Desarrollo de Aplicaciones WEB por Componentes – Código Libre

Development of WEB Applications of The Component – Open Source

Medina Castillo Sergio Arturo[1], Chaparro Anaya Martha Lucila[2]

Facultad de Ingeniería, Universidad Cooperativa de Colombia, Bucaramanga, Colombia,
sergio.medina@campusucc.edu.co

Facultad de Ingeniería, Universidad Cooperativa de Colombia, Bucaramanga, Colombia,
martha.chaparro@campusucc.edu.co

Recibido: 13/08/2012 • Aprobado: 12/11/2012

Resumen

En la actualidad, el desarrollo de software no parte de cero; por el contrario, ya se cuenta con un conjunto de herramientas suministradas por *frameworks*, lo que permite el desarrollo más rápido de aplicaciones, factor relevante e indispensable para apoyar procesos de mejoramiento continuo en busca de mayores niveles de competitividad en esta sociedad globalizada.

En todos los aspectos, la evolución de las aplicaciones web, ya sea en código libre o propietario, está desarrollándose velozmente, porque proporciona niveles de comunicación de servicios, interoperabilidad y acceso a clientes internos y externos que permiten soportar la administración de diferentes procesos de negocio.

Palabras Clave: *código libre, componentes, sistema, software*

Abstract

Nowadays software development not starting from scratch, however already has a set of tools provided by frameworks, which enables faster application development, relevant and indispensable factor for supporting continuous improvement processes seeking higher levels of competitiveness in this global society.

In all respects the development of Web applications, whether open source or proprietary, is developing rapidly, by providing service levels of communication, interoperability, access to internal and external customers that allows management support different business processes.

Keywords: *hardware, open source, software, system*

I. Introducción

Bajo el contexto internacional de competitividad, es necesario dar una respuesta ágil y oportuna a las necesidades del sector productivo y es responsabilidad de las instituciones de educación superior el asumir este reto; es por ello, que en la Universidad Cooperativa de Colombia se está trabajando un proyecto de desarrollo de componentes de software que permitan que el tiempo en suministrar una solución software sea mínimo, siguiendo parámetros de estandarización, para la reutilización del mismo.

Se desea utilizar las herramientas de código libre para desarrollar una serie de componentes normalizados y estandarizados que se conviertan en herramientas de apoyo para la producción de software de manera rápida y eficiente. Su desarrollo se basa en paradigmas y tecnologías orientados a objetos y estándares de desarrollo propios de la Universidad Cooperativa de Colombia Fig. 1.

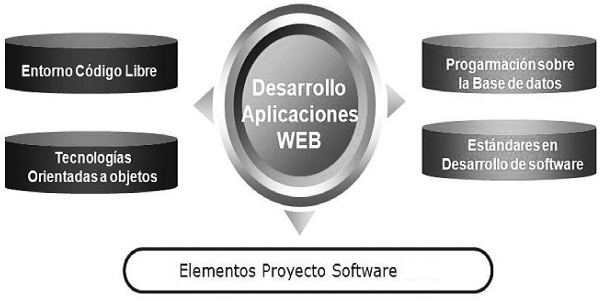

Fig. 1 *Elementos desarrollo de Software*

Uno de los propósitos de la estandarización del proceso de desarrollo de software es permitir la generación de aplicativos que sean fáciles de mantener en el tiempo, con vidas útiles amplias, generando un proceso eficiente de escalabilidad e interoperabilidad con otros aplicativos, lo cual redunda en grandes beneficios en tiempo y eficiencia en la compilación y producción de información válida para la toma de decisiones.

Otro propósito, es unificar metodologías, modelos y herramientas en la construcción de software Fig. 2.

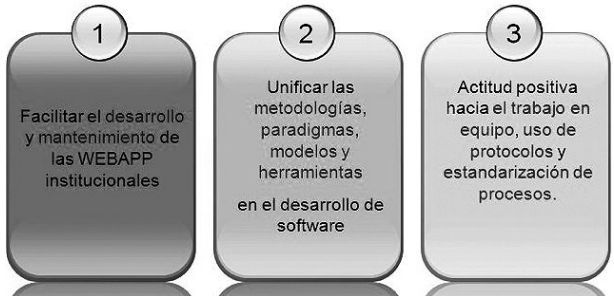

Fig. 2 *Propósitos Estándar desarrollo software*

En este proyecto se tomaron, como caso de estudio para el desarrollo de software, algunos procesos académicos y administrativos de la Universidad Cooperativa de Colombia, sede Bucaramanga, los cuales sirvieron de soporte para la generación del estándar de desarrollo y de los componentes que hoy se presentan y utilizan como herramientas para el proceso rápido de aplicativos web, bajo entorno de código libre; se utilizó como herramienta de programación el lenguaje PHP y como sistema administrador de base de datos (DBMS), *PostgreSql*, involucrando en ellos la programación orientada a objetos y la programación sobre la base de datos a través de funciones, procedimientos y disparadores (*Trigger*).

II. Marco de Referencia

En la Fig. 3 se presentan las bases conceptuales necesarias para el desarrollo de software: la Teoría General de Sistemas, que involucra los conceptos de sistema y sistema de información; la Ingeniería web, sus paradigmas, modelos, técnicas y herramientas; el estándar GNU de código abierto; teorías de diseño gráfico aplicadas en la realización de páginas web y el modelo relacional de base de datos, en sus aspectos de organización y manejo de la información, que comprenden el concepto de programación en la base de datos y lenguajes de programación orientados a entornos web, e involucran el paradigma de la programación orientada a objetos.

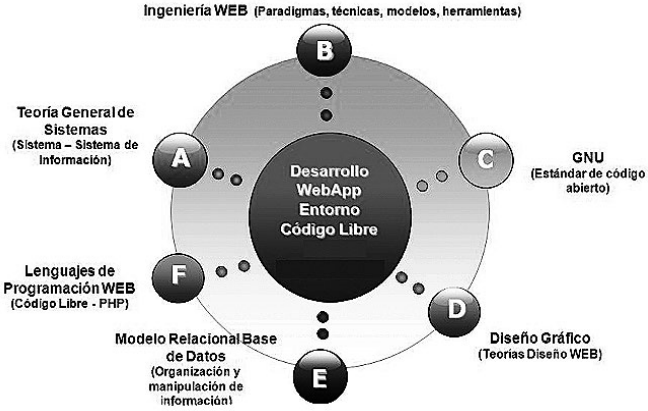

Fig.3 *Bases Conceptuales*

A continuación, una breve presentación de estas bases conceptuales.

A. Teoría general de sistemas

La teoría de sistemas (TS) es una rama específica de la teoría general de sistemas (TGS) y constituye el enfoque sistémico.

La teoría no soluciona problemas, pero produce formulaciones conceptuales que pueden crear condiciones de aplicación en la realidad. Desde la teoría general de sistemas, las propiedades de los sistemas no pueden describirse significativamente en términos de sus elementos separados, sino desde el todo; es decir, la compresión de los sistemas sólo ocurre cuando se estudian globalmente, involucrando todas las interdependencias de sus partes.

La TGS se fundamenta en tres premisas básicas:

- Los sistemas existen dentro de sistemas.
- Los sistemas son abiertos.
- Las funciones de un sistema dependen de su estructura.

Dentro de la teoría general de sistemas, se contempla el sistema como un conjunto o disposición de elementos que están organizados para cumplir una función determinada. El concepto de sistema hace referencia a una unidad, a un todo integrado, organizado a un conjunto cuyas propiedades y características son producto de las relaciones y conexiones entre sus partes y el todo con el entorno en el cual se encuentra. “Comprender las cosas sistémicamente significa literalmente colocarlas en un contexto, establecer la naturaleza de sus relaciones” [1].

Los sistemas tienen fronteras, que los diferencian del ambiente. Esas fronteras pueden ser físicas o conceptuales. Si hay algún intercambio entre el sistema y el ambiente a través de estas, el sistema es abierto; de lo contrario, es cerrado. El ambiente es el medio externo que envuelve física o conceptualmente a un sistema; es decir, el sistema interactúa con el ambiente. Todo lo que pasa del ambiente al sistema se conoce como entradas y lo que el sistema envía al ambiente, como salidas Fig. 4.

