INSTITUTO SUPERIOR POLITÉCNICO "JOSÉ ANTONIO ECHEVERRÍA"

FACULTAD DE INGENIERÍA INDUSTRIAL

Centro de Estudios de Ingeniería y Sistemas

INTERFACES DINÁMICAS EN APLICACIONES WEB MULTIPLATAFORMAS

TRABAJO PARA OPTAR POR EL TÍTULO DE INGENIERÍA EN INFORMÁTICA

Autor: Yubismel Perdomo Velázquez

Tutor: Ing. Wilber Linares Frómeta

Centro Principal de Automatización

de las FAR cpa@unicom.co.cu

Ciudad de La Habana,

Cuba Junio, 2006

Resumen

Una aplicación de software está compuesta generalmente por tres componentes: La capa de Tecnología que maneja la presentación e interfaces con el sistema operativo, la red y otras herramientas de software. La capa de las reglas de negocio, frecuentemente comprende el componente más costoso en el proceso de implementación. La capa de datos es utilizada como almacén para la información generada. Este trabajo se centrará principalmente en la capa de presentación, específicamente en las interfaces de usuario.

Actualmente la forma de manejar las interfaces está basada en elementos estáticos, predefinidos; sin la posibilidad de ser configurables o personalizados. El diseño se realiza de forma independiente en cada uno de los sistemas que se desarrollan en la organización. El dinamismo de las mismas se reduce a mostrar a los usuarios solamente las opciones a las que tiene derecho a partir de su perfil.

Con el presente trabajo se dará un paso de avance en la automatización del proceso de presentación de la información al permitir la creación y gestión de forma dinámica de las interfaces, generando los elementos de esta capa. Controlando estrictamente las operaciones disponibles para cada uno de los usuarios en pendencia de los privilegios con que cuenten ellos en el sistema. Permitiendo una personalización total en el formato de presentación de los datos.

Índice

Índice de tablas

Índice de figuras

Introducción

Dentro de los objetivos de un software podemos establecer dos vertientes: por un lado el software debe realizar las tareas para las que fue diseñado, es decir, cumplir los requisitos funcionales que le fueron encomendados, y por otra parte el sistema debe ser manejable, fácil de aprender y de comprender.

Se ha hecho necesaria la búsqueda de nuevas formas de interactuar con las aplicaciones para que sean más apropiadas a cada usuario. Esto ha motivado el diseño de sistemas que sean capaces de adaptarse a las variadas características de los usuarios. Lo ideal seria proporcionar interfaces fáciles de entender y usar por parte del usuario mediante la utilización de técnicas de interacción más compatibles con sus características, permitiéndole mejorar el rendimiento durante la utilización de la aplicación.

Para ofrecer al usuario interfaces más adecuadas a sus características, habilidades o preferencias, se necesitan también interfaces que mantengan un alto grado de usabilidad en cada una de las posibles plataformas donde el usuario puede hacer uso de la aplicación, y en cada uno de los entornos donde dicha interacción se pueda llevar a cabo. Aquellas aplicaciones capaces de adaptarse a los distintos usuarios, plataformas, y entornos donde se produce la interacción se denominan adaptativas. El diseño de aplicaciones cuyas interfaces de usuario sean adaptativas requiere la modificación de las actuales técnicas para la creación de interfaces, introduciendo los mecanismos necesarios para el diseño de las capacidades de adaptación de la interfaz de usuario dentro del ciclo de desarrollo tradicional.

Ocasionalmente la comunicación entre los clientes y los desarrolladores no es buena por esta razón no siempre se logran capturar correctamente los requerimientos para la construcción del software, También puede suceder que

la comunicación funcione perfectamente pero existen ocasiones en las que los usuarios no saben realmente que es lo que necesitan. Todo esto puede afectar la funcionalidad esperada del sistema. Por esto durante la construcción del mismo pueden surgir constantes modificaciones en los requerimientos. El efecto de este problema podría disminuirse si se desarrolla el sistema cumpliendo los requerimientos descritos y además de ello se brinda una funcionalidad adicional que permita la adaptabilidad del mismo a las características de cada usuario en particular.

Actualmente las aplicaciones web cuentan con interfaces que aunque en muchos casos son dinámicas y han sido bien diseñadas, no siempre se adaptan a las necesidades de todos los usuarios de forma óptima. Este trabajo tiene como finalidad el manejo de interfaces gráficas en aplicaciones web, se basa en la generación de componentes como menús y formularios, de forma tal que la integración de estos elementos permita la creación de interfaces dinámicas, lográndose una mayor flexibilidad en cuanto a funcionalidad y adaptabilidad. De esta forma, una vez terminado el sistema y puesto en funcionamiento, serían los usuarios los que configurarían cada una de las opciones disponibles para así ajustarlas a sus características y a sus necesidades.

Situación problémica y problema a resolver.

En la actualidad existen en la institución una serie de dificultades que constituyen la situación problémica a las que se hará frente con el presente trabajo.

- La forma de presentación de la información varía con frecuencia.
- Existe rigidez en la manera de presentación de la información.
- Al producirse una variación en la forma de presentación de la información, obligatoriamente hay que rehacer en gran medida lo que se ha hecho para adecuarlo a estas variaciones.

Lo anterior está provocado, básicamente, por la no existencia de independencia entre los datos y la forma de visualizar los mismos, esto podría definirse como la causa que provoca tal situación problémica y por ende el problema a resolver; puede mencionarse también la falta de una óptima interacción entre los diferentes módulos que componen las aplicaciones. Con respecto a esto puede decirse que un elevado por ciento de las modificaciones en las aplicaciones se realizan por cambios producidos en la presentación de la información que las mismas manejan, de manera que de no existir una concepción e implementación correcta de estas aplicaciones, que permitan minimizar el impacto de estos cambios en los productos de software realizados, se corre el riesgo de la pérdida de utilidad del producto desarrollado o, en el mejor de los casos, el empleo de personal y el uso de un tiempo prolongado en la adecuación del software a los nuevos cambios en los requerimientos.

Al asumir esta tarea consideramos que es de vital importancia, para resolver nuestra situación problémica, lograr la separación de los datos y la lógica del negocio de todo lo concerniente a la forma en que se presentarán a los usuarios finales las informaciones por ellos requeridas, apreciamos también la utilidad de realizar lo anterior de forma tal que se corresponda con las características muy

particulares de la entidad en cuestión. Es útil señalar, la necesidad de lograr facilidades que permitan, tanto a los usuarios finales, como a aquellos encargados de la administración de las aplicaciones, una personalización en determinado grado, de la forma y la apariencia con que se visualizarán las interfaces.

Aportes prácticos esperados del trabajo.

Este trabajo brindará un valioso aporte a la institución al permitir nuevas concepciones en lo referente al manejo de la capa de presentación, los datos y las informaciones que son tratadas por los productos de software, permitirá, además, dar un salto cualitativo en la manera en que se gestionan las diferentes posibilidades que tienen los usuarios finales en las aplicaciones desarrolladas en la entidad, aporta un ahorro de tiempo sustancial al brindar a los usuarios la facilidad de determinar y personalizar sus interfaces.

Objeto de estudio.

En aras de resolver la problémica existente, consideramos que debemos prestar especial atención al estudio del desarrollo de herramientas que permitan la gestión dinámica de las interfaces de usuario en las aplicaciones.

Para ello hemos de centrarnos específicamente en aspectos como la arquitectura multicapa que se utiliza en las aplicaciones que pueden beneficiarse de ser divididas en capas de componentes. La suma de las capas forma el todo. La integridad de cada una queda independiente de las otras.

Una aplicación típica está compuesta generalmente por tres componentes: La capa de Tecnología (infraestructura) que maneja la presentación e interfaces con el sistema operativo, la red y otras herramientas de software. La capa de las Reglas de Negocio, frecuentemente comprende el componente más costoso en

el proceso de implementación. La capa de datos es utilizada como almacén para la información generada.

En las aplicaciones Multicapas, los tres componentes están contenidos en capas totalmente independientes y separadas. Esta separación entre las capas de tecnología, reglas de negocio y de datos, posibilita utilizar, reemplazar o re-utilizar cada componente, en nuevas combinaciones conforme a requisitos empresariales específicos y dinámicos. No sucediendo así en los sistemas que no son multicapas, en estos, los componentes de tecnología y reglas de negocio, están compilados conjuntamente y no pueden ser separados. Dado el carácter dinámico del negocio, las reglas de negocio cambian. El usuario efectúa modificaciones en el código del programa y re-compila a fin de responder a los nuevos requisitos empresariales. Paralelamente, el proveedor realiza modificaciones a los códigos a fin de incorporar la nueva tecnología de Información. Estos dos desarrollos independientes son incompatibles entre Sí.

Campo de acción.

La capa de infraestructura (interfaces) es la encargada de la presentación de un conjunto de objetos visuales y llevar a cabo el procesamiento de los datos producidos por la interacción de los usuarios con dicha capa. Esta, debe describir y administrar las alternativas técnicas para el manejo dinámico de gráficas, menús, formularios y el análisis de la información.

Una forma de interactuar con la información puede ser mediante el uso de un CMS. Otra puede ser usando un motor de plantillas. En esta, el mecanismo de plantillas sirve para separar el código de presentación del resto del código de la aplicación Web. Consiste en codificar todo lo que tenga que ver con la presentación en una serie de plantillas de código HTML (u otro lenguaje de presentación), con expresiones sencillas intercaladas para comunicarse con el resto de la aplicación y poder mostrar datos dinámicos. Un motor de plantillas es el que se encarga de hacer la traducción a HTML.

Este mecanismo es independiente de la metodología MVC (Model – View – Controller o Modelo Vista-Controlador) y puede utilizarse sin ella, pero combinando las dos cosas se consigue una estructura de código muy organizada y elegante. De esta forma, las vistas simplemente se ocupan de extraer la información necesaria del modelo y comunicársela a las plantillas, y estas simplemente de dar un formato visual a esa información, añadiendo la información estática pertinente.

Aunque así se complica algo la construcción del sistema al introducir un nuevo nivel, en las plantillas el código de presentación queda mucho mejor aislado que en las vistas de una aplicación MVC clásica (sin plantillas), de forma que se facilita la tarea de los diseñadores web.

Hipótesis.

Si realizáramos un producto de software que permita la creación y el manejo dinámico de las interfaces web se podrá garantizar la independencia entre los datos y la forma de presentación de los mismos.

Objetivos del trabajo

General:

- Diseñar un producto de software que permita la gestión dinámica de interfaces Web.

Específicos:

- Separar en diferentes capas las funcionalidades del software.
- Desarrollar una personalización de la interfaz para el usuario final.

Tareas desarrolladas para cumplir los objetivos.

1. Realizar el análisis y crítica de la forma en que se maneja actualmente la interfaz de usuario en la entidad.
2. Desarrollar el análisis y diseño de un producto que permita la

manipulación de forma dinámica de la interfaz.

Capítulo 1 Fundamentación

El Proceso Unificado de Desarrollo (RUP). Es un proceso de ingeniería de software basado en la modelación de sistemas informáticos usando tecnología orientada a objetos, provee un enfoque disciplinado para asignar tareas y responsabilidades dentro de una organización desarrolladora o cualquier proyecto de software. Entre sus características se encuentra el desarrollo del software de manera iterativa, usa arquitecturas basadas en componentes, verifica continuamente la calidad del producto, propone la creación de artefactos como herramientas visuales con el uso de UML. Siempre se conoce el estado del proyecto, las inconsistencias entre análisis, diseño e implementación se detectan tempranamente, el cliente obtiene resultados a corto plazo, y las pruebas se concentran en los aspectos de mayor riesgo.

En la elaboración de este documento se utiliza el Proceso Unificado de Desarrollo (RUP) y como notación el Lenguaje Unificado de Modelación (UML). Este es un lenguaje gráfico para visualizar, especificar, construir y documentar los artefactos de un sistema con gran cantidad de software. UML proporciona una forma estándar de escribir los planos de un sistema, cubriendo tanto las cosas conceptuales, tales como procesos del negocio y funciones del sistema, como las cosas concretas, tales como las clases escritas en un lenguaje de programación específico, esquemas de bases de datos y componentes de software reutilizables. [7]

1.1 Introducción

En lo adelante se brinda una visión general de los aspectos relacionados con los sistemas basados en la tecnología web, y los conceptos necesarios para el estudio de los que están construidos bajo una arquitectura multicapa; las características de cada tipo de herramienta, así como la descripción de los principales conceptos asociados al objeto de estudio y que son necesarios para entender el funcionamiento de las demás áreas.

1.2 Objeto de estudio

1.2.1 Objetivos estratégicos de la organización

El objetivo estratégico de la organización es la preparación del país para la defensa y la lucha armada ante una posible agresión extranjera. Dentro de este objetivo general nuestro trabajo se concentra en el aumento del control, mediante la automatización de todos los recursos materiales, financieros y humanos de la entidad.

1.2.2 Flujo actual de los procesos

En la actualidad la manera en que se manejan las interfaces de usuario es totalmente basada en elementos estáticos, predefinidos; sin la posibilidad de ser configurables o personalizados. El diseño de las interfaces se realiza de forma independiente en cada uno de los sistemas que se desarrollan en la organización. El dinamismo de las mismas se reduce a mostrar a los usuarios solamente las opciones a las que tiene derecho a partir de su perfil.

1.2.3 Análisis crítico de la ejecución de los procesos

En la actualidad los procesos orientados al tratamiento dinámico de las interfaces de usuario son prácticamente inexistentes en la entidad, lo anterior nos lleva a la causa fundamental de la existencia de nuestra situación problemática y por ende las consecuencias derivadas de la misma. Como se ha explicado anteriormente en la organización cambian con frecuencia las necesidades de modificación de las interfaces, lo cual redunda en una revisión y reconfección de un elevado por ciento del trabajo realizado. Todo esto lleva a pérdidas de tiempo, ineficiencia, empleo de personal en esta labor, que en ocasiones no es el mismo que desarrolló los productos de software, con las implicaciones que esto conlleva. Otro elemento a señalar es la rigidez que se impone en la manera de presentación de las informaciones a los usuarios.

Súmesele a lo anterior el hecho de que las implementaciones no están basadas en modelos multicapas que garanticen la separación de las interfaces de la lógica de negocio. La configuración gráfica que tiene el sistema no siempre se ajusta, de la mejor manera, a las necesidades de los usuarios finales.

1.3 Procesos objeto de automatización

Se automatizará el proceso de presentación de la información. Para permitir la creación y gestión de forma dinámica de las interfaces, generando los elementos de la capa de interfaz. Controlando estrictamente las operaciones disponibles para cada uno de los usuarios en dependencia de los privilegios con que cuenten ellos en el sistema. Permitiendo una personalización total en el formato de presentación de los datos.

1.4 Sistemas automatizados existentes vinculados al campo de acción

En estos momentos no existen en la organización Sistemas Automatizados que den respuesta a los elementos planteados en el campo de acción.

