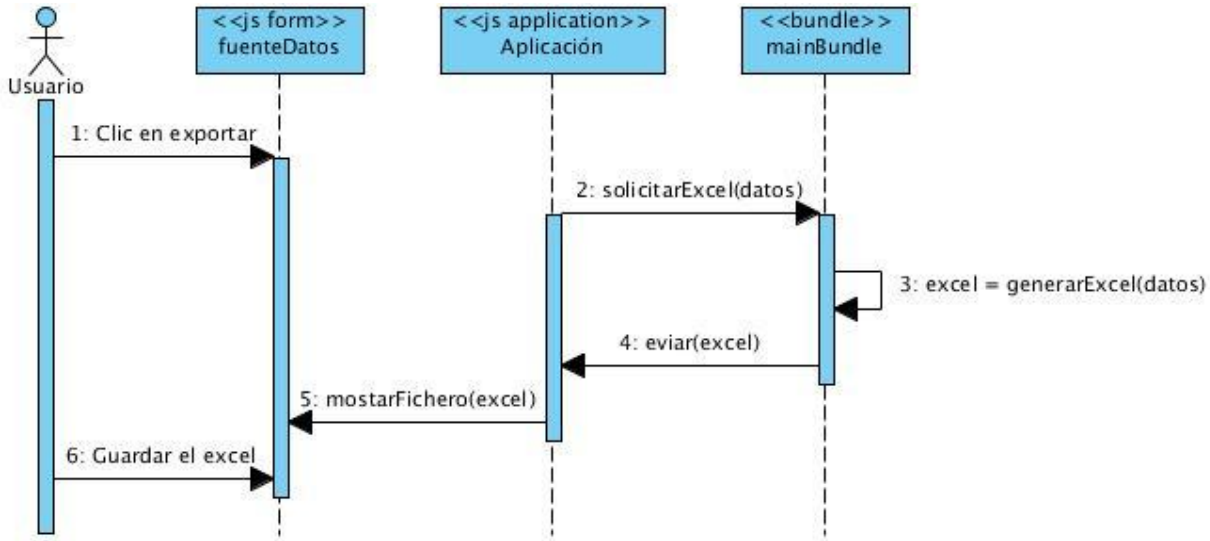


Figura 25 Diagrama de secuencia del caso de uso Construir_Gráfico sección Exportar conjunto de datos