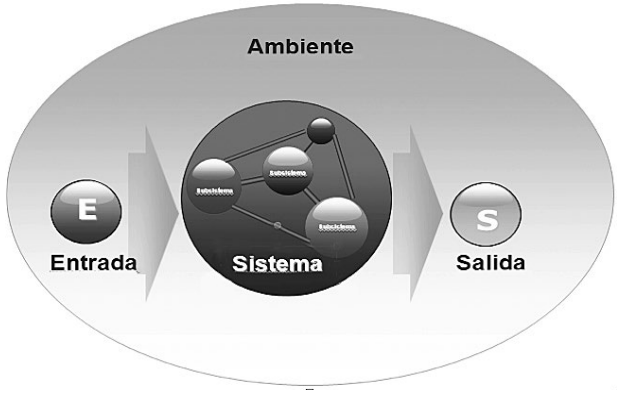

Fig. 4 *Entorno del Sistema*

Los sistemas de información son uno de los muchos tipos de sistemas que se definen como un conjunto de elementos interrelacionados, con el propósito de prestar atención a las necesidades de información de una organización, para elevar el nivel de conocimientos que permitan apoyar la toma de decisiones y el mejoramiento continuo.

Para el desarrollo de los sistemas de información existen una serie de modelos o paradigmas propuestos por la ingeniería del software, que han venido evolucionando: desde los lineales secuenciales, hasta los interactivos, encontrándose en esta clasificación nuevas prácticas para el desarrollo del software.

B. Ingeniería web

En el proceso de ingeniería web-IWEB, generalmente se manejan modelos de proceso de desarrollo de software de tipo incremental, lo que permite elaborar rápidamente versiones de *WebApp*.

Para el desarrollo de las *WebApp* que suelen ser controladas por el contenido, por el aspecto estético y el tiempo de respuesta, se necesita un equipo multidisciplinario de personas que trabajen en actividades diferentes, pero complementarias y de manera paralela.

C. Marco de trabajo para la web.

Es evidente la necesidad de aplicar gestiones sólidas y principios de ingeniería en el desarrollo de las *WebApps*; para lograrlo, es vital construir un marco de referencia IWEB que trabaje en conjunto con un modelo de proceso eficaz, caracterizado por las actividades y las tareas de ingeniería Fig. 5.

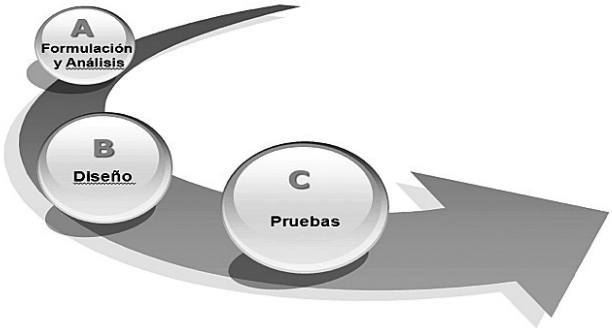

Fig. 5 *Modelo IWEB*

El paradigma o modelo de procesos de IWEB contempla las siguientes etapas:

1. Formulación y análisis de sistemas basados en Web.

La formulación y el análisis de sistemas y aplicaciones basadas en web representan una sucesión de actividades de ingeniería web. Se comienza con la identificación de las metas globales para la *WebApp* y se termina con el desarrollo de un modelo de análisis o especificación de los requisitos del sistema.

Formulación: se sugiere una serie de preguntas que deberían formularse y responderse al comienzo de la etapa de formulación, entre las cuales se encuentran:

¿Cuál es la motivación principal para la *WebApp*?

¿Por qué es necesaria la *WebApp*?

¿Quién va a utilizar la *WebApp*?

Las respuestas a estas preguntas deben ser lo más concisas posibles y, así mismo, tener en cuenta, las metas específicas para el sitio web que se ha de desarrollar. Las metas se pueden categorizar de la siguiente manera:

Metas informativas. Indican la intención de proporcionar el contenido específico para el usuario final.

Metas aplicables. Indican la habilidad de realizar algunas tareas dentro de la *WebApp*.

Al terminar el desarrollo de las metas y de los perfiles de los usuarios, la actividad de formulación se centra en la afirmación del ámbito para la *WebApp* con las etapas de planificación y análisis. En la primera, se realiza la distribución de las actividades en el tiempo y, en la segunda, se *crea* un modelo de análisis completo para la *WebApp*; se elabora el ámbito definido durante la etapa de formulación, contemplando cuatro tipos de análisis diferentes: *análisis de contenido, análisis de interacción, análisis funcional y análisis de configuración.*

En el análisis de contenido se realiza la identificación del espectro completo del contenido que se va a proporcionar; en el análisis de la interacción se hace la descripción detallada de la interacción del usuario y la *WebApp*; en el análisis funcional, los casos de uso creados como parte del análisis de interacción, definen las operaciones que se aplicarán en el contenido de la *WebApp* e implicarán otras funciones de procesamiento, y en el análisis de configuración se

efectúa una descripción detallada del entorno y de la infraestructura en donde reside la *WebApp* y del grado de utilización de la base de datos que se va a utilizar.

2. Diseño de aplicaciones basadas en web.

El diseño debe establecerse como un problema comercial inmediato, mientras se va definiendo una arquitectura de la aplicación que le permita la habilidad de evolucionar con mayor rapidez.

Con el objetivo de realizar un diseño eficaz basado en web, se deberá trabajar reutilizando cuatro elementos técnicos:

Principios y métodos de diseño. La modularidad eficaz, la elaboración paso a paso y cualquier otra característica de diseño de software conducirá a sistemas y aplicaciones basados en web, más fáciles de adaptar, mejorar, probar y utilizar.

Configuraciones de diseño. Son un enfoque genérico, problemas que se pueden adaptar a una variedad más amplia de los problemas específicos.

Plantillas. Son utilizadas para proporcionar un marco de trabajo esquemático de cualquier configuración de diseño o documento por utilizar dentro de una *WebApp*.

Diseño arquitectónico. Este tipo de diseño se centra en la definición de la estructura global hipermedia para la *WebApp*, y en aplicación de las configuraciones de diseño y plantillas constructivas para popularizar la estructura.

Patrones de diseño. Pueden aplicarse en nivel jerárquico, nivel de componentes y nivel de hipertexto. Estos últimos se centran en el diseño de las características de navegación que permiten al usuario moverse por el contenido de la *WebApp* fácilmente.

Diseño de navegación. Se definen las rutas de navegación que le permitan al usuario acceder al contenido y a los servicios de la *WebApp*. Para que se pueda llevar a cabo se debe, como primera medida, identificar la semántica de la navegación para diferentes usuarios del sitio y definir, en segunda instancia, la mecánica para lograr la navegación.

En una *WebApp* el manejo variará de acuerdo con los roles o perfiles de usuarios. Cada uno de estos roles se pueden asociar a otros niveles de acceso al contenido y de servicios diferentes.

Diseño de la interfaz. Es la primera impresión de una *WebApp*. Es una parte muy importante que debe cumplir con ciertas características mínimas para evitar que el usuario abandone el sitio web.

3. Pruebas de las aplicaciones basadas en web.

El enfoque de las pruebas para *WebApps* no varía con relación a los procesos encontrados en la Ingeniería de Software.

Se aplican pruebas a la unidad de componentes de proceso seleccionado y a las páginas web para verificar el cumplimiento de las diferentes funcionalidades y el manejo del contenido.

Se construye la arquitectura y se realizan pruebas de integración.

D. Proceso de desarrollo por componentes.

El modelo de proceso para la ingeniería del software basada en componentes se centra en la reutilización del software; este surge debido a la exigencia del medio dinámico que requiere respuestas de calidad en el menor tiempo posible, y a que se dispone de desarrollos de software de muy alta calidad que hace que se puedan usar nuevamente.