1.5 Fundamentación de los objetivos

General:

- Diseñar un producto de software que permita la gestión dinámica de interfaces Web.

Específicos:

- Separar en varias capas las funcionalidades del software. Para lograr así la independencia entre los datos y la interfaz de usuarios. Aplicando la metodología de programación multicapas, de esta forma quedan separadas la capa de presentación y la de la lógica del negocio y si ocurriera un cambio en una de esta no habría que hacer modificaciones considerables sobre la otra.

- Desarrollar una personalización de la interfaz para el usuario final. De esta forma podemos eliminar la rigidez que existe en la forma de presentación de la información, creando interfaces más funcionales y atractivas al usuario.

1.6 Tendencias y tecnologías actuales

Con el avance de las tecnologías y de Internet, cada vez hay más instituciones que orientan sus aplicaciones hacia la tecnología web. Dicha tecnología está basada en la arquitectura cliente-servidor. El cliente hace una petición al servidor para que le envíe una página web, este analiza la extensión de la página solicitada, y la devuelve al cliente una vez que la localiza en su sistema de archivos y ejecuta el intérprete de dicha extensión. La variedad de lenguajes y formatos en los que nos podemos encontrar codificadas las páginas es bien variada, así encontramos extensiones como HTML, ASP, PHP, XML, etc.

Editores de páginas Web.

Entre los editores de paginas Web más utilizados actualmente se encuentran FrontPage, GoLive(de Adobe), Zend Studio, Eclipse y Dreamweaver este último es un editor que permite componer una página de forma visual y además poder realizar la edición manual del código fuente de la misma. Ofrece varias herramientas para optimizar y gestionar el sitio completo, incluso un cliente FTP con opciones para mantener sincronizados el sitio local y remoto. Añade algunas funciones predefinidas de Javascript, que facilitan la inclusión de controles y efectos habituales de los lenguajes de scripting para los no programadores.

Zend Studio

Este es un programa de la compañía Zend, impulsores de la tecnología de servidor PHP, orientada a desarrollar aplicaciones web, en lenguaje PHP, además de servir de editor de texto para páginas PHP, proporciona una serie de ayudas que pasan desde la creación y gestión de proyectos hasta la depuración

de código. Está escrito en Java, por esto a veces supone que no funcione tan rápido como otras aplicaciones de uso diario. Sin embargo, esto ha permitido a Zend lanzar con relativa facilidad y rapidez versiones del producto para Windows, Linux y MacOS, aunque el desarrollo de las versiones de este último sistema se retrase un poco más.

Zend Studio consta de dos partes en las que se dividen las funcionalidades de parte del cliente y las del servidor. Las dos se instalan por separado, la del cliente contiene el interfaz de edición y la ayuda. Permite además hacer depuraciones simples de scripts, aunque para disfrutar de toda la potencia de la herramienta de depuración habrá que disponer de la parte del servidor.

FrontPage, creado hace ya muchos años, ha tenido multitud de versiones que han ido mejorando su funcionamiento. Está orientado a personas inexpertas y sin conocimientos de HTML. Sus capacidades son semejantes a las de otros editores, como el crear mapas de imágenes, gestionar la arborescencia de las páginas del sitio.

Al ser un producto Microsoft, está orientado a construir páginas optimizadas para Internet Explorer. Por esta misma razón. Conseguir páginas que se vean bien en otros navegadores puede ser complicado con este programa, lo que constituye un serio inconveniente.

GoLive nos permite crear sitios profesionales sin ser webmasters expertos. Con el podemos escribir prácticamente toda la página, sin haber visto nunca el código fuente. En muchos aspectos, es muy parecido a Macromedia Dreamweaver.

Lenguajes de programación

Un lenguaje de programación muy utilizado para crear programas encargados de realizar acciones dentro del ámbito de una página web lo constituye **JavaScript**. Con este se pueden crear efectos visuales, para mostrar contenido de forma dinámica. También nos permite, ejecutar instrucciones como respuesta a las acciones del usuario, con lo que podemos crear páginas interactivas.

JavaScript permite la programación de pequeños scripts, pero también de programas más grandes, con elementos de orientación a objetos, con funciones, estructuras de datos complejas, etc. Además, pone a disposición del programador todos los elementos que forman la página web, para que éste pueda acceder a ellos y modificarlos dinámicamente.

Dentro de los lenguajes *Script* para el desarrollo de aplicaciones Web se encuentra **Perl.** Quien es el decano de los lenguajes de desarrollo Web. En 1998 casi todo Internet estaba hecho con Perl, poco a poco esto ha cambiado.

Perl es uno de los lenguajes más poderosos y versátiles que existen en el mundo. Posee una gran cantidad de librerías para hacer de todo y a lo largo de los años se ha reunido una enorme documentación sobre su uso. Por otro lado Perl, cuenta con una sintaxis poco intuitiva. Debido a esto, resulta un poco difícil de dominar.

ASP (*Active Server Pages)* es una solución de Microsoft basada en Visual Basic. La principal ventaja es que hay un flujo constante de trabajo para estos desarrolladores. Sin embargo, las tendencias actuales pronostican un decremento de los servidores de Microsoft y un aumento en los sistemas Linux y BSD. Además ASP es un sistema con nula portabilidad pues requiere necesariamente de un servidor Windows, con todas las implicaciones que esto trae consigo.

JSP (*Java Server Pages*) (Páginas de Servidor Java) Es una tecnología orientada a crear páginas web con programación en Java, altamente tipificado con el que se puede desarrollar código bien estructurado y orientado a objetos.

La tecnología Java se ha posicionado bien en las grandes empresas por el marketing y el respaldo de SUN, pero no necesariamente por sus ventajas ante una necesidad concreta con JSP podemos crear aplicaciones web que se ejecuten en variados servidores web, de múltiples plataformas, ya que Java es en esencia un lenguaje multiplataforma.

Python. Este es rápido, intuitivo, con una hermosa sintaxis y, libre (100% GPL). Está pensado para programar clases desde el inicio, lo que lo hace ideal para la programación orientada a objetos. Las empresas futuristas del ramo de la tecnología (como Google) lo están tomando como su lenguaje base, Uno de los problemas de Python es su escasa documentación y número de aplicaciones, no obstante se espera que eso cambie en el próximo par de años.

PHP (acrónimo de PHP: Hypertext Preprocessor). Su facilidad de uso, la rapidez de su motor y su alianza con MySQL lo han convertido en casi un estándar de la red. Su presencia, en un impresionante número de servidores lo ha llevado a estar muy por encima de cualquier otro lenguaje script. PHP es sumamente escalable, si consideramos "escalable" como la capacidad de un sistema de aumentar el número de usuarios aumentando sus recursos y sin perder ninguna de sus ventajas. Es un lenguaje de alto nivel impregnado en páginas HTML y ejecutado en el servidor, donde se encuentra almacenado el script. Permite la conexión a diferentes tipos de servidores como: MySQL, Postgres, Oracle, ODBC, IBM DB2, Microsoft SQL Server y SQLite; permitiendo la creación de Aplicaciones Web muy robustas. Incluye funciones de correo electrónico, gestión de bases de datos, gestión de archivos, tratamiento de imágenes, tratamiento de cookies, accesos restringidos, comercio electrónico o para propósito general (funciones matemáticas, explotación de cadenas, de fechas, corrección ortográfica, compresión de archivos. A esta inmensa librería cabe ahora añadir todas las funciones que se van creando por necesidades propias y que luego son reutilizadas en otros sitios, y todas aquellas intercambiadas u obtenidas en foros o sitios especializados. Es un software libre, por lo que su empleo trae por consiguiente un ahorro económico.

La arquitectura cliente servidor está estrechamente relacionada con **SQL**. **SQL** (Structured Query Language), es un lenguaje de alto nivel, no procedural, normalizado y que permite la consulta y actualización de los datos almacenados en bases de datos. Se ha convertido en el estándar de manipulación de datos en SGBD.

Herramientas case:

Visual Paradigm for UML Es una herramienta CASE que utiliza "UML" como lenguaje de modelaje. Posee entorno de creación de diagramas para UML, diseño centrado en casos de uso y enfocado al negocio. Usa un lenguaje estándar, común a todo el equipo de desarrollo facilitando la comunicación. Posee capacidades de ingeniería directa (versión profesional) e inversa. Genera modelo y código que permanece sincronizado en todo el ciclo de desarrollo. Está disponible en múltiples versiones, para cada necesidad y para varias plataformas.

Rational Rose es una herramienta CASE desarrollada por los creadores de UML (Booch, Rumbaugh y Jacob-son), cubre todo el ciclo de vida de un proyecto: concepción y formalización del modelo, construcción de los componentes, transición a los usuarios y certificación de las distintas fases y entregables. El navegador UML de Rational Rose permite establecer una trazabilidad real entre el modelo (análisis y diseño) y el código ejecutable. Facilita el desarrollo de un proceso cooperativo en el que todos los agentes tienen sus propias vistas de información (vista de Casos de Uso, vista Lógica, vista de Componentes y vista de Despliegue).

Gestores de bases de datos:

Oracle es un sistema de administración de base de datos se considera como uno de los sistemas más completos, destacando su soporte de transacciones, estabilidad, escalabilidad, además de ser multiplataforma.

Su mayor defecto es su enorme precio (está en dependencia de las versiones y licencias). Otro aspecto desfavorable es la seguridad de la plataforma, y las políticas de suministro de parches de seguridad. Aunque su dominio en el mercado de servidores empresariales ha sido casi total hasta hace poco, desde hace algún tiempo sufre la competencia del Microsoft SQL Server de Microsoft y de la oferta de otros con licencia libre como PostgreSQL, MySql.

Otro sistema gestor de bases de datos relacional utilizado actualmente es **MySQL**, licenciado bajo la licencia GPL de GNU. Este aprovecha la potencia de sistemas multiprocesador. Dispone de API's en gran cantidad de lenguajes (C, C++, Java, PHP, etc). Permite utilizar gran cantidad de tipos de datos para las columnas, gestión de usuarios y contraseñas. Mantiene un muy buen nivel de seguridad en los datos.

Actualmente también se utiliza **PostgreSQL**, servidor de base de datos relacional, libre, liberado bajo la licencia BSD Además de los tipos base, también soporta otos tipos de datos como: fecha, monetarios, elementos gráficos y otros. Permite la creación de tipos propios. Incorpora una estructura de datos array, funciones de diversa índole, manejo de fechas, geométricas, orientadas a operaciones con redes. Admite declaración de funciones propias, así como la definición de disparadores. Soportando el uso de índices, reglas y vistas. Incluye herencia entre tablas. Con el se puede lograr tanto la gestión usuarios, como de los permisos asignados a cada uno de ellos.

En la administración de grandes volúmenes de contenido, las páginas Web estáticas son cosa del pasado. Para tener éxito hoy en día, es necesario crear aplicaciones que se puedan actualizar con frecuencia, para cumplir estos objetivos, algunas veces se requiere un sistema de administración de contenido **CMS** (Content Management System). Un CMS es una herramienta que permite crear y modificar el contenido de una página web, con poco o nada de conocimiento técnico. Mambo es un ejemplo de CMS que está basado en el

lenguaje de programación PHP y base de datos SQL de código abierto. Basa todo su aspecto en plantillas (*templates)* o temas (*themes).* [5]

Otra tendencia actual es el uso de generadores de plantillas pues es común que en grandes proyectos el rol de diseñador gráfico y el de programador sean cubiertos por personas distintas, sin embargo la programación en PHP tiende a combinar estas dos labores en una persona y dentro del mismo código lo que trae consigo grandes dificultades a la hora de cambiar alguna parte del diseño de la página, pues se tiene que escarbar entre los scripts para modificar la presentación del contenido. Una forma de solucionar esto es usando **smarty.** Este es un motor de plantillas para PHP, cuyo objetivo es separar el contenido de la presentación en una página Web.

Básicamente es un motor de plantillas que nos da la posibilidad de tener la presentación de nuestros proyectos totalmente separada del código de la aplicación. Esto permite hacer cambios rápidos y sin complicaciones sobre dicha presentación, sin afectar en modo alguno a la parte lógica del programa.

1.7 Análisis crítico de las fuentes y bibliografías utilizadas

Utilizaremos **JavaScript** por su compatibilidad con la mayoría de los navegadores modernos, además de permitirnos crear los efectos necesarios en las páginas para definir interactividades con los usuarios. Por ser un lenguaje de programación bastante sencillo y pensado para hacer las cosas con rapidez, a veces con ligereza. Es un lenguaje que se puede aprender con facilidad y puede ser utilizado en toda su potencia con sólo un poco de práctica.

También utilizaremos **PHP** por ser un lenguaje multiplataforma. Con capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad. Por leer y manipular datos desde diversas fuentes, incluyendo

datos que pueden ingresar los usuarios desde formularios HTML, por su capacidad de expandir su potencial utilizando la enorme cantidad de módulos (llamados ext's o extensiones). Por, ser Libre, lo que se presenta como una alternativa de fácil acceso para todos y permitir el uso de las técnicas de programación orientada a objetos. [4]

Será utilizado **Dreamweaver** como editor Web porque genera un código HTML bastante limpio y compatible con la mayoría de los navegadores. Incluye varias herramientas añadidas que permiten aumentar la productividad. Se integra perfectamente con otras aplicaciones de Macromedia, como Fireworks o Flash, creándose todo un entorno de trabajo autosuficiente. La coordinación entre el modo de edición visual y el modo de edición manual esta muy lograda permitiendo a los programadores y diseñadores experimentados añadir elementos al HTML no soportados por Dreamweaver.

También será utilizado **Zend Studio** porque es multiplataforma. Permite ejecutar código SQL y conectarse con distintos motores, la conectividad que ofrece con la DB es muy buena. Puede mostrar los resultados de consultas, posee auto completado de código, información contextual, un depurador muy bueno y plantillas de código editables. Es bueno en el manejo de proyectos, clases y funciones.

Como gestor de bases de datos será usado **PostgreSQL**. Para este el máximo tamaño de una base de datos está limitado solo por la capacidad de almacenamiento del hardware (No es viable para su uso con grandes bases de datos, a las que se acceda continuamente) Para su instalación no requiere de gran cantidad de memoria ni espacio en disco y además es multiplataforma haciéndolo idóneo para su uso en sitios web que posean alrededor de 500.000

peticiones por día. Aunque carece de un conjunto de herramientas que permitan una fácil gestión de los usuarios y de las bases de datos que contenga el sistema y aunque la velocidad de respuesta que ofrece ante bases de datos pequeñas puede parecer un poco deficiente, es un gestor magnífico, que posee una gran escalabilidad ajustándose muy bien a nuestros requerimientos.

1.8 Conclusiones

Hasta aquí se ha hecho un estudio de la situación relacionada con las interfaces de usuario en la institución. Se analizaron las actuales tendencias de las tecnologías vinculadas con dicha situación, para poder tomar un punto de partida y así tener un criterio que permitiera decidir cuales herramientas elegir, que estrategia seguir y que filosofía de trabajo desarrollar. A partir de esto se trazaron los objetivos de este trabajo.

