DISEÑO DE ESTRUCTURAS DINÁMICAS PARA OFRECER GRÁFICOS Y PANTALLAS EMERGENTES EN ENTORNO WEB

J. Armando Millas Larios

Marcos S. Cil González

Álvaro Rodríguez Marín

PROYECTO

DE SISTEMAS INFORMÁTICOS DE INGENIERÍA EN INFORMÁTICA

UNIVERSIDAD COMPLUTENSE DE MADRID

DEPARTAMENTO DE SISTEMAS INFORMATICOS Y COMPUTACIÓN

DIRECTOR: Miguel Ángel Blanco Rodríguez

Junio 2014, Madrid

AUTORIZACIÓN

Los autores de este proyecto autorizan a la Universidad Complutense de Madrid a difundir a utilizar el presente trabajo de investigación hasta el punto actual de desarrollo, tanto en la aplicación como la memoria únicamente con fines académicos, no comerciales y mencionando expresamente a sus autores. También autorizan a la Biblioteca de la UCM a depositar el trabajo en el Archivo Institucional E-Prints Complutense.

AUTORES

J. Armando Millas Larios

Marcos S. Cil González

Álvaro Rodríguez Marín

FECHA

20/06/2014

AGRADECIMIENTOS

Queremos agradecer este trabajo a nuestras familias y amigos que nos han apoyado en los momentos más difíciles y de mayor esfuerzo a lo largo de la carrera.

Por otra parte, mencionar a los distintos profesores que nos han impartido las distintas asignaturas que nos han hecho formarnos de manera académica y personal.

Finalmente, damos las gracias a nuestro director de proyecto, Miguel Ángel Blanco Rodríguez sin cuya orientación este trabajo no habría sido posible.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE ABREVIATURAS

RESUMEN

La finalidad de nuestro proyecto ha sido la creación de un sistema que permite desarrollar de forma simple sobre él una aplicación basada en web con carga importante de base de datos. La funcionalidad de este sistema se ha enfocado en ofrecer capacidad automatizada para mostrar interfaces web y gráficos que interaccionan y recogen la información de una base de datos, así como son capaces de modificarla. Por lo tanto, nuestro desarrollo está orientado al tratamiento de la base de datos de la aplicación que se quiera construir, de forma que toda la funcionalidad gire en torno a ésta. Para ello, se requiere la configuración en nuestro sistema de las acciones que se deseen permitir en la aplicación que se va a crear. En concreto, estas acciones tienen la siguiente estructura: mediante un formulario generado automáticamente por el sistema, el usuario introduce los parámetros; a continuación, el sistema ejecuta la acción sobre la base de datos; por último, los resultados de la ejecución son mostrados en forma de tabla o gráfico al usuario. Además, ofrecemos distinción entre usuarios para que ciertos usuarios puedan realizar solo ciertas acciones.

La intención a lo largo de todo el desarrollo ha sido siempre permitir la mayor flexibilidad posible, de modo que se pudiese realizar el mayor número posible de acciones distintas, formas de representación, etc. Aunque el estado actual del proyecto no permite toda la funcionalidad que desearíamos, el haber sido pensado con la flexibilidad en mente permite que se pueda extender de forma natural en la mayoría de los aspectos que aún no están incluidos. También se aplica, a la hora de hablar de la flexibilidad, la intención de reusabilidad que hemos pretendido otorgar a todos los elementos del sistema, especialmente el uso de una misma Acción para varios objetivos en función de sus parámetros y otros aspectos de la configuración. Por ejemplo, permitir usar una misma acción para mostrar un gráfico o una tabla o crear una nueva acción a partir de otra solamente agregando un parámetro, sin tener que configurar una acción desde el principio en ninguno de los dos casos.

El proyecto tiene aún por delante una fase de extensión pues la complejidad en la configuración de las acciones se ha demostrado que es excesiva. En esta fase, se propone mejorar ese aspecto principalmente para hacer un sistema realmente útil a la hora de crear aplicaciones web.

PALABRAS CLAVE

Base de datos, Aplicación web, Interfaz automática, Configuración de funcionalidad, Flexibilidad

ABSTRACT

The main objective of this project is to create a system that allows developing in a simply way a web application by using numerous databases as the main focus. The functionality of this system has been focused on offering an automatized capacity to show multiple interfaces' web. Thanks to these webs, there are show graphics that interact and gather information from a database as well as they modify it.
Therefore, our development is aimed at treating the database application that wants to be built so that all the functionality rotates around it. In order to accomplish this, a configuration of the numerous actions that allow the different uses of the application that will be created in our system is required. Specifically, these actions have the following structures: a form will be created in an automatic way in which from than point on the user should introduce the corresponding parameters. Afterwards, the system will run the actions to the database. Finally, the results made by the run will be shown to the user in a graphic organizer or table. There is the possibility to distinguish among various types of users in order to limit the different actions of each type of user.
As we have stated before, the aim of this project during all its development has been to allow the best available flexibility so that different actions or ways of representation can be achieved. Although we know that the current functionality is limited, due to the high flexibility in the project developed, the potential extensibility of most of the aspects not included will take place naturally. The flexibility mentioned before will also allow the reutilize of the elements of the system; especially the use of the same action for various purposes depending on its parameters and other aspects of the configuration. For example, allowing the use of the same action to display a graph or a table. This procedure also allows the use of a single action to create a new action from another simpler action by just adding a parameter, without preconditioning in either cases.
Due to the extent complexity of these configurations of the actions, our project has a prior extension’s phase very important. During this phase, it is aimed to improve that aspect mainly to make a really useful system when creating web applications.

KEYWORDS

Database, Web application, Automatic interface, Funcionality configuration, Flexibility

CAPÍTULO 1. INTRODUCCIÓN

MOTIVACIÓN

A lo largo de la historia, la sociedad ha recurrido a diversos sistemas de gestión de almacenamiento de datos hasta la aparición de las bases de datos en 1963.Una base de datos, desde el punto de vista informático, es un sistema formado por un conjunto de datos almacenados en discos que permite el acceso directo a ellos y un conjunto de programas que manipulen ese conjunto de datos. Hoy en día, casi cualquier aplicación tiene como base de su funcionamiento una base de datos, sin irnos más lejos, no podríamos acceder a cualquier sistema de reserva de hoteles, recibir clases a distancia, cualquier sistema de venta online. [3]

Funcionamiento

No es tan simple como colocar la base de datos en un servidor web y ya está. Todo empieza cuando el servidor web recibe una petición a través del navegador. El servidor web no puede acceder directamente al contenido de la bases de datos, si no que tiene que acceder al servidor de bases de datos, el cual recibe una petición del servidor web y la ejecuta. El servidor de bases de datos devuelve al servidor web los datos pedidos, junto con un código de error. El servidor web recibe los datos del servidor de bases de datos y los procesa. El servidor web envía el resultado de su procesamiento al navegador en un formato de página web conocido. [4]

Ventajas de las bases de datos en entorno web y donde se usan

- Disponibilidad Temporal y Geográfica con una interfaz común
- Posiblidad de interactuar con el contenido de una base de datos en un navegador web

Esto, junto con la intención de desarrollar una aplicación que nos muestre la información de una base de datos, hace de nuestro proyecto una herramienta útil que ocuparía este vacío existente de aplicaciones web para obtener la información de una base de datos.

El proyecto tiene el objetivo de desarrollar una aplicación web que ofrezca una solución para mostrar elementos emergentes con el contenido de una base de datos.

La aplicación en sí consta de una única página web dinámica que muestra una serie de menús variables en función del usuario activo. A través de las opciones disponibles en dichos menús, se despliegan una serie de cuadros de texto con los datos de entrada necesarios en cada caso. Y a partir de estos, se muestra una ventana con el resultado de la acción seleccionada.

ESTADO DEL ARTE

Dos de los pilares de nuestro proyecto, orientación a base de datos e implementación web, son ampliamente usados hoy en día por desarrollos de todo tipo. Además, están íntimamente relacionados de manera que la gran mayoría de webs hace uso de una base de datos para gestionar toda la información que manejan. Nuestro sistema se basa en esta realidad, y la amplifica de forma que la unión entre web y base de datos ya venga implícita en la forma de desarrollar una aplicación. El objetivo es el de facilitar este desarrollo web, en nuestro caso abstrayendo la unión entre base de datos e interfaz, por lo tanto, entra en la categoría de sistemas pensados para que se implemente sobre ellos. Actualmente existen gran diversidad de tecnologías cuya finalidad es la de facilitar la creación de cualquier tipo de aplicación sobre ellas. En esta, caen todos los motores, frameworks o editores. Podemos llegar al extremo de incluir los lenguajes de programación de alto nivel (es decir sus compiladores o interpretes) en esta categoría, pero al margen de esto, en este tipo de categorización nos estamos refiriendo a tecnologías que suelen ser más especializados, por ejemplo: Unity (motor de videojuegos) [16].

Cabe diferenciar otra categoría de programas que, aunque su objetivo es ayudar al diseñador/programador, no se desarrolla con su tecnología si no por medio de ellos mismos. A esta pertenecería Eclipse, por ejemplo. Diferenciamos por tanto que estos programas ofrecen facilidades, pero el desarrollo final (el código en el caso de Eclipse), sería el mismo que si se hubiese hecho sin este tipo de aplicaciones.

Una vez hecha esta categorización, nos vamos a centrar en aquellas que están especializadas en web y bases de datos, pues son las que comparten objetivo con nuestro proyecto. Entre estas, las más básicas son los editores de páginas web, como por ejemplo FrontPage (que ya está fuera de producción) [17]. Este tipo de programas están pensados para crear visualmente páginas web y que sea el programa el que traduzca a HTML (o JavaScript) la visualización que se ha diseñado. En general no ofrecen más que eso, páginas web de cara al cliente, es decir, que son directamente renderizadas por el navegador, sin entrar en procesado por parte del servidor ni acceso a bases de datos.

Unas soluciones más actuales y completas son las que ofrecen todos los sistemas denominados Sistemas de Gestion de Contenidos, CMS (Content Management System). En este caso, estamos hablando de plataformas con las que se pueden crear todo tipo de páginas web, y que cuentan con multitud de módulos para distintos propósitos. Gracias

a estos, se puede llegar a implementar una solución web sin necesidad de llegar a escribir una sola línea de código. Para el desarrollo de los propios módulos sí que se necesita programar. Los CMS sí que trabajan con bases de datos, aunque suele ser una base de datos propia en la que se almacenan los contenidos que se van creando para la página. No suelen ser sistemas pensados por tanto para unir con una base de datos ya existente. La limitación de los CMS está en lo que ofrecen sus módulos y por tanto no se suelen usar para proyectos que requieran innovación en la funcionalidad que ofrecen (Por ejemplo no están pensados para ser la plataforma en la que programar un videojuego para web). Además tampoco se suelen usar en proyectos de gran tamaño. Sus ventajas ya las hemos nombrado: incluso personas no iniciadas en la programación pueden crear un sitio web con ellos y facilitan una estructura robusta y probada (por toda la gran cantidad de webs que hacen uso de estos sistmeas) para empezar a desarrollar a los que sí quieren programar más allá de lo ofrecido en los módulos existentes.

Ejemplos de CMS son WordPress [18], Joomla [19] o Drupal [20]; estos tres están enfocados a la creación de webs de propósito general. Además existen multitud de CMS más especializados, por ejemplo en vBulletin [21] para la creación de foros, BoxBilling [22] para tiendas online, etc.

En el campo del desarrollo web, como en casi todos los relacionados con la informática, tenemos también los denominados "frameworks" y "APIs". Estos son conjuntos de librerías (nos referimos a código utilizable por un programador) que proporcionan las funciones básicas y las más repetidas a la hora de desarrollar una web. Normalmente el uso de un framework o API implica que el desarrollo se haga sobre él será en el mismo lenguaje que en el que el framework está construido. En aquellas que se usan para el código de lado del cliente (interpretado por el navegador), esto no causa mayor impacto pues HTML y JavaScript son los estándares usados y no existe alternativa. Sin embargo del lado del cliente, antes de elegir un framework se debe determinar con que lenguaje (PHP, Java, Phyton, etc) se va a realizar el proyecto.

El término "framework" se suele aplicar más a los sistemas que proporcionan cobertura para un proyecto en general. De esta manera un framework actúa con el modelo Inversión de Control, es decir, que el framework proporciona una estructura de programa (más o menos flexible) y es el código del framework el que llama al código de desarrollado específico para la aplicación. De esta manera se promueven buenas prácticas de programación y se proporciona una solución completa para la aplicación [23].

Por otro lado las "APIs" o "librerías" se utiliza comúnmente para las aplicaciones que resuelven un problema más concreto en el desarrollo. El término "APIs" quiere decir en

realidad Interfaz de Programación de Aplicaciones ("Aplication Programming Interface") lo cual hace referencia a las funciones que ofrecen una librería. De forma correcta se debería hablar del API de una librería, y el sistema que proporciona la funcionalidad sería la propia librería. Sin embargo se ha extendido el significado de la palabra y muchas veces de usa para denominar a la propia librería. Además dado su significado real, también hablamos de APIs para llamar a los puntos de unión que ofrece un sistema mayor que permite su uso incrustado en otras páginas. Con esto nos referimos por ejemplo al API que proporciona Facebook que ofrece funcionalidad entre otras muchas para sus botones "me gusta" y la capacidad de hacer login con la cuenta de su sistema [24]. Google también ofrece APIs de este tipo, siendo especialmente interesante la de GoogleMaps [25].

En el sentido de librerías, existen para miles de propósitos distintos. Nosotros mismos hemos hecho uso de dos APIs para el desarrollo de nuestro proyecto Highcharts para gráficas y Dhtmlx para interfaces, las cuales describiremos con más detalle en el próximo capítulo. En la programación del lado del cliente, una de la liberías más usadas es JQuery, que proporciona una forma más simple de programar sobre JavaScript, siendo a veces discutido si constituye un framework o no dado que abarca en sus mejoras gran cantidad de los objetivos de JavaScript. Nosotros hemos optado por incluirlo como librería por no cumplir el modelo de Inversion de Control [26]. Otras librerías que ofrecen funcionalidad interesante son Three.js para la generación de gráficos 3D o Box2dWeb para la creación de juegos de navegador.

Desde el lado del servidor, ponemos como ejemplo la libería cURL [27], para PHP que se usa para transferir datos en varios protocolos, y aplicandolo a web en Http. También se puede considerar de esta forma a las funciones que ofrecen los servidores Apache o Apache Tomcat, (para programación en PHP y Java respectivamente) que dan acceso a la funcionalidad que permiten como servidor web más allá del propio procesado del código en servidor [28].

Al hablar de frameworks, como ya hemos dicho, tenemos que hablar de lenguajes de programación web de cara a servidor. En este grupo podemos encontrar también multitud de propuestas distintas, llegando a haber más de una para cada lenguaje de desarrollo web. Por ejemplo nos encontramos con Ruby on Rails para el lenguaje Ruby [29]; Symfony [30], Yii [31] y CodeIgniter [32] para PHP ; Spring [33] para Java (JSP y Servlets) y Django [34] y Pylons [35] para Phyton por poner algunos ejemplos. Entre ellas se diferencian por su nivel de complejidad (que influye en la curva de aprendizaje del framework) y su potencia, siendo normalmente directamente proporcionales. Las que se denominan como "de uso empresariales" suelen ser los más completos y más complejos y están pensados para proyectos de gran tamaño (El mejor ejemplo de esto es Symfony que incluso cuenta con una versión reducida para proyectos más simples llamada Silex [36]).

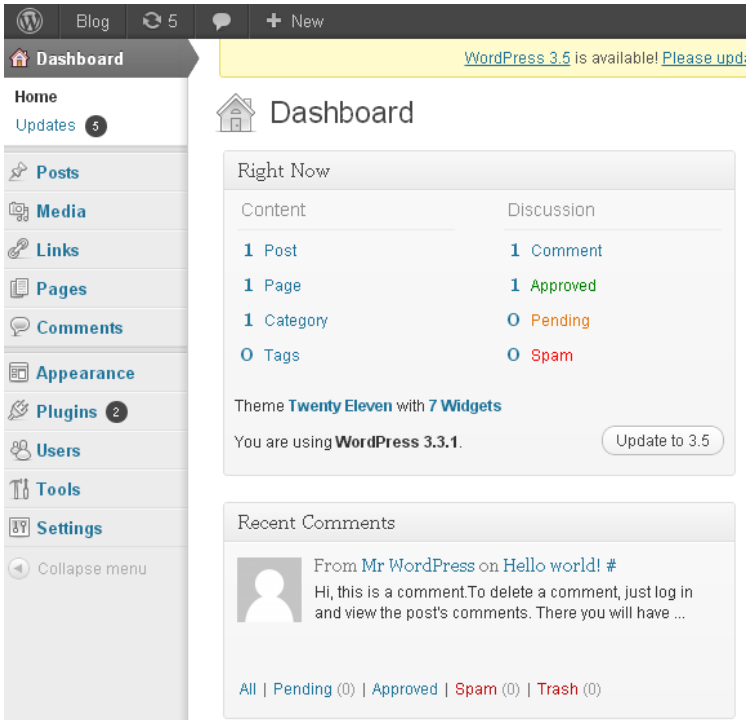

ILUSTRACIÓN 1: PANEL DE CONTROL DEL CMS WORDPRESS

Con respecto a bases de datos, muchos de los lenguajes como PHP proporcionan nativamente formas de conectar con ellas. No obstante los frameworks simplifican esto incluyendo como parte del desarrollo la conexión con los datos. Además también existen liberías con este mismo objetivo, el simplificar la obtención y manejo de los datos almacenados. La más curiosa por su planteamiento es Hibernate [37], para Java, la cual hace una mapeo directo entre tablas y objetos del programa. Otro de los propósitos que cumplen este tipo de sistemas es el de abstraer el tipo base de datos (MySQL, Oracle, etc) de la implementación dada en el sistema.

Estos son las alternativas más comunes que tenemos a la hora de embarcarnos en un desarrollo web, (si no queremos partir de cero por supuesto). Además vamos a hablar de ciertos programas que también tienen cabida por compartir características con nuestro sistema.

En primer lugar queremos destacar un programa que siendo web y enfocado a base de datos, que sin embargo caería en la categoría que hemos descrito antes de programas que ayudan pero no permiten desarrollar sobre ellos. Este es el phpMyAdmin [38], un gestor de bases de datos online.

Otro tipo de programa que queremos comentar son los llamados ERP, sistemas de planificación de recursos empresariales (“ Enterprise Resource Planning”). Estos se proporcionan como una solución completa para todas las necesidades administrativas que una empresa pueda necesitar. Son usados principalmente por empresas de tamaño mediano a grande. La realidad es que dado la enorme diversidad de necesidades que una empresa puede requerir, estos sistemas permiten una gran flexibilidad y se pueden englobar en la categorización de plataforma sobre la que desarrollar una solución concreta que hemos hecho. Además están muy enfocados a bases de datos dado la gran cantidad de información que una empresa grande maneja. Por tanto estas plataformas permiten la extensión a los requisitos especificados ya sea usando programación o sin usarla por medio de programas anexos al ERP.

