ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

DESARROLLO DE UNA APLICACIÓN PARA GENERAR REPORTES PERSONALIZADOS UTILIZANDO SOFTWARE LIBRE

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN INFORMÁTICA MENCIÓN INGENIERIA DE SOFTWARE

AUTOR:

JOHN ALDRIN ZUMBA ROSERO

Jazr99@yahoo.es

DIRECTORA: ING. SANDRA PATRICIA SANCHEZ GORDON

Sandra_sanchezg@yahoo.com

Quito, diciembre 2014

DECLARACIÓN

Yo, John Aldrin Zumba Rosero, declaro bajo juramento que el trabajo aquí descrito y detallado es de mi plena autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y que he consultado las referencias bibliográficas que se incluyen en el documento.

A través de la presente declaración cedo el derecho de propiedad intelectual correspondiente a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

_____

John Aldrin Zumba Rosero

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por John Aldrin Zumba Rosero, bajo mi supervisión.

_____

Ing. Sandra Sánchez

DIRECTORA DE PROYECTO

AGRADECIMIENTO

Agradezco al creador de todas las cosas por protegerme durante toda mi vida y darme el impulso para superar los obstáculos y dificultades para llegar hasta donde he llegado.

A mi padre y madre por enseñarme a nunca rendirme ante los problemas, por enseñarme que el amor es la fuerza más grande que existe, por haberme forjado como la persona que soy actualmente.

A mi esposa, por su paciencia, comprensión, dedicación, fuerza, amor y tan solo por ser tal y como es.

A mis hijas por su entereza y su compresión de no estar junto a ellas y por enseñarme a enfrentar las dificultades con alegría.

DEDICATORIA

A esas personas importantes en mi vida, que siempre estuvieron prestas a brindarme toda su ayuda y que hicieron todo en la vida para que yo pudiera lograr mis sueños, por motivarme y darme la mano cuando sentía que el camino se terminaba, a ustedes por siempre desde el fondo de mi alma.

CONTENIDO

INDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

El presente proyecto se enfoca en el desarrollo de una aplicación de información que permita generar y visualizar información almacenada en distintas bases de datos. Para lo cual se realiza un estudio del desarrollo de aplicaciones de software libre, una descripción de las diferencias que existen entre las diferentes bases de datos a utilizar en el presente trabajo, una descripción de las herramientas utilizadas y un adecuado análisis y diseño del sistema.

El CAPÍTULO 1 detalla las características más relevantes del software libre para el desarrollo web, sus antecedentes y categorías. Detalles importantes sobres las aplicaciones web y su aplicación.

El CAPÍTULO 2 presenta una comparación entre sistemas de gestión de bases de datos que se utilizarán para el desarrollo del presente trabajo.

El CAPÍTULO 3 contiene las etapas de construcción del sistema basado en el patrón de desarrollo MVC, además detalla la aplicación a un caso de estudio y obtención de resultados.

El CAPÍTULO 4 muestra las conclusiones y recomendaciones obtenidas como resultado del proyecto.

PRESENTACIÓN

En nuestro tiempo, la aparición de nuevas tecnologías basadas en las computadoras, la multimedia e Internet ha hecho que esté disponible gran cantidad de información y que el conocimiento se pueda difundir rápidamente.

Muchas de las cosas que realizamos habitualmente dentro del desarrollo de aplicaciones están siendo implementadas en un entorno orientado hacia la Web. Pero la duda es si eso es realmente lo mejor que se puede hacer con estas nuevas tecnologías. Al mismo tiempo que han ido aparecido un conjunto de variantes de solución al diseño y a la arquitectura de las aplicaciones de todo tipo. No obstante la aplicación de dichas variantes al desarrollo de software actual se vuelve un poco compleja y en ocasiones no compatibles con las características de este tipo de aplicaciones.

El presente proyecto pretende ser una herramienta diseñada para apoyar la labor de los administradores de bases de datos y usuarios finales en las tareas de obtener información de varias bases de datos, tomando en cuenta el ahorro en tiempo y recursos que implica la generación de la información pertinente.