Capítulo 2 Modelo del dominio

2.1 Introducción

El proceso de negocio que se está estudiando, tiene muy bajo nivel de estructuración, por lo que la determinación de sus fronteras no es una tarea sencilla. De tal manera no se pude determinar con exactitud quienes son las personas que inician ni quienes son las que desarrollan las actividades en cada proceso. Por esto se utilizará un modelo del dominio, el que permitirá mostrar los principales conceptos que se manejan en el dominio del sistema en desarrollo. Lo anterior se realiza con el objetivo de ayudar a los usuarios, clientes, desarrolladores y demás interesados a utilizar un vocabulario común para poder entender el contexto en el que se emplaza el sistema.

2.2 Definición de las entidades y los conceptos principales

A continuación se describirán los conceptos significativos dentro del entorno donde trabaja el sistema.

- **Usuario:** Describe a cualquier usuario final que interactúa con el sistema.
- **Tipo de usuario:** Agrupación de varios usuarios de acuerdo a sus privilegios.
- **Sistema:** Sistema general que está compuesto por todos los módulos.
- **Módulo:** División funcional de el sistema en partes mas pequeñas que se componen de un grupo de funcionalidades semejantes.
- **Permiso:** Un determinado privilegio que se le puede asignar a los usuarios.
- **Interfaz:** Componente que estará disponible para cada usuario, con el que dicho usuario podrá interactuar con la aplicación.

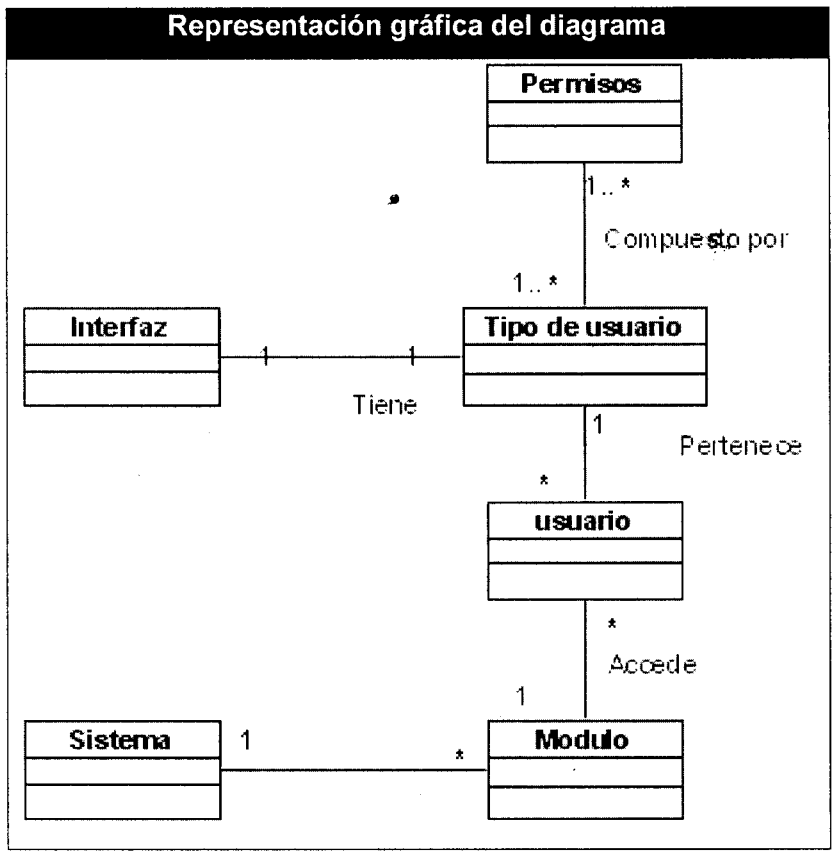
2.3 Representación del modelo del dominio

Un modelo conceptual tiene como objetivo identificar y explicar los conceptos significativos en el dominio de problema, identificando los atributos y las asociaciones existentes entre ellos. Puede ser visto como una representación de las cosas, entidades, ideas, conceptos objetos del mundo real o dominio de interés. Dichas entidades u objetos no son componentes de software. También podría ser considerado como un diccionario visual de abstracciones, conceptos, vocabulario e información del dominio de problema. No es absolutamente correcto o incorrecto, su intención en ser útil sirviendo como una herramienta de comunicación. El modelo conceptual es una técnica que permite modelar algunos aspectos relacionados con las entidades y sus relaciones, pero no todos.

Pueden utilizarse distintas notaciones para representar un modelo conceptual. Aquí para tal fin se utilizará el diagrama de clases de UML. Esta notación puede utilizarse durante varias etapas del proceso de software para representar distintas cosas.

Limitaciones:

Dado un problema es posible confeccionar muchos modelos conceptuales, no necesariamente consistentes entre sí. La elaboración del modelo conceptual es una tarea que requiere una interpretación de la realidad, y en dicha interpretación juega un rol importante la subjetividad de quien realiza la tarea. [1]



El proceso de desarrollo de software debe incluir en sus primeras etapas la construcción del modelo conceptual. Dicha tarea suele plantarse como parte

del análisis de requerimientos. La identificación de requerimientos debe acompañarse con la identificación de las entidades o conceptos involucrados en los mismos. De igual manera, la identificación de entidades puede contribuir con la identificación de requerimientos.

2.4 Conclusiones

En este capitulo se realizó un estudio para determinar los procesos de negocios que son llevados a cabo en la institución. Aquí se determinaron los conceptos significativos así como la relación existente entre ellos. Finalmente se modelaron en un diagrama de dominio para lograr que los clientes y desarrolladores utilicen un lenguaje común con el que puedan entenderse entre si y comprender el contexto en el que se desarrolla el sistema.

Capítulo 3 Requisitos

3.1 Introducción

Antes de desarrollar cualquier aplicación de software hay que saber quienes serán los usuarios que interactuarán con ella para luego determinar como debe funcionar el sistema. Los requerimientos especifican acciones que el sistema debe ser capaz de realizar, sin tomar en consideración ningún tipo de restricción física. Por lo general se describen mejor a través del modelo de casos de uso y los casos de uso como tal.

Un caso de uso es una parte de la funcionalidad del sistema. El modelo de casos de uso representa la funcionalidad completa de este, guiando el proceso software y describiendo lo que debe hacer el sistema para cada usuario. Basado en el modelo de casos de uso se pueden crear los modelos de diseño e implementación.

En este capitulo se definirán los actores que van a interactuar con el sistema y se desarrollarán los diagramas de casos de uso del sistema. Para establecer el correcto funcionamiento de la aplicación se determinarán los requerimientos y luego se hará la descripción de los casos de uso para una mejor comprensión.

3.2 Actores del sistema a automatizar

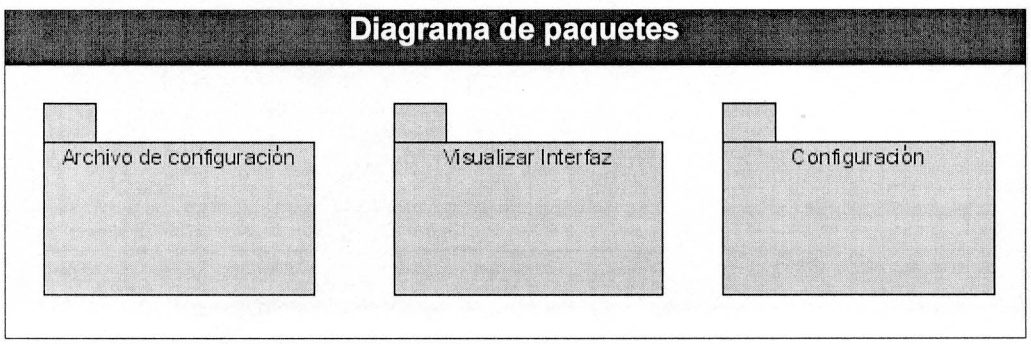
Un actor representa a un usuario humano u otro sistema que interactúa con el sistema bajo análisis.

Tabla 1. Definición de actores del sistema a automatizar

Nombre del actor	Descripción
Usuario	Usuario final del sistema, persona que interactúa con el sistema.
Sistema externo	Parte funcional en la que está dividido el sistema, esta puede solicitarle algún servicio al modulo de interfaz.

3.3 Paquetes y sus relaciones

Para el desarrollo de este trabajo fue necesario dividirlo en tres paquetes: Archivos de configuración que agrupa los casos de uso relacionados con la gestión de los archivos y variables de configuración. Visualizar interfaz, relacionado con las funciones de generación y visualización de componentes e interfaces. Por ultimo el paquete configuración, contiene los casos de uso relacionados con la personalización de la interfaz.



3.4 Diagrama de casos de uso del sistema a automatizar

Un caso de uso representa un gránulo funcional del sistema, relatado una secuencia de acciones que uno o más actores llevan a cabo sobre este para obtener un resultado de valor significativo.

Figura 1- Caso de uso: Visualizar interfaz

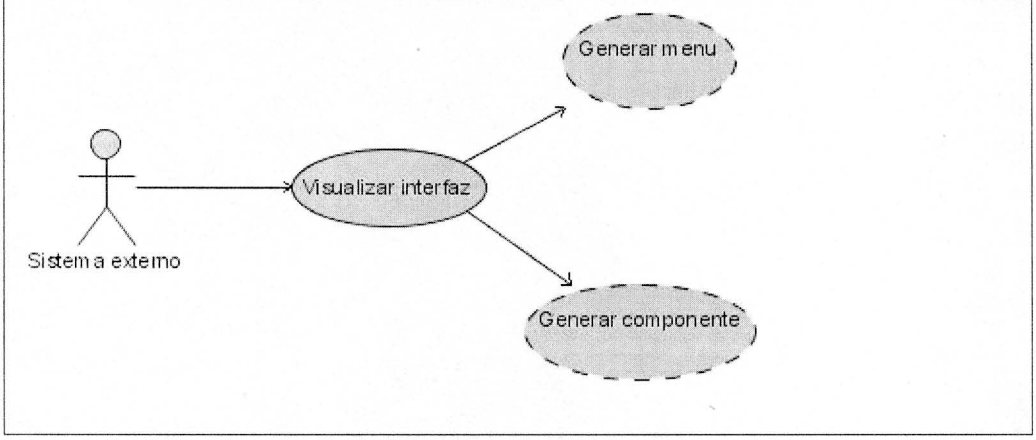


Figura 2- Caso de uso: Configurar estilos

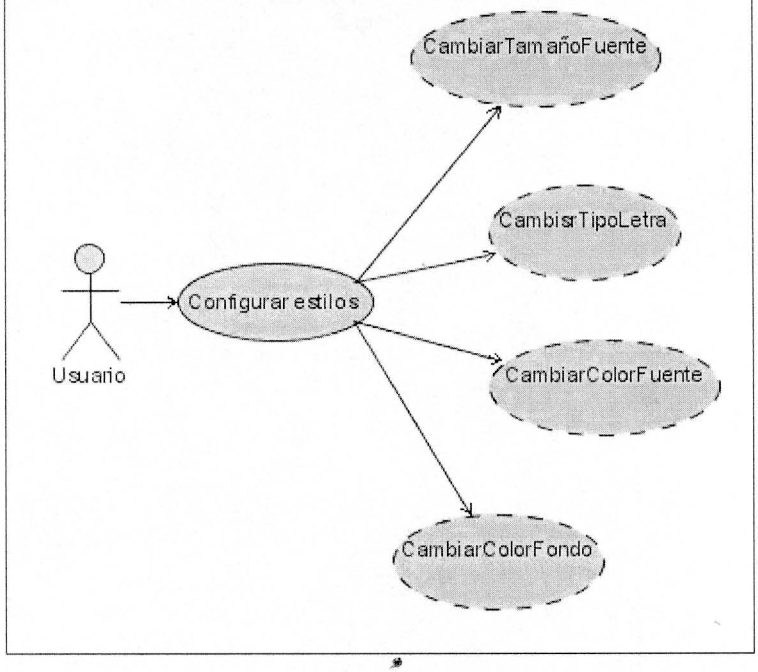


Figura 3- Caso de uso: Guardar configuración

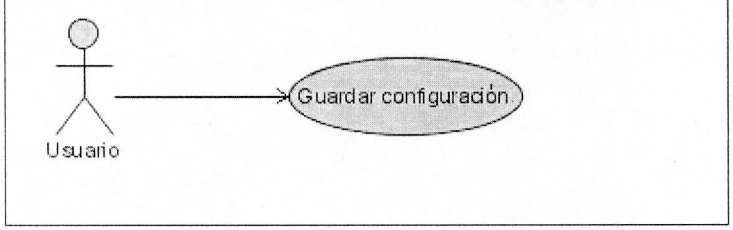


Figura 4- Caso de uso: Configurar Componentes

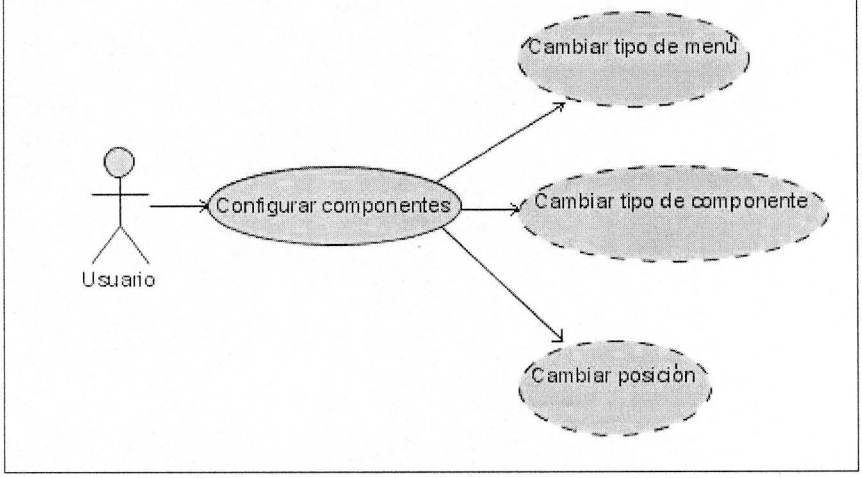
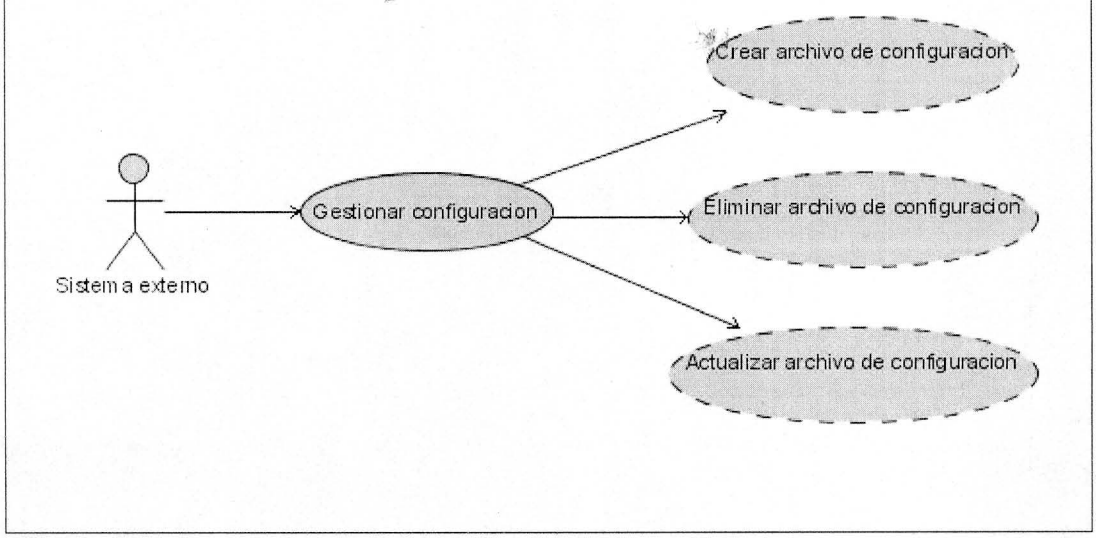


Figura 5- Caso de uso: Gestionar configuración



3.5 Definición de los requisitos

Funcionales

Mostrar interfaz con las opciones permitidas para cada usuario en dependencia de su perfil.