Una vez se establecen los requerimientos del software y se elige el diseño arquitectónico, se busca qué software puede ser utilizado, es decir se trabaja en “componer” el sistema, a partir de una de tres alternativas: compra de componentes reutilizables, selección a partir de las bibliotecas existentes o construcción de nuevos componentes, no sin antes pensar en la forma de eliminar el requisito o cambiarlo por otro, cuya funcionalidad se pueda realizar por un componente software.

En la ingeniería del dominio se crea un modelo de dominio de aplicación que se utiliza como base para el análisis de los requisitos y con ello se determina el conjunto de componentes de software que se pueden reutilizar. La Fig. 6 ilustra un modelo de proceso típico que acopla la ingeniería de software basado en componentes.

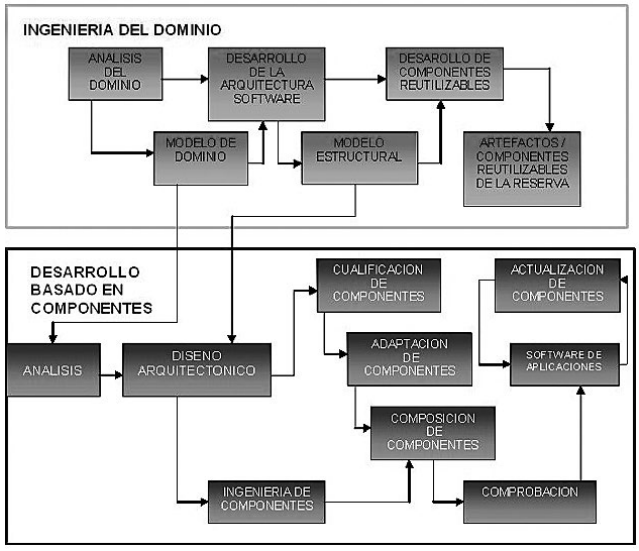

Fig. 6 *Modelo Desarrollo por componentes*

1. Ingenieria del dominio. El objetivo de la ingeniería del dominio es identificar, construir, catalogar y diseminar un conjunto de componentes de software que tienen aplicación en el software requerido dentro de un dominio de aplicación en particular. Se desea establecer mecanismos para el desarrollo de software siguiendo estándares que permitan compartir componentes, es decir, reutilizarlo.

2. Desarrollo por componentes. El desarrollo basado en componentes es una actividad de la ingeniería del software basada en componentes ISBC que tiene lugar en paralelo a la ingeniería del dominio, en donde el equipo del software refina el estilo arquitectónico adecuado para el modelo de análisis de la aplicación que se va a construir.

Una vez que se ha establecido la arquitectura, se debe popularizar mediante los componentes que están disponibles en bibliotecas de reutilización, y/o que se han diseñado para satisfacer las necesidades del cliente. De ahí que el flujo de una tarea de desarrollo basada en componentes, tenga dos caminos posibles. Cuando los componentes reutilizables están disponibles para una integración futura en la arquitectura, deben estar cualificados y adaptados. Cuando se requieren componentes nuevos, deben diseñarse. Los componentes resultantes, entonces, se «componen» (se integran) en una plantilla de arquitectura y se comprueban a conciencia.

3. Cualificación, adaptación y composición. La ingeniería del dominio proporciona la biblioteca de componentes reutilizables necesarios para la ingeniería del software, basada en componentes. Algunos de estos se desarrollan dentro de ella misma; otros se pueden extraer de las aplicaciones actuales y aun otros se pueden adquirir de terceras partes. La existencia de componentes reutilizables no garantiza que estos puedan integrarse fácilmente, o de forma eficaz, en la arquitectura elegida para una aplicación nueva. Esta es la razón por la que se aplica una sucesión de actividades de desarrollo basada en componentes cuando se ha propuesto que se utilice un componente.

La cualificación de componentes asegura que un componente candidato llevará a cabo la función necesaria, encajará además «adecuadamente» en el estilo arquitectónico especificado para el sistema y exhibirá las características de calidad (por ejemplo, rendimiento, fiabilidad, usabilidad) necesarias para la aplicación.

4. La adaptación de componentes. Lo ideal sería que la ingeniería del dominio creara una biblioteca de componentes que pudieran integrarse fácilmente en una arquitectura de aplicaciones. La implicación de una «integración fácil» consiste en que se hayan implementado los métodos consecuentes de la gestión de recursos para todos los componentes de la biblioteca; existan actividades comunes, tales como la gestión de datos para todos los componentes, y se hayan implementado interfaces dentro de la arquitectura y con el entorno externo de manera consecuente.

La tarea de composición de componentes requiere ensamblar componentes cualificados, adaptados y diseñados con el fin de popularizar la arquitectura establecida para una aplicación; para poderla realizar se debe establecer una infraestructura donde los componentes estén unidos a un sistema operacional. La infraestructura (normalmente una biblioteca de componentes especializados) proporcionará un modelo para la coordinación de componentes y servicios específicos que hacen posible coordinar unos componentes con otros, y llevar a cabo las tareas comunes.

5. Ingeniería de componentes. Como se ha señalado anteriormente, el proceso de ISBC fomenta la utilización de los componentes de software existentes. Sin embargo, hay ocasiones en que estos se deben diseñar. Es decir, los componentes de software nuevos deben desarrollarse e integrarse con los componentes ya existentes y los de desarrollo propio. Dado que estos componentes nuevos van a formar parte de la biblioteca de desarrollo propio de componentes reutilizables, deberían diseñarse para su reutilización. No hay nada de mágico en la creación de componentes de software que se pueden reutilizar. Conceptos de diseño tales como abstracción, ocultamiento, independencia funcional, refinamiento y programación estructurada, junto con los métodos orientados a objetos, pruebas y métodos de verificación de corrección, contribuyen a la creación de componentes de software que son reutilizables.

E. GNU-Estándar de código abierto.

GNU/Linux es, a simple vista, un Sistema Operativo. Como sistema operativo, es muy eficiente y tiene un excelente diseño. Es multitarea, multiusuario, multiplataforma y multiprocesador; en las plataformas *Intel core* en modo protegido, protege la memoria para que un programa no pueda hacer caer al resto del sistema; carga sólo las partes que se usan en el programa; comparte la memoria entre programas aumentando la velocidad y disminuyendo el uso de memoria; usa un sistema de memoria virtual por páginas; utiliza toda la memoria libre para cache; permite usar bibliotecas enlazadas, tanto estática como dinámicamente; se distribuye con código fuente; tiene un sistema de archivos avanzado, pero puede usar los de los otros sistemas, y soporta redes tanto en TCP/IP como en otros protocolos.

Los componentes de un sistema GNU/Linux no están en el dominio público, ni son *shareware*. Son lo que se llama “software libre”. Esto significa que el código fuente está disponible a todo el que lo quiera y siempre lo estará. El software libre puede ser vendido o regalado, a discreción de todo aquel que posea una copia, aunque a todo aquel que lo distribuye se le obliga a hacerlo con el código fuente. Todo esto está reglamentado por la Licencia Pública General GNU (GPL). Esta licencia se encarga de que GNU/Linux permanezca siempre libre.

F. Diseño gráfico – Teoría aplicada a páginas web

Se puede definir el diseño gráfico como el proceso de programar, proyectar, coordinar, seleccionar y organizar una serie de elementos para producir objetos visuales destinados a comunicar mensajes específicos a grupos determinados.

El diseño gráfico busca transmitir las ideas esenciales del mensaje de forma clara y directa, usando para ello diferentes elementos gráficos que den forma al mensaje y lo hagan fácilmente entendible por los destinatarios del mismo. El diseño gráfico no significa crear un dibujo, una imagen, una ilustración, una fotografía. Es algo más que la suma de todos esos elementos; no obstante, para poder conseguir comunicar visualmente un mensaje de forma efectiva, el diseñador debe conocer a fondo los diferentes recursos gráficos a su disposición y tener la imaginación, la experiencia, el buen gusto y el sentido común necesarios para combinarlos de forma adecuada.