Por último, aunque estos programas existen desde los años 80, actualmente se está efectuando (o se ha efectuado ya, dependiendo del programa concreto del que hablemos) una migración a entorno web en la interfaz de todos ellos. Los ejemplos más famosos para este tipo de aplicaciones son SAP y Oracle EBS [39].

A continuación, podemos identificar que puntos en común comparte nuestro sistema con todas estas alternativas ofrecidas. Nuestra idea desde el principio fue proporcionar un sistema capaz de ser usado sin demasiados conocimientos del funcionamiento de la web. En ese aspecto, nuestro proyecto toma el espíritu de los CMS, no solo por la simplicidad de la implantación sino también por la capacidad de agregar nuevos módulos que hagan cosas nuevas dentro del propio entorno del sistema. Sin embargo nuestro punto de vista con respecto al uso de la base de datos y el cumplimiento de requisitos no se parece tanto al de los CMS y se aproxima más al enfoque de un ERP. Aunque nosotros no proporcionamos ninguna funcionalidad ya prevista para las finanzas, los requisitos en nuestro sistema giran entorno a la base de datos tal y como lo hacen en un ERP y el apartado visual, que es uno de los puntos fuertes de los CMS queda bastante al margen. Por último, nuestro planteamiento se aleja del de frameworks y librerías pues en estos, como hemos visto, es imprescindible el uso de código en un lenguaje específico. Nuestro sistema trata de abstraerse de eso y funciona con configuración directa sobre la propia base de datos del sistema. No obstante, al igual que en el caso de los modulo de CMS, ciertos aspectos también tienen que ser programados para agregar nuevos aspectos al proyecto.

Por último hay un tercer aspecto, además de base de datos e implementación web, en el que nuestro proyecto destaca. Esto es el enfoque que tiene desde el principio a poder mostrar la información en gráficas. Como ya hemos visto existen muchas librerías y sistemas que proporcionan gráficas para entornos webs, entre ellas Highcharts la cual hemos usado nosotros. También existen programas para la generación de gráficas fuera de las páginas web. Sin ir más lejos la famosa suit ofimática de Microsoft, Word, ofrece la capacidad de generarlos e incrustarlos en cualquiera de los documentos que es capaz de editar, y teniendo como origen de datos una base de datos Access o una tabla de

Excel. Esta capacidad no es exclusiva de Word y cualquier suit ofimática también la tiene (como OpenOffice en el mundo del software libre). Los CMS y los ERP de los que hablábamos antes, tienen también la capacidad para producir gráficos. Los primeros, mediante módulos, con el origen de datos siendo los propios contenidos de la web o un origen estático. Los segundos ofrecen esta capacidad en los programas adjuntos encargados de visualización de datos, (como Oracle Reports), siendo el origen de los datos la base de datos del sistema. En este punto también se asemejan los ERP a nuestro desarrollo, en el que la capacidad para gráficas está orientada a los datos contenidos en la base de datos cliente.

CAPÍTULO 2. ENTORNO DE DESARROLLO

TECNOLOGÍAS USADAS

DHTMLX

Es una librería de JavaScript utilizada para la construcción de interfaces de aplicaciones webs dinámicas. Gracias a su arquitectura modular,los distintos componentes de construcción se separan en distintos bloques(tablas, árboles,barra de herramientas, menú, calendario,etc) con diferentes funcionalidades pudiendo combinarlos en una interfaz basada en Ajax. La librería se distribuye bajo la licencia GNU y licencias comerciales. [7]

Las principales características son:

-Estructura modular: Cada uno de los componentes son independientes y autosuficientes de manera que se pueden utilizar de forma individual para asegurar una parte específica de la aplicación, por ejemplo, el menú de navegación,datagrid o selector de fecha.

-Funcionalidad tipo escritorio: Arrastrar y soltrar, edición online, validación de datos, interactividad a través de Ajax. El comportamiento de los componentes puede ser similar a los elementos de la interfaz de usuario estándar de Windows.

- Comunicación cliente-servidor: El único requisito es la gestión correcta de los componentes(XML,JSON,SCV,etc.).Compatible con cualquier lenguaje base de datos y está preparado para administrar la comunicación entre cliente y servidor con PHP, .NET , Java.

-Visual Designer: Permite a los desarrolladores construir las interfaces de la aplicaciones web en un entorno visual,es decir, sin código.

-Skin Builder: El aspecto de la interfaz es totalmente personalizable gracias a Internet SkinBuilder, que genera los CSS , archivos e imágenes necesarios para proporcionar el tema elegido por los desarrolladores.

-Cross-browser : Se pueden construir interfaces web basadas en Ajax compatibles con todos los navegadores modernos (Mozilla Firefox,Chrome,Internet Explorer, Opera y Safari).

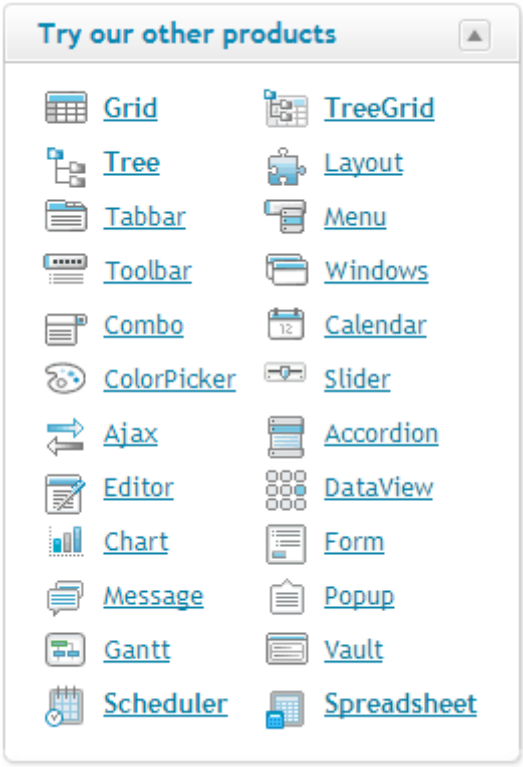

ILUSTRACIÓN 2 : DISTINTOS MÓDULOS OFRECIDOS POR DHTMLX.

JAVASCRIPT

JavaScript es el lenguaje interpretado orientado a objetos desarrollado por Netscape que se utiliza en millones de páginas web y aplicaciones de servidor en todo el mundo. JavaScript de Netscape es un superconjunto del lenguaje de scripts estándar de la edición de ECMA-262 3 (ECMAScript) que presenta sólo leves diferencias respecto a la norma publicada.

Contrariamente a la falsa idea popular, JavaScript no es "Java interpretativo". En pocas palabras, JavaScript es un lenguaje de programación dinámico que soporta construcción de objetos basado en prototipos. La sintaxis básica es similar a Java y C++ con la intención de reducir el número de nuevos conceptos necesarios para aprender el lenguaje. Las construcciones del lenguaje, tales como sentencias if, y bucles for y while, y bloques switch y try ... catch funcionan de la misma manera que en estos lenguajes (o casi).

JavaScript puede funcionar como lenguaje procedimental y como lenguaje orientado a objetos. Los objetos se crean programáticamente añadiendo métodos y propiedades a lo que de otra forma serían objetos vacíos en tiempo de ejecución, en contraposición a las definiciones sintácticas de clases comunes en los lenguajes compilados como C++ y Java. Una vez se ha construido un objeto, puede usarse como modelo (o prototipo) para crear objetos similares.

Las capacidades dinámicas de JavaScript incluyen construcción de objetos en tiempo de ejecución, listas variables de parámetros, variables que pueden contener funciones, creación de scripts dinámicos (mediante eval), introspección de objetos (mediante for ... in), y recuperación de código fuente (los programas de JavaScript pueden decompilar el cuerpo de funciones a su código fuente original).

Los objectos intrínsecos son Number, String, Boolean, Date, RegExp y Math. [8]

JSON

JSON(JavaScript Object Notation) es un formato de intercambio de datos de pequeña capacidad. Su análisis resulta sencillo para las máquinas y su manejo tanto de lectura y escritura es muy fácil. Es completamente independiente pero usa convenciones propias de lenguajes como C,C++,C#,Java,JavaScript,etc.Estas propiedades hacen que JSON sea el lenguaje de intercambio de datos ideal. [9]

Está construido sobre dos estructuras:

- Una conjunto de pares nombre/valor que pueden ser un objeto,registro,tabla hash.
- Una lista ordenada de valores que pueden ser un array, un vector, una lista o una secuencia.

Estos son estructuras de datos universales que son soportados por casi todos los lenguajes de programación modernos . Esto hace que el formato de los datos sea intercambiable entre los lenguajes que están basados en dichas estructuras.

En JSON,dichas estructuras tienen el siguiente formato:

Un objeto es un conjunto desornado de pares nombre/valor. Comienza con '{' y termina con '}'. Cada nombre va precedido de ':' y los pares de nombre/valor separados por comas.

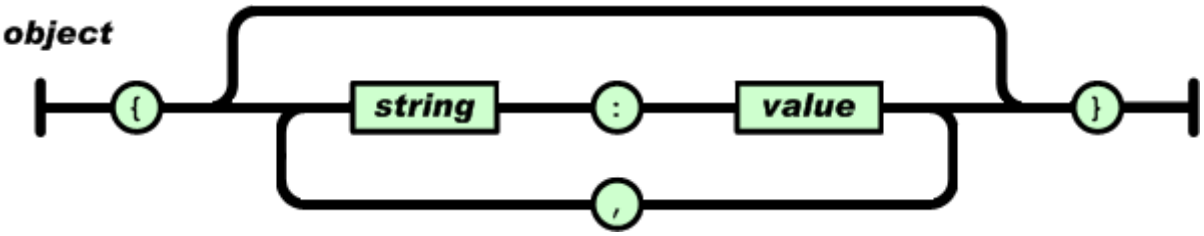

ILUSTRACIÓN 3

Un array es un conjunto ordenado de valores.Comienza con un '[' y termina con ']' con los valores separados por comas.

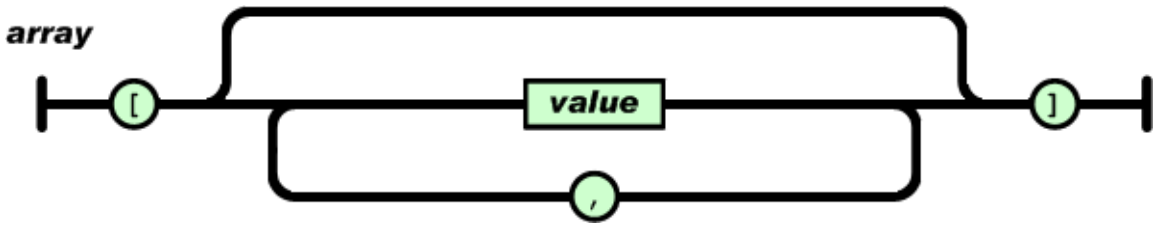

ILUSTRACIÓN 4

Un valor puede ser un string entre comillas dobles, o un número, o un booleano o un objeto o un array. Estas estructuras pueden estar anidadas.

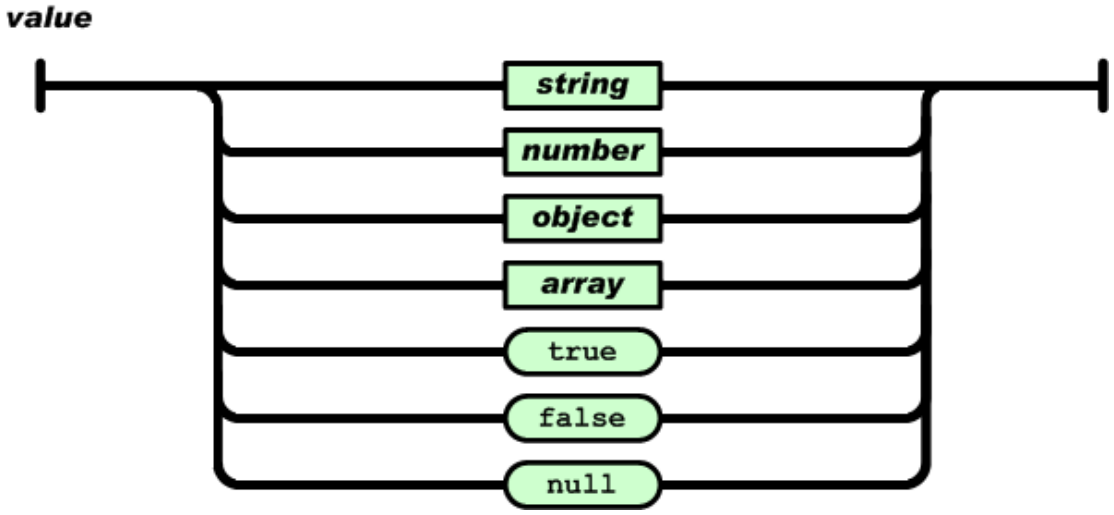

ILUSTRACIÓN 5

Un número sería muy parecido a un número de Java o C, salvo que los formatos octal y hexadecimal no se usan.

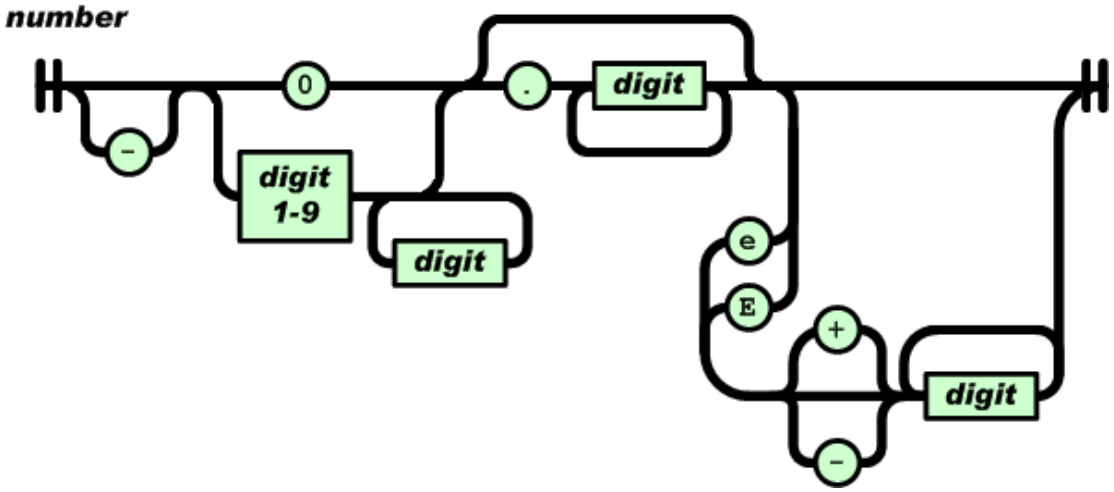

ILUSTRACIÓN 6

Un string es una secuencia de ceros o caracteres Unicote, entre comillas dobles. Un caracter es muy parecido a un string de Java o C

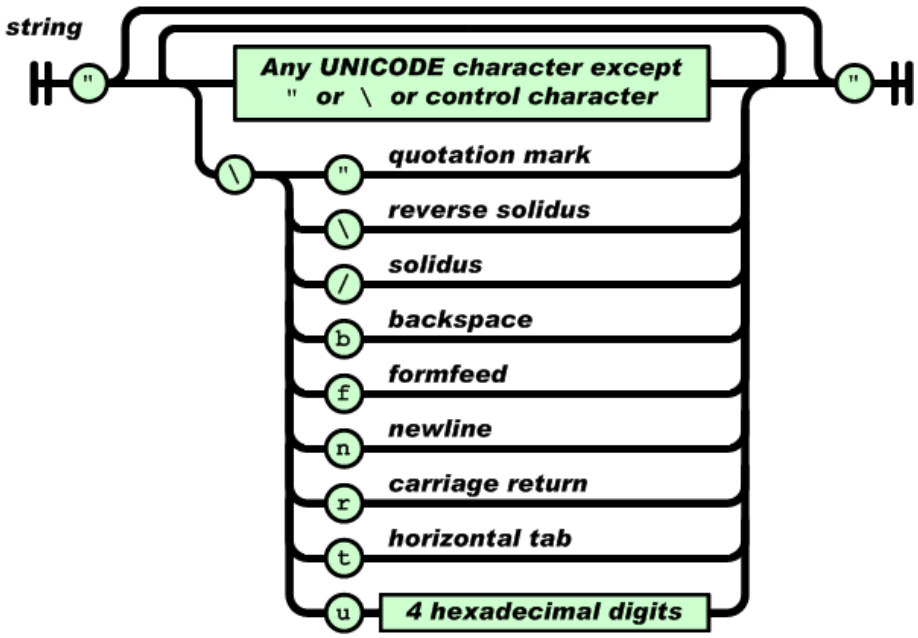

ILUSTRACIÓN 7

HIGHCHARTS

HighCharts es una librería escrita en JavaScript que ofrece de manera sencilla la posibilidad de añadir gráficos interactivos a tu aplicación web. [10]

Características

- Compatible con todos los navegadores modernos tanto de dispositivos móviles como fijos.

- Una de las características clave es que HighCharts te permite acceder al codigo fuente y editarlo tanto si te encuentras en una licencia gratuita como si no. Esto permite una gran flexibilidad y facilidad a la hora de las posibles modificaciones.

- Está basado únicamente en tecnologías de buscadores nativos y no requiere plugins como Flash o Java. Necesita dos archivos JS para ejecutarse: highcharts.js core and cualquiera de jQuery, MooTools o Prototype framework.

- Soporta numerosos tipos de gráficos. Muchos de estos pueden ser combinados en un único gráfico.

- La sintaxis sencilla no requiere habilidades de programación especiales.

- A través de la API puedes añadir,eliminar o modificar series de puntos o ejes incluso después de la creación del gráfico. En combinación con jQuery,MooTools o la API Prototype's Ajax, se consiguen gráficos con actualización constante con values del servidor, suministrados por el usuario…

- A veces, se quiere comparar variables que no estan en la misma escala,highcharts te permite asignar multiples ejes para comparar distintos tipos de variables.

- Al pasar el ratón sobre un punto de la gráfica se tiene la opción de incorporar una descripción emergente con información sobre dicho punto.

- Puedes exportar tu gráfico a diferentes formatos como PNG,JPG,PDF o SVG y imprimirlos directamente desde la página web.

- Highcharts coge los datos externos en un array de JavaScript. los cuales pueden ser redefinidos en un objeto local, en un archivo independiente o en otro lugar.Además, los datos puede ser manipulados de cualquier forma y las llamadas a las funciones suelen devolver los datos en un array.

- Los gráficos con ejes cartesianos puede ser convertidos a gráficos en polares o radianes fácilmente.

- También nos da la opción de revertir los ejes, ante la necesidad de que el eje X aparezca de manera vertical,como por ejemplo, en un gráfico de barras. Y todas las etiquetas pueden rotar en cualquier ángulo.

MYSQL

MySQL con unos 100 millones de copias es el mayor gestor de bases de datos de código abierto. Con su gran velocidad y facilidad de uso ha llegado a ser la principal elección para apalicaciones web,web 2.0 porque elimina la mayoria de problemas relacionados con la inactividad,mantenimiento y administración de aplicaciones modernas.