1. Cada usuario dispone de un perfil con todas las opciones disponibles en dependencia de sus permisos.
2. Cada usuario cuenta con su propia interfaz gráfica.

La fuente puede ser configurada.

3. Cambiar color de fuente.
4. Cambiar tamaño de la fuente.
5. Cambiar Tipo fuente
6. Cambiar estilo de la fuente

El color del fondo de la interfaz de usuario puede ser modificado.

7. Cambiar el color del fondo.

Configuración de componentes.

8. El usuario puede seleccionar el tipo de componente en el que se visualice la información.
9. Seleccionar tipo de menú en el que se deben mostrar las opciones disponibles.
10. El usuario puede modificar la posición de los componentes, según sus necesidades y la funcionalidad del módulo.

Deben existir las opciones necesarias para:

11. Permitir guardar el estado de la interfaz grafica.

No funcionales

Apariencia

12. Los colores a utilizar estarán dentro de las tonalidades utilizadas en la institución.
13. La resolución a utilizar será 800x600.
14. El idioma a utilizar será solo el español.

Usabilidad

15. Disponibilidad de ayuda en línea.
16. Los usuarios deben tener un conocimiento básico en trabajo con sistemas operativos (Windows, Linux).
17. El sistema estará disponible 24 horas.

Rendimiento

18. El tiempo de respuesta debe ser inferior a 5 segundos.
19. El tiempo de ejecución de un hipervínculo debe ser inferior a 5 segundos.

Portabilidad

20. El sistema puede ser utilizado sobre las plataformas Linux o Windows.

Software

Para el cliente:

21. Navegador Mozilla Firefox.
22. Sistema operativo Linux, Windows 98 o superior.

Para el servidor:

23. Sistema operativo Windows Advanced Server (2000 o superior) o Linux en cualquiera de sus distribuciones.
24. Un servidor Apache v2.0 o superior con modulo PHP5 disponible, este debe estar configurado con la extensión "pgsql" incluida.
25. Un servidor de base de datos PostgreeSQL v8.0 o superior.

Hardware

Para el servidor:

26. Requerimientos mínimos: Procesador Pentium III a 1GHz de velocidad de procesamiento y 1Gb de memoria RAM.
27. Al menos 40Gb de espacio libre en disco duro.
28. Tarjeta de red.

Para el cliente:

29. Requerimientos mínimos: Procesador Pentium II a 133Mhz con 128 Mb de memoria RAM.
30. Tarjeta de red.

3.6 Descripción de los casos de uso

Tabla 2. Descripción del caso de uso Visualizar interfaz de usuario

Caso de Uso:	**Visualizar interfaz de usuario**
Actores:	Sistema externo (Inicia).
Propósito: Visualizar interfaz de usuario con todas las operaciones disponibles para un usuario determinando.	
Resumen: El caso de uso se inicia cuando el Sistema externo solicita visualizar interfaz para un usuario determinado, enviando el nombre del usuario para el cual se debe mostrar dicha interfaz. El sistema localiza el perfil y el archivo de configuración de ese usuario. Se leen los datos del perfil, se localiza la interfaz y se generan los componentes dinámicos. Se leen los valores de las variables de configuración y se aplican a la interfaz. El caso de uso finaliza cuando la interfaz es visualizada.	
Precondiciones :	Existe el perfil de usuario.

Flujo normal de eventos **Sección "Generar menú"**			
Acción del actor:		Respuesta del sistema:	
1	Sistema externo solicita generar menú.	1.2	El sistema busca el perfil de usuario y lee las variables donde aparecen las opciones de menú.
		1.3	El sistema busca el archivo de configuración de componentes y lee de ahí el valor de las variables de configuración de

			menú.
		1.3	Genera el menú con las opciones leídas.
		1.5	El sistema visualiza el menú.
Sección "Generar componente"			
Acción del actor		**Respuesta del sistema**	
2	El sistema externo solicita generar componente.	2.1	El sistema busca el archivo de configuración y lee de ahí el valor de las variables para encontrar la configuración que debe ser aplicada al componente.
		2.2	El sistema genera el componente y lo visualiza en la interfaz de usuario aplicándole la configuración leída anteriormente.
Referencias:	R2		
Poscondiciones:	-		
Curso alterno de los eventos			
Sección " Generar menú"			
Acción del Actor		**Respuesta del Sistema**	
		1.1	El sistema no encuentra el archivo de configuración.
		1.2	Se genera el menú con la configuración por defecto.
Sección "Generar componente"			
		2.1	No se encuentra la configuración para el componente.

		2.2	Se genera el componente con la configuración por defecto.
Requerimientos Especiales:			

Tabla 3. Descripción del caso de uso Configurar componente

Caso de Uso:		**Configurar Componente**	
Actores:		Usuario(Inicia)	
Propósito: Permitirle al usuario configurar los componentes de su interfaz gráfica para mejorar la funcionalidad y ajustarla a sus necesidades.			
Resumen: El caso de uso se inicia cuando un usuario solicita configurar componentes, este puede seleccionar el tipo de menú en el que se podría mostrar una determinada información, modificar la posición de un componente seleccionado o cambiar dicho componente por otro equivalente. Finalizando el caso de uso cuando se aplican los cambios y se guarda la configuración.			
Precondiciones:			
Curso normal de los eventos **Sección "Cambiar tipo de menú"**			
Acción del actor		Respuesta del sistema	
1	El usuario solicita cambiar tipo de menú.	1.2	El sistema localiza el archivo de configuración, lee las opciones de menú y muestra una interfaz con los tipos disponibles.
1.3	El usuario selecciona el tipo que desea utilizar.	1.4	El sistema aplica los cambios y guarda la configuración.
Sección "Cambiar tipo de componente"			
Acción del actor		**Respuesta del sistema**	
2	El usuario solicita cambiar tipo de componente.	2.1	El sistema localiza el archivo de configuración, lee las opciones

			de componentes y muestra una interfaz con los tipos de disponibles.
2.2	El usuario selecciona el tipo deseado.	2.3	El sistema aplica los cambios y guarda la configuración.
Sección "Cambiar posición"			
Acción del actor		**Respuesta del sistema**	
3	El usuario solicita cambiar posición de un componente.	3.1	El sistema localiza el archivo donde se guarda la configuración de los componentes, muestra la interfaz resaltando los componentes que pueden ser movidos.
3.2	El usuario selecciona un componente y modifica su posición.	3.3	El sistema aplica los cambios y guarda la configuración.
Referencias:	R9,R10		
Poscondiciones:			
Requerimientos Especiales:	-		

Tabla 4. Descripción del caso de uso Guardar configuración

Caso de Uso:	**Guardar configuración**
Actores:	Usuario (Inicia).
Propósito: Permitirle al usuario guardar todos los cambios hechos mediante la configuración de la interfaz.	
Resumen: El caso de uso se inicia cuando un usuario solicita guardar el estado de su perfil	

gráfico el sistema localiza todos los componentes configurables, determina su estado, guardando este valor en los archivos de configuración finalizando de esta forma el caso de uso.			
Precondiciones:	Se ha modificado la interfaz gráfica.		
Curso normal de los eventos			
Acción del actor		**Respuesta del sistema**	
1	El usuario solicita guardar el estado de su interfaz.	2	El sistema localiza todos los componentes configurables dentro la interfaz, determina el valor de cada una de las variables que definen su estado. Finalmente se guardan estos valores en los archivos de configuración.
Referencias:	R11		
Poscondiciones:	Se han actualizado las variables del archivo de configuración de usuario.		
Requerimientos Especiales:	-		

Tabla 5. Descripción del caso de uso Gestionar archivo de configuración

Caso de Uso:	**Gestionar archivo de configuración**
Actores:	Sistema externo (Inicia).
Propósito: Permitir la gestión del archivo de configuración asociado a un usuario.	
Resumen: El caso de uso se inicia cuando el Sistema externo solicita crear, eliminar o actualizar los archivos de configuración de interfaz de algún usuario. El sistema localiza el archivo y realiza sobre él la operación solicitada, si no existe lo crea, finalizando de esta forma el caso de uso.	

Precondiciones:			
Flujo Normal de Eventos **Sección "Crear archivo de configuración"**			
Acción del Actor		**Respuesta del Sistema**	
1	El Sistema externo solicita crear el archivo de configuración de un usuario determinado.	1.1	El sistema verifica si el archivo existe, de ser así lo elimina.
		1.2	El sistema crea el archivo de configuración de ese usuario generando las variables de configuración con los valores por defecto.
Sección "Eliminar archivo de configuración"			
2	El Sistema externo solicita eliminar el archivo de configuración de un usuario	2.1	El sistema busca los archivos de configuración del usuario y los elimina.
Sección "Actualizar archivo de configuración"			
3	El Sistema externo solicita actualizar el archivo de configuración de un usuario determinado.	3.1	El sistema busca el perfil de ese usuario, lee los datos de ese perfil.
		3.2	En dependencia de los datos leídos, el sistema elimina las variables que ya no guardan relación con la interfaz de usuario.
		3.3	Crea las nuevas variables con los valores por defecto.
Referencias:	**R1, R2**		

Poscondiciones:	
Requerimientos especiales	

Tabla 6. Descripción del caso de uso Configurar estilos

Caso de Uso:	**Configurar Estilos**
Actores:	Usuario (Inicia).
Propósito:	Que el usuario pueda cambiar las opciones de l estilos en su interfaz.
Resumen: El caso de uso se inicia cuando el usuario decide cambiar el estilo de su interfaz para ajustarla a sus necesidades. El sistema identifica las opciones que pueden ser modificadas y brinda el soporte para hacerlo. El usuario modifica las opciones que desea. Este puede cancelar la operación en cualquier momento o solicitar guardar los cambios una vez que haya hecho las modificaciones deseadas. Finalizando así el caso de uso.	
Precondiciones:	El usuario ha entrado al sistema y posee una interfaz gráfica.

Flujo Normal de Eventos **Sección "Cambiar color de fuente"**			
Acción del Actor		**Respuesta del Sistema**	
1	El usuario solicita configurar estilos.	1.1	El sistema muestra la interfaz con las diferentes opciones que se pueden configurar.
2	El usuario accede la opción Cambiar color de fuente	2.1	El sistema lee del archivo de configuración los posibles colores que pueden ser aplicados y los muestra.

3	El usuario selecciona el color que desea cambiar y solicita aplicar	3.1	El sistema aplica el color seleccionado, localiza el fichero de configuración y actualiza las variables relacionadas con el color de fuente.
Sección: "Cambiar tamaño de la fuente"			
5	El usuario solicita configurar estilos	5.1.	El sistema muestra la interfaz con as diferentes opciones que puede configurar.
6	El usuario solicita cambiar tamaño de la fuente.	6.1	El sistema muestra al usuario la interfaz con los diferentes valores que puede seleccionar.
7	El usuario escoge un valor y guarda los cambios.	7.1	El sistema localiza el archivo de configuración y dentro de este las variables relacionadas con el tamaño de la fuente actualizando estas con los nuevos valores.
Sección: "Cambiar Tipo de Letra"			
8	- El usuario solicita configurar estilos.	8.1	El sistema muestra la interfaz con las diferentes opciones que puede configurar.
9	El usuario solicita Cambiar Tipo de Letra.	9.1	El sistema localiza el archivo de configuración y dentro de este las variables relacionadas con el tipo de letra. Muestra una interfaz con los tipos disponibles.
10	El usuario selecciona el tipo de letra que desea cambiar y solicita guardar.	10.1	- El sistema aplica este tipo a la interfaz y actualiza la variable relacionada con este en el archivo de configuración.

Sección: "Cambiar el color del fondo"			
12	El usuario solicita configurar estilos	12.1	.-El sistema muestra la interfaz con las diferentes opciones que puede configurar.
13	El usuario solicita Cambiar Color de fondo.	13.1	– El sistema localiza el archivo de configuración lee los colores de fondo que se encuentran disponibles y los muestra.
14	El usuario selecciona el color que desea cambiar y solicita guardar o cancelar la configuración.	14.1	- El sistema aplica este color a la interfaz y actualiza las variables de relacionadas con el color.
Sección: "Cambiar estilo de Fuente"			
16	.-El sistema solicita configurar estilos	16.1	- El sistema muestra la interfaz con las diferentes opciones que puede configurar.
17	El usuario solicita Cambiar Estilo de Fuente.	17.1	El sistema busca el archivo de configuración le de este los estilos que están disponibles y luego los muestra en una nueva interfaz.
18	El usuario selecciona el color que desea cambiar y solicita guardar o cancelar la configuración.	18.1	El sistema aplica a la interfaz el color que fue seleccionado luego actualiza las variables relacionadas con este dentro del archivo de configuración.
Referencias:	R3, R4, R5, R6, R7		
Poscondiciones:-			
CURSO ALTERNO DE LOS EVENTOS			

ACCIÓN DEL ACTOR		RESPUESTA DEL SISTEMA	
4.1	El usuario cancela la configuración		
		4.2	El sistema visualiza el menú inicial.
7.1	El usuario cancela la configuración		
		7.2	El sistema visualiza el menú inicial.
11.1	El usuario cancela la configuración		
		11.2	El sistema visualiza el menú inicial.
15.1	El usuario cancela la configuración		
		15.2	El sistema visualiza el menú inicial.
19.1	El usuario cancela la configuración		
		19.2	El sistema visualiza el menú inicial.
Requerimientos especiales	-		

3.7 Conclusiones

En este capitulo se describieron los actores, sus roles, la funcionalidad del sistema, se crearon y descripción de los casos de uso y se determinaron los requisitos funcionales y no funcionales. Ahora se puede pasar a la siguiente fase de desarrollo.