El principal componente de toda composición gráfica es, pues, el mensaje que se ha de interpretar, la información que se desea hacer llegar al destinatario a través del grafismo. El diseño

gráfico de una página web es tan solo una parte del diseño de la misma, ya que, además, hay que considerar un conjunto más o menos extenso de condicionantes que van a limitar la libre creatividad del diseñador.

En primer lugar, las páginas web se deben descargar de un servidor web por medio de Internet, por lo que el ancho de banda de las conexiones de los usuarios va a ser un factor clave en la velocidad de visualización. Los elementos gráficos son archivos cuyo peso depende del formato en que se guarden; estudios realizados demuestran que el tiempo máximo de espera de un usuario en la descarga de una página suele ser de unos 10 segundos, pasados los cuales, este prefiere abandonar el sitio y buscar otro más rápido. Por lo tanto, el número de elementos gráficos que se pueden introducir en una página web queda bastante limitado, teniendo que buscar alternativas mediante el uso imaginativo de fuentes y colores.

Otro aspecto por tener en cuenta es la visualización de las páginas web en unas aplicaciones específicas: los navegadores web, que imponen grandes limitaciones al diseño de las mismas. La ventana de un navegador es eminentemente rectangular, con unas medidas concretas (dadas por la resolución empleada por el usuario en su monitor) y con unas capacidades de interpretación de colores que varían mucho según el ordenador usado, el sistema operativo, el monitor y la tarjeta gráfica. Estos factores imponen fuertes limitaciones al diseñador web, quien siempre debe buscar que sus páginas puedan ser visualizadas correctamente por el mayor número de usuarios. Para intentar solventar estas diferencias, debe trabajar a la vez con varios navegadores, diseñando sus páginas de tal forma que la interpretación de ellas sea similar en todos, lo que impone nuevas limitaciones al diseño.

Por otra parte, una página web no es un diseño gráfico estático; esta contiene diferentes elementos que tienen la capacidad de interaccionar con el usuario, entre otros, menús de navegación, enlaces, formularios. Además, no existe de forma aislada, sino que hace parte de un conjunto de páginas inter-relacionadas entre sí (el sitio web), que deben presentarse al usuario con el mismo "estilo", aunque su funcionalidad sea muy diferente. A esto hay que sumarle que las páginas diseñadas deben construirse en un lenguaje específico, el HTML, el cual, por sí mismo, es muy limitado, lo que hace que el diseñador web tenga que estar siempre pensando si la interfaz que está diseñando gráficamente podrá construirse posteriormente.

Por último, una página web suele ocultar, en la mayoría de los casos, una serie de procesos complejos que se ejecutan sin que el usuario sea consciente de ellos (ejecución de códigos de lenguajes de programación tanto en cliente como en servidor, acceso a bases de datos en servidores remotos, etc.), procesos que añaden tiempo a la presentación de las páginas y que muchas veces suelen afectar, de forma importante, el diseño de estas.

G. Modelo relacional de bases de datos

En el modelo relacional, los datos son percibidos por el usuario como una colección de tablas. La representación de datos mediante tablas hace más comprensible y entendible su manejo; en términos relacionales las tablas se componen de filas y columnas.

El modelo relacional define reglas en tres áreas principales:

1. Estructura de datos. Estas reglas definen cómo se almacenan las columnas en las tablas, el tamaño y tipo de dato asociado con cada columna y cuáles columnas serán usadas como llaves, índices, etc.

2. Integridad de los datos. Las reglas que aseguran que los datos están completos y son consistentes en toda la base de datos.

3. Manipulación de los datos. Reglas que definen las maneras de accesar los datos. Con base en

la teoría matemática de los conjuntos, se han definido tres operaciones relacionales básicas para la recuperación de datos:

Selección (*Select*)

Proyección (*Project*)

Unión (*Join*)

Las reglas de estas operaciones conforman las bases del lenguaje relacional SQL (*Structured Query Language*).

1. Estructura lógica. Una Base de Datos Relacional - RDB se conforma de tablas, índices y diccionario. La estructura lógica de una RDB se implementa mediante la definición de índices y tablas. Una tabla se define por los datos que contiene, y las inter-relaciones de los datos, por el contenido mismo de los datos.

Una tabla está compuesta de filas y columnas. Una fila contiene campos con información que está relacionada y una columna, campos que representan la misma información de diferentes elementos lógicos; es decir, una fila contiene los diferentes campos y la columna, las diversas ocurrencias de un campo.

2. Índices. Se definen opcionalmente para cada tabla, buscando que el acceso a los datos de tabla sea más eficiente; por defecto, una tabla se accede secuencialmente. Sin embargo, cuando las tablas contienen muchos registros y se requiere accesarlos aleatoriamente, se pueden definir índices para mejorar el desempeño. Cuando se utilizan los índices, se puede garantizar el valor único de un campo clave o llave.

3. Estructura física. El modelo físico de la RDB es menos complejo y por lo general es transparente para el usuario. Es responsabilidad del administrador de la Base de Datos - DB controlar el uso de los dispositivos de almacenamiento, por consideraciones de desempeño de la DB. El DBMS utiliza bloques físicos, llamados páginas, para el almacenamiento de los datos.

H. Lenguajes de programación web

Un lenguaje de programación se puede definir como el conjunto de símbolos y reglas semánticas y sintácticas que especifican su estructura y el significado de sus expresiones; es utilizado para controlar el comportamiento lógico y físico de una máquina. Mediante un lenguaje de programación se le indica al computador las acciones que debe realizar, se le suministran los datos con que debe trabajar, se le indica la forma de procesar, transmitir y guardar los datos y la manera de generar las entregas de información, bajo una variada gama de circunstancias. Todo programa escrito en un determinado lenguaje de alto nivel, debe pasarse a un lenguaje de máquina, para que pueda ser ejecutado por el procesador y esto se puede lograr de dos maneras: por una parte, mediante un programa que va adaptando las instrucciones conforme son encontradas ,(una instrucción a la vez); a este proceso se le llama interpretar y a los programas que lo hacen se les conoce como intérpretes; y, de otro lado, traduciendo todo el programa fuente a su equivalente escrito en lenguaje de máquina; este proceso se denomina compilar y el programa traductor, compilador.

Existen varias clases de lenguajes de programación libre, tales como, Perl, Pitón, Tcl, etc.; sin embargo, para el desarrollo de *WebApp* uno de los más populares es PHP.

PHP es un lenguaje interpretado de propósito general, ampliamente usado y que está diseñado especialmente para desarrollo web. Generalmente, se ejecuta en un servidor web, tomando el código en PHP como su entrada y creando páginas web como salida. Puede ser alojado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas.

III. Estándar de Desarrollo de Software Web Bajo Entorno de Código Libre de la Universidad Cooperativa de Colombia – Bucaramnaga

Para facilitar el desarrollo de software en la Universidad Cooperativa de Colombia, se definieron una serie de estándares que permiten unificar criterios en lo relacionado con nomenclatura, estructura en los espacios de almacenamiento, documentación del software, interfaz gráfica, estándares en diseño, modelación y seguimiento del desarrollo de software, paradigmas de programación, políticas de seguridad, salidas del sistema y herramientas para el desarrollo. A continuación se presentan los estándares en los aspectos mencionados.

A. Estándar 1: nomenclatura

Se deben especificar los nombres de los archivos, campos, variables y demás elementos propios de la programación, con nombres descriptivos, claros, siempre en minúscula y que representen el valor almacenado.

1. Nombres de archivos: nombre descriptivo; si es compuesto utilizar la barra de piso (_), para separarlos. No utilizar caracteres especiales; en lo posible letras, siempre en minúscula y su longitud máxima será de 20 caracteres (prever que la longitud del nombre y la ruta no sea superior a 100 caracteres). Ejemplos: contrato.pdf, foto_91258632.jpg.