Es utilizado por las empresas más importantes para conseguir un gran ahorro económico y energético.

Su principal programa es MySQL Enterprise, que consiste en un conjunto de pruebas software, herramientas de monitorización y un gran soporte Premium disponible a través suscripción anual muy asequible.

Es también clave para LAMP el sistema de infraestructura de Internet formado por Linux,Apache,Perl,PHP o Python. [11]

Los objetivos de MySQL son:

-Ser el mejor y más usado gestor de bases de datos en el mundo de las apliaciones online.

-Disponible y asequible para todos.

-Gran facilidad de uso.

-Estar en una mejora constante sin perder la rapidez y la seguridad.

-Sin errores.

-Adentrar a las personas en la filosofía del código abierto.

AJAX

Usando JavaScript,un página en HTML puede asincronamente realizar peticiones al servidor cargando el contenido de los documentos en XML,HTML,JSON o texto plano.La tecnología de JavaScript puede usar el contenido para actualizar o modificar el Documento Objeto Modelo (DOM).

El término Asynchronous JavaScript Technology and XML(Ajax) ha emergido recientemente para describir este modelo de interacción.

Lo que hace a Ajax único es que el cliente tiene el control específico del contenido de la página. La página interactúa gracias a la tecnología JavaScript basada en eventos como la carga de un documento,Mouse click o cambio de enfoque. Éstas permiten separar entre la presentación lógica y los datos. Tradicionalmente, las aplicaciones del lado del servidor estaban enfocadas en generar documentos HTML para todos los eventos del cliente resultantes de las llamadas al servidor. [12]

Algunos usos de las interacciones Ajax son los siguientes:

-Validación de datos en tiempo real.

-Autocompletado

-Carga en petición.

-Efectos y controles sofísticados en la interfaz de usuario.

-Actualización constante de datos y del servidor.

- Presentación parcial.

-Mashups

-Página como una aplicación

JAVA

Java es un lenguaje de programación y una plataforma informática comercializada por primera vez en 1995 por Sun Microsystems. Hay muchas aplicaciones y sitios web que no funcionarán a menos que tenga Java instalado y cada día se crean más. Java es rápido, seguro y fiable. Desde portátiles hasta centros de datos, desde consolas para juegos hasta súper computadoras, desde teléfonos móviles hasta Internet. [13]

Java ha sido probado, ajustado, ampliado y probado por toda una comunidad de desarrolladores, arquitectos de aplicaciones y entusiastas de Java. Java está diseñado para permitir el desarrollo de aplicaciones portátiles de elevado rendimiento para el más amplio rango de plataformas informáticas posible. Al poner a disposición de todo el mundo aplicaciones en entornos heterogéneos, las empresas pueden proporcionar más servicios y mejorar la productividad, las comunicaciones y colaboración del usuario final y reducir drásticamente el costo de propiedad tanto para aplicaciones de usuario como de empresa. Java se ha convertido en un valor impagable para los desarrolladores, ya que les permite:

-Escribir software en una plataforma y ejecutarla virtualmente en otra

-Crear programas que se puedan ejecutar en un explorador y acceder a servicios Web disponibles

-Desarrollar aplicaciones de servidor para foros en línea, almacenes, encuestas, procesamiento de formularios HTML y mucho más

-Combinar aplicaciones o servicios que utilizan el lenguaje Java para crear aplicaciones o servicios con un gran nivel de personalización

-Escribir aplicaciones potentes y eficaces para teléfonos móviles, procesadores remotos, microcontroladores, módulos inalámbricos, sensores, gateways, productos de consumo y prácticamente cualquier otro dispositivo electrónico. [14]

SERVLETS

Java Servlet es una tecnología utilizada para la mejora de los servidores web.

Abastece de componentes basados en metodos independientes para la construcción de aplicaciones web, sin limitaciones de desarrollo de programas CGI. Servlets son servidores y plataformas independientes que dejan libremente a elección del desarrollador la mejor estrategia para sus servidores,plataformas y herramientas.

Toda la familia de Java API tiene acceso a Servlets. Hoy en día, es muy popular para la interacción y construcción de web dinámicas. [15]

ECLIPSE

Eclipse es una comunidad creada originalmente por IBM en 2001 para todo el mundo cuyo deseo es colaborar con la comercialización de software de código abierto.

Sus proyectos están enfocados a la construcción de una plataforma de desarrollo de código abierto extensible a frameworks, herramientas. Originalmente se rige por los términos de la licencia Common Public License para posteriormente obedecer a los términos de la Eclipse Public License.La Free Software Foundation declaró a ambas licencias de software libre incompatibles con la licencia GNU GPL.

Los widgets de Eclipse están implementados por una herramienta de widget para Java llamada Standard Widget Toolkit, a diferencia de la mayoría de la aplicaciones Java, que usan las opciones estándar Abstract Window Toolkit(AWT) o Swing. La interfaz de usuario de Eclipse también tiene una capa GUI intermedia llamada JFace, la cual simplifica la construcción de aplicaciones basadas en SWT.

El entorno de desarrollo integrado (IDE) de Eclipse emplea módulos (en inglés *plug-in*) para proporcionar toda su funcionalidad al frente de la plataforma de cliente enriquecido, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no. Adicionalmente Eclipse permite el uso de otros lenguajes de programación como son C/C++ y Python, permite a Eclipse trabajar con lenguajes para procesado de texto como Latex, aplicaciones en red como Telnet y sistema de gestión de bases de datos. Se provee soporte para Java y CVS en el SDK de Eclipse. Y no tiene por qué ser usado únicamente con estos lenguajes, ya que soporta otros lenguajes de programación.

En cuanto a las aplicaciones clientes, Eclipse provee al programador con frameworks muy ricos para el desarrollo de aplicaciones gráficas, definición y manipulación de modelos de software, aplicaciones web, etc. Por ejemplo, GEF (Graphic Editing Framework - Framework para la edición gráfica) es un plugin de Eclipse para el desarrollo de editores visuales que pueden ir desde procesadores de texto wysiwyg hasta editores de diagramas UML, interfaces gráficas para el usuario (GUI), etc. Dado que los editores producidos por GEF "viven" dentro de Eclipse, además de poder ser usados conjuntamente con otros plugins, hacen uso de su interfaz gráfica personalizable y profesional.

El SDK de Eclipse incluye las herramientas de desarrollo de Java, ofreciendo un IDE con un compilador de Java interno y un modelo completo de los archivos fuente de Java. Esto permite técnicas avanzadas de refactorización y análisis de código. Mediante diversos plugins estas herramientas están también disponibles para otros lenguajes como C/C++ (Eclipse CDT) y en la medida de lo posible para lenguajes de script no tipados como PHP o Javascript. [2]

Características:

- Eclipse dispone de un editor de texto con resaltado de sintaxis para los diferentes lenguajes de programación soportados.
- Compilación en tiempo real.
- Pruebas unitarias con JUnit
- Control de versiones con CVS
- Integración con Ant, asistentes para creación de proyectos,clases,tests,etc…
- El lenguaje de programación utilizado en Eclipse es Java con un 92 % de cuota.

MYSQL WORKBENCH

MySQL Workbench es una herramienta visual de diseño de bases de datos, administración o desarrollo de las mismas.Provee de modelado de datos, desarrollo SQL y una gran variedad de herramientas de administración para la configuración del servidor. Disponible para distintos sistemas operativos.

Diseño

MySQL Workbench permite al administrador de las bases de datos y al desarrollador visualmente diseñar, modelar, generar y gestionar las bases de datos. Esto incluye todo lo que un modelador de datos pueda necesitar para crear complejos modelos ER. Simplifica el diseño y el mantenimiento de la base de datos, así como automatiza el tiempo.

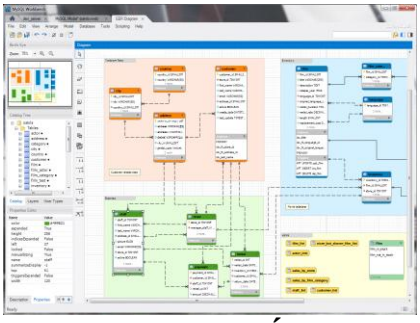

ILUSTRACIÓN 8

Desarrollo

MySQL Workbench proporciona herramientas visuales para la creación, ejecución y optimización de las consultas SQL. El Panel de Conexiones de la Base de Datos permite a los desarrolladores gestionar fácilmente las conexiones de la base de datos. Por último, el Buscador de Objetos da un acceso instantáneo a los esquemas y objetos de estas mismas.

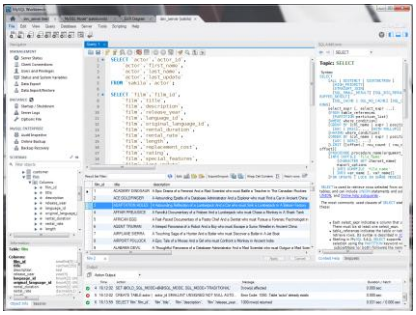

ILUSTRACIÓN 9

Administración

MySQL Workbench provee de una consola visual para la sencilla administración de MySQL y beneficiarse de una visibilidad dentro de las bases de datos mayor. Los desarrolladores y administradores de la base de datos pueden usar las herramientas visuales para la configuración de servidores, la administración de usuarios, tareas de recuperación y la inspección de los datos auditados. [5]

XAMPP

XAMPP es una distribución de Apache completamente gratuita y fácil de instalar que contiene MySQL, PHP y Perl. Mucha gente conoce de primera mano que no es fácil instalar un servidor web Apache y la tarea se complica si le añadimos MySQL,PHP y Perl. El objetivo de XAMPP es crear una distribución fácil de instalar para desarrolladores que se están iniciando en el mundo Apache. XAMPP viene configurado por defecto con todas las opciones activadas. XAMPP es gratuito tanto para usos comerciales como no comerciales. En caso de usar XAMPP comercialmente, asegúrate de que cumples con las licencias de los productos incluidos en XAMPP.

XAMPP es una compilación de software libre (similar a una distribución de Linux). Es gratuita y puede ser copiada libremente de acuerdo con la licencia GNU GPL. Únicamente la compilación de XAMPP está publicada bajo la licencia GPL. Cada uno

de los componentes incluidos tiene su propia licencia y deberías consultarlas para conocer qué es posible y que no. En el caso de uso comercial deberás consultar las licencias individuales, en particular MySQL. Desde el punto de vista de XAMPP como compilación, el uso comercial es gratuito. [6]

Características

-Entorno de desarrollo PHP más popular

-Fácil instalación y configuración

-Completamente gratuito.

-Soportado por diversos sistemas operativos.

METODOLOGÍA

En este apartado describimos el modelo de organización que hemos seguido en todo el proceso de construcción de este proyecto. Como en cualquier proceso de ingeniería de software se diferencian varias etapas. Todas ellas tienen su capítulo dedicado, en el que se explica en profundidad lo realizado a lo lardo del tiempo empleado en ella. Aquí nos limitamos a describir el paso por todas ellas y como fuimos organizando nuestro trabajo en cada una.

La primera fase del proyecto fue la de especificación de requisitos. La mayoría de ellos fueron tomados de las descripciones que nuestro tutor nos proporcionaba y otros fueron autoimpuestos por nosotros al ir conociendo el objetivo del sistema. Durante este periodo tuvimos reuniones a menudo para ir concretando, entre el equipo y el tutor, los detalles del proyecto. Una vez tuvimos claro el alcance pasamos a la fase de diseño.

En esta etapa, tuvimos que decidir en principio las líneas generales en las que íbamos a dar solución a lo anteriormente proyectado. Fue en este periodo en el que decidimos que tecnología utilizaríamos y que lenguajes se adaptaban mejor al proyecto y a nuestra forma de trabajar. También elegimos que entorno de desarrollo usaríamos, el cual ya hemos descrito en este mismo capítulo. Tras determinar estas lineas globales de diseño, dividimos el trabajo de diseño en los tres elementos principales que habíamos obtenido del análisis en general del proyecto entero (estos son la interfaz, el motor de querys y las

páginas de procesado). Una vez dividido, en cada apartado se realizó el diseño más pormenorizado. En esta etapa se redujeron drásticamente la cantidad de reuniones, especialmente tras haber determinado los apartados que mencionábamos. Tras la etapa de diseño, cuando teníamos en mente que había que desarrollar, comenzamos con su implementación.

Durante la implementación, continuamos con la división en apartados que habíamos marcado anteriormente. Aunque se trató de aislar lo máximo posible la implementación en cada uno de las lineas de desarrollo, los módulos que construíamos estaban pensados para trabajar juntos por lo que necesitamos ciertas reuniones y puestas en común del desarrollo llevado a cabo. A lo largo de esta fase, pudimos comprobar que la linea que desarrollaba el motor de querys, necesitaba más trabajo que las otras dos, por lo que en el último periodo de la implementación nuestro trabajo se enfocó en este apartado. Una vez hecha la implementación del sistema, pasamos a una subetapa en la que implementamos la aplicación de prueba sobre nuestra propia plataforma. En un sistema de las características del nuestro, esto constituye en realidad la realización de pruebas. No obstante separamos una cuarta etapa de evaluación de los resultados obtenidos.

Tras realizar la aplicación que ponía a prueba el funcionamiento del desarrollo, nos planteamos que aspectos faltaban. Ayudados por reuniones con el tutor, determinamos que aspectos que no habían resultado satisfactorios teníamos que incluir en otro proceso de diseño y desarrollo y cuales plantear como extensiones futuras al proyecto.

Esta segunda etapa de desarrollo en la que principalmente se eliminaron bugs y se añadieron detalles que mejoraban el software, está incluida sin diferenciación en cada capítulo, pues habitualmente no constituye cambios de peso suficiente como para destacarlos.

Conocimientos

Ya hemos nombrado las características principales de nuestro sistema. Por lo tanto podemos nombrar que conocimientos hemos aplicado a lo largo del desarrollo del proyecto.

En primer lugar, para la organización hemos aplicado conocimientos de Ingeniería del Software, los cuales en general también son aplicables para cualquier otro desarrollo. En particular nos sirvieron para planear las etapas que hemos descrito en el anterior

apartado. En dicha asignatura también se nos inculcan valores como la buena gestión de un proyecto, especificación de requisitos, así como la asignación de distintos roles a los miembros del grupo. Por otra parte, ha sido fundamental el apoyo que nos ha proporcionado el conocimiento en asignaturas aplicadas a la programación más concretamente Programación Orientadas a Objetos o Laboratorio de Programación III, en las cuales se imparte de manera teórica y práctica el lenguaje de programación Java, necesario para desarrollar el uso de JavaScript y más concretamente la librería utilizada como es Dhtmlx. En cuanto al manejo de bases de datos, los conceptos necesarios se impartieron en la asignatura Bases de Datos y Sistemas de Información, sin la cual el entendimiento de MySQL habría sido imposible. Por último,el conocimiento del uso de diferentes librerías y en especial HighCharts lo adquirimos en la asignatura Ingeniería de Sistemas Basados en Conocimiento, donde en una de las prácticas se nos requería el uso de la API de Twitter y el posterior uso de un gestor de gráficos a partir de los datos obtenidos

CAPÍTULO 3. ESPECIFICACIÓN DE REQUISITOS

En este capítulo comenzamos a describir nuestra aplicación con la exposición de los requisitos que en un primer paso especificamos. Como en cualquier proyecto software, estos fueron cumplidos en mayor o menor grado de éxito en las posteriores fases de diseño e implementación. En el capítulo de conclusiones analizamos entre otras cosas si los requisitos aquí definidos han sido cumplidos aceptablemente.

Como ya hemos dicho nuestra aplicación es un sistema genérico que permite implementar otras aplicaciones sobre él. Por ello nos referiremos a cualquiera de estas aplicaciones que se podrían construir sobre la nuestra como "aplicación cliente" o "aplicación final". Estas aplicaciones tendrán sus propios requisitos y casos de uso, que en su fase de implementación se tendrán que traducir en registros sobre nuestro sistema.

Por tanto al analizar los requisitos de nuestro sistema indicaremos además en qué puntos se relaciona con un supuesto diseño de aplicación cliente. (Veremos un ejemplo de este segundo en el apartado de Ejemplo de Aplicación desarrollada sobre la plataforma).

En primer lugar desarrollaremos los conceptos básicos que usaremos en la especificación de nuestra aplicación.

Acción: Toda la funcionalidad que vaya a ofrecer la aplicación cliente se definirá en forma de acciones. Estas representan una interacción con la base de datos, ya sea en forma de obtención, modificación, creación o eliminación de datos. Además los resultados de esta interacción se presentan al usuario a través de la interfaz del programa. Como veremos, una misma acción podrá ser configurada para ejecutarse con distintos parámetros o de forma que sus resultados se presenten de distintas maneras. En principio no se define límite a lo que se debe poder configurar para que una acción puede realizar, asemejándolo a la potencia del lenguaje SQL. En la fase de diseño analizaremos el nivel real de potencia que podemos llegar a obtener.

Rol: El concepto de Rol permite agrupar Acciones con sus respectivas configuraciones de manera que estas representen la funcionalidad que se le quiere otorgar a un tipo de usuario en la aplicación cliente. Por lo tanto, en nuestro sistema todo usuario tendrá asignado un Rol, el cual determinará qué Acciones se muestran en la interfaz. Esta estructura permite definir una misma Acción con distinta configuración (parámetros y presentación) para el mismo o para distintos Roles. Lo que otorga mayor flexibilidad a la hora de configurar los requisitos de la aplicación cliente sobre nuestro sistema.

Esta sección muestra los distintos perfiles de requisitos que hemos definido en el diseño de la aplicación:

Usuario final. Este representa a la persona que trabajará con las interfaces, acciones y gráficos devueltos por la aplicación cliente. Como es normal, existirán distintos tipos de usuarios finales que requieran distinta funcionalidad, la cual normalmente se debería especificar como distintos Roles en nuestra aplicación.

Los usuarios finales no requieren conocimientos de SQL ni de la configuración de la aplicación. Para este perfil no debería haber diferencia entre usar un diseño hecho sobre nuestro sistema o un desarrollo desde cero, puesto que harán uso de aquello que se haya especificado en la aplicación cliente.

Los requisitos de este grupo son el acceder a la aplicación (y salir) y ejecutar Acciones. Esto último implica tanto poder editar los parámetros que se le pasan a la Acción como visualizar los datos devueltos de la manera apropiada. Todo ello se llevará a cabo a través de la interfaz web del sistema. Por último como un requisito no vital añadimos que el usuario pueda configurar la interfaz para que se adapte a sus necesidades, como visualizar la forma de introducir un parámetro de la forma que prefiera según el tipo de dato (textfield, combobox, lista, etc).

Configurador. Este perfil es el de la persona encargada de llevar a cabo la asignación de Roles y la configuración de las distintas Acciones con sus querys correspondientes. Por lo tanto el Configurador implementará sobre nuestro sistema las especificaciones de la aplicación cliente. Para esta tarea se le presupone un conocimiento de SQL y del manual de configuración de nuestra aplicación pero no necesariamente de su programación interna.