Capítulo 4 Descripción de la solución propuesta

4.1 Introducción

El diseño es el proceso en el que se aplican distintas técnicas y principios con el propósito de definir un producto con los suficientes detalles para permitir su realización física. Pretende construir un sistema que satisfaga determinada especificación y se ajuste a las limitaciones impuestas por el medio. Respetando los requisitos sobre forma, rendimiento utilización de recursos.

El modelo lógico de datos es la abstracción de los objetos que encontraremos en una aplicación particular y los cuales pueden ser convertidos en elementos de bases de datos mientras que el modelo físico es el resultado de llevar el modelo lógico al sistema elegido para implementar la base de datos. En esta fase la estructura de las entidades, atributos y relaciones dependerá del manejador.

En este capitulo se presentan los diagramas de clases del diseño, se crea el modelo lógico de datos, el modelo físico y el diagrama de despliegue para mostrar las relaciones físicas entre los componentes hardware y software en el sistema final, la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes software. También se exponen los principios de diseños utilizados.

4.2 Diagrama de clases del diseño

4.2.1 Paquete Configuración

El diagrama de clase es el diagrama principal de análisis y diseño para un sistema. Durante el análisis del sistema, el diagrama se desarrolla buscando una solución ideal. Durante el diseño, se usa el mismo diagrama, y se modifica para satisfacer los detalles de las implementaciones. A continuación se muestran los diagramas de clases de nuestro sistema. Para lograr un mayor nivel de claridad en los diagramas se ha representado la clase Smarty sin detalles. Esta puede ser consultada en los anexos.

Figura 6- Configuras estilos

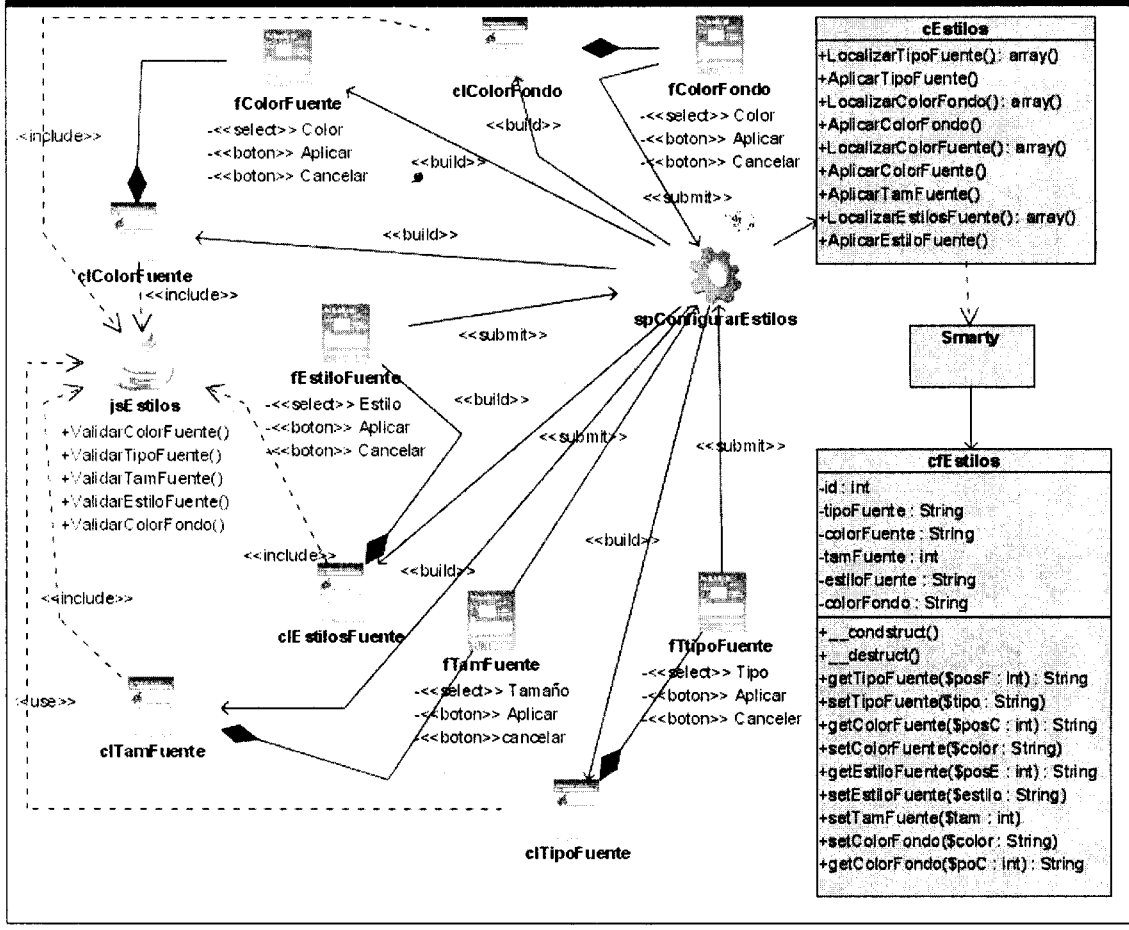
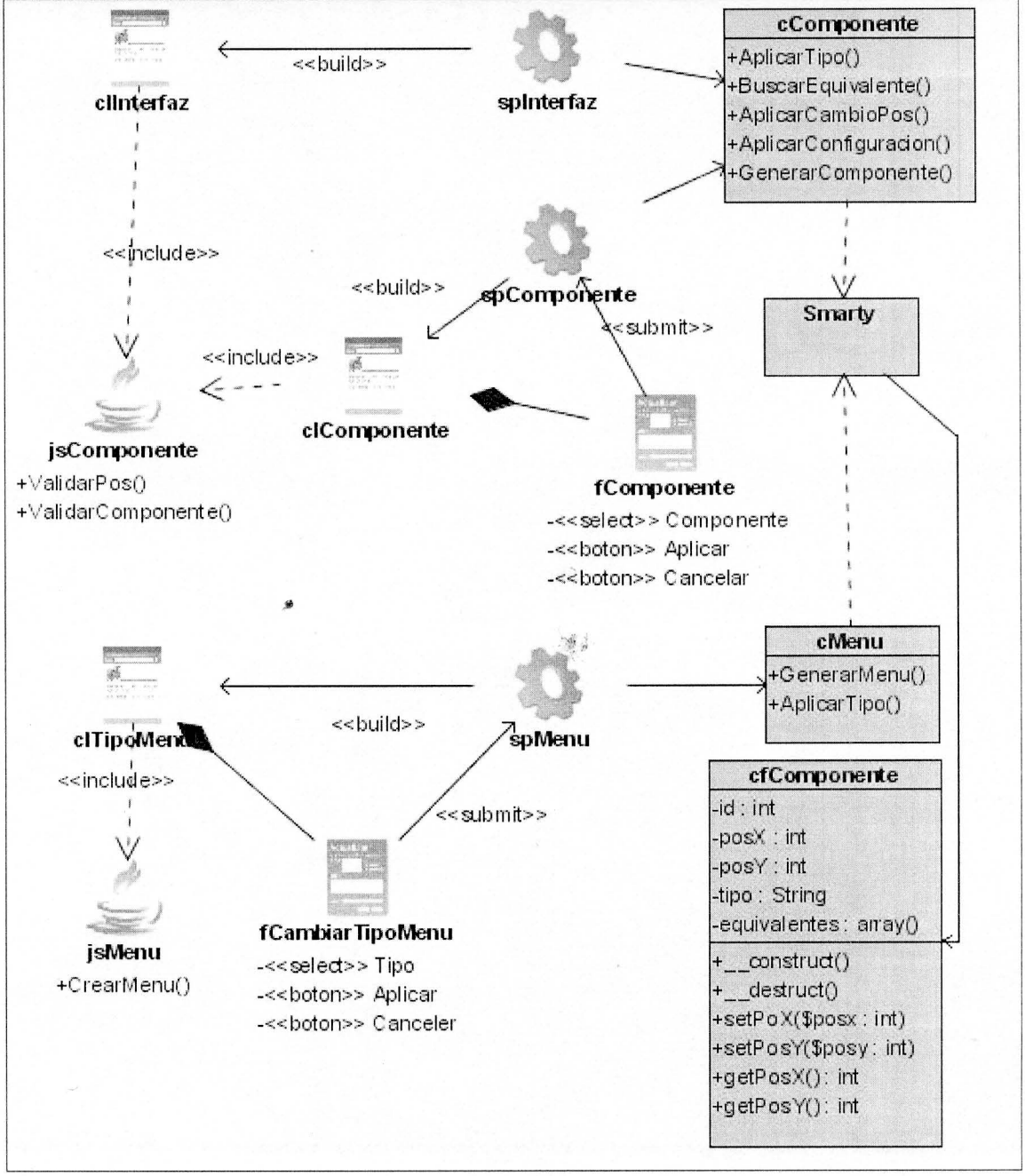


Figura 7- Configurar componentes



4.2.2 Paquete Archivos de Configuración

Figura 8- Gestionar configuración

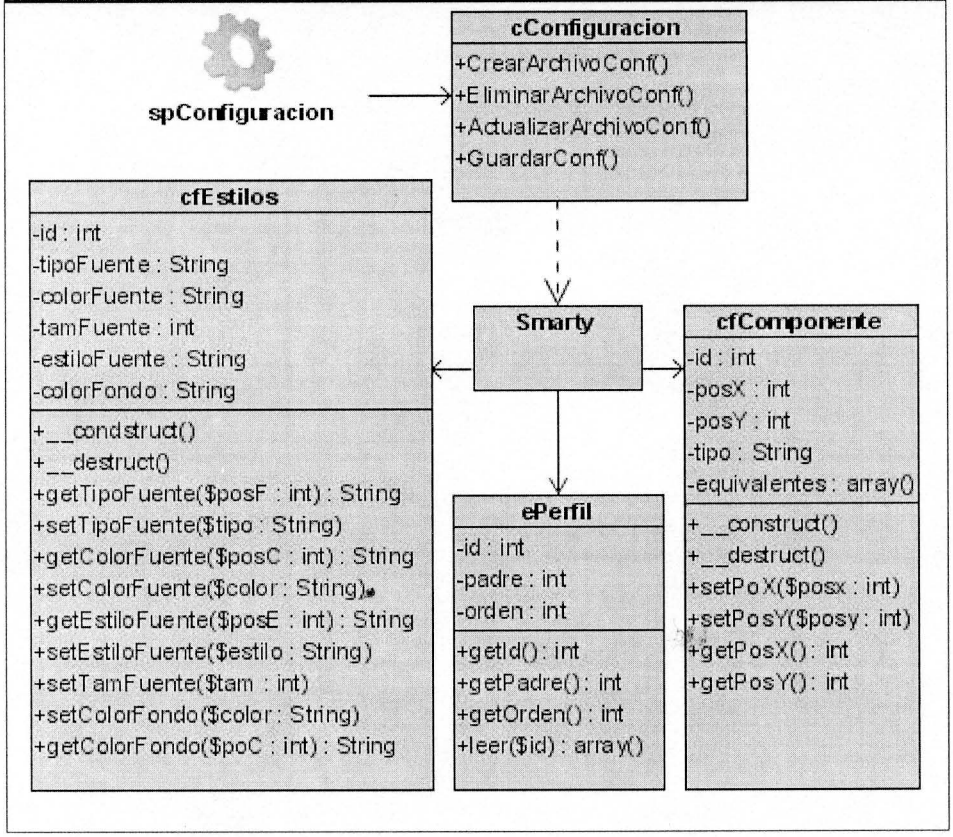
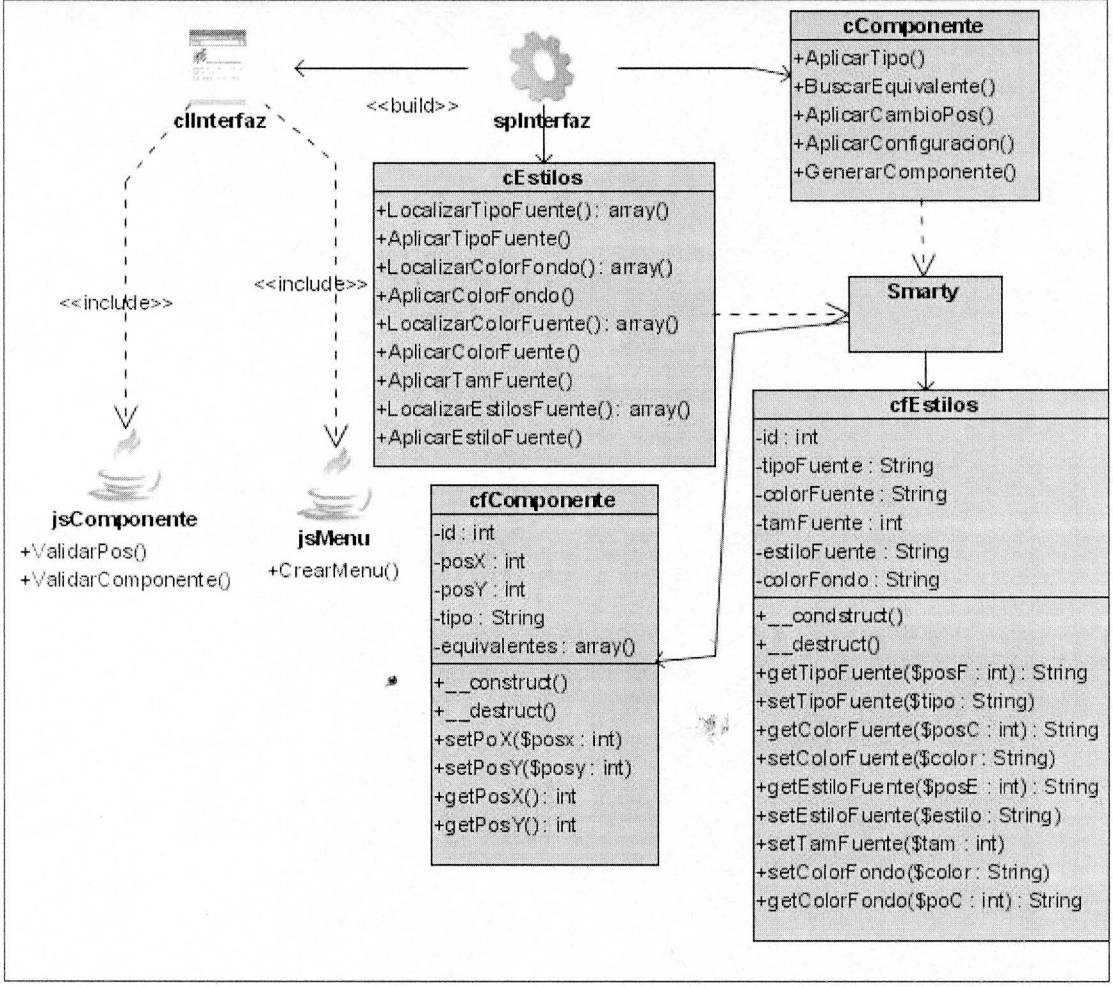
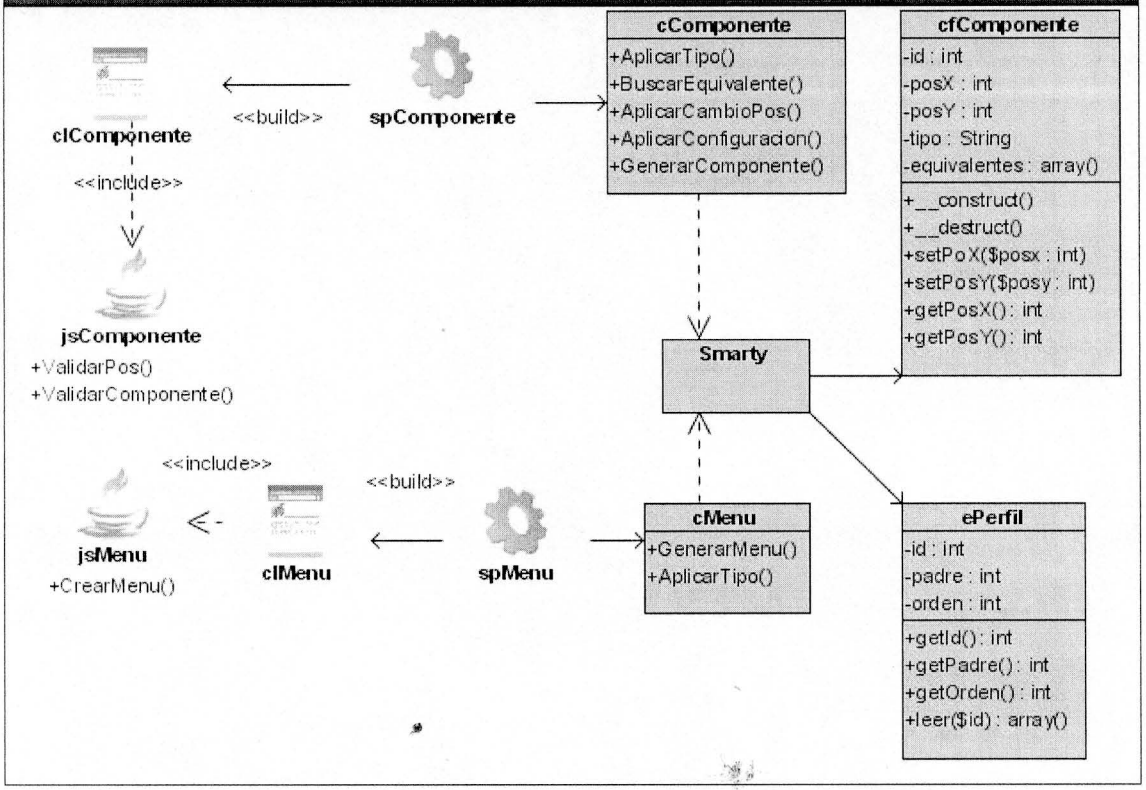