Para los nombres de las tablas y campos de la base de datos, se debe tener en cuenta el siguiente estándar:

2. Nombre de la tabla: nombre descriptivo; si es compuesto utilizar la barra de piso (_), para su separación. No utilizar caracteres especiales, en lo posible letras, siempre en minúscula. Ejemplos:

Administrador, detalle_compra, Notas, estudio_formales

3. Nombre campos: el nombre de los campos debe ser de ocho caracteres, de los cuales, los tres primeros identifican la tabla y los otros cinco se utilizan para el campo; siempre en minúscula. Ejemplos:

grucodes: Código del estudiante en la tabla de grupo.

doccedul: Cédula en la tabla del docente

Las variables de memoria, utilizadas en los programas, deben tener nombres que identifiquen su función con suficiente claridad, siempre en minúscula. Ejemplos: tipo_asign, sw_agregar, etc.

4. Nombres y longitudes de campos genéricos y llaves: este estándar va dirigido a aquellos campos genéricos que son utilizados por varias tablas de la Base de Datos de las diferentes aplicaciones y a la estructura de datos de campos, llaves primarias y foráneas.

Con respecto a los campos que utilizan varias aplicaciones y los campos llaves, se referencia el siguiente estándar (Tabla 1).

TABLA I
Campos Genéricos de las Webapp

Descripción	Nombre Campo	Tipo	Longitud
Nombres de persona	Nombre (De acuerdo a nomenclatura ya establecida)	Texto o tipo carácter	25
Apellidos de persona	Apellido(De acuerdo a nomenclatura ya establecida)	Texto o tipo carácter	25
Campos de descripción tales como ciudad, programa, barrio, departamento, sección, nombre empresa, dirección etc.	Ciudad (De acuerdo a nomenclatura ya establecida)	Texto o tipo carácter	40
Campos codificados como Llaves primarias o foráneas tales como: códigos, nits, cedula, identificadores etc.	De acuerdo a nomenclatura ya establecida	Generalmente se manejan de tipo texto o carácter	15
Campos de descripción para el de rutas de archivo, emails etc.	Email (De acuerdo a nomenclatura ya establecida)	Texto o carácter	100
Campos de Observaciones, apuntes, conclusiones, recomendaciones etc (Nota: si la aplicación necesita mas espacio se manejan varios campos de observación)	Observación (De acuerdo a nomenclatura ya establecida)	Texto, carácter o memo	256
Los campos de dominio finito, tales como genero (femenino, masculino), estado (activo, inactivo) Lo relativo a letras siempre será minúscula	estado	Texto o carácter	1

B. Estándar 2: estructura de los espacios de almacenamiento

Para el almacenamiento físico de la información propia de un sistema, bajo entorno web, se presenta la siguiente estructura de directorios y subdirectorios Fig. 7.

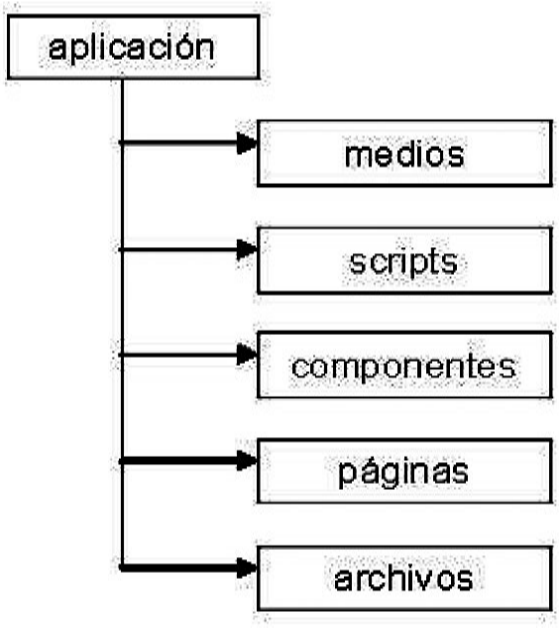

Fig. 7 *Estructura de almacenamiento*

1. Medios: almacenamiento de los elementos multimedia de la aplicación (imágenes, textos, sonidos, animaciones y vídeo).

2. Scripts: almacenamiento del código PHP propio de la aplicación.

3. Componentes: almacenamiento de las clases y rutinas generales aplicadas a todas las aplicaciones web desarrolladas.

4. Páginas: almacenamiento de las páginas web de la aplicación.

5. Archivos: documentos importantes de la aplicación o de subir y bajar en web.

C. Estándar 3: documentación del software

Programas. Cada programa que se desarrolle de una aplicación, debe iniciar con comentarios en los cuales este se documente, especificando el Nombre de la aplicación y programa, fecha de elaboración, fecha de la última actualización, su objetivo, autor y sus entradas y salidas. Ejemplo en PHP Fig. 8.

```
<?php

// Nombre Aplicación:        Notas
// Nombre Módulo:            Estudiante
// Nombre Programa:          Visualiza_notasest.php
// Objetivo:                 Presentar la información de las
notas en cada corte
// Entradas:                 Tabla programa académico,
estudiante, materias y grupos
// Salidas:                  Código y nombre del programa
académico, código y nombre del estudiante, asignatura, nota de
corte, nota de investigación
// Autor:                    María Medina
// Fecha elaboración:        Febrero 1 de 2006
// Fecha modificación:       Marzo 16 de 2006
// Programador Modifica:     Juan Pérez
// Observaciones:            Utiliza los componentes Conexión
y SQL para realizar la conexión y la sentencia SQL.

?>
```

Fig. 8 *Ejemplo de documentación en PHP*

1. En medio impreso. Para cada programa se debe tener un documento con la definición y las características propias del mismo, junto con las actualizaciones realizadas a lo largo de su existencia y además la documentación de los componentes. El formato de documentación del programa, debe tener los siguientes elementos:

- Identificación del programa, que comprende:
  - Nombre de la aplicación.
  - Nombre del módulo.
  - Nombre del programa.
  - Autor (programador creador).
  - Fecha de elaboración.
- Objetivo(s) (uno general o varios específicos).
- Entradas, bases de datos, tablas, variables pedidas por el navegador, parámetros, etc.
- Salidas, actualizaciones sobre tablas de bases de datos, información en el navegador, etc.
- Validaciones, cuando sean programas de captura de información, escribir el tipo de validación efectuada, datos numéricos, valores nulos, dominios finitos, etc.
- Acciones, operaciones SQL realizadas sobre la base de datos.
- Componentes utilizados.

  Observaciones.

Debe tener la firma del autor y ser verificado por el director del proyecto.

El formato de actualización del programa, debe tener los siguientes elementos:

- Identificación del programa, que comprende:
  - Nombre de la aplicación.
  - Nombre del módulo.
  - Nombre del programa.
  - Programador (realiza la modificación).
  - Fecha de modificación.
- Descripción de la modificación.
- Entradas, bases de datos, tablas, variables pedidas por el navegador, parámetros, etc.
- Salidas, actualizaciones sobre tablas de bases de datos, información en el navegador, etc.
- Validaciones: cuando sean programas de captura de información, escribir el tipo de validación efectuada, datos numéricos, valores nulos, dominios finitos, etc.
- Acciones: operaciones SQL realizadas sobre la base de datos.
- Componentes utilizados.
- Observaciones.

Debe tener la firma del programador que realiza la modificación y ser verificado por el director del proyecto.

El formato de documentación de los componentes, debe tener los siguientes elementos:

- Identificación del componente, que comprende:
  - Nombre del componente.
  - Tipo: (clase, función o procedimiento).
  - Lenguaje (Php, javascript, etc.).
  - Autor (programador creador).
  - Fecha de elaboración.
- Objetivo(s) (uno general o varios específicos).
- Entradas, parámetros de entrada.
- Salidas, parámetros de salida.
- Breve descripción del componente.
- Componentes utilizados, si utiliza otros componentes.
- Observaciones.

Debe tener la firma del autor y ser verificado por el director del proyecto.

2. Documentación programa. La documentación del programa debe presentar información que permita identificar: aplicación, módulo, autor, fecha, objetivos, entradas, salidas, validaciones, acciones y componentes que maneja Tabla 2.