Una vez se haya hecho la especificación de la aplicación final, el Configurador tendrá que evaluar si las herramientas de las que dispone en la versión estándar de nuestro sistema son suficientes para implementar dicha especificación (en el capítulo de implementación se describe la funcionalidad que proporcionamos).

Los requisitos del Configurador son los que describimos de manera más extensa pues como decimos, interactúa con el motor principal del sistema para configurarlo. Debe poder administrar (crear, modificar y eliminar) las acciones, los usuarios, los roles y poder seleccionar el origen de los datos, es decir indicar al sistema cual es la base de datos de la aplicación cliente. Dentro de la administración de acciones se incluye todo lo relativo a su configuración, como la selección de parámetros y forma de visualización de la respuesta o la definición de que proceso se lleva a cabo al ejecutarla. En la administración de roles y usuarios destacamos la asignación del rol a cada usuario y de acciones a los roles. Con respecto a la interfaz de usuario, el Configurador podrá

determinar de que forma le aparecen al usuario los parámetros a introducir, y la organización de las acciones en distintos menús que permitan una mejor navegación.

Desarrollador. El rol de Desarrollador es el encargado de hacer modificaciones mayores a nuestra aplicación para añadirle funcionalidad extra más allá de la proporcionada por el desarrollo estándar. Este es el caso al que nos referíamos antes en el que el Configurador determine que necesita alguna característica que no está incluida para la implementación de la aplicación cliente. Los Desarrolladores si que conocerán el funcionamiento interno del sistema, permitiéndoles hacer los cambios necesarios para adaptarlo a las necesidades de la aplicación final.

En general esperamos que los cambios requeridos sean el añadido de nuevas formas de presentación de los datos, nuevas funciones con las que construir las Acciones o nuevos elementos para mejorar la interfaz.

Además los Desarrolladores serán los responsables de mantener el sistema y los datos en caso de fallos.

Este grupo no genera requisitos como tal, pues su propia definición indica que serán capaces de modificar directamente el código del sistema.

Finalmente la interfaz de la aplicación estará contenida en una única página que se actualice dinámicamente sin necesidad de recargarse. La interfaz contará con menús que permitan acceder a las acciones. Una vez se elija una acción, deben aparecer un formulario donde introducir los parámetros necesarios y un botón para ejecutarla. Finalmente los resultados se visualizarán en la misma web permitiendo visualizar a la vez el resultado de varias acciones.

REQUISITOS DE LAS PRUEBAS

En este apartado especificamos una serie de requisitos para el desarrollo de una demo de prueba a través de la cual analizaremos la realización de los objetivos marcados en el proyecto.

En general, para cualquier prueba de nuestro proyecto necesitaremos una base de datos “cliente”. Tampoco hace falta que sea muy compleja, ya que se trata tan solo de un ejemplo, pero podría serlo.

Con esto dicho pasamos a concretar las pruebas:

- Obtención de datos a través de una tabla.
- Obtención de datos a través de un gráfico.

- Inserción en la base de datos.

PRUEBA 1:

Consistirá en añadir una acción a uno de los menús en la cual, con un parámetro de entrada se montará una query para la base de datos que nos devolverá la información necesaria formateada para la api dhtmlx con la que devolver una tabla en una ventana auxiliar.

-Prerrequisitos: Con figuración de la acción en la base de datos de la aplicación.

-Datos de entrada: Un parámetro introducido por el usuario.

-Datos de salida: Una tabla en una ventana de dhtmlx.

PRUEBA 2:

En esta prueba realizamos algo similar a la anterior, salvo porque en esta la aplicación se encargará de darnos los datos con el formato adecuado para la api highcharts con el fin de obtener un gráfico con la información.

-Prerrequisitos: Con figuración de la acción en la base de datos de la aplicación.

-Datos de entrada: Un parámetro introducido por el usuario.

-Datos de salida: Un gráfico de highcharts en una ventana de dhtmlx.

PRUEBA 3:

La tercera prueba consiste en insertar elementos en una tabla ya existente de la base de datos de prueba.

-Prerrequisitos: Configuración de la acción en la base de datos.

-Datos de entrada: Campos de la tabla a la que se quiera añadir el nuevo elemento.

-Datos de salida: El elemento es añadido a la tabla, se puede ver la instrucción sql mediante la que se ha llevado a cabo la operación.

CAPÍTULO 4. DISEÑO

En este capítulo mostramos el diseño que propusimos para nuestro sistema y la forma en la que da solución a los requisitos anteriormente descritos.

Presentamos los siguientes elementos de la aplicación, los cuales en su mayoría ya han sido bocetados en la especificación y aquí se desarrolla su diseño y se explica su manera de funcionar e interaccionar entre sí:

Acciones: para adaptar este concepto al diseño necesitamos definir qué interacciones con la base de datos permitimos y como las estructuramos para mantenerlas en el sistema. Decidimos permitir dos tipos de acciones:

Acciones configuradas. Este tipo de acciones crearán una query SQL según lo guardado por el Configurador. Esta información, necesaria para la query, que describimos a continuación se guardará en la base de datos de nuestro sistema. Los tres aspectos principales de la configuración serán el conjunto de tablas sobre el que se construirá la sentencia select, el grupo de campos que devolverá y el grupo de uniones y filtros que se aplicarán en el where. Como vemos para estas acciones no permitimos group by ni realizar operaciones sobre los campos, lo cual aunque limita la potencia que ofrecemos, da una visión más simple del sistema al Configurador.

El Configurador, además de seleccionar los campos para el select y el where tendrán que designar su origen. En ambos grupos de campos, los valores que pueden tomar son fijos, rellenos por el usuario o rellenos por la query. En los campos con valor fijo, este es indicado directamente por el Configurador. En los rellenos por el usuario, el valor lo tomará del indicado por el usuario en la interfaz. En los rellenos por la query, se especificará que ese campo tomará el valor que devuelva la ejecución de las query. Además las uniones necesarias para el where serán de la siguiente forma:

Campo Query -operador- campo Query

Campo Query -operador- campo Usuario

Campo Query -operador- campo Fijo

Campo Usuario -operador- campo Fijo

De esta manera cubrimos todas las posibles uniones o filtros que el Configurador pueda necesitar hacer. En principio implementaremos el operador igual ('=') pudiéndose ampliar fácilmente a otros como veremos en el capítulo de Conclusiones.

Toda la información relativa a los campos y uniones se guardará en la base de datos de nuestro sistema de forma a anexa a la de la propia acción. Además permitiremos reutilizar grupos de campos para distintas acciones y ampliar los grupos de campos por medio de herencia, esto es, crear un grupo de campos nuevo que tenga los ya definidos en un grupo anterior añadiendo uno o varios campos más. A continuación mostramos un ejemplo que muestre lo anteriormente dicho sobre campos y uniones:

Supongamos que queremos configurar una acción que devuelva información de un cliente del cual el usuario proporciona el dni. La información del cliente se guarda en la tabla Cliente (nombre, apellido, dni). Además necesitamos que las filas devueltas por esta query vengan marcadas por un texto, por ejemplo "cliente:". Configuraríamos la tabla clientes como única en la acción. En el grupo de campos select: campo "cliente:" Fijo; campo nombre Query; campo apellido Query; campo dni Query. En el grupo de campos where: campo dni Query -operador ('=')- campo dni Usuario. De esta manera deberíamos obtener la select: *Select "cliente:", nombre, apellido, dni from Cliente where dni=$dni;* (Siendo $dni la variable que almacena la información proporcionada por el usuario para el campo $dni).

Este ejemplo sencillo, ilustra la forma básica de trabajar de nuestro motor de querys. En un ejemplo más complejo, con varias tablas (o incluso varios esquemas de base de datos) en las que el mismo nombre de campo pueda aparecer en varias de ellas, el funcionamiento requeriría diferenciar el origen de los datos. Para tratar esto nuestro sistema proporcionará alias a las tablas y a los campos de forma que sean únicos (por ejemplo usando los identificadores de los objetos en la base de datos). Este tema será más ampliamente tratado en el capítulo de Implementación.

Además con respecto a las uniones, para sistemas sencillos, el Configurador podrá guardar en el sistema cómo se unen dos tablas y el sistema, siempre que aparezcan esas tablas en las seleccionadas para la query, automáticamente incluirá las sentencias necesarias en el where. Sin embargo, para bases de datos (o querys) algo más complejas esto no es viable y por tanto, no podremos hacer uso de esta característica. Por esto, cada Acción configurada llevará guardado si se usa esta opción o no.

Las acciones no siempre son una select sobre la base de datos. Nuestro sistema también implementará en un principio la instrucción Insert (y en el capítulo Conclusiones veremos las líneas a seguir para incluir el Update y Delete). Para el Insert, usaremos la select devuelta por el motor, de forma que la estructura final sea:

Insert Tabla(campo1,campo2) select …

Los campos que incluiremos de la tabla en la que insertar serán todos menos los que estén marcados como autoincremento. Por tanto si hay campos que permiten null y queremos dejarlo con este valor, el configurador usará el valor null como fijo en la select.

La otra opción para acciones son las que se basan directamente en un código SQL que se ejecutará tal cual, sin modificaciones. Por tanto, en esta opción no permitiremos parámetros con los que el usuario pueda interactuar con la acción ya que no es procesada por el motor de querys del sistema. Debido a las limitaciones impuestas a las acciones configurables, damos esta segunda vía para que las acciones que requieran una complejidad extra no sean del todo inviables en nuestra plataforma.

Menu: El objetivo de este elemento es únicamente ordenar la lista de acciones de las que dispone un usuario. Cada menú tendrá un nombre y para mostrar la acción en un menú usaremos el elemento ContenidoMenú. El ContenidoMenú asociará Menús con Acciones.

Rol: El concepto de rol es fácilmente adaptable al diseño. Simplemente, cada usuario tiene un Rol asociado. Usamos el mismo elemento, ContenidoMenú, para asociar cada uno de estos contenidos con un Rol. Así, a un usuario solo le aparecerán los Menús que tengan ContenidosMenú asociados con el Rol de este usuario.

ContenidoMenu: Este elemento, como acabamos de ver, proporciona la funcionalidad para el sistema de Roles y Menús. Además, hemos querido darle una flexibilidad extra al permitir cambiar desde este nivel el grupo de campos con el que se ejecutará una Acción. Así, aunque la Acción tenga asociados sus grupos de campos predefinidos, el Configurador puede hacer excepciones incluyendo una forma distinta de ejecutarla.

Páginas de Procesado: Este elemento se introduce para dar respuesta a lo especificado sobre las distintas representaciones de los datos obtenidos por la query. Cada

ContenidoMenú tendrá asociada una página de procesado encargada de dar una forma visual adecuada a los datos. La página se indicará como una dirección web a la que la aplicación enviará los datos obtenidos como Parámetros. A su vez, la página de procesado devolverá a la página principal los datos que haya generado en un formato entendible por esta última.

Para proporcionar mayor funcionalidad aún a las páginas de procesado necesitamos incluir un nuevo tipo de grupo, aquel que incluye los parámetros que se le pasen a la página junto a los datos de la query. De esta forma los Desarrolladores pueden implementar una interacción mayor con el usuario por medio de este elemento, así como diseñarlas de forma que sea útil reutilizarlas con distintos parámetros fijos que varíen su comportamiento. Las páginas de procesado dotan de gran flexibilidad al sistema: aunque en principio están pensadas para el apartado visual de los datos, pueden usarse como formas de post-procesado en cualquier sentido e incluso hacer llamadas de páginas entre sí.

Desde el punto de vista de la seguridad, hemos elegido que la select generada por el motor la ejecute este mismo sin que en ningún momento se pase código SQL como parámetro. Esto asegura que no se pueden introducir sentencias ajenas al motor en ningún punto de la ejecución.

Como avanzamos en la especificación la implementación de páginas de procesado nuevas es responsabilidad de los Desarrolladores.

Cumpliendo con la especificación, los resultados de las Acciones se pueden visualizar de dos formas: como gráficos o como tablas. Por lo tanto, en nuestro sistema por defecto implementaremos dos páginas de procesado, una para cada una de las opciones. Además, el tipo de gráfico proporcionado será el de multibarras que se adapta a bastantes tipos de información.

Tipos de campos: Para permitir cierta personalización de las interfaces, los usuarios y los configuradores serán capaces de elegir como se representa un campo de entrada en la interfaz. Los tipos de campos se definirán con un código que la página principal en la que se integra la interfaz sea capaz de comprender. Por tanto, la creación de nuevos tipos de campos es responsabilidad del Desarrollador, pues necesita añadir código en esta página.

Cada campo registrado en el sistema tendrá asociado un tipo por defecto. Sin embargo, el Configurador podrá definir que para cierto Rol un campo tenga asociado otro tipo de campo. Finalmente, el usuario podrá definir que para sí mismo un campo aparezca de con otro tipo el cual prevalecerá sobre las otras dos configuraciones.

El tipo de campo por defecto será el textField, es decir, un lugar donde escribir en una única línea. Además será el único implementado por el sistema estándar, siendo tarea del Desarrollador agregar los necesarios para la aplicación cliente.

CAPÍTULO 5. IMPLEMENTACIÓN

TABLAS

En este capítulo tratamos la implementación dada al diseño hecho en el anterior. Describiremos las tablas creadas en la base de datos del sistema, las páginas que componen nuestra aplicación y como es el flujo de datos ante una petición de ejecución de una Acción. También hablaremos de como funciona la interfaz y el uso que hemos dado a las APIs dhtmlx y HighCharts.

Hemos hecho uso de una base de datos MySQL para la implementación de la base de datos. Todos los elementos del sistema están definido por un identificador entero único el cual será la forma de referenciar unos elementos a otros.

Tabla Accion:

En esta se registran todas las Acciones definidas en el sistema.

idGCampoPredef: Define el grupo de campos usado en el where. (Unión con GCampo).

idGCampoSelectPredef: Define el grupo de campos usado en el select. (Unión con GCampo).

IdClaseAccion: Especifica el tipo de acción, esto es Select o Insert (Delete y Update en caso de ser agregado a la funcionalidad). (Unión con ClaseAccion).

IdTablaAccion: Es el identificador de la tabla a la que se hace referencia cuando la Acción sea un Insert. En caso de ser Select se ignora. (Unión con Tabla).

Tabla GCampo:

Guarda los grupos de campos para las acciones de forma recursiva.

-idGCampo: Identificador del campo “actual”.

-idGCampoBase: Id del ”siguiente” campo.

Tabla ContenidoGCampo:

-idContenidoGCampo.

-idGCampo:

-idCampo:

-idValorCampo: Valor del campo.

-idAliasSelectValorCampo:

-idClaseContenidoGCampo:

-idCampoUnion;

Tabla ValorCampo:

Valor del campo indicado en ContenidoGCampo.

-idValorCampo.

-valor.

Tabla ClaseContenidoGCampo:

Tipos de clases de los campos.

-idContenidoGCampo.

-nombre.

Tabla Campo:

Contiene los detalles de un campo

-idCampo.

-idTipoCampoPredef: Id del tipo de campo en caso de no tener ninguno definido en el usuario.

-nombre.

-idTabla: id de la tabla a la que pertenece el campo??

-idClaseCampo: id que hace referencia a de que tipo es el campo.

-autoincremento: indica si el campo se autoincrementa??

Tabla ClaseCampo:

Indica de qué clase es el campo (de momento “normal).

-idClaseCampo.

-nombre.

-descripción.

Tabla TipoCampo:

Guarda el tipo de los campos (entrada, botononones..)

-idTipoCampo.

-Tipo.

Tabla ClaseAcción

Tipo de acción.

-idClaseAccion: Identificador de la clase de acción.

-nombre.

-descripción.

Tabla TablaAccion:

Relaciones entre las acciones y la tabla a la que se refieren en caso de ser un Insert.

-idTabla: Tabla de la base de datos para el Insert.

-idAcción: Acción a la que se refiere.

Tabla Tabla:

En esta tabla tenemos todas las tablas utilizadas por las acciones y la base de datos en que aparecen.

-idTabla.

-idBaseDatos.

-nombre.

Tabla BaseDatos:

Guarda el id de cada base de datos y su nombre.

-idBaseDatos.

-nombre.

Tabla ContenidoMenu:

Es una de las tablas más importantes. En ella se registran los menús y submenús para cada rol.

-idContenidoMenu.

-idRol: Id del rol al que corresponde la acción.

-idMenu: Submenú al que corresponde la acción.

-idAccion: Acción que se ejecutará.

-idGCampo: Grupo de campos que corresponderán al where??

-nombre.

-idGCampoSelect: Grupo de campos que corresponderán al select.

-idPaginaProcesado: Referencia a la página jsp que ha de procesar la respuesta de la base de datos para darles el formato adecuado (JSON para la api dhtmlx, JSON para la api Highcharts…)

-descripcion.

Tabla Rol:

Distintos tipos de usuario para la aplicación.

-idRol.

Tabla Usuario:

Almacena los usuarios de la aplicación, con una referencia a su rol correspondiente.

-idUsuario.

-idRol.

Tabla Menu:

Contiene los submenús para rellenar el menú de cada usuario.

-idMenu.

-nombre.

Tabla UsuarioTipoCampoPredef:

Tipo predefinido para los campos en función de cada usuario.

-idUsuario.

-idCampo.

-idTipoCampo.

-idContenidoMenu.

Tabla RolTipoCampoPredef:

Tipo predefinido para los campos en función de cada rol.

-idRol.

-idCampo.

-idTipoCampo.

-idContenidoMenu.

Tabla QueryAccion:

Relaciona las querys directamente con el contenido del menú.

-idQuery.

-idContenidoMenu.

Tabla Query:

Querys para las acciones que van directamente a la base de datos.

-idQuery.

-queryText: La query en texto plano.

Tabla PaginaProcesado:

Guarda la dirección en la que se encuentran las páginas de procesado para los datos procedentes de la base de datos.

-idPaginaProcesado.

-direccion.

-descripcion.

INTERFAZ

En la implementación de la interfaz usando Dhtmlx, tuvimos que elegir que componentes usar de todos los que ofrece la librería. Tras algunas pruebas con los componentes dhtmlxMenu y dhtmlxToolbar, nos decantamos por el componente dhtmlxAccordion. La decisión estuvo determinada por el boceto presentado en la especificación. Con el formato en acordeón, pretendíamos mantener el aspecto en forma de grandes botones y listas que refleja esa primera aproximación, aunque de una manera más condensada para dejar más espacio al formulario.

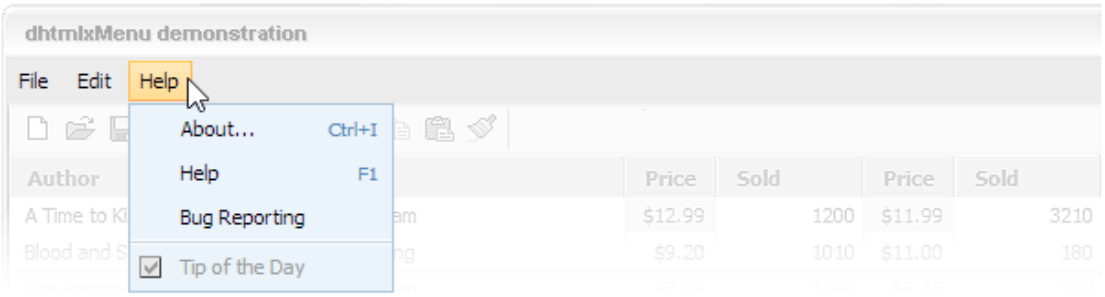

ILUSTRACIÓN 10 : EJEMPLO DE LA APARIENCIA DE DHTMLXMENU.