Figura 9- Guardar configuración



4.2.3 Paquete Visualizar Interfaz

Figura 10- Visualizar interfaz



4.3 Principios de diseño

Un aspecto que debemos tener en cuenta a la hora de desarrollar cualquier aplicación es su interfaz. Con el transcurso del tiempo esta ha ido adquiriendo relevancia, principalmente en aplicaciones web. El éxito de la aplicación esta estrechamente ligado con la calidad de su interfaz, quien juega un rol significativo en la usabilidad del sistema. Muchas veces se diseñan interfaces gráficamente atractivas, pero no siempre se logra que tengan la funcionalidad requerida lo que provoca en la mayoría de los casos que los usuarios abandonen nuestro sistema.

El diseño de nuestro sistema esta basado en los usuarios, garantizando los principios de usabilidad ofreciendo interfaces atractivas, sencillas y fáciles de

usar. Brindando privacidad, seguridad y garantía de la información. Flexibilidad en cuanto a forma de uso. Disponiendo de mecanismos para disminuir los riesgos de errores. Evitando acciones redundantes que dificulten la visión de los elementos claves dentro de la interfaz o los objetivos de trabajo.

Dentro del trabajo en equipo se deben administrar y fijar los estándares de nombres, interfaces, mensajes, rutinas reusables, el glosario de nombres. Una parte muy importante que no debe hacerse a un lado es el estándar de codificación. La unificación del código permite que todo el equipo tenga consistencia y ahorra tiempo en la lectura y revisión entre desarrolladores [6] a continuación se mostrará el estándar de codificación definido para nuestro proyecto.

Estándar de codificación en proyectos WEB, lenguaje PHP y Gestor PostgreSQL.

Las carpetas del sitio estarán en minúsculas y español, y tendrán la siguiente estructura:
clases
estilos
js
sesiones
inclusión
imágenes

No deben existir ficheros con muchos subniveles en el sitio, todos los directorios se deben encontrar en la raíz del sitio.

Se debe mantener para el lenguaje JavaScript los mismos estándares de código que en PHP.

Formularios y Proyectos **Objetivo:** Nombrar los formularios y los proyectos de forma estándar para todas las aplicaciones.		
Apariencia de formularios y proyectos	Primera letra en mayúscula	Los nombres de los formularios y proyectos comenzarán con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará la notación PascalCasing*. Ejemplo: MiProyecto.
Carpetas	Los nombres de los directorios serán todos en minúscula y como tendrán la estructura explicada anteriormente.	
Nombre de formularios y proyectos	Relacionados al propósito	El nombre empleado, debe permitir que con sólo leerlo se conozca el propósito del formulario o proyecto. Ejemplo: Sistema para la Demanda de la lucha armada, Sidem.
Clases y Objetos **Objetivo:** Nombrar las clases e instancias de las mismas de forma estándar para todas las aplicaciones.		

Apariencia de clases y objetos	Primera letra en mayúscula	Los nombres de las clases y las instancias de las mismas deben comenzar con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará notación PascalCasing*. Ejemplo: MiClase(). ***Debemos decir que en ocasiones se utilizarán prefijos para denotar el tipo de la clase siempre en minúsculas. Preferiblemente prefijos de una sola letra.***
Nombre de clases y objetos	Relacionados al propósito	El nombre empleado, debe permitir que con sólo leerlo se conozca el propósito de la clase o instancia de la misma. Para el caso de las instancias es recomendable que se denoten así: Para la clase: Nomumedida su instancia será $Oumedida, de forma tal que la primera letra indique que es un objeto y el resto, la clase a la que pertenece.
Apariencia de atributos	Primera letra en minúscula	El nombre que se le da a los atributos de las clases debe comenzar con la primera letra

		en minúscula, en caso de que sea un nombre compuesto se empleará notación CamelCasing**.
Nombre de atributos	Nemotécnicos	El nombre empleado, debe permitir que con sólo leerlo se conozca el propósito del mismo dentro de la clase. Ejemplo: $nTabla, este atributo denota el nombre de una tabla.
Apariencia de las funciones	Primera letra en mayúscula	Los nombres de las funciones deben comenzar con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará notación PascalCasing*. Ejemplo: function BuscaUnidad(). Si son funciones que obtienen un dato se emplea el prefijo **get** y si fijan algún valor se emplea el prefijo **set.**
Nombre de las funciones	Nemotécnicos	El nombre empleado, debe permitir que con sólo leerlo se conozca el propósito de la misma dentro de la clase.
Declaración de parámetro en funciones	Agrupados por tipos Poner los string 1 numéricos 2, además, agrupar según	Los parámetros que se le pasan a las funciones se recomienda sean declarados de forma tal que estén agrupados por el tipo de dato que

	valores por defecto	contienen. Ejemplo: BuscaUnidad($nTabla (string), $nCampos(string), $kIndice (entero)).
Variables y constantes		
Apariencia de variables	Primera letra en minúscula.	El nombre que se le da a las variables debe comenzar con la primera letra en minúscula, en caso de que sea un nombre compuesto se empleará notación CamelCasing**.
Apariencia de constantes	Todas sus letras en mayúscula	Se deben declarar las constantes con todas sus letras en mayúscula.
Nombres de las variables y constantes	Nemotécnicos	El nombre empleado, debe permitir que con sólo leerlo se conozca el propósito de la misma. Ejemplo: $nFields.
Declaración de constantes y asignación a variables	Una por cada línea	Se recomienda declarar una constante por cada línea y con las asignaciones a las variables sucede lo mismo. Ejemplo: define("CONSTANT1","value1"); define("CONSTANT2","value2"); $nTabla='nomproducto'; $kIndice=0;
Identación **Objetivo:** Lograr una estructura uniforme para los bloques de código así como para los diferentes niveles de anidamiento.		
0 espacios en	Require	No se empleará ningún espacio

blanco desde la izquierda en	Include Class	en blanco desde la izquierda para las instrucciones antes mencionadas. Se tomará como inicio de la página el tag PHP <?
2 espacio en blanco desde la izquierda en	Function Define	Se dejarán dos espacios en blanco desde la izquierda en las instrucciones antes mencionadas.
2 espacio en blanco desde la referencia en	Inicio y fin de bloque	Se recomienda dejar dos espacios en blanco desde la instrucción anterior para el inicio y fin de bloque {}. Lo mismo sucede para el caso de las instrucciones If, else, For, While, Do While, Switch, Foreach.
Niveles de anidación	Hasta 5 niveles	Se recomienda emplear hasta 5 niveles de anidación en instrucciones If, For, While.

Ejemplo de identación

```
<?
require ('class/Interface.php');

class MiClase
{
  function BuscaUnidad($nTabla, $nFields, $kIndice)
  {
    if ($nTabla)
    {
      ...
```

```
    }
    for (...)
    {
      ...
    }
  }
}
?>
```

Comentarios, separadores, líneas y espacios en blanco **Objetivo:** Establecer un modo común para comentar el código de forma tal que sea comprensible con sólo leerlo una vez.		
Ubicación de comentarios	Al inicio de cada clase o función y al final de cada bloque de código.	Se recomienda comentar al inicio de la clase o función especificando el objetivo de la misma así como los parámetros que usa (especificar tipos de dato, y objetivo del parámetro) entre otras cosas. Y se comenta también cuando se cierran los ciclos, clases, instrucciones if y otras.
Separador de instrucciones	Se emplea el punto y coma.	Se recomienda usar el separador al final de cada instrucción y no en la línea de abajo. Ejemplo: define ("CONSTANT","value1");
Líneas en blanco	Se emplean antes de cada función.	Se recomienda dejar una línea en blanco antes de la definición de cada función para dar claridad al código.

Espacios en blanco	Entre operadores lógicos y aritméticos.	Se recomienda usar espacios en blanco entre estos operadores para lograr una mayor legibilidad en el código. Ejemplo: $nTabla = 'nomproducto'; if (($nTabla) && ($nFields))
Bases de Datos, Tablas, esquemas y Campos		
Apariencia de la BD	Primera letra en mayúscula	Los nombres de las BDs deben comenzar con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará notación PascalCasing*. Ejemplo: ContMaterial.
Nombres de las BDs	Nemotécnicos y relacionados al propósito.	El nombre empleado, debe permitir que con sólo leerlo se conozca el propósito de la misma.
Apariencia de los esquemas	Todas las letras en minúscula.	El nombre a emplear para los esquemas debe escribirse con todas las letras en minúscula para evitar problemas con el Case Sensitive del gestor. Ejemplo: create schema 'finanzas';
Nombres de los esquemas	Nemotécnicos y relacionados al propósito.	El nombre empleado, debe permitir que con sólo leerlo se conozca el propósito del mismo.

Apariencia de las tablas	Todas las letras en minúscula.	El nombre a emplear para las tablas debe escribirse con todas las letras en minúscula para evitar problemas con el Case Sensitive del gestor. Ejemplo: create table 'nom_producto';
Nombres de las tablas	Nemotécnicos y relacionados al propósito. Además clasificando las tablas por su tipo.	El nombre empleado, debe permitir que con sólo leerlo se conozca el propósito del mismo. Se deben clasificar las tablas por su tipo, es decir por los datos que contienen se le coloca un prefijo, se pueden clasificar en: Nomencladores, tablas de datos, de auditoría, de seguridad, de configuración etc... Ejemplo: Nomencladores nom_... Auxiliares aux_... Datos dat_... Históricas his_... Seguridad seg_... Temporales tmp_... Configuración cfg_...
Apariencia de los campos	Todas las letras en minúscula.	El nombre a emplear para los campos debe escribirse con todas las letras en minúscula

		para evitar problemas con el Case Sensitive del gestor. Ejemplo: add field 'idproducto';
Nombre de los campos	Nemotécnicos En caso de identificadores, emplear id...(Ejemp: idmunic), este seria igual en la tabla de datos que lo emplea.	El nombre empleado, debe permitir que con sólo leerlo se conozca el propósito del mismo. Además se debe incluir un comentario en la descripción del mismo. Ejemplo: cantemb: cantidad de embalajes.
Nombre de las llaves primarias	Nemotécnicos empleando prefijos.	Se nombrarán las llaves primarias de forma que se vea de qué tabla es y que es primaria. Ejemplo: pk_cuenta. (Llave primaria de la tabla cuenta). Si es una llave compuesta se coloca el prefijo y en nemotécnico los campos que la forman.
Nombre de las llaves foráneas.	Nemotécnicos empleando prefijos.	Se nombrarán las llaves foráneas de forma que se vea de qué tabla es y que es foránea. Ejemplo: fk_cuenta. (Llave foránea de la tabla cuenta). Si es una llave compuesta se coloca el prefijo y en nemotécnico los campos que la forman.
Nombre de las	Nemotécnicos	Se nombrarán las secuencias

secuencias	empleando prefijos.	de forma que se vea de qué campo es y que es una secuencia. Ejemplo: seq_idcuenta. (Secuencia del campo idcuenta).
Restricciones Únicas y de Chequeo	Nemotécnicos empleando prefijos.	Ejemplo: (u_ o c_) + nombre del campo que la emplea. Ejemplo: u_idmunic... c_cuenta...
Nombres de las funciones, triggers, y vistas	Prefijos + Nemotécnicos	El nombre empleado, debe permitir que con sólo leerlo se conozca el propósito del mismo. Ejemplo: ft_ Funciones de triggers. vw_ Vistas Ejemp: ft_calcidentificador

- ***Notación PascalCasing:** Los identificadores y nombres de variables, métodos y funciones están compuestos por múltiples palabras juntas iniciando cada palabra con letra mayúscula. Ejemplo: NotacionPascalCasing.
- ****Notación CamelCasing:** Los identificadores y nombres de variables, métodos y funciones están compuestos por múltiples palabras juntas iniciando cada palabra con letra mayúscula excepto la primera palabra que debe iniciar con minúscula. Ejemplo: notacionCamelCasing.