TABLA II
DOCUMENTACIÓN PROGRAMA

Aplicación:
Módulo:
Programa:
Autor:
Fecha:
OBJETIVO (S)
ENTRADAS
SALIDAS
VALIDACIONES
ACCIONES
COMPONENTES
OBSERVACIONES
Elaborado por — Revisado por

D. Estándar 4: interfaz gráfica

Estructura de la interfaz. Hace referencia a la ubicación y contenidos de información en la página que el usuario observa Fig. 9.

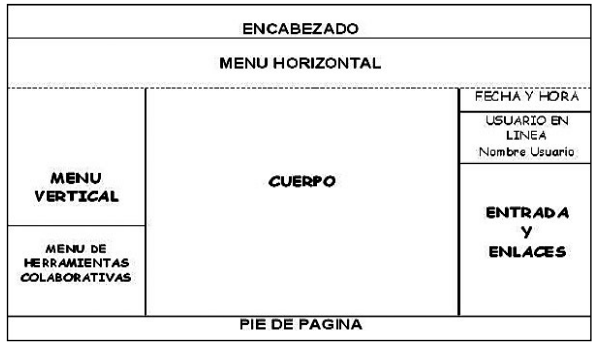

Fig. 9 *Interfaz Gráfica*

1. Colores. Los colores institucionales que se manejan son:

- Gama de los verdes.
- Color blanco.
- Colores propios del escudo de la institución.

2. Encabezado. Debe contemplar el concepto y las políticas institucionales, siguiendo los lineamientos de los diseños de la página de la www.ucc.edu.co

3. Menú horizontal. Este menú contiene las opciones típicas de página tales como: inicio, contáctenos, mapa del sitio, términos de uso.

4. Menú vertical. **C**ontendrá las opciones propias de la aplicación que se esté desarrollando; no se manejan botones aislados, sino una plantillas de botones.

5. Menú de herramientas colaborativas. Presentará opciones de foro, chat, agenda, dependiendo de la aplicación.

6. Pie de página. Presentará los derechos de autor, condiciones y/o términos de uso.

7. Fecha y hora. Mostrará la fecha y hora del sistema

8. Usuarios en línea. Mostrará un número que corresponde a la cantidad de usuarios en línea conectados a la aplicación.

9. Personalización del usuario. Mostrará el nombre del usuario que se encuentra utilizando la aplicación.

10. Entradas y enlaces. Permiten el ingreso a la aplicación mediante una identificación y una contraseña. Además, muestran unos hipervínculos gráficos a opciones adicionales que ofrece la página, respetando los diseños de la página de la www.ucc.edu.co

11. Cuerpo. Obedece al diseño que se tenga en cada aplicación.

12. Texto (fuente, tamaño y colores de texto). El texto debe ser claro, sencillo, preferiblemente en minúsculas, evitar el uso de mayúscula fija. La fuente se recomienda que sea legible (letra imprenta) y que admita caracteres tales como tildes y ñ /Ñ. (se recomienda el uso de la fuente Tahoma). El tamaño del texto debe variar cuando se trata de títulos, de contenidos, de hipervínculos (evitar la monotonía sin caer en el exceso). El color de la fuente debe jugar con las gamas de la interfaz; la fuente tendrá un color oscuro, si el fondo es claro, y lo contrario.

13. Navegación (botones, hipervínculos, ubicación). Se usarán plantillas de botones para el manejo del menú vertical; los hipervínculos estarán tanto en texto como en ícono. Cuando una temática implique el manejo de varias páginas, deben utilizarse botones de avanzar y retroceder, siempre los mismos y en el mismo sitio de la interfaz.

14. Medios (peso de imágenes, colores). Evita el uso de imágenes pesadas (máximo de 4 KB); los íconos deben identificar la opción que están representando; deben diseñarse teniendo en cuenta los colores de la interfaz, y ser originales. Identificar dentro de la interfaz las imágenes que son hipervínculos y aquellas que hacen parte de la presentación de la página. Por ejemplo, mediante el uso de efectos de animación en los hipervínculos.

E. Estándar 5: paradigma de programación-(Programación Orientada a Objetos)

La Programación Orientada a Objetos (POO) es el estándar de programación para las aplicaciones web que se desarrollen, separando el software reutilizable como componente y el software propio o exclusivo de la aplicación, con base en la estructura de las carpetas antes mencionadas.

F. Estándar 6: estándares de modelación

El estándar para el modelamiento de los procesos se realizará a través de la orientación a objetos, utilizando los diagramas de UML que sean pertinentes con el software por desarrollar.

Para el modelamiento de datos se seguirá el esquema estructurado, mediante el uso de la metodología de la normalización definida en el modelo relacional.

G. Estándar 7: estándares en el seguimiento al desarrollo Web

Independientemente del modelo de desarrollo que se utilice, se deben evidenciar cada una de las etapas propias del paradigma. En ellas como mínimo deben generase documentos propios de la etapa y actas que den soporte de la presentación y aceptación por parte de los involucrados en la aplicación.

A continuación se presentan los estándares en el seguimiento al desarrollo, enunciando actividades básicas que requieren evidencias.

1. Definición de requerimientos. Una de las actividades básicas y fundamentales dentro del desarrollo del software es el estudio de su comportamiento, el cual se realiza mediante el modelamiento, utilizando los diagramas de caso de uso para representar los requerimientos funcionales del software.

Se tiene como estándar en la definición de requerimientos la presentación de los siguientes documentos:

- Diagramas de casos de uso.
- Listas de requerimientos de entradas, de procesamiento y de salida.
- Acta de presentación y aprobación firmada por los comprometidos en el desarrollo (grupo de desarrollo) y el usuario final (quién o quiénes reciben la aplicación).

2. Diseño. Dentro del seguimiento a la etapa del diseño se tendrán en cuenta la evaluación de aspectos relacionados con la interfaz, modelo de datos, procesos, seguridad y respaldo. A continuación se hará referencia a los elementos de seguimiento en los aspectos anteriormente mencionados.

3. Diseño de la interfaz. Se tiene como estándar en el diseño de la interfaz la presentación de los siguientes documentos:

- Prototipo del diseño de la interfaz.
- Mapa de navegación completo de toda la aplicación.
- Acta de presentación y aprobación firmada por los comprometidos en el desarrollo, (grupo de desarrollo) y usuario final (quien o quienes reciben la aplicación).

4. Base de datos. Como estándar en la definición del modelo de datos se tiene la presentación de los siguientes documentos:

- Modelo conceptual de la base de datos (modelo entidad relación).
- Descripción de cada una de las tablas de la base de datos, donde se especifique: nombre del campo y su estructura de dato (tipo de dato y longitud), descripción del campo, definición de llaves primarias y foráneas, validación de integridad para campos (dominios finitos, validaciones propias de la aplicación).
- Acta de presentación y aprobación firmada por los comprometidos en el desarrollo, (grupo de desarrollo y usuario final (quién o quiénes reciben la aplicación).

5. De procesos. Se tiene como estándar en la definición de procesos la presentación de los siguientes documentos:

- Diagrama de clases.
- Lista de procesos.
- Acta de presentación y aprobación firmada por los comprometidos en el desarrollo (grupo de desarrollo y usuario final (quien o quienes reciben la aplicación)
- Diagramas de UML que se consideren necesarios.

6. Seguridad y respaldo. Se tiene como estándar en la definición de políticas de seguridad y procedimientos de mantenimiento a la base de datos, la presentación de los siguientes documentos:

- Definición de las políticas de seguridad propias del software.
- Definición del procedimiento y periodicidad para realizar copias de respaldo y opción de restauración de la base de datos.
- Definición de estrategias de auditoría para el software.

- Acta de presentación y aprobación firmada por los comprometidos en el desarrollo (grupo de desarrollo) y usuario final (quien o quienes reciben la aplicación).