ILUSTRACIÓN 11: EJEMPLO DE LA APARIENCIA DE DHTMLXTOOLBAR

Este será el componente usado para el menú principal, mientras que los submenús estarán incluidos en un dhtmlxGrid (con una sola columna) que proporciona la apariencia de lista.

Para el formulario de las acciones, usamos dhtmlxForm, que actúa de forma similar a un formulario estandar de html pero con una interfaz mejorada y con funciones ya preparadas para la actualización dinámica.

Por último mantuvimos la idea de usar el componente dhtmlxWindow para la visualización de los resultados. Dentro de estas ventanas, incrustaremos los gráficos del API Highcharts o las tablas representadas también con dhtmlxGrid.

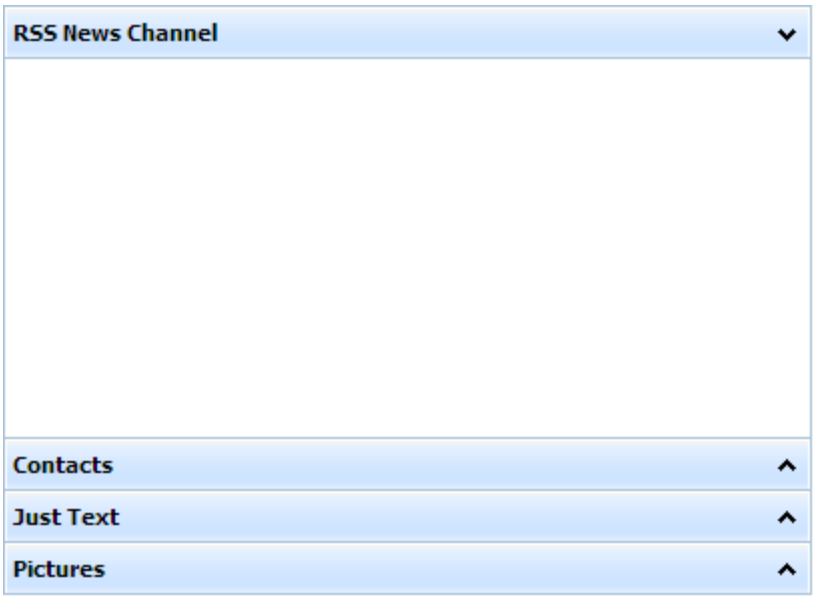

ILUSTRACIÓN 12: EJEMPLO DE LA APARIENCIA DE DHTMLXACCORDION.

Tras la elección de los componentes, el siguiente paso en el desarrollo de la interfaz, fue la interacción entre ellos y la carga dinámica de datos. Dado que los nombres de los menús, de las acciones, etc. se almacenan en la base de datos, necesitábamos cargarlos dinámicamente. Dhtmlx permite definir distintos eventos, (click, dobleclick, arrastrar, etc) de los cuales solo usamos el click. A grandes rasgos, cuando se hace click en uno de los elementos este llama a una página jsp que le proporciona la información del elemento nuevo que se ha de crear. A contiuación detallamos más este proceso junto con la estructura que forman las páginas del sistema:

Sales	Book		Price	Delivery terms		Bestseller
				In Store	Shipping	
1500	The Partner	John Grisham	$12.99	☑	2 days	◉
1300	The Village	Ivan Bunin	$11.66	☐	24 Hours	○
800	Eugene Onegin	Alexandr Pushkin	$11.20	☑	24 Hours	○
700	The Winter's Tale	William Shakespeare	$19.31	☑	1 Hour	○
700	Misery	Stephen King	$7.70	☐	na	○
650	The Captain's Daughter	Alexandr Pushkin	$10.21	☐	2 days	○
500	It	Stephen King	$9.70	☐	na	○
400	Cousin Bette	Honore de Balzac	$0	☑	1 Hour	○
350	The Green Mile	Stephen King	$11.10	☑	24 Hours	○
250	The Black Sheep	Honore de Balzac	$16.00	☑	1 Hour	○
150	Father Goriot	Honore de Balzac	$9.99	☐	2 days	○
-80	Lost Illusions	Honore de Balzac	$8.10	☑	na	○
-100	Hamlet	William Shakespeare	$5.99	☑	1 Hour	○
-100	Boris Godunov	Alexandr Pushkin	$7.15	☑	1 Hour	○
-200	The Rainmaker	John Grisham	$7.99	☐	2 days	○
-300	Dark Avenues	Ivan Bunin	$14.96	☑	1 Hour	○
-120(	The Dark Half	Stephen King	$0	☐	2 days	○

ILUSTRACIÓN 13 : EJEMPLO DE LA APARIENCA DE DHTMLXGRID.

El sistema está constituido por 2 páginas principales: Principal.jsp y Respuesta.jsp.

La primera, Principal.jsp, contiene todo el código necesario para la interfaz y la comunicación de ésta con el motor de querys. Esta página hace uso de otras tres (además de Respuesta.jsp) para refrescarse dinámicamente según el usuario vaya haciendo uso de los menús.

Al iniciarse la página Principal.jsp se hace una llamada a la página JSONAcordeonMenu.jsp que devuelve el JSON requerido para la creación del menú principal.

Al hacer click en un elemento se hace una petición a la página JSONGridSubmenu.jsp la cual devuelve la información, en forma de JSON, necesaria para la creación del submenú que se ha seleccionado.

Cuando se hace click en un elemento del submenú la petición llega a la página JSONFormForm.jsp que devuelve el JSON que contiene los datos del formulario para la acción solicitada.

La página Respuesta.jsp es la que contiene el código del Motor de querys. Se hace una llamada a Respuesta.jsp cuando hacemos click en el botón de aceptar en una acción. Tras el proceso correspondiente al Motor y a las páginas de procesado, el control volverá a Principal.jsp donde se crearán los componentes dhtmlxWindow para mostrar el resultado.

En caso de ser el resultado una tabla, el API dhtmlx cuenta con métodos para incluir directamente cualquiera de sus componentes dentro de una de sus ventanas. Así pues, hacemos uso de esto y no tenemos más que rellenar la tabla con los resultados recibidos.

Si el resultado es un gráfico, el módulo de ventanas es capaz de representar dentro de ellas cualquier contenedor html. Como puede haber varios gráficos a la vez no podemos determinar un contenedor fijo que sea el que tome la ventana, sino que tenemos que crear dinámicamente un " " nuevo por cada gráfico que se genere. Lo realizamos por medio de código JavaScript, numeramos cada contenedor y asignamos cada uno a una ventana nueva. De esta manera el gráfico aparece dentro de la ventana y se desplaza con ella.

PÁGINAS DE PROCESADO

Como ya se explicó en el capítulo de diseño, la forma en la que solucionamos la necesidad de representar los datos obtenidos de la base de datos de forma distinta para las distintas acciones son las páginas de procesado. Gracias a ellas, podemos tratar la información de distintas maneras para que la página principal la pueda usar como sea conveniente en cada caso.

Las páginas de procesado las implementamos en forma de páginas .jsp en las que podemos generar el código HTML eficazmente. En general, consisten en la creación de un JSON con el formato adecuado para cada acción que requiera la página principal.

Más concretamente, contamos con las siguientes:

Página de procesado para tablas (PáginaProcesadoTabla.jsp)

En esta tratamos los datos devueltos por la base de datos para formar un JSON compatible con las tablas de la api dhtmlx.

Página de procesado para gráficos (PáginaProcesadoGrafico.jsp)

Aquí generamos un JSON con la estructura conveniente para los gráficos de la api highcharts, más en concreto para los de barras (de momento).

Motor

El funcionamiento de la página Respuesta.jsp, que constituye el Motor de querys, es el más complejo del sistema. Dado que su objetivo es crear una instrucción SQL, vamos a dividir su explicación en las partes de las que consta una de estas instrucciones: select, from, where. Además si se trata de un insert, añadimos la definición de los campos de la tabla, es decir que tendremos la estructura insert (<campos>) from select <...>.

Comenzamos con el from. Para determinar que tablas se incluyen basta con ver cuales tiene asociada la Acción en la tabla TablaAccion. Como cada tabla tiene asociada a que base de datos pertenece, la llamada a las tablas será de la forma <baseDatos>.<Tabla>. Además el sistema proporciona automáticamente el alias a cada tabla de la forma idBaseDatos_idTabla.

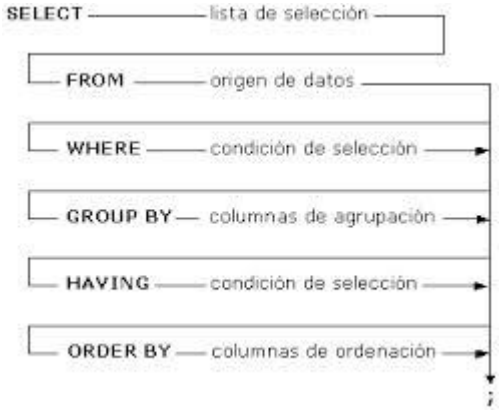

ILUSTRACIÓN 14 : ESTRUCTURA COMPLETA DE UNA CONSULTA. EN NUESTRO CASO SOLO USAMOS LAS TRES PRIMERAS CLÁUSULAS.

Para el select tomamos el grupo de campos GrupoCampoSelect. De este sacamos que campos se incluirán en el select. Como los campos están relacionados con su respectiva tabla, podemos obtener el origen de los datos usando el alias que antes hemos definido para cada una. Atendiendo a la clase del campo tomaremos la decisión de especificar el valor directamente para los rellenos por el cliente o por el Configurador, o poner el nombre del campo en la tabla si queremos obtener el valor de la base de datos. Por último el alias que queremos que tome el campo en el select viene determinado por el Configurador, y en caso de no especificar se toma el nombre del campo en la tabla.

Para el where el funcionamiento es similar. Se obtienen los campos del GrupoCampo. En este caso cada uno de ellos constituye un filtro. Los alias que necesitamos vienen determinados por el mismo sistema que en el select y también se atiende de igual manera a la clase de campo para determinar si incluir valores fijos o referencias a la

base de datos. En este caso, el alias del propio campo es ignorado pues no tiene sentido un alias para los campos usados en el where.

Para la construcción del insert, tenemos especificado en que tabla se va a insertar en la misma tabla de Accion (campo idTablaAccion). Por lo tanto seleccionamos todos los campos que están relacionados con esa tabla. Si alguno de ellos está marcado como autoincrementable, el sistema no lo incluirá por determinar que está pensado para no ser insertado a mano sino llevar la cuenta de manera autónoma.

Por último contamos con un sistema que trata de unir las tablas presentes en la Acción de forma automática. Si la Acción está marcada para utilizar esta capacidad, se accederá a la tabla unionTabla y por cada fila que coincida con dos de las tablas que usa la Acción, se creará un filtro con los datos proporcionados.

Finalmente Respuesta.jsp es la responsable de ejecutar la query en el servidor MySQL y devolver el resultado. Este se genera en forma de JSON con información sobre el nombre de las cabeceras. A continuación se pasa el control a las páginas de procesado. En el contenidoMenu que haya llamado a la Acción que se acaba de ejecutar, se especifica la página de procesado que tiene asociada. A esta página se le pasará el JSON que se generó anteriormente como parámetro. Si hay especificado un grupo de campos en el GrupoCamposParametros, también se le pasaran todos los que contenga ese grupo. Seguirán la forma que explicamos para el Select sobre las clases de campos. En caso de no haber ninguna página de procesado indicada, no se mostrará resultado de la Acción, aunque esta haya sido ejecutada.

SEGURIDAD

Seguridad en la implementación.

Al definir usuarios y roles, se supone de forma implícita que necesitamos asegurar que las restricciones definidas por el Configurador se cumplen. Esto es, que un usuario cuyo rol no tiene asignada una Acción, no puede ejecutarla. El hecho de no incluirla en los menús que se mostrarán al usuario no basta, pues se podría modificar el código en el cliente para cambiar identificadores de Acción u otros parámetros a la petición al Motor de querys que no deseamos que el usuario pueda controlar. Por ello necesitamos realizar ciertas comprobaciones antes de ejecutar la Acción que se nos pide en el Motor. Estas son:

Comprobar que hay un usuario logueado en la sesión desde la que se solicita la Acción. Cualquier petición que se realice sin este requisito será desechada automaticamente.

Obtener su rol y comprobar que ese rol tiene asociada la Acción que se quiere realizar. Al ser esta asociación a nivel de ContenidoMenu, comprobamos con esta acción también que no se han modificado los parámetros de llamada para la Acción. Además en caso de tener más argumentos de los esperados el sistema solo accederá a aquellos

que espera según los registros del sistema. Por último, si los parámetros son menos, el sistema devolverá un error.

Por otro lado, al generar consultas a la base de datos, queremos evitar la posibilidad de Injección de SQL. Para esto, se limitan en los parámetros de entrada los caracteres permitidos. Además, a la hora de pasar el flujo del programa a las páginas de procesados, esta solo obtendrán un JSON con los resultados de la ejecución de la query, y no el código que ha generado esos resultados. Las páginas de procesado no están pensadas para hacer peticiones a la base de datos, aunque se podrían hacer en caso de ser necesario. Lo que hemos evitado en todo momento es pasar el código SQL que se va a ejecutar como parámetro entre páginas, pues esto se considera un fallo de seguridad importante en un sistema web.

PRUEBA

Para poder enseñar el funcionamiento de nuestro proyecto, vamos a realizar una pequeña prueba a través de unas tablas de ejemplo:

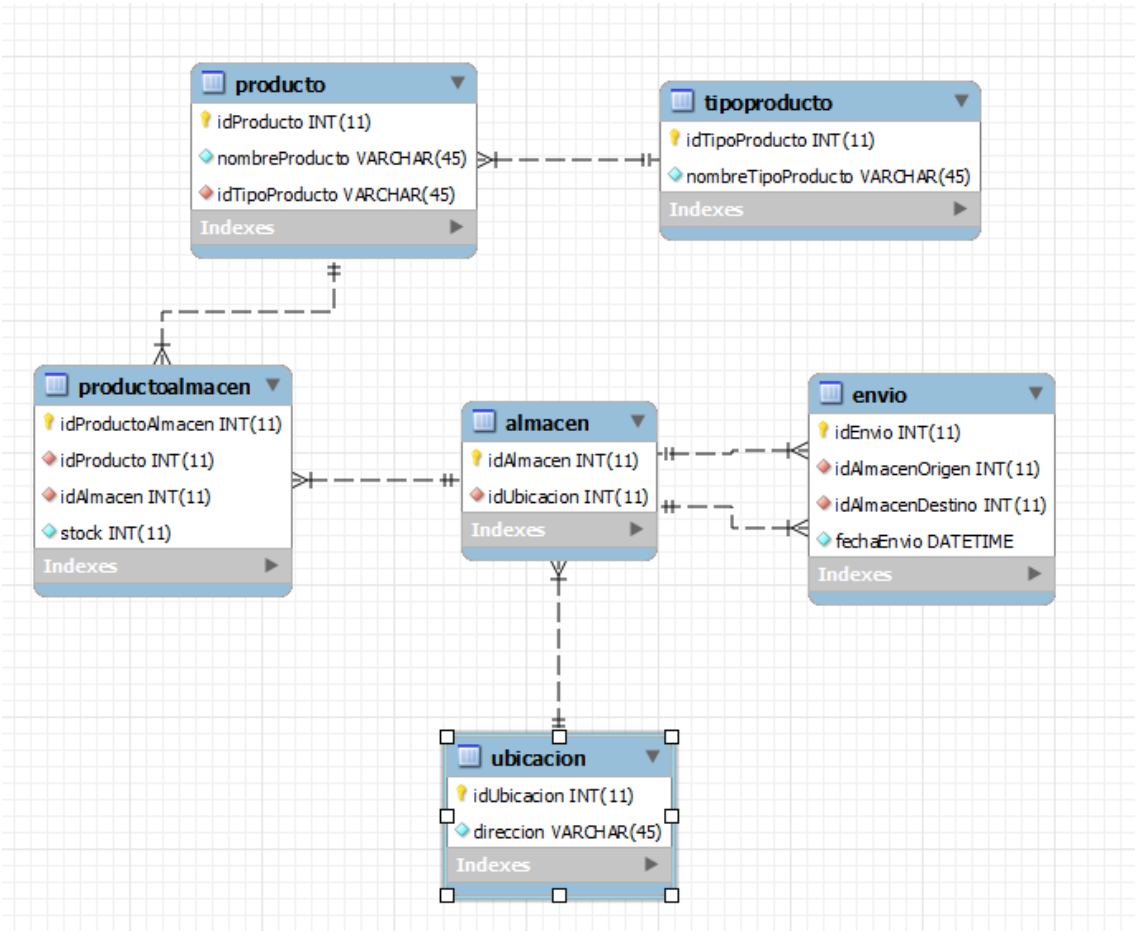

ILUSTRACIÓN 15 : BASE DE DATOS DE LA PRUEBA.

Como podemos ver tenemos seis tablas:

Producto

Es una tabla con los productos de una supuesta tienda de hardware, en ella recogemos:

-idProducto.

-nombreProducto.

-idTipoProducto: Clave ajena del tipo de producto de la tabla “TipoProducto”.

TipoProducto

En esta tenemos los posibles tipos de producto para el ejemplo:

-idTipoProducto.

-nombreTipoProducto.

ProductoAlmacen

Aquí guardamos las relaciones entre productos y almacenes para saber en qué almacen están qué productos.

-idProductoAlmacen.

-idProducto: Clave ajena de la tabla “Producto”.

-idAlmacen: Clave ajena de la tabla “Almacen”.

-stock.

Almacen

Tabla para recoger los almacenes de la tienda.

-idAlmacen.

-idUbicacion: Clave ajena de la tabla “Ubicacion”.

Ubicación

-idUbicacion.

-direccion.

Envio

En esta tenemos los datos necesarios para guardar los envíos realizados por la tienda.

-idEnvio.

-idAlmacenOrigen: Clave ajena de la tabla “Almacen”.

-idAlmacenDestino: Clave ajena de la tabla “Almacen”.

-fecha.

Mediante este sencillo diseño de una base de datos para nuestra supuesta tienda de hardware en la que disponemos de los datos de los productos, los almacenes y los datos de los envíos realizados entre ellos; podemos mostrar tanto la configuración de la aplicación como los resultados finales de la misma.

CONFIGURACIÓN

Siguiendo los pasos desarrollados en el apartado de “Uso del sistema” vemos lo que hemos de introducir en la base de datos:

Registramos en la base de datos la información referente a la base de datos, las tablas de la misma y sus campos en las tablas “BaseDatos”, “Tabla” y “Campo” respectivamente.