CAPÍTULO1
SOFTWARE LIBRE PARA DESARROLLO WEB

1.1 ANTECEDENTES DEL SOFTWARE LIBRE

1.1.1 GENERALIDADES

El software libre se caracteriza por su código abierto y, por lo general, es gratuito. Pero “software libre” no significa “no comercial”. Un programa libre debe estar disponible para uso comercial, desarrollo comercial y distribución comercial.

Al ser abiertos, toda persona puede acceder a su código fuente ya sea para copiarlo, distribuirlo, estudiarlo, cambiarlo y mejorarlo. Así, alguien que necesita un software para desarrollar alguna tarea específica, puede tomar un código fuente existente y modificarlo en función de sus fines.

Los desarrolladores de software libre integran una amplia comunidad que tiene como principio fundamental la cooperación entre sus miembros.

1.1.2 SIGNIFICADO

El software es libre si sus usuarios y desarrolladores gozan de todas estas libertades:

- La libertad de usar el programa, con cualquier propósito.
- La libertad de estudiar cómo funciona el programa y adaptarlo a sus necesidades. El acceso al código fuente es una condición previa para esto.
- La libertad de distribuir copias, con lo que se puede ayudar a otros.
- La libertad de mejorar el programa y hacer públicas las mejoras a los demás, de modo que toda la comunidad se beneficie. El acceso al código fuente es un requisito previo para esto.

Software Libre implica la libertad de distribuir copias, sea con o sin modificaciones, en forma gratuita o cobrando un monto por la distribución, a cualquiera y en cualquier lugar.

Software Libre significa, entre otras cosas, que no hay que pedir o pagar permisos.

Software Libre implica la libertad de hacer modificaciones y utilizarlas de manera privada en trabajo u ocio, sin siquiera tener que anunciar que dichas modificaciones fueron realizadas. Cuando se publican y difunden los cambios no hace falta avisar a nadie en particular ni de ninguna manera en particular.

La libertad de distribuir copias debe incluir tanto las formas binarias o ejecutables del programa como su código fuente, sean o no versiones modificadas. Distribuir programas de modo ejecutable es necesario para que los sistemas operativos libres sean fáciles de instalar.

Para que las libertades de hacer modificaciones y de publicar versiones mejoradas tengan sentido, se debe tener acceso al código fuente del programa. Por lo tanto, la posibilidad de acceder al código fuente es una condición necesaria para el Software Libre.

Para que estas libertades sean reales, deben ser irrevocables mientras no se haga nada incorrecto; si el desarrollador del software tiene el poder de revocar la licencia aunque no haya motivos, el software no es libre.

1.1.3 ORIGEN

El software libre nació de la mano del propio software en la década de los años 60. En ese entonces, las gigantescas máquinas a las que llamaban computadoras hacían uso de programas cuyo código fuente estaba a la vista de todos y se podía distribuir libremente. Esto provocó que ya en esos tiempos, desde el punto de vista

de la informática, existiera una pequeña comunidad de científicos y programadores que intercambiara código, a la vez que informes de errores e ideas. El software por entonces no era más que un valor añadido a las computadoras de la época y se solía distribuir gratuitamente por los fabricantes.

La situación cambió radicalmente con el descenso del precio de las máquinas y sus componentes (hardware) y la progresiva necesidad de un software más potente y con mayores funcionalidades, lo que dio lugar a que incluso había gente que estaba dispuesta a pagar dinero por él. Esto que en sí no es necesariamente malo, provocó sin embargo un giro radical en la industria informática: las primeras compañías exclusivamente dedicadas a la creación de software aparecieron y se hicieron fuertes en el mercado.

Para poder obtener mayores beneficios económicos y estratégicos, estas empresas limitaron hasta más no poder lo que el usuario podía hacer con el software que estas empresas creaban.

No fue hasta mediados los años 80, cuando Richard Stallman formalizó las ideas básicas del movimiento del software libre que está revolucionando la industria del software.