H. Estándar 8: estándares de salidas del sistema

Se tiene como estándar en la definición de las salidas del sistema la presentación de las siguientes recomendaciones:

- Todo informe debe tener encabezado y pie de página. En el encabezado siempre irá la siguiente información:

1. Nombre de la institución, y NIT parametrizados (se toman desde una tabla), y alineados a la izquierda.
2. Fecha y hora de impresión del informe, alineados a la derecha.
3. Titulo del informe en centro de la hoja, Si el informe maneja una periodicidad, este debe estar incluido en este título.
4. En el pie de página se debe presentar la siguiente información: usuario que genera el informe, alineado a la izquierda, número de página alineado a la derecha, mostrar página actual y el número total de páginas del documento (ejemplo: Página 2 de 5) alineado a la derecha.
5. Con respecto al tipo de fuente se recomienda tahoma, de tamaño 12.
6. Se recomienda respetar márgenes para permitir una buena presentación y facilidad de archivo del documento.
7. La generación de informes y consultas debe ser muy versátil; es decir, diseñar una serie de variables, con opciones de ordenamiento y agrupación, y número de copias, en una misma interfaz gráfica, que permitan de manera dinámica filtrar la información de la base de datos, con el fin de evitar una lista muy amplia y poco eficiente de informes y consultas.
8. El generador de informe debe permitir el almacenamiento en formato digital del mismo, con opción de enviar a correo electrónico.
9. Se debe realizar un breve comentario acerca del informe o la consulta, que visualice su utilidad, a quién va dirigido, el contenido y forma de presentación del mismo.
10. Se recomienda el manejo de informes estadísticos utilizando gráficos de barras, áreas, líneas y tortas.
11. Las consultas que se generan inicialmente en la pantalla, deben tener la opción de imprimirse.
12. Se recomienda que los informes que sean de carácter formal como recibos de pago de matrícula, por ejemplo, se trabajen bajo formato pre impreso.
13. Todo informe y consulta debe ser documentado de la siguiente manera:
    - Identificación del informe.
    - Periodicidad.
    - Número de copias.
    - Usuario a los que va dirigido.
    - Usuario generador del informe.
    - Descripción del informe.
    - Tipo de informe (pre impreso o no). En caso de ser en formato pre impreso debe llevar el número del documento pre impreso.
    - Dependencia encargada de archivar el informe.
    - Lista de direcciones de correo a los que se les debe enviar el informe.

I. Estándar 9: políticas de seguridad.

Para las aplicaciones web desarrolladas para la Universidad Cooperativa de Colombia, se definen unos aspectos de seguridad mínimos que garanticen la integridad, consistencia y confiablidad de la información a través de la implementación de estrategias de seguridad, tales como, manejo de sesiones y permisos por perfiles de usuario.

1. Manejo de sesiones. La sesión es el conjunto de páginas por las cuales navega el usuario dentro de una aplicación web (*WebApp*). Esta es única por cada navegante y su seguridad se convierte en elemento fundamental para garantizar su funcionalidad y operabilidad.

En el manejo de la sesión, se definen dos elementos de seguridad, concernientes con la autentificación del usuario, que busca la validación de la existencia de un usuario válido en cada página de una sesión, y el tiempo en página, que busca la definición de tiempos máximos en cada una de las páginas de la sesión, evitando dejar abierto por mucho tiempo la *WebApp*, lo que ocurre normalmente por descuido u olvido de los usuarios finales.

2. Manejo de perfiles. En una aplicación web, bajo el esquema de la orientación a objetos, se definen actores o perfiles que utilizan de manera particular la *WebApp*; es decir, cada perfil maneja unos permisos sobre determinados procesos de la *WebApp*, lo que garantiza confiabilidad en la información, ya que su uso está definido y controlado para cada uno.

J. Estándar 10: herramientas

Las herramientas seleccionadas se encuentran en el ámbito del software libre Tabla 3.

TABLA III
HERRAMIENTAS

Herramienta	Descripción	Tipo Licencia
PostgreSQL	Administrador de Bases de Datos (DBMS)	Libre
PHP 5	Lenguaje de programación para que facilita el manejo de la programación orientada a objetos	Libre
Wampserver	Emulador de servidor WEB, para el desarrollo local de la WEBAPP	Libre
Suite ADOBE MACROMEDIA	Herramientas para la edición y diseño de las páginas WEB y programas de la WEBAPP	Licencia

K. Estándar 11: construcción WebApp

Con respecto al estándar de construcción que se ha de emplear, se utilizan dos tecnologías importantes, como lo son, la programación orientada a objetos y la programación sobre la base de datos mediante el uso de funciones, procedimientos y disparadores (*trigger*), que redundan en eficiencia y seguridad para el desarrollo en ambiente web y facilitan el mantenimiento futuro de la *WebApp*.

Para la programación de cada uno de los procesos, la aplicación estructura tres grandes aspectos estándares de aplicación de las tecnologías mencionadas. El primero de ellos, es la función, procedimiento o disparador (*trigger*) de la base de datos, que contiene las operaciones en términos de SQL y PL-SQL, los cuales permiten la realización del proceso específico, generando mayor seguridad y eficiencia al centralizar el almacenamiento y procesamiento sobre la base de datos.

El segundo aspecto hace referencia a las clases involucradas en dicho proceso, a través de sus atributos y, sobre todo, de sus métodos. Es importante anotar que bajo el estándar definido, los métodos de las clases se encargarán de ejecutar la función o procedimiento correspondiente al proceso, en la base de datos.

Finalmente, el tercer aspecto hace referencia al *script* de código fuente que involucra la interfaz gráfica con el usuario, en el cual se implementarán todos estos aspectos de la Interfaz gráfica de los usuarios ¨GUI¨ del proceso y la incorporación de las clases y de los componentes que se utilizarán dentro del mismo. Se puede apreciar este esquema en la Fig. 10.

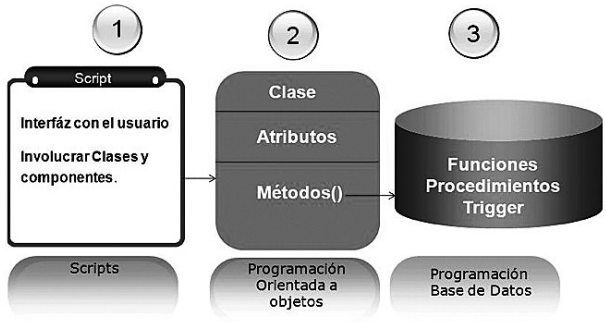

Fig. 10 *Estándar en construcción de WebApp*

K. Estándar 12: Metodología Aplicación WebApp

En la Universidad Cooperativa de Colombia UCC – Seccional Bucaramanga, se utiliza para la construcción de la programación *WebApp*, una metodología que involucra unos pasos coherentes, los cuales permiten realizar un proceso ordenado e incremental, usando de una manera más eficiente los diferentes componentes y estándares estipulados a nivel de desarrollo de software Fig. 11.

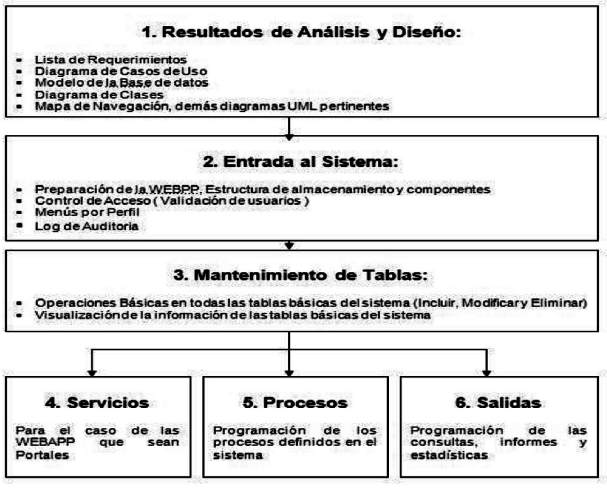

Fig. 11 *Metodología construcción WebApp*

Esta metodología comprende una serie de etapas, partiendo de un diseño completo, claro, orientado a objetos y diagramado mediante UML. Estos pasos son:

1. Claridad sobre el análisis y diseño realizado y diagramado con UML.

2. Entrada al sistema, que implica actividades de preparación de la *WebApp*, en cuanto a la definición de la estructura del sitio web, creación del esquema de la base de datos, creación de la plantilla para la interfaz y manejo de la aplicación, programación de las funciones en la base de datos para el control de acceso, registro de auditoría y demás propias de la entrada al software. Además, la programación del diagrama de clases y los *scripts* relacionados con la entrada.