4.3.1 Ayuda

En el sistema general esta compuesto por varios módulos, cada uno de estos gestiona la ayuda de forma independiente. De esta forma se encontrará accesible en todas las aplicaciones en dependencia del módulo al que el usuario esté accediendo en un momento determinado. Será fácil de localizar y brindara las opciones necesarias para lograr que los usuarios se familiaricen con el sistema y puedan utilizarlo de forma eficiente. A esta ayuda se le suma un complemento por parte del módulo de interfaz. Este complemento estará estrechamente relacionado con la personalización de la interfaz brindando la información necesaria para que los usuarios puedan utilizar las opciones de configuración de interfaces.

4.4 Tratamiento de errores

Semejante a lo que sucede con la ayuda ocurre con el tratamiento de errores. Cada uno de los módulos que compone el sistema posee su propio tratamiento de errores. En cuanto al modulo de interfaz se puede mencionar que adiciona su propio método estrechamente relacionado con el trabajo con ficheros y la recuperación de datos. Las funciones definidas en la clase Smarty permitirán darle solución a la mayoría, también se han definido otras funciones JvaScript que se encargaran del control de los errores relacionados con la visualización de componentes.

4.5 Diseño de la base de datos

El modelado de datos es el proceso de ordenar los datos y sus relaciones con el fin de desarrollar el modelo lógico. Los objetivos que se pretenden son conseguir

estructuras de datos flexibles, estables y normalizados además de separar procesos de los datos.

La modelación los datos procesados por un sistema de información se realiza en diferentes niveles consecutivos de abstracción:

- **Nivel Conceptual**: a este nivel se realiza una formalización de los datos almacenados en el sistema mediante una descripción de las entidades, los atributos de estas entidades y las posibles relaciones entre ellas. Este modelo se realiza durante la fase de análisis del sistema.
- **Nivel Lógico**: mientras que el modelo conceptual es independiente del tipo de software de gestión de información, en el nivel lógico se realiza la adaptación de aquel modelo al tipo de sistema de gestión de datos que se vaya a utilizar. Al final se obtiene un modelo lógico de registros que representa la estructura de los datos en dicho sistema. Este modelo se realiza durante la fase de diseño del sistema, se suele completar con información adicional sobre el volumen de los datos y la forma de acceso a los mismos.
- **Nivel Físico**: a este nivel se debe determinar cómo se organiza físicamente el almacenamiento de los datos en ficheros. Todos estos detalles se pueden ignorar, ya que son competencia del sistema de gestión de datos que se utilice.

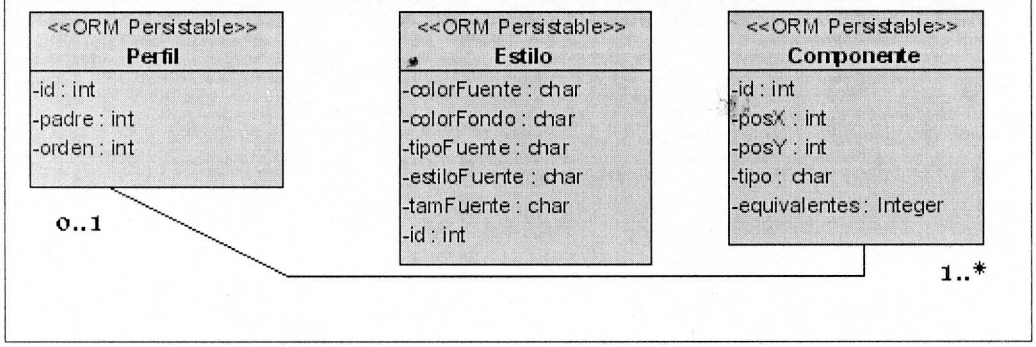
Con la modelación de los datos, además de especificar las características de la información, se pretende conseguir la simplificación de las estructuras definidas, buscando y eliminando los elementos de datos redundantes, para lo cual se produce una reorganización de estas estructuras para eliminar repeticiones, proceso que se conoce con el nombre de normalización.

Este proyecto, como se expuso anteriormente está formado por varios módulos relacionados entre sí. Nuestro módulo tiene la responsabilidad de gestionar las interfaces de usuario. La que está estrechamente relacionada con la capa de presentación y con la configuración de los componentes visuales de interfaz. Como se conoce, en la capa de datos es donde se almacena el contenido que

será procesado y luego visualizado en la capa de presentación. En este proyecto cada módulo es el responsable de diseñar su base de datos y determinar la forma de su información así como de aportar dichos datos al modulo de interfaz, de esta forma la información persistente que manipulamos en el modulo de interfaz es la relacionada con las variables de configuración de las interfaces de usuario. Smarty incorpora un mecanismo para el tratamiento de dichas variables. Este mecanismo esta basado en ficheros de configuración por tanto la información persistente, la guardamos en dichos ficheros. Esto trae algunas ventajas por ejemplo evita la realización de consultas por lo tanto disminuye el acceso a la base de datos.

4.5.1 Modelo lógico de datos

Figura 11- Diagrama de clases persistentes.



Para proceder al diseño de los ficheros y de las bases de datos del sistema, se debe convertir previamente el modelo conceptual que incluía tipos de entidades y relaciones con atributos asociados, en un modelo lógico que únicamente considere tipos de registros compuestos por campos de datos.

4.5.2 Modelo físico de datos

El modelo físico de datos es el resultado de llevar el modelo lógico al sistema elegido para implementar la BD. En esta fase la estructura de las entidades, atributos y relaciones dependerá del manejador. Cada uno de los elementos del modelo lógico se tiene que transformar en un elemento del modelo físico. En algunos casos la transformación es directa porque el concepto se soporta igual en ambos modelos, pero otras veces no existe esta correspondencia, por lo que es necesario buscar una transformación que conserve lo mejor posible la semántica, teniendo en cuenta los aspectos de eficiencia que sean necesarios en cada caso. [3]

El último paso en la relación con los datos que utilizará un sistema de información, consiste en la elección de la organización física que soporte los métodos de acceso a los datos establecidos anteriormente. Durante el diseño físico también se seleccionan las claves de acceso a los ficheros y se eligen las claves alternativas. Se crean, por tanto, los ficheros índices para posibilitar accesos alternativos.

4.6 Diagrama de despliegue

Un diagrama de despliegue se define como una colección de nodos y arcos. Donde un nodo puede ser un elemento físico que existe en tiempo de ejecución y representa un recurso computacional que, por lo general tiene memoria y capacidad de almacenamiento. Con el siguiente diagrama de despliegue pretendemos describir la topología del sistema que se está modelando, mostrando la configuración de nodos que participan en la ejecución y de los componentes que residen en ellos.

Nuestro sistema está formado por una arquitectura web clásica, uno o varios clientes y un servidor web.

Figura 12- Diagrama de despliegue

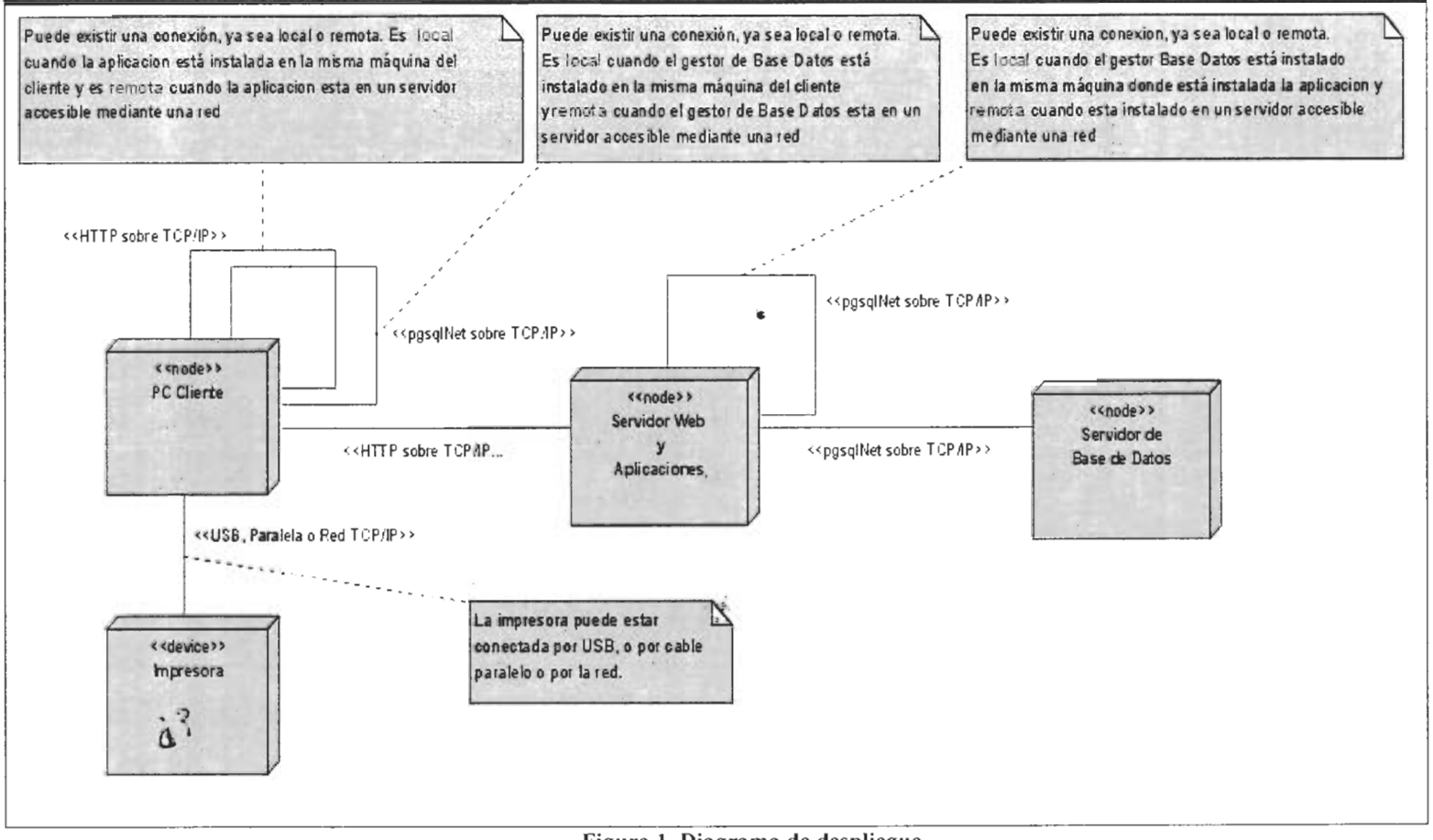


Figura 1. Diagrama de despliegue

4.7 Conclusiones

En este capitulo se presento el diagrama de clases del diseño. Se destacaron los principios de diseño a seguir. Se creó el modelo lógico y físico de datos, se construyó el diagrama e despliegue. Explicando la lógica del negocio del sistema mediante el uso de diagramas de clases.

Capítulo 5 Estudio de factibilidad

5.1 Introducción

Antes de acometer la realización de algún proyecto debemos saber si su implementación será factible o no por esta razón es necesario realizar el estudio de factibilidad. De esta forma se puede determinar el tiempo estimado de realización, cantidad de personas necesarias para desarrollarlo edemas del costo del proyecto.

En este capitulo se realiza dicho estudio de factibilidad utilizando COCOMO. Empleando el método de estimación basado en casos de uso, este permite documentar los requerimientos de un sistema en términos de actores y casos de uso.

5.2 Planificación basada en casos de uso

La especificación de los requerimientos mediante casos de uso ha probado ser uno de los métodos más efectivos para capturar la funcionalidad de un sistema. Este hecho se puede apreciar en algunas metodologías actuales ampliamente difundidas, como el Proceso Unificado de Rational (Rational Unified Process) donde se propone especificar la funcionalidad de los sistemas mediante la utilización de casos de uso. El método de casos de uso permite documentar los requerimientos de un sistema en términos de actores y casos de uso.

Tipo de actor	Descripción	Factor de peso	Actores	Total
Simple	Sistema con sistema a través de interfaz de programación.	1	1	1
Medio	Sistema con sistema mediante protocolo de interfaz basada en texto.	2	0	0
Complejo	Persona que interactúa con el sistema mediante interfaz gráfica.	3	1	3
			UAW	4

UAW = S(Factor * Actores)

Tipo de CU	Descripción	Peso	Cantidad de CU	Total
Simple	El caso de uso tiene de 1 a 3 transacciones.	5	1	5
Medio	El caso de uso tiene de 4 a 7 transacciones.	10	3	30
Complejo	El caso de uso tiene más de 8	15	1	15

	transacciones.			
			UUCW	50

UUCW = Sumatoria (Factor * CantCU)

UUCP = UAW + UUCW

Factor	Descripción	Peso	Valor asignado	Total
T1	Sistema distribuido	2	5	10
T2	Tiempo de respuesta	1	2	2
T3	Eficiencia del usuario final	1	4	4
T4	Funcionamiento Interno complejo	1	2	2
T5	El código debe ser reutilizable	1	4	4
T6	Facilidad de instalación	0,5	1	0,5
T7	Facilidad de uso	0,5	4	2
T8	Portabilidad	2	4	8
T9	Facilidad de cambio	1	3	3
T10	Concurrencia	1	5	5
T11	Incluye objetivos especiales de seguridad	1	4	4
T12	Provee acceso directo a terceras partes	1	0	0
T13	Se requieren facilidades especiales de entrenamiento de usuarios	1	1	1
			Sumatoria	45,5
TCF = 0.6 + 0.01 * Sumatoria(Peso * Valor)			TCF	1,055

Factor	Descripción	Peso	Valor asignado	Total
E1	Familiaridad con el modelo de proyecto utilizado	1,5	3	4,5

E2	Experiencia en la aplicación	0,5	3	1,5
E3	Experiencia en la orientación a objetivos.	1	4	4
E4	Capacidad del analista líder.	0,5	5	2,5
E5	Motivación.	1	5	5
E6	Estabilidad de requerimientos	2	3	6
E7	Personal Part–Time	-1	1	-1
E8	Dificultad del lenguaje de programación	-1	3	-3
			Sumatoria	19,5

EF = 1.4 - 0.03 * Sumatoria(Peso * Valor)	EF	0,815
Factor de conversión	CF	20
UCP = UUCP * TCF * EF	UCP	46,43055
E = UCP * CF	E	928,611

Actividad	**Porcentaje %**	**Horas-Hombres**
Análisis	10	232,15275
Diseño	20	464,3055
Implementación	40	928,611
Pruebas	15	348,229125
Sobrecarga (otras actividades)	15	348,229125
Total	100	2321,5275

Esfuerzo Total (Horas--Hombre)	ET1	2321,5275
Esfuerzo Total (Mes--Hombre)	ET2	14,3304167
Salario Promedio	SM	50
Cantidad de Hombres	CH	4
Costo Hombre--Mes	CHM	200
Costo Total	Costo	2866,0

La realización de este proyecto supone un periodo de alrededor de 14 meces para un solo trabajador que utilice 6 horas diarias de trabajo utilizando 27 días

de cada mes, por tanto para un equipo que está formado por cuatro integrantes y suponiendo que los cuatro trabajen de forma uniforme este se lograría desarrollar en tres meses y medios con un costo de $2866. Por tanto puede resultar factible su desarrollo.