El software libre, tal y como lo conocemos hoy, dio sus primeros pasos con un manifiesto en favor de la libertad de expresión y un proyecto conocido hoy mundialmente, el proyecto GNU.

Ha sido el binomio Internet-software libre el que ha propiciado uno de los cambios más radicales de las últimas décadas. La mayor parte de la infraestructura de Internet se sustenta sobre código libre, mientras que las posibilidades que ofrece Internet han sido vitales para el pleno desarrollo del software libre como elemento tecnológico. Sin embargo, mientras el cambio tecnológico basado en Internet ha tenido una fuerte implantación en el mundo occidental, la mentalidad ligada al

software libre está tardando algo más en calar en la sociedad. Pero no cabe duda de que paulatinamente va ganando en importancia.

1.2 CATEGORÍAS

En esta sección se describen las categorías de software libre cuyos términos encontramos esparcidas por todo Internet. Esta categorización es en función de la libertad o falta de ella que reciben los usuarios de ese programa para poder usarlo y/o modificarlo.

1.2.1 SOFTWARE LIBRE O "FREE SOFTWARE"

Permite al usuario copiar, modificar y distribuir copias del programa. El código fuente está disponible. Las compañías pueden distribuir copias por una cantidad de dinero.

El término "Free software" es con frecuencia mal traducido e interpretado. La palabra inglesa 'free' puede significar dos cosas: gratis o libre. Lo que se quiere decir con "free software" es software libre, y por eso en español se usa ambos términos indistintamente y no software gratis.

Con software libre la libertad que no se tiene es la de restringir estos derechos a otros usuarios, es decir, la libertad de eliminar libertades. Si se distribuye una copia o una modificación de un programa libre todos los usuarios disponen de los derechos antes citados.

Si un programa es libre, entonces puede ser potencialmente incluido en un sistema operativo libre tal como GNU, o sistemas GNU/Linux libres.

1.2.2 DOMINIO PÚBLICO

Este software no está protegido bajo copyright. Se dice entonces que un programa es de dominio público cuando la persona que lo ha realizado no tiene ningún derecho sobre él. Por lo tanto, no percibe ninguna compensación económica ni tiene control sobre su uso, modificación o distribución. Algunas copias, distribuciones o modificaciones del programa podrían no hacerse libres. Esto significa que algunas personas "no cooperativas" podrían tomar el programa y convertirlo en propietario, o sea que pueden hacer cambios, muchos o pocos, y distribuir el resultado como un producto de su propiedad, y por lo tanto las personas que reciban ese programa con esas modificaciones no tendrán la libertad que el autor original le dio.

Aunque en ocasiones se utiliza el término "dominio público" para identificarlo con "software libre" esto no es correcto. Dominio público es un término legal y significa "ausente de copyright" o "sin derechos reservados".

1.2.3 COPYLEFT

Una categoría especial de software libre que permite el uso, copia y modificación a los usuarios, y que no permite a los distribuidores restricciones a estas libertades cuando modifiquen, copien o distribuyan el programa. Esto asegura que cada copia será free software. Copyleft es un concepto general como copyright y debe existir una licencia que lo plasme en términos legales.

Un ejemplo de Copyleft es el compilador GNU para C++. La palabra surge de buscar un opuesto de la palabra copyright (derechos reservados) para ejemplificar que en vez de quitar libertad a los usuarios, este tipo de categoría garantiza esa libertad.

Para cubrir un programa con "Copyleft", primero se reservan los derechos, luego se le añaden términos de distribución los cuales son un instrumento legal.

1.2.4 FUENTE ABIERTA U OPEN SOURCE

El término software de "Fuente Abierta" u "Open Source" es usado por algunas personas para dar a entender más o menos lo mismo que software libre donde el acceso al código fuente es apenas un pre-requisito para dos de las cuatro libertades que definen al Software Libre. Muchas personas no entienden que el acceso al código fuente no es suficiente. "Software Libre" evita caer en esa confusión.

1.2.5 SOFTWARE LIBRE QUE NO ESTA PROTEGIDO POR COPYLEFT

Existe software que permite a los usuarios las