3. Mantenimiento de tablas, proceso estándar pre programado en la plantilla, que permite el manejo de las operaciones básicas sobre una tabla, como lo es la inclusión, modificación y eliminación, junto con la posibilidad de visualización y búsqueda de información.

4. Desarrollo de las diferentes opciones diseñadas en la *WebApp*, clasificadas según los componentes que se utilizarán dentro de ellas, de la siguiente manera:

- Salidas (PDF, EXCEL, navegador).
- Procesos propios de la WebApp donde se utilicen los estándares definidos y tecnologías para manejo más eficiente de la información, entre otras, tecnología AJAX.
- Servicios, tales como procesos colaborativos: foros y chat, manejos de archivos sobre un servidor, galería de imágenes, etc.

IV. Componentes de Software Bajo Entorno de Código Libre

Los componentes son módulos prediseñados que ejecutan funciones específicas dentro de la construcción de software, tales como, conexiones con la base de datos, generación de salidas en diferentes formatos y generación de gráficos estadísticos que pueden ser utilizados en cualquier aplicación web, reduciendo significativamente los tiempos de programación.

Para el caso particular de la Universidad Cooperativa de Colombia, se utilizan componentes que cumplan con el estándar de desarrollo; es decir, el paradigma de la programación orientada a objetos y que permitan, si es pertinente, el uso de la programación en la base de datos. Algunos de los componentes utilizados fueron realizados por el grupo de desarrollo de la institución y otros, tomados de la comunidad de código libre en el ciberespacio y ajustados al estándar de desarrollo definido.

A. Componentes desarrollados en la UCC.

El grupo de desarrollo de software web en ambiente de código libre, ha creado los siguientes componentes:

1. Conexión con base de Datos (PHP – POSTGRESQL y PHP – MYSQL), que permite no solo la conexión de PHP con las bases de datos POSTGRESQL y MYSQL, sino la ejecución de cualquier sentencia SQL con las validaciones necesarias.

2. Validación de la sesión en lo referente a la autentificación del usuario y al tiempo máximo de inactividad en página y el manejo de los permisos a las diferentes opciones que ofrece una aplicación web

3. Manejo de combos genéricos que permitan la visualización y selección de datos con cualquier sentencia *Select* y además la interfaz con procesos de Tecnología AJAX.

4. Proceso de mantenimiento de tablas, que permita las operaciones básicas sobre una tabla (incluir, modificar y eliminar), junto con las opciones de búsqueda por cualquier campo y la visualización de la información, paginando los datos.

B. Componentes adaptados de la Comunidad de Código Abierto

La comunidad de código libre en el ciberespacio ofrece una gama inmensa de componentes que realizan diversas tareas y, por lo tanto, facilitan la construcción de software. Sin embargo, muchos de estos componentes no se encuentran diseñados sobre tecnologías orientadas a objetos y, por ello, algunos fueron tomados de la comunidad de código abierto y adaptados al estándar de la Institución. Los componentes tomados fueron:

1. Componente para le generación de salidas en formato PDF, compuesto por la clase FDPF y los estilos fuentes necesarios para su utilización. Mediante este componente se construyó un prototipo de informe, con funciones genéricas de encabezado y pie de página y un esquema parametrizado que pueda ser reutilizado en cualquier otro informe que necesite la salida en este formato.

2. Componente para le generación de salidas en formato Excel, compuesto por la clase Excel y los estilos fuentes necesarios para su utilización. Mediante este componente se construyó un prototipo de informe, con un esquema parametrizable que pueda ser reutilizado en cualquier otro informe que necesite la salida en este formato.

3. Componente para la generación de gráficos estadísticos, *Libchart*, compuestos por clases y estilos que involucran los diferentes tipos, tales como barras, líneas, áreas, tortas, con los cuales se crearon programas prototipos que puedan ser reutilizados.

4. Componente para el manejo de archivos sobre un servidor, que permite subir y bajar archivos de un servidor web, ajustado en su totalidad al estándar de desarrollo de la institución, parametrizado para los tipos de archivos por manejar y con las validaciones necesarias para este tipo de proceso.

5. Prototipo para el manejo de galería de imágenes, con todo el estándar definido por la institución, las validaciones sobre formatos y tamaño de imágenes que permite ser reutilizado en cualquier aplicación web que requiera este tipo de actividad.

6. Componente para el manejo de foros, ajustado a los estándares de la institución, que permite la generación y validación de la vigencia de un foro, así como la participación y conclusiones del mismo.

7. Componente para el manejo de listas de correo, adaptado al estándar de la institución, parametrizable, lo que permite su reutilización en cualquier aplicación web que requiera de esta actividad.

8. Prototipo para la implementación de la tecnología AJAX en la visualización de información, que utiliza el estándar de la institución, parametrizable, de fácil uso en cualquier aplicación web que necesite optimizar la visualización de la información.

V. Conclusiones

El desarrollo por componentes y la programación orientada a objetos permiten la reutilización del software, fundamental para la industrialización de este.

La estandarización es el único camino de unificación de criterios que permite el manejo de políticas de calidad en el desarrollo de software.

Cuando se trabaja con un equipo de ingenieros desarrolladores de software se debe tener un estándar bien definido, que contemple todas las etapas, desde la ingeniería de requerimientos, pasando por las etapas de análisis, diseño, desarrollo, pruebas y documentación.

Cuando se trabaja con herramientas libres es necesario contar con un *framework* que facilite la etapa de desarrollo.

Aunque la comunidad de desarrolladores de software con herramientas de código libre, está en crecimiento y cuenta ya con el apoyo de grandes empresas de Hardware, aún se tiene un gran campo de acción, pues quedan grandes retos por trabajar.

Referencias

[9] F. Capra. Frijot (1998). La trama de la vida. Anagrama, Barcelona, 1998.

[10] ERL, Thomas. Service-Oriented Architecture : Concepts, Technology and Design : Prentice Hall,. 792 p., 2005.

[11] C. Larman. UML y Patrones, Prentice Hall, México. 2002.

[12] R.S. Pressman, "Ingeniería del Software, un enfoque práctico", 5ª Edición. Mc Graw Hill, 2002.

[13] R. S. Pressman, "Ingeniería del Software, un enfoque práctico", 6ª Edición. Mc Graw Hill, 2006.

[14] A. Martínez. "Una metodología para el diseño de sistemas de información, basada en el estudio de sistemas blandos". *Revista Espacios*. ISSN 0798-1015 versión impresa, 2007.

[15] Ministerio de Administraciones Públicas. Documentación. Métricas V3. http://www.csi.map.es/csi/metrica3/index.html . 2006. Jacobson, I., Booch, G., RUCCaugh, J. "El Proceso Unificado de Desarrollo", Addison Wesley, 2000.

[16] Koch, N."Software Engineering for Adaptive Hypermedia Applications. Reference Model, Modeling Techniques and Development Process". PhD. Thesis, Ludwig-Maximilians-Universität München, 2001.

[17] Sommerville, I. "Software Engineering", 6th ed., Addison Wesley, 2001.

[18] Turk, D., France, R. y Rumpe, B. Limitations of Agile Software Processes. En Proceedings of 4th International, 2002.

[19] Conference on eXtreme Programming and Agile Processes in Software Engineering, XP 2002.

[20] Alghero, S., Italy, April 2002, pp. 43-46, 2002.

[21] Ward, S., Kroll, P., "Building Web Solutions with the Rational Unified Process: Unifying the Creative Design. 2008.