Para llevar esto a cabo tenemos el siguiente script .sql:

“insert into basedatos values (100, 'demo');

insert into tabla values (100, 100, 'almacen');

insert into tabla values (101, 100, 'envio');

insert into tabla values (102, 100, 'producto');

insert into tabla values (103, 100, 'productoalmacen');

insert into tabla values (104, 100, 'tipoproducto');

insert into tabla values (105, 100, 'ubicacion');

insert into campo values (1000,1,'idAlmacen',100,1,1);

insert into campo values (1001,1,'idUbicacion',100,1,0);

insert into campo values (1010,1,'idEnvio',101,1,1);

insert into campo values (1011,1,'idAlmacenOrigen',101,1,0);

insert into campo values (1012,1,'idAlmacenDestino',101,1,0);

insert into campo values (1013,1,'fecha',101,1,0);

insert into campo values (1020,1,'idProducto',102,1,1);

insert into campo values (1021,1,'nombreProducto',102,1,0);

insert into campo values (1022,1,'idTipoProducto',102,1,0);

insert into campo values (1030,1,'idProductoAlmacen',103,1,1);

insert into campo values (1031,1,'idProducto',103,1,0);

insert into campo values (1032,1,'idAlmacen',103,1,0);

insert into campo values (1033,1,'stock',103,1,0);

insert into campo values (1040,1,'idTipoProducto',104,1,1);

insert into campo values (1041,1,'nombreTipoProducto',104,1,0);

insert into campo values (1050,1,'idUbicacion',105,1,1);

insert into campo values (1051,1,'direccion',105,1,0); ”

En él dejamos la base de datos del sistema preparada para comenzar a crear las acciones.

Creación de las acciones:

-Ahora añadimos el insert correspondiente a la acción:

“INSERT INTO `mydb`.`accion`

(`idAccion`,`idGCampoPredef`,`idClaseAccion`,`idGCampoSelectPredef`,`idTablaAccion`,`idGCampoParametroPredef`,`unionAuto`,`nombre`,`descripcion`)

VALUES

(100,101,1, 100,null,102,null, "select productos de un almacen","selecciona todos los productos de un almacen por su id");”

-Creación de la select y configuración de los campos:

Para la select definimos las tablas que necesitará la acción (lo que sería el “from”):

“INSERT INTO `mydb`.`tablaaccion`(`idTabla`,`idAccion`) VALUES (102,100),(103,100);”

Las columnas seleccionadas (es decir, el “select”) y las condiciones (el “where”):

“INSERT INTO `mydb`.`contenidogcampo`

(`idGCampo`,`idCampo`,`idValorCampo`,`idAliasSelectValorCampo`,`idClaseContenidoGCampo`,`idCampoUnion`,`idContenidoGCampo`)

VALUES

--select

(100,1021,1,null,2,null,100), --nombreProducto

(100,1033,1,null,2,null,101), --stock

--where

(101,1032,1,null,3,null,102), --idAlmacen=pA.idAlmacen

(101,1031,1,null,2,1020,103) --pA.idProducto=p.idProducto”

En estas cabe destacar que se hace un insert por cada condición de la forma “campo=campo”.

-Añadir al menú:

Para que la acción aparezca en el menú correspondiente debemos añadirla a la tabla correspondiente indicando también la página de procesado que necesite:

“INSERT INTO 'mydb'.'contenidomenu' VALUES (100, 1, 1, 100, null, 'select productos de almacen', null, 1, null, null);”

Con esta acción en concreto añadimos una acción al menú 1 para obtener una tabla con los productos del almacén indicado por el usuario.

Además de esta, configurado en el script adjunto (…), añadimos otra para obtener un gráfico de los mismos datos y otra para inserter productos.

CAPÍTULO 6. USO DEL SISTEMA

USO CONFIGURADOR

Como dijimos, no hemos implementado una interfaz para las tareas asignadas al Configurador, de modo que en esta sección se describe la forma correcta para realizar las configuraciones sobre el sistema, especialmente incluir una nueva Acción. Además de servir como una referencia de uso para este perfil, esta guía puede usarse de la misma manera para saber de qué forma debería manipularse la base de datos de nuestro sistema a la hora de desarrollar una interfaz para el Configurador.

Asumimos que todos los campos, tablas y bases de datos que vayamos a usar en nuestra Acción están ya registrados en nuestro sistema y que podemos obtener sus identificadores en las tablas: Campo, Tabla y Base de Datos respectivamente.

-Creación de una Acción.

Vamos a organizar la creación de la Acción de forma que finalmente obtengamos un script SQL que ejecutemos sobre la base de datos.

En primer lugar creamos los grupos de campos que necesitamos para la Acción:

Suponemos que todos los grupos creados son independientes y no queremos heredar de uno anterior. En caso de querer usar esta opción deberíamos indicar su id en lugar de null. Además si no queremos usar alguno de los grupos no necesitamos crearlo. Esto es especialmente aplicable para el grupo de campos Parámetros, pues las páginas de procesado incluidas no requieren ningún parámetro adicional.

Los identificadores idGCampoSelect, idGCampoWhere, idGCampoParametros serán números enteros elegidos de forma que no coincidan con otros Grupos de Campos.

INSERT INTO `mydb`.`gcampo`

(`idGCampo`, `idGCampoBase`)

VALUES

(

idGCampoSelect, null

),

(

idGCampoWhere, null

),

(

idGCampoParametros, null

);

Creamos la Acción:

Para ello usaremos los identificadores de Grupo de Campos anteriores. Si hemos decidido prescindir de alguno, usaremos el identificador 1 correspondiente a un Grupo de Campos sin elementos. De nuevo el idAccion tendrá que ser elegido de forma que no esté ya usado por otra Acción. Para el idClaseAccion usaremos 1 para una select y 2 para un insert. El idTablaAccion será el identificador de la tabla sobre la que realizar el insert y en caso de ser una select pondremos null. Por último para hacer uso de la característica de unión automática de tablas pondremos 1 en unionAuto.

INSERT INTO `mydb`.`accion`

(`idAccion`,

`idGCampoPredef`,

`idClaseAccion`,

`idGCampoSelectPredef`,

`idTablaAccion`,

`idGCampoParametroPredef`,

`unionAuto`,

`nombre`,

`descripcion`)

VALUES

(

idAccion,

idGCampoWhere,

<1 ó 2>, -- 1 para select y 2 para insert.

idGCampoSelect,

idTablaAccion,

idGCampoParametros,

<0 ó 1>, -- 1 para usar union automática.

nombre,

descripcion

);

Ahora empezamos la creación de la select como tal. En primer lugar seleccionaremos las tablas. Haremos un insert por cada tabla que queramos incluir:

INSERT INTO `mydb`.`tablaaccion`

(`idTabla`,

`idAccion`)

VALUES

(

idTabla,

idAccion,

);

A continuación vamos a proceder a la configuración de los campos. Empezaremos por los campos del Grupo de Campos del where. Por cada filtro o unión haremos un insert, asegurandonos de nuevo que el id asignado no está usado ya. Como vimos, podremos configurarlos como fijos, rellenos por la query o rellenos por el usuario Para ello usaremos el idClaseContenidoGCampo: 1 – Fijo, 2 – Relleno por la query, 3 – Relleno

por el usuario. El idCampo, será el id del campo al que estamos haciendo referencia en la tabla Campo.

Además para usar alias en el select o para hacer un filtro con una constante, necesitaremos rellenar ese valor en la tabla ValorCampo y asignar el id resultante en idValorCampo o idAliasSelectValorCampo. Si no vamos a usar esto, podemos dejarlo a null, el valor será ignorado en ese caso. Si el filtro va a ser con otro campo, usaremos idCampoUnion para especificarlo.

INSERT INTO `mydb`.`contenidogcampo`

(`idGCampo`,

`idCampo`,

`idValorCampo`,

`idAliasSelectValorCampo`,

`idClaseContenidoGCampo`,

`idCampoUnion`,

`idContenidoGCampo`)

VALUES

(

idGCampoWhere o idGCampoSelect o idGCampoParametros

idCampo,

idValorCampo,

idAliasSelectValorCampo,

<1, 2 ó 3>, --1 – Fijo, 2 – Relleno por la query, 3 – Relleno por el usuario.,

idCampoUnion,

idContenidoGCampo

);

Finalmente, para que la acción aparezca en la interfaz, necesitamos insertar en la tabla ContenidoMenu. Supondremos que el menú ya está creado así como el rol al que se va a

asignar y escribiremos sus identificadores en el insert. En este insert podremos especificar otros grupos de campos en vez de los predefinidos que hemos puesto al insertar la accion. Como en principio estamos haciendo una acción desde la base, podemos dejarlos a null para que tomen los predefinidos. El idContenidoMenu no deberá existir como ya estamos acostumbrados. El nombre será el que se le presente al usuario en la interfaz y la descripción proporciona información extra sobre el objetivo de esa acción. Por último especificaremos la página de procesado que deseamos para la acción. Es en este punto en el que estamos decidiendo si queremos que los resultados se muestren como un gráfico o como una tabla. Los identificadores asignados de base a las páginas de procesado son 1 – tabla, 2 – gráfico.

```
INSERT INTO `mydb`.`contenidomenu`
(`idContenidoMenu`,
`idRol`,
`idMenu`,
`idAccion`,
`idGCampo`,
`nombre`,
`idGCampoSelect`,
`idPaginaProcesado`,
`idGCampoParametro`,
`descripcion`)
VALUES
(
idContenidoMenu,
idRol,
idMenu,
idAccion,
null,
nombre,
```

null,

idPaginaProcesado, -- 1 – tabla, 2 – gráfico, otro id para páginas creadas.

null,

descripción

);

Con esto tendríamos completada nuestra Acción e incluida en la interfaz.

Como hemos visto a lo largo del proceso, podemos partir de una Acción ya hecha o un Grupo de Campos para simplificarlo. Por ejemplo, si solo queremos configurar una Acción con un distinto tipo de representación solo tendremos que asociar en un nuevo Contenido Menu, la misma Acción con otra página de procesado. Si queremos llamar a una misma Acción con un parámetro más, podemos crear un nuevo Grupo de Campos que herede del de la Acción y añada el parámetro. Después simplemente configurariamos un nuevo Contenido Menu con la misma Acción pero suministrando el identificador del nuevo Grupo, con lo que el Motor tomaría este y no el predefinido en la Acción.

USO DESARROLLADOR

En esta sección, proporcionamos las localizaciones de los elementos que más comúnmente serán creados o cambiados por los desarrolladores y proporcionamos indicaciones para esta tarea.

La forma más directa y sencilla de hacer modificaciones a nuestro sistema son las páginas de procesado. Las que se proporcionan de base se encuentran al mismo nivel que las páginas principales, pero pueden situarse en cualquier ruta válida dentro del servidor. Además para que sean reconocidas por el sistema y se puedan asociar a Acciones, deben incluirse su ruta con un identificador en la tabla paginaProcesado. Este número será el que se use cuando se configuren las Acciones.

Para implementar una página de procesado tenemos que tener en cuenta los siguientes aspectos:

Entrada de información a la página de procesado:

El motor de querys hace una petición a la página de procesado asociada a la Acción con el resultado en forma de JSON como parámetro. Este JSON tiene dos elementos en la raíz, cabeceras y datos. El primero es una lista del nombre de las cabeceras de los datos y el segundo es una lista, de listas conteniendo los datos, siendo las listas más internas lo equivalente a una tupla o una fila de la base de datos. El JSON tendía el siguiente aspecto por tanto:

{"cabeceras":["col1","col2","col3"],

"datos":[["val11","val12","val13"],["val21","val22","val23"]]}

Además se incluyen como parámetros los campos determinados en el grupo de campos GCampoParametros. Los parámetros no se pasan como en una petición GET y POST (es decir no se recogen, en código java con: request.getParameter) pues la petición sigue siendo la misma. No se hace una petición nueva sino que se mantiene la enviada por el usuario y se agregan atributos a esta petición que las páginas de procesado pueden recoger. (Con la instrucción: request.getAttribute). En resumen, se usan atributos de petición en lugar de parámetros propiamente dichos.

Salida de información a Principal.jsp:

La página Principal.jsp está preparada actualmente para generar tablas y gráficos. Por lo tanto la página de procesado envía un JSON con los campos necesarios para la configuración añadiendo uno que distinga. Este es "tipoRespuesta". Si este campo contiene "grafico" la página principal actuará como si recibiera los datos correspondientes a un gráfico y si recibe "tabla" como si recibiera los de una tabla.

Por tanto si nuestro objetivo es ampliar con un nuevo tipo de gráfico, la página de procesado debería seguir devolviendo "grafico" en tipoRespuesta pues el API Highcharts ya incluye la distinción entre gráficos como campo necesario para la creación de estos en el JSON. Simplementa deberíamos crear una página similar a la actual de gráficos (además la estructura del JSON para highcharts es similar para todos ellos, por lo que se podría reutilizar código con llamadas a varias páginas de procesado).

Si el objetivo es crear una nueva estructura alternativa a la tabla, incluida en dhtmlx, habría que integrarla en la interfaz, de modo que deberíamos modificar Principal.jsp para que reconociera una nueva etiqueta e hiciese lo necesario para gestionar un nuevo tipo de presentación.

Por último como guía para cualquier otra modificación se puede partir del código en Principal.jsp utilizado para gráficos, en los que se crea un nuevo elemento en la estructura HTML de la página para trasladarla a una de las ventanas de dhtmlx. En estas ventanas se mostrará cualquier otro elemento de visualización que se pretenda crear.

Como ejemplo ilustrativo, propongamos algo ligeramente más complejo: dejar que el usuario elija el gráfico que desee. Para esto, usaremos el grupo de campos GCampoParametros, en el que el usuario especificará el gráfico deseado. Esta información se pasará a la página de procesado determinada. En esta implementación proponemos crear una página de procesado de gráficos principal, que en realidad no genere información de salida sino que según este parámetro, derive a otras en las que se implementa el gráfico determinado. La que se asociaría a la Acción por tanto sería la página principal de gráficos.

Para cambios en los tipos de campo, la página indicada es JSONFormForm.jsp, donde se procesan los campos que se mostrarán en el formulario. Actualmente la página toma como nombre del tipo de campo para el API dhtmlx el que aparezca en la base de datos del sistema como nombre del tipo de campo. Si nos basta con los campos ofrecidos por dhtmlx, simplemente añadiriamos el nombre que toma en el API el tipo de campo que deseamos incluir en la tabla TipoCampo.

Si queremos poder poner distintos nombres a los campos con el objetivo de distinguir sus distintas variantes, deberíamos añadir casos en la página que decíamos antes, de forma que se distinga en función del nombre y se apliquen la configuración correspondiente.

Por último para cambios en el propio Motor recomendamos usar las tablas denominadas clase. (estas son las que comienzan por clase seguido por el nombre del elemento al que dan la distinción). El objetivo es añadir en estas tablas nuevos clases con las que relacionar las modificaciones que hagamos y al cambiar el código en las distintas páginas, respetar el funcionamiento para las clases ya proporcionadas.

INSTALACIÓN

Describimos a continuación la forma indicada para instalar el sistema. Los requisitos necesarios son un servidor web Apache-Tomcat y una base de datos MySQL. Si se tiene

una base de datos ya desarrollada para la aplicación cliente, esta tiene que estar en el mismo servidor MySQL que en el que se va a instalar el sistema.

El primer paso es iniciar el servicio MySQL y exportar el fichero "instalacionBBDD.sql", proporionado. Esto creará la estructura de la base de datos de nuestro sistema así como las opciones incluidas por defecto. En este punto, si ya tenemos preparada la base de datos cliente, la podemos exportar a este servidor MySQL.

Proporcionamos también un script que genera en las tablas BaseDatos, Tabla y Campo la información correspondiente a la base de datos cliente que le indiquemos. Si, como decimos, la base de datos de la aplicación cliente ya está preparada, podemos ejecutar este script ahora para dejar preparada la base de datos del sistema. En caso de no estarlo, recomendamos continuar con la instalación y ejecutar este script, una vez lo esté y antes de empezar a desarrollar Acciones en el sistema.

Una vez preparada la base de datos, tenemos que alojar el sistema en el servidor. Para un entorno de pruebas podemos hacerlo en localhost o si disponemos de un alojamiento con Apache-Tomcat disponible, podemos subirlo allí. El fichero viene proporcionado en forma de instalacionSistema.zip en la que se incluye el código, y en forma de fichero WAR para su subida directa al servidor (instalacionSistema.war. En esta última solo se incluye un ejecutable).

Tras esto ya tendríamos disponible el sistema. Podemos acceder a http://localhost:8080/Proyecto/Principal.jsp para acceder a la aplicación (en caso de haberla instalado en un servidor local). Observaremos un panel sin menús pues aún no están configurados. A partir de aquí comenzaría la labor de configuración de Acciones, Usuarios, etc.

RESULTADOS DE LAS PRUEBAS

Como ya hemos comentado anteriormente, para poder mostrar el funcionamiento del proyecto de una forma más clara, decidimos hacer una serie de pruebas con una base de datos de ejemplo. Dicha base de datos también viene más atrás, en el apartado "Prueba"

del capítulo 5. Y, como se puede observar, representa la base de datos de una supuesta tienda de hardware.

Ahora pasamos a detallar los resultados de las pruebas:

Prueba 1:

Para la primera prueba hemos de configurar la acción correspondiente en la base de datos de la aplicación. A continuación vamos a la página principal, seleccionamos en el menú 1 la acción "select productos de almacen", introducimos el id del almacén y pulsamos el botón "Enviar".

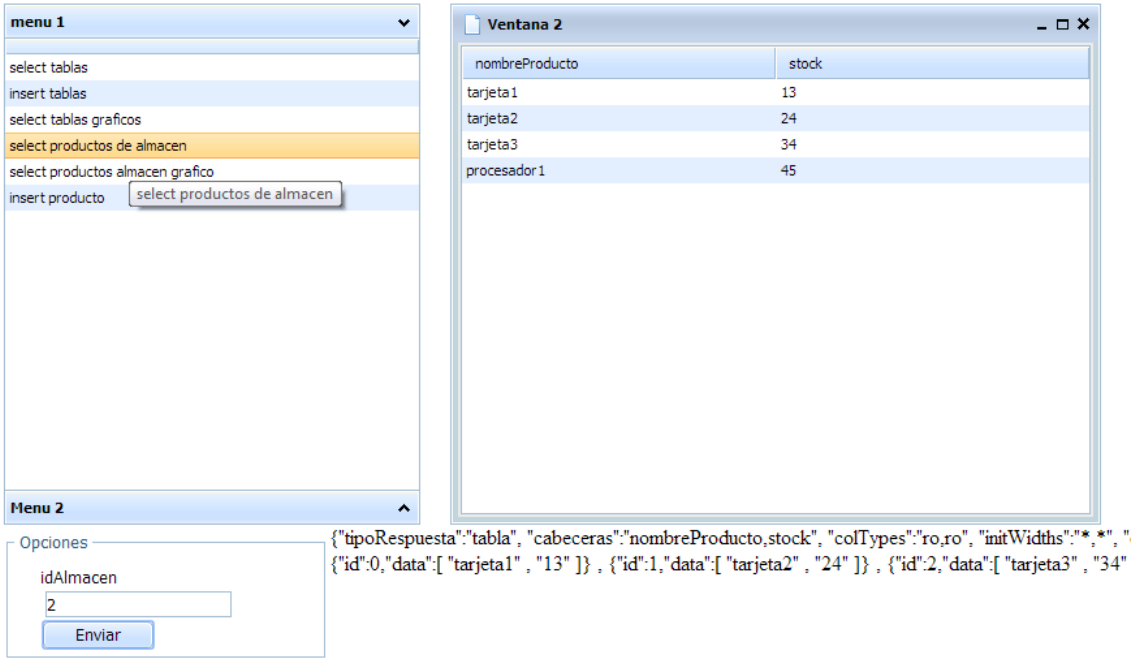

ILUSTRACIÓN 16 : RESULTADO PRUEBA 1.