5.3 Beneficios tangibles e intangibles

La implantación de esta aplicación trae consigo una serie de beneficios, pues disminuirá los costos de producción de otras aplicaciones semejantes que requieran el manejo de interfaces dinámicas y sean para el uso de la institución en cuestión. Le aportará grades beneficios a los usuarios al brindarles una interfaz bastante funcional y hasta cierto punto configurable de acuerdo a sus necesidades trayendo consigo el ahorro de tiempo.

5.4 Análisis de costos y beneficios

Para el desarrollo y puesta en funcionamiento de este sistema no se necesitan grandes gastos de recursos ni de tiempo de desarrollo. Se han utilizado herramientas de software libre lo que trae consigo un considerable ahorro relacionado con los pagos de licencias. Como se utiliza una arquitectura basada en capas, un cambio en los requerimientos no incurriría en grandes gastos de mantenimiento. La portabilidad del sistema garantizará la adaptación a varios sistemas operativos incurriendo en gastos mínimos. Por lo antes expuesto se ha llegado a la conclusión que la construcción de este sistema es factible.

5.5 Conclusiones

En este capítulo se analizó la factibilidad de realización del sistema, se determinó el costo de producción del mismo, el tiempo que se estimó en que debía de estar listo el sistema, el esfuerzo que debía realizar el equipo de desarrollo y la cantidad de personas necesarias para la realización del sistema.

Conclusiones

En el presente trabajo investigativo se presentó el diseño de un producto de software para dar tratamiento a las interfaces de usuarios en aplicaciones Web. Se analizaron aspectos relacionados con las tendencias actuales de desarrollo Web también se estudió la forma en que se manejaba este tema en la institución determinando las principales dificultades que dieron origen al desarrollo de este trabajo.

Se expusieron los conceptos generales teóricos que permitieron ubicar a cualquier persona que consultara este trabajo en el tema que se estaba tratando. Aquí se explicaron las definiciones que serían la base sobre la que se sustentó el resto del documento. En esta sección se incluyó una breve descripción sobre el estado del arte de la situación problemática abordada y de la tecnología propuesta para su solución. Se realizó un estudio para determinar los procesos de negocios que son llevados a cabo en la institución determinando los conceptos significativos así como la relación existente entre ellos.

Teniendo en cuenta la experiencia del equipo de trabajo en este tipo de proyecto, se puede decir que el trabajo realizado ha sido de gran aporte para la institución como para la formación de cada uno de los miembros del equipo. De esta forma se han cumplido en gran medida los objetivos trazados.

Recomendaciones

Luego de haber desarrollado esta fase del trabajo y basado en la experiencia adquirida se harán las siguientes recomendaciones:

- Desarrollar una herramienta que permita generar las interfaces básicas para cada uno de los módulos que forman el sistema.
- Como el desarrollo de este trabajo ha sido solo una fase dentro de un proyecto general que es mucho más abarcador, se recomienda dar continuación al mismo, de esta forma podrá ofrecer todos los resultados deseados.

Referencias bibliográficas

[1]-Craig Larman, *UML y Patrones*

Miguel Katrib Mora, *El Proceso Unificado de Desarrollo.*

[2]-Booch, G., Rumbaugh, J., Jacobson, I. *"El Lenguaje Unificado de Modelado"*. Addison-Wesley. 1999.

[3]-*Técnicas y prácticas*. Ministerio de Administraciones Públicas

Monrose S, Ciudad F. *EMBRIOCIM – Enciclopedia de Embriología Médica – Colección GALENOMEDIA.* Trabajo de Diploma para optar por el título de Ingeniero Informático, Instituto Superior Politécnico "José Antonio Echeverría", Ciudad de la Habana, Julio de 2004.

[4]-Espinosa A. *Sistema para la administración unificada de usuarios.* Trabajo de Diploma para optar por el título de Ingeniero Informático, Instituto Superior Politécnico "José Antonio Echeverría", Ciudad de la Habana, junio del 2004.

[5]-Leyva M. *Sitio Web del grupo de estudio de desastres.* Trabajo de Diploma para optar por el título de Ingeniero Informático, Instituto Superior Politécnico "José Antonio Echeverría", Ciudad de la Habana, junio del 2005

[6]- *Simulación digital de procesos constructivos* http://micigc.uniandes.edu.co/Investigaciones20y 20Desarrollo/investigaciones2004-10/actualizacion ISPLAN.pdf

Bibliografía

1- *Modelo del dominio.* http://iie.fing.edu.uy/ense/asign/desasoft/Teorico2/ Dominio.pdf. (10-3-06)

2- *Diseño con dreamweaver MX 2004.* http://www.ciberaula.com/curso/ dreamweaver/que_es/ (17-03-06).

3- *PostGreSQL vs. MySQL.* http://www.netpecos.org/docs/mysql_ postgres/ x57.html (17-03-06).

4- *¿Qué es PHP?.* http://www.ulfix.net/content/view/417/114/ (24 03 2006).

5- *Diagramas de Casos de Uso.* http://www-gris.det.uvigo.es/~avilas/UML/ node25.html (30-03-06).

6- *Identificar Casos de Uso del sistema.* http://www.cs.ualberta.ca/~pfiguero/ soo/metod/ requerimientos. html#act1 (30-03-06).

7- *Fase de Planificación y Especificación de Requisitos.* http://www.clikear.com/ anuales/uml/faseplanificacion.asp (30-03-06).

8- *Documentation.* http://smarty.php.net/docs.php

9- *Páginas dinâmicas.* http://www.desarrolloWeb.com/manuales/7/.. (02-2006).

10- *CM SR atings.* http://www.opensourcecms.com/index.php?Option=content& task= view&id =388. (02-2006).

11- *Ventajas de incorporar un CMS a tu Web.* http://proyectoblong.blogspot.com/ 2005/09/ventajas-de-incorporar-un-cms-tu-web.html (02-2006).

12- *Programación Web.* http://lenguajes-de-programacion.com/programación - web.shtml (02-2006).

13 *Desarrollo de una Aplicación en tres Capas con VS .NET.* http://www.microsoft.com/spanish/msdn/comunidad/mtj.net/voices/art140.asp (03-06-06).

14 Morris L, Duplessis D. *Sistema Automatizado de Gestión de Colecciones.* Trabajo de Diploma para optar por el título de Ingeniero Informático, Universidad de Holguín "Oscar Lucero Moya", Holguín, junio del 2004.

15 Martínez D. *Sitio Web de la dependencia de transporte de la universidad de Holguín.* Trabajo de diploma para optar por el título de ingeniero en sistemas informáticos. Universidad de Holguín "Oscar Lucero Moya", Holguín, junio del 2004.

16 Cleger S. *INTRANET CORPORATIVA HOTEL PLAYA PESQUERO.* Trabajo para optar por el título de Ingeniero Informático. Universidad de Holguín "Oscar Lucero Moya", Holguín, junio del 2004.

17 Kanzaki T. *EDITOR DE ECUACIONES MULTIMEDIA INTELIGENTE PARA ÓPTIMAWEB.* Trabajo de Diploma para optar por el título de Ingeniero Informático, Instituto Superior Politécnico "José Antonio Echeverría", Ciudad de la Habana, junio del 2004.

18 María N, Saborit Y. *SISTEMA DE CATALOGACIÓN Y RECUPERACIÓN DE RECURSOS DE INFORMACION Hubble.* Trabajo de Diploma para optar por el título de Ingeniero Informático, Instituto Superior Politécnico "José Antonio Echeverría", Ciudad de la Habana, junio del 2004.

Glosario de términos

Cookies Pequeños trozos de información textual que el servidor Web envía al navegador y que el navegador devuelve sin modificar cuando visita más tarde la misma site o dominio.

CMS (Content Management System). Un CMS es una herramienta que permite crear y administrar el contenido de una página Web.

MVC (Model – View – Controller) modelo de tres capas, que consiste en desacoplar los tres niveles de complejidad de la solución software: capa Model (Datos), capa View (interfaz) y capa Controller (lógica de negocio).

PHP (Personal Home Page) Es un lenguaje interpretado de alto nivel impregnado en páginas HTML y ejecutado en el servidor.

SGBD Sistemas de gestión de bases de datos.

Smarty Motor de plantillas para PHP.

SQL (Structured Query Language), Lenguaje de alto nivel, no procedural, normalizado, que permite la consulta y actualización de los datos de base de datos relaciónales.

Anexo 1 Clase Smarty

Clase Smarty

Smarty
$template_dir = 'c:\Inetpub\Smarty\templates'
$compile_dir = 'c:\Inetpub\Smarty\templates_c'
$config_dir = 'c:\Inetpub\Smarty\configs'
$plugins_dir = array('plugins')
$debugging = false
$error_reporting = null
$debug_tpl = ''
$debugging_ctrl = 'NONE'
$compile_check = true
$force_compile = false
$caching = 0
$cache_dir = 'c:\Inetpub\Smarty\cache'
$cache_lifetime = 3600
$cache_modified_check = false
$php_handling = SMARTY_PHP_PASSTHRU
$security = false
$secure_dir = array()
$security_settings = array('PHP_HANDLING' => false, ...
$trusted_dir = array()
$left_delimiter = '{'
$right_delimiter = '}'
$request_vars_order = 'EGPCS'
$request_use_auto_globals = true
$compile_id = null
$use_sub_dirs = false
$default_modifiers = array()
$default_resource_type = 'file'
$cache_handler_func = null
$autoload_filters = array()
$config_overwrite = true
$config_booleanize = true
$config_read_hidden = false

Clase Smarty. continuación

```
$config_fix_newlines = true
$default_template_handler_func = ''
$compiler_file = 'c:\Archivos de programa\PostgreSQL\8.0\include\libs\Smarty_Comp...
$compiler_class = 'Smarty_Compiler'
$config_class = 'Config_File'
$_tpl_vars = array()
$_smarty_vars = null
$_sections = array()
$_foreach = array()
$_tag_stack = array()
$_conf_obj = null
$_config = array(array('vars' => array(), 'files' => array()))
$_smarty_md5 = 'f8d698aea36fcbead2b9d5359ffca76f'
$_version = '2.6.10'
$_inclusion_depth = 0
$_compile_id = null
$_smarty_debug_id = 'SMARTY_DEBUG'
$_smarty_debug_info = array()
$_cache_info = array()
$_file_perms = 0644
$_dir_perms = 0771
$_reg_objects = array()
$_plugins = array(                    'modifier'    => array(),                    ...
$_cache_serials = array()
$_cache_include = null
$_cache_including = false
```

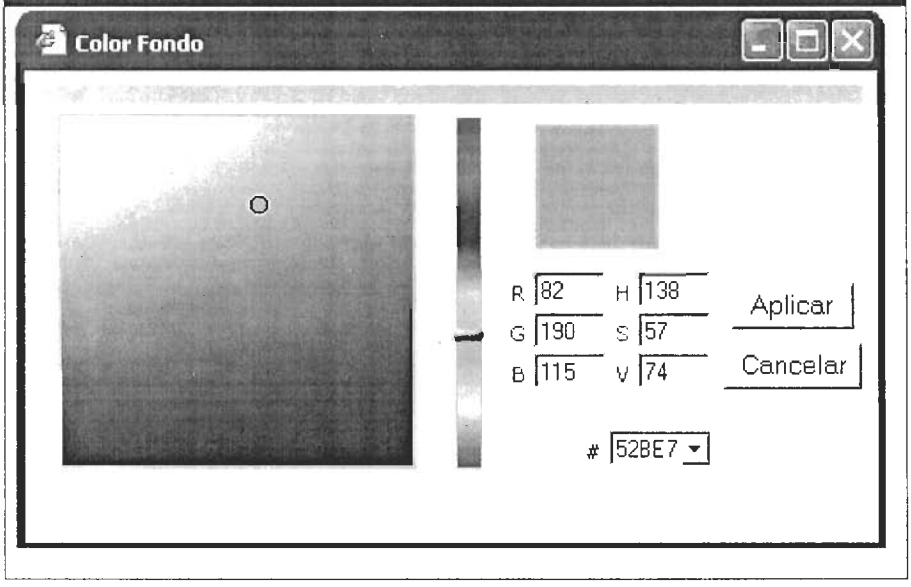
```
Smarty()
assign($tpl_var, $value = null)
assign_by_ref($tpl_var, $value &)
append($tpl_var, $value = null, $merge = false)
append_by_ref($tpl_var, $value &, $merge = false)
clear_assign($tpl_var)
register_function($function, $function_impl, $cacheable = true, $cache_attrs = null)
```

Clase Smarty. Final

```
unregister_function($function)
register_object($object, $object_impl &, $allowed = array(), $smarty_args = true, $blo...
unregister_object($object)
register_block($block, $block_impl, $cacheable = true, $cache_attrs = null)
unregister_block($block)
register_compiler_function($function, $function_impl, $cacheable = true)
unregister_compiler_function($function)
register_modifier($modifier, $modifier_impl)
unregister_modifier($modifier)
register_resource($type, $functions)
unregister_resource($type)
register_prefilter($function)
unregister_prefilter($function)
register_postfilter($function)
unregister_postfilter($function)
register_outputfilter($function)
unregister_outputfilter($function)
load_filter($type, $name)
clear_cache($tpl_file = null, $cache_id = null, $compile_id = null, $exp_time = null)
clear_all_cache($exp_time = null)
is_cached($tpl_file, $cache_id = null, $compile_id = null)
clear_all_assign()
clear_compiled_tpl($tpl_file = null, $compile_id = null, $exp_time = null)
template_exists($tpl_file)
get_template_vars($name = null) &
get_config_vars($name = null) &
trigger_error($error_msg, $error_type = E_USER_WARNING)
display($resource_name, $cache_id = null, $compile_id = null)
fetch($resource_name, $cache_id = null, $compile_id = null, $display = false)
config_load($file, $section = null, $scope = 'global')
get_registered_object($name) &
clear_config($var = null)
_get_plugin_filepath($type, $name)
_is_compiled($resource_name, $compile_path)
_compile_resource($resource_name, $compile_path)
_compile_source($resource_name, $source_content &, $compiled_content &, $cach...
_get_compile_path($resource_name)
_fetch_resource_info($params &)
_parse_resource_name($params &)
_run_mod_handler()
_dequote($string)
_read_file($filename)
_get_auto_filename($auto_base, $auto_source = null, $auto_id = null)
_unlink($resource, $exp_time = null)
_get_auto_id($cache_id = null, $compile_id = null)
_trigger_fatal_error($error_msg, $tpl_file = null, $tpl_line = null, $file = null, $line = nu...
_process_compiled_include_callback($match)
_smarty_include($params)
```

Anexo 2 Interfaces de usuario.

Interfaz Seleccionar Color de fondo



Interfaz Seleccionar Color de fuente