En la imagen podemos observar como la salida es la esperada. Una tabla con los productos del almacén indicado en una ventana auxiliar.

Prueba 2:

Configuramos nuestra segunda acción en la base de datos y pasamos a probarla desde la aplicación del proyecto. Basta con ir al menú 1 y seleccionar la acción "select productos

almacen grafico", rellenar el id del almacén del que queramos obtener los productos y pulsar el botón "Enviar".

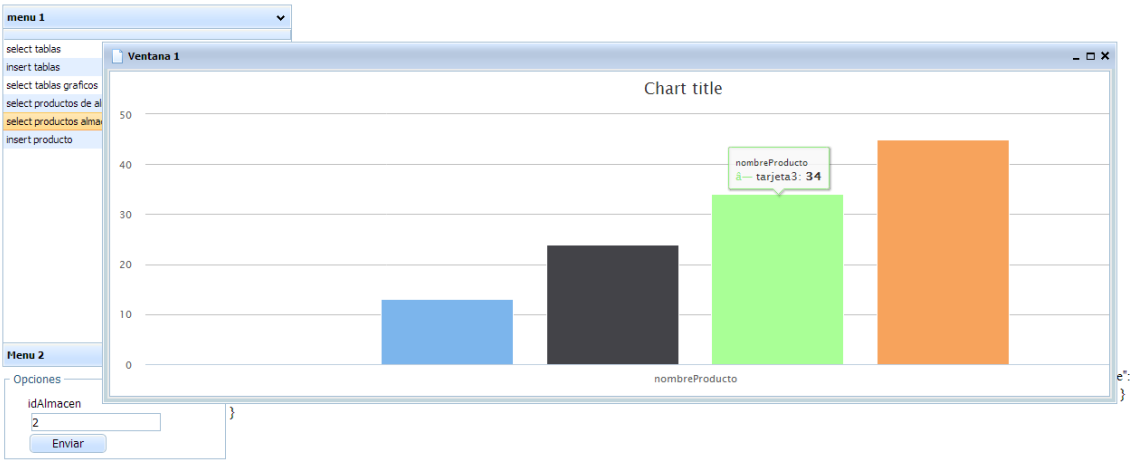

ILUSTRACIÓN 17 : RESULTADO PRUEBA 2.

Podemos observar como el resultado es el esperado, un gráfico con los productos que se encuentran en dicho almacén. Mostrado en una ventana auxiliar.

Prueba 3:

Tras haber configurado la acción para insertar nuevos productos, pasamos a la página principal de nuestra aplicación, seleccionamos la acción adecuada ("insert producto") y rellenamos los campos adecuadamente. Ahora pulsamos el botón de enviar y el producto será añadido a la base de datos, esto lo podemos comprobar, por ejemplo, inspeccionando la base de datos.

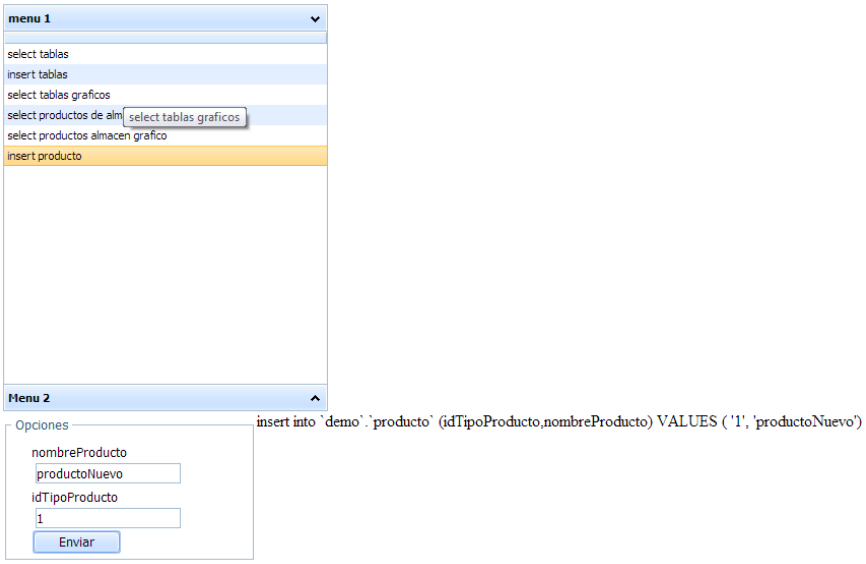

ILUSTRACIÓN 18 : RESULTADO PRUEBA 3

7. CONCLUSIÓN

En este capítulo vamos a exponer las conclusiones obtenidas de la realización de este proyecto. Hablaremos de la forma en la que hemos cumplido los requisitos que nos habíamos puesto, como también nuestras expectativas en cuanto al alcance del proyecto. Trataremos también todas las vías de desarrollo que hemos dejado abiertas y finalmente los problemas que nos hemos encontrado en la realización del trabajo y los puntos que consideramos mejorables en él.

Como hemos visto a lo largo de esta memoria, podríamos dividir nuestra aplicación en tres partes: interfaz, motor de querys y control de usuarios. La primera y la tercera se definían claramente en los requisitos mientras que la segunda nace en la fase de diseño a partir del concepto de Acción el cual también estaba incluido en la especificación. Podemos decir por tanto que la toma de requisitos nos ha servido como una hoja de ruta bastante eficaz a la hora de realizar el proyecto. A continuación analizamos en detalle cada una de estas partes:

La interfaz obtenida finalmente consideramos que cumple adecuadamente su cometido. Gracias al API dhtmlx pudimos construirla de forma sencilla como vimos en implementación y de forma que obtuvimos resultados estéticamente aceptables. Este API ofrece de base una interfaz en colores azules neutros y que a nuestro juicio resulta moderna y atractiva, y además como vimos en la descripción que ofrecimos, permite usar “skins” para cada uno de los elementos de interfaz que se usen. Las skins son combinaciones nuevas de colores y efectos que permiten variar la forma en que se visualizan los elementos, lo cual permite dar un toque diferente a la apariencia usando una de las muchas que existen en internet o incluso personalizarla a fondo creando una skin propia.

La interfaz cumple además con nuestra intención de ser un elemento suficientemente independiente del resto de la aplicación. Esto quiere decir que en caso de ser necesario se podría diseñar una interfaz nueva, y seguir usando la misma funcionalidad del resto del sistema. El único punto que está inevitablemente unido al modo en que se organiza la base de datos es el concepto de tipo de campo que se usa para personalizar como se pide la entrada de datos al usuario.

Además, con poco trabajo por parte de un programador web, se podría incrustar nuestra aplicación en cualquier página web, aunque no ha sido nuestra prioridad en el desarrollo el potenciar este aspecto de adaptación.

Por otro lado la parte de Páginas de Procesado, es una herramienta muy potente que añadimos en la fase de diseño en un principio para cubrir la necesidad de mostrar gráficos distintos. Aunque como vimos finalmente solo desarrollamos un tipo de gráficos, la ampliación a otros es casi trivial. Las páginas de procesado son uno de los puntos que consideramos más interesantes de nuestro sistema pues sobrepasan el objetivo con el que fueron diseñadas y permiten gran flexibilidad a la hora de procesar y mostrar los datos obtenidos.

El motor de querys es el punto que se describía de forma más abstracta en los requisitos. Esto es especialmente patente en la potencia que queríamos otorgar a las Acciones. Mientras que en los requisitos la equiparamos al lenguaje SQL, más tarde fuimos limitándola según iba creciendo la complejidad del sistema. Las principales limitaciones son la falta de ciertos elementos muy usados en SQL: group by, funciones como nvl o if, sentencias update y delete y operadores distintos del igual. Esto ocurrió debido a que en la especificación únicamente se hablaba del concepto de acción y es en la fase de desarrollo en la que decidimos incluir el elemento del motor de querys.

Nuestra motivación para desarrollar un sistema capaz de construir querys fue especialmente tener una forma de combinar los parámetros que el Configurador quería que los usuarios pudiesen pasar a una Acción, con la funcionalidad que se quería implementar con esa Acción. Esto llevó a generar un sistema entero de querys en el que se pudiese señalar que campos son los que debían aparecer al usuario como parámetros. Además pensamos que esta forma de construir una query podría facilitar al rol del Configurador su tarea.

Finalmente según iban complicándose las estructuras necesarias para mantener las querys de las Acciones en la base de datos, comenzamos a pensar que quizás estábamos haciendo lo contrario a lo pretendido y estabamos complicando la tarea de configurar Acciones. Este hecho, unido a que no podíamos ofrecer toda la potencia del lenguaje SQL que habíamos nombrado en los requisitos, fue lo que nos llevó a diseñar un tipo de Acción en la que se pudiese directamente incluir código SQL aunque perdiendo la posibilidad de incluir parámetros. Con este tipo de Acción cubríamos un doble objetivo: simplificar al Configurador la creación de Acciones complejas con el motor de querys y permitir aquellas que fuesen imposibles con éste.

Sin duda este es uno de los puntos que peor valoramos de nuestro diseño pues nos hubiera gustado poder diseñar un sistema que no tuviese que recurrir a este tipo de Acciones como veremos en la sección de Desarrollos futuros.

Por otro lado al problema de la potencia de las querys se le unió la falta del desarrollo de una interfaz para el Configurador. Dado que en los requisitos no se especificaba como iba a interaccionar este perfil con el sistema, dejamos la creación de esta interfaz como una tarea secundaria que finalmente no llegó a realizarse. Pensamos también en usar nuestro propio sistema de motor de querys para generar la interfaz, pero aunque fuese viable teóricamente, en la práctica nos fue imposible por el problema anteriormente descrito de la potencia no equiparable a SQL. Finalmente decidimos no proporcionar interfaz y que la configuración se hiciese a través de las cargas a la base de datos directamente, con lo que se aumentaba la complejidad de nuevo de la configuración del sistema.

Con control de usuarios nos referimos a todo lo especificado en cuanto a Roles, Menus y Usuarios. En este apartado los requisitos han sido cumplidos en su totalidad, permitiendo la gestión de usuarios y roles que prentendíamos. Además como vimos en el diseño, la forma en que permitimos configurar los ContenidosMenu y los Tipos de Campo dotan al sistema de mucha flexibilidad, lo cual es siempre deseable en una aplicación de la naturaleza de la nuestra.

Tras este analisis exhaustivo de lo especificado y diseñado, debemos indicar que en nuestra opinión nuestro sistema, aunque es operativo en el sentido de que funciona tal y como describimos, no es todavía usable de forma que los perfiles que detallábamos en los requisitos, en especial Configuradores, puedan usarlo de forma eficiente. Esto es debido a la complejidad acumulada especialmente en la parte del motor de querys. Sin embargo, dado que el problema es la complejidad, las soluciones necesarias para hacer de este un sistema válido no necesitan reformar el código actual sino facilitar, con la implementación de una interfaz para Configurador por ejemplo, la forma de trabajar con el sistema.

Podemos distinguir por tanto las siguientes áreas en las que realizar extensiones a nuestro sistema:

Interfaz para Configurador. Por los motivos que hemos explicado anteriormente, es necesario para constituir una aplicación usable, la implementación de una interfaz para realizar las tareas del Configurador. La forma en la que se puede hacer esta interfaz es muy diversa, puede ser desde una interfaz directa a cada tabla (con lo que algunos de los problemas de usabilidad continuarían presentes) hasta un sistema más complejo que a ojos del Configurador solo trate con los elementos que el conoce, es decir Acciones, Roles, etc.

Nuestra idea siempre ha sido la simplificación de la forma de uso por lo que nos inclinamos más por la segunda opción. Además creemos que la inclusión de varios niveles de complejidad en la interfaz de configuración harían mucho más amigable el sistema. Esto es, que para las configuraciones más comunes, se pueda acceder a menús simples y directos y que las que requieran más complejidad tengan sus interfaces específicas, quizás reservadas para los Configuradores más experimentados.

La forma de simplificar la configuración en nuestro sistema es en muchos de los casos tomar las opciones por defecto en lugar de que lo haga el propio Configurador. Por ejemplo se podría desarrollar una interfaz para crear acciones rápidamente en la cual siempre se crearan grupos de campos nuevos (ignorando la herencia) y se usaran siempre los grupos por defecto de la acción (sin especificar nada en el contenidoMenu).

Para terminar con este punto, consideramos adecuado señalar la adecuación de los asistentes paso a paso para nuestro sistema. Unos cuantos asistentes que ayudaran a las principales configuraciones explicando que se está haciendo en cada uno de los pasos ayudarían enormemente a familiarizarse con la aplicación en los primeros usos.

Incremento de la potencia del Motor de querys. Ya hemos visto que aspectos del lenguaje SQL se han quedado sin implementar en nuestro sistema. Esta extensión agruparía su progresiva inclusión en el Motor de querys. Para las más simples, como la inclusión de delete y update, bastaría con añadir la forma en la que se organizaría el código para estas instrucciones. Para otras como el group by es posible implementarlas agregando relaciones entre objetos de la base de datos. Y por último para la inclusión de funciones necesitariamos revisar la forma en que los campos de las tablas y los filtros para cada Acción se generan, con lo que tendríamos que rediseñar en cierta medida el Motor entero.

Mejoras en la interfaz de usuario. Incluimos en este apartado la creación de mayor diversidad de gráficos y tipos de campos. Con la inclusión del group by en la potencia SQL que veíamos en el punto anterior, se daría pie al uso de muchos más tipos de gráficas que las proporcionadas actualmente.

Por otro lado entre los tipos de campos que se pueden desarrollar destacan aquellos que necesitan acceder a la base de datos cliente (como una lista o un combobox), pues esta operación no es trivial como sí lo sería la inclusión de un calendario como tipo de campo. Para realizar esto se necesita revisar la forma en que se crea el formulario de las acciones.

Además para el ejemplo de las listas se plantearía otro problema: el uso de identificadores en las tablas de las bases de datos. Con esto queremos decir que si hacemos una lista con los identificadores el usuario final no podrá saber fácilmente que elemento esta seleccionando y si lo hacemos con un nombre (por ejemplo) este no tiene porqué ser un identificador único a la hora de realizar una Acción.

Mejorar la extensibilidad. Finalmente este punto trata de como facilitar la tarea de un Desarrollador que quiera incluir cambios más profundos que la implantación de nuevos gráficos o nuevos tipos de campos. Por ejemplo podemos pensar que se puede necesitar dejar de usar el API Highcharts para pasar a usar cualquier otro de los existentes. Tal y como se presenta al estructura del sistema actualmente, para ello se requeriría cambios en distintos puntos del código.

El objetivo de esta extensión sería plantearse esta posibilidad y encapsular el código de cada una de las APIs y módulos que definen nuestro sistema para que se pudiesen usar, cambiar o crear nuevos cada uno por su lado.

APÉNDICE

Highcharts(Tipos de gráficos soportados)

Highcharts soporta diferentes tipos de gráficos que pueden ser mostrados de manera significativa y que posteriormente hablaremos detalladamente de cada uno de ellos. [1]

Line Chart

Line chart se representa por una serie de puntos conectados por líneas. Es utilizado a menudo para visualizar datos que cambian a lo largo del tiempo.

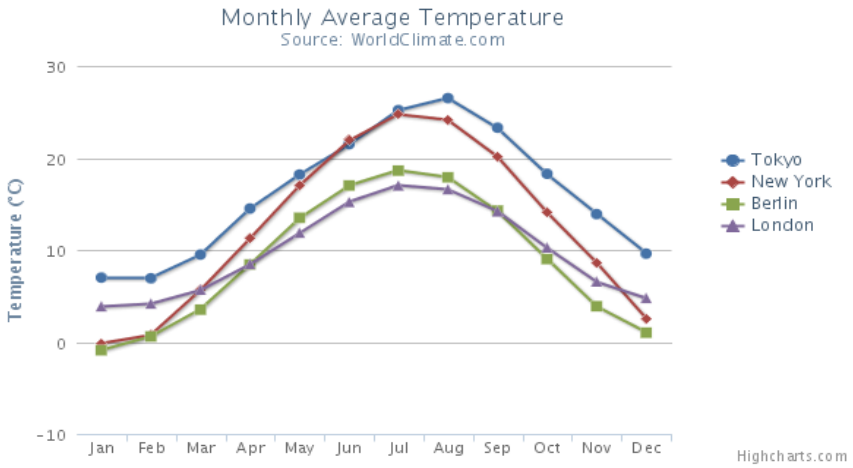

ILUSTRACIÓN 19 : LINE CHART.

Spline Chart

Spline chart dibuja una linea curva entre los puntos de una serie de datos.

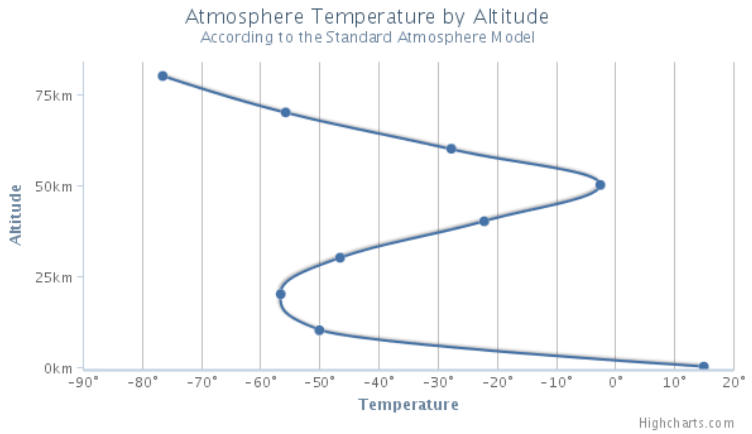

ILUSTRACIÓN 20 : SPLINE CHART.

Area Chart

Realiza las mismas funciones que Line chart pero rellenando el área comprendida entre las líneas.

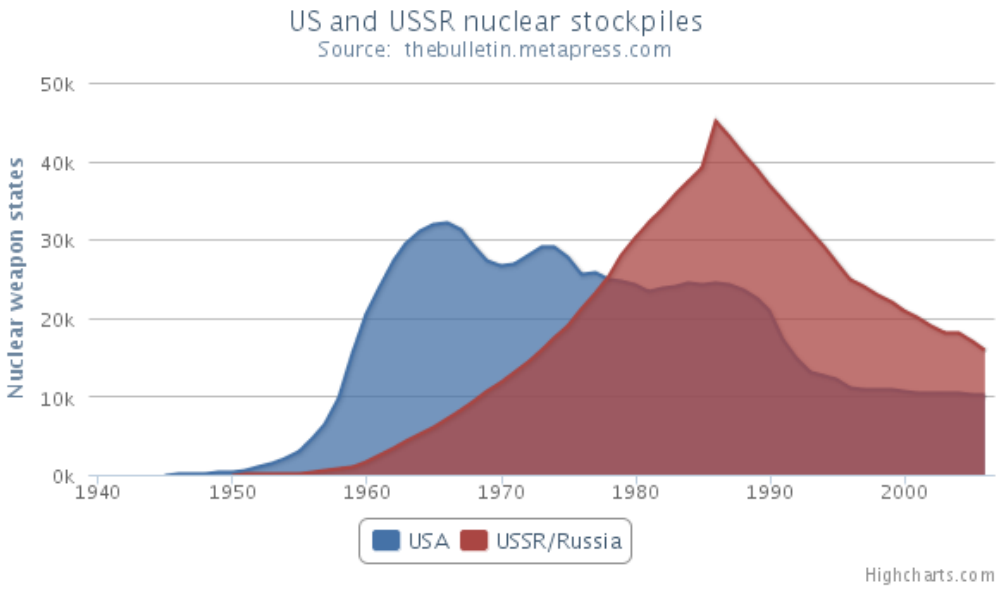

ILUSTRACIÓN 21 : AREA CHART.

AreaSpline Chart

Es lo mismo que Area Chart, sólo que la linea es una ranura en vez de líneas rectas.

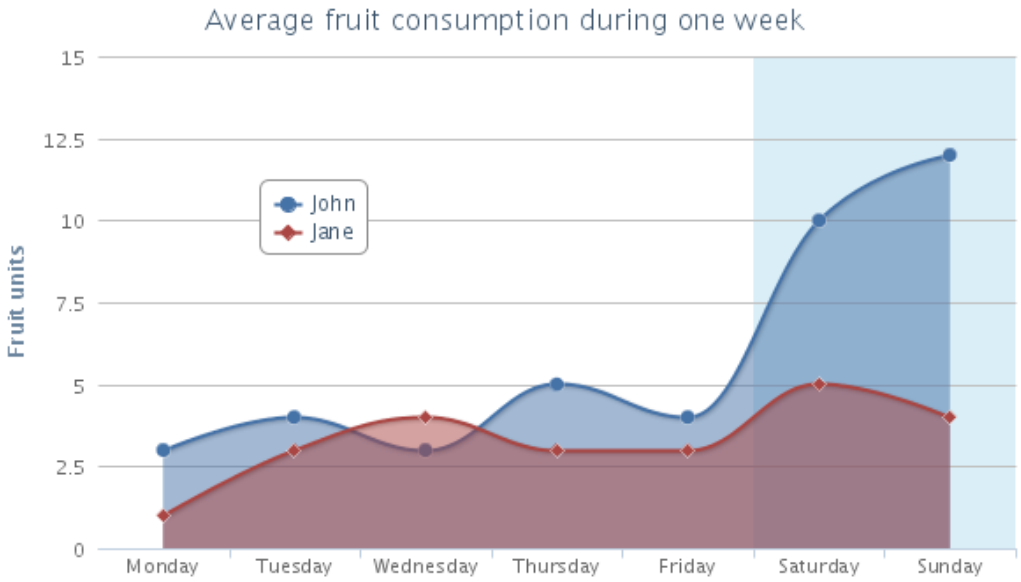

ILUSTRACIÓN 22 : AREASPLINE CHART.

Column Chart

Column chart muestra los datos en barras verticales.

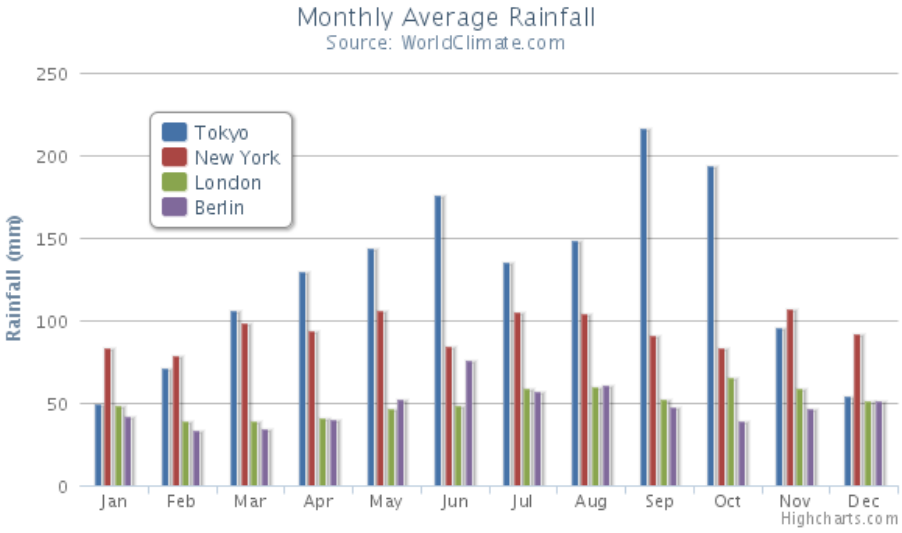

ILUSTRACIÓN 23 : COLUMN CHART.

Bar Chart

Es exactamente lo mismo que column chart únicamente que los ejes x e y están cambiados.

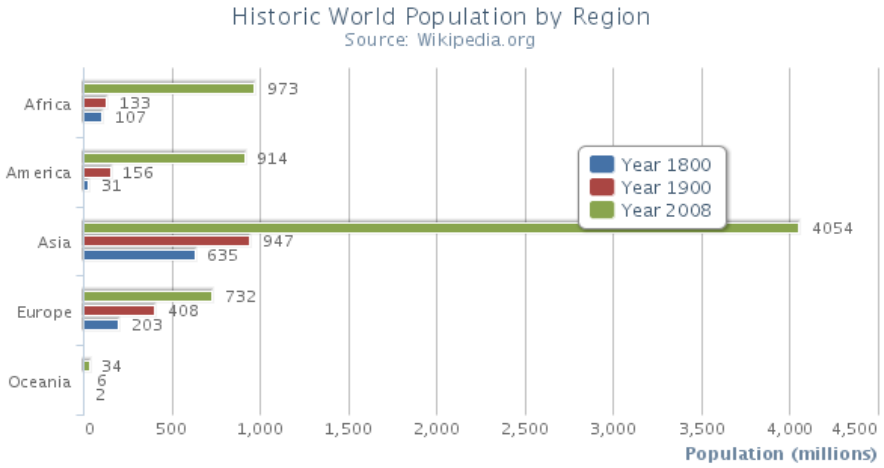

ILUSTRACIÓN 24 : BAR CHART.

Pie Chart

Pie chart es un gráfico circular que se divide en diferentes sectores en proporción a la cantidad del total que representa.

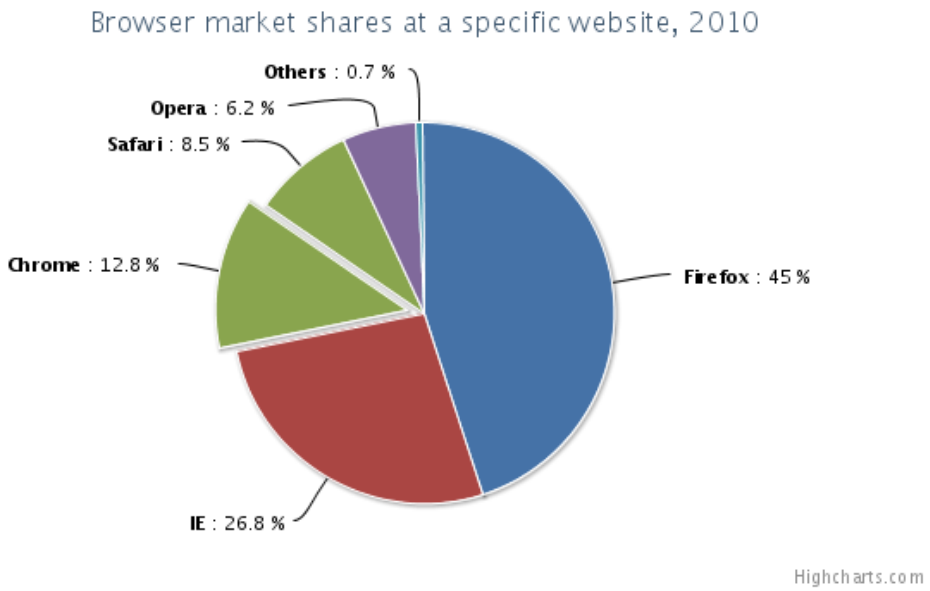

ILUSTRACIÓN 25 : PIE CHART.

Scatter Chart

Dibuja un punto por cada dato en series y sin conexiones entre ellos.

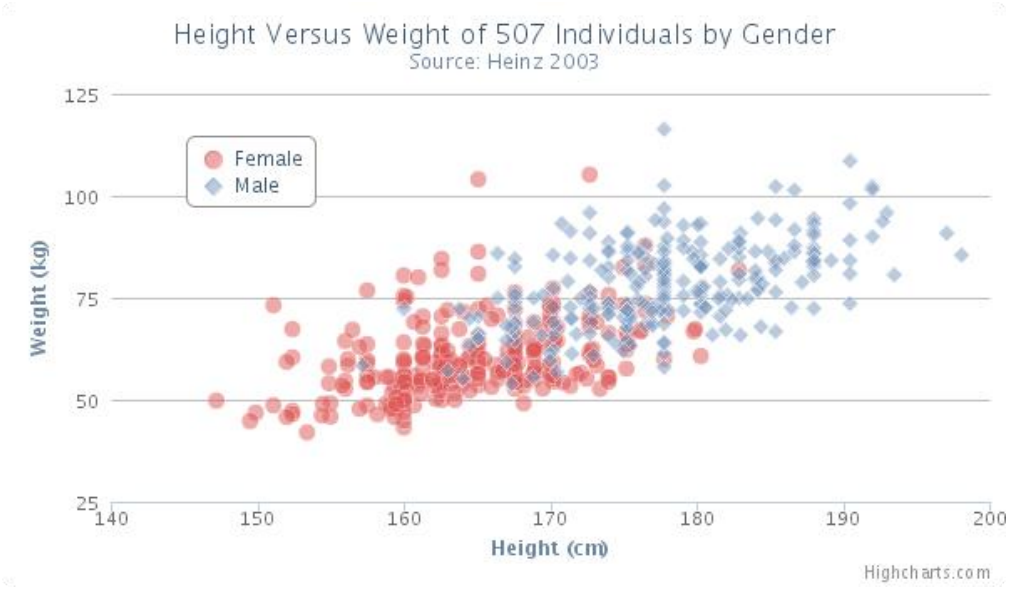

ILUSTRACIÓN 26 : SCATTER CHART.

Polar Chart

Polar charts requiere el archivo highcharts-more.js para su utilización.

Se tiene ua gran preocupación por reutilizar las opciones existentes para el diseño del conjunto de opciones de polar charts. El resultado es que únicamente tenemos que cambiar los ejes X-Y a coordenadas polares.

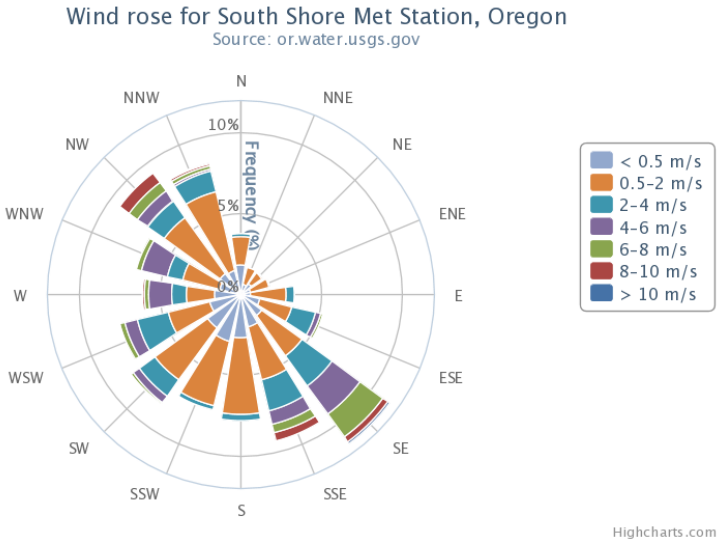

ILUSTRACIÓN 27 : POLAR CHART.

Angular Gauges

También conocidos como marcadores que proporcionan una gran visualización en cuadros de mando. Como con polar charts, el objetivo principal es ampliar las opciones existentes de otros tipos de gráficos.

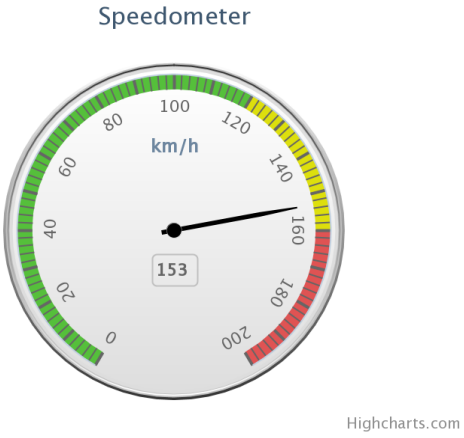

ILUSTRACIÓN 28 : ANGULAR GAUGES.

Range Series

Range series requiere el archivo highcharts-more.js para su utilización.

Dentro de este tipo de gráficos encontramos subtipos como “arearange”,”areasplinerange” y “columnrange”. Los puntos de range series pueden ser definidos por objetos ({x: 0, low:1, high: 9}) o por arrays ([0,1,9]). En cualquier caso, el valor de x puede estar omitido.

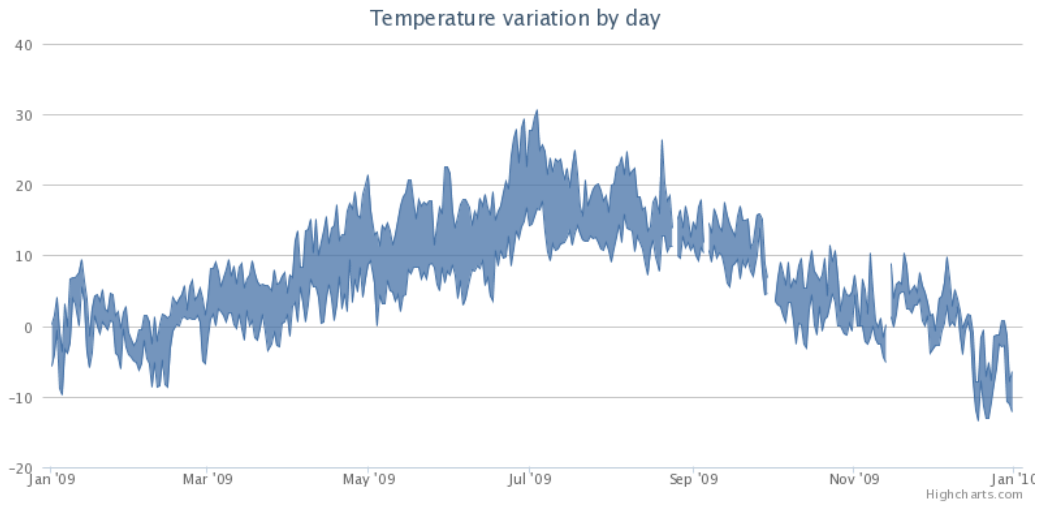

ILUSTRACIÓN 29 : RANGE SERIES.

BIBLIOGRAFÍA

[1] Chart types,Highcharts.

Available: http://www.highcharts.com/docs/chart-and-series-types/chart-types

[2] What is Eclipse and the Eclipse Foundation?

Available: http://www.eclipse.org

[3] Blog Historia de la Informática (4 enero 2011)

Available: http://histinf.blogs.upv.es/2011/01/ 04/historia-de-las-bases-de-datos/

[4] Luján Mora Sergio,Bases de datos en entorno Internet. Universidad de Alicante.curso 2001-2002 .

Available: http://rua.ua.es/dspace/bitstream/10045/13987/2/1a-bases-datos-internet.pdf

[5] MySQL Products.

http://www.mysql.com/products/workbench/

[6] Acerca de,Apache Friends.

Available: https://www.apachefriends.org/es/about.html

[7] JavaScript Component Library for Building Rich Web Apps,DHTMLX Suite

Available: http://dhtmlx.com/docs/products/dhtmlxSuite/index.shtml

[8] Mozilla Developer Network. Acerca de JavaScript.(9 nov 2011).

Avalable: https://developer.mozilla.org/es/docs/JavaScript/Acerca_de_JavaScript

[9] Introducing JSON

Available: http://www.json.org/

[10] What is highcharts?, Highcharts product

Available: http://www.highcharts.com/products/highcharts

[11] About MySQL

Available: http://www.mysql.com/about/

[12] Murray Greg. Asynchronous JavaScript Technology and XML (Ajax) With the Java Platform (9 jun 2005)

Available: http://www.oracle.com/technetwork/articles/javaee/ajax-135201.html

[13] ¿Qué es la tecnología Java y para qué la necesito?

Available: http://www.java.com/es/download/faq/whatis_java.xml

[14] Conozca más sobre la tecnología Java

Available: http://www.java.com/es/about/

[15] Java Servlet Technology Overview

Available: http://www.oracle.com/technetwork/java/javaee/servlet/index.html

[16] Motor de juegos Unity

Available: http://unity3d.com/unity

[17] FrontPage

Available: http://donwebayuda.com/que-es-frontpage/

[18] WordPress

Available: http://es.wordpress.org/

[19] Joomla!

Available: http://www.joomla.org/about-joomla.html

[20] Drupal

Available: https://drupal.org/

[21] vBulletin

Available: http://www.vbulletin.com/en/learn-more/

[22] BoxBilling

Available: http://www.boxbilling.com/

[23] Martin Fowler. Inversion de Control

Available: http://martinfowler.com/bliki/InversionOfControl.html

[24] Facebook Developers

Available: https://developers.facebook.com/

[25] API Google Maps

Available: https://developers.google.com/maps/?hl=es

[26] JQuery

Available: http://jquery.com/

[27] cURL

Available: http://curl.haxx.se/

[28] Apache Server

Available: http://httpd.apache.org/

[29] Ruby on rails

Available: http://rubyonrails.org/

[30] Symfony

Available: http://symfony.es/

[31] Yii

Available: http://www.yiiframework.com/

[32] Codeigniter

Available: http://ellislab.com/codeigniter/user-guide/

[33] Spring

Available: http://spring.io/

[34] Django

Available: https://www.djangoproject.com/

[35] Pylons

Available: http://www.pylonsproject.org/

[36] Silex

Available: http://silex.sensiolabs.org/

[37] Hibernate

Available: http://hibernate.org/

[38] Php myadmin

Available: http://www.phpmyadmin.net/home_page/index.php

[39] Oracle EBS

Available: http://www.oracle.com/es/products/applications/ebusiness/index.html?ssSourceSiteId=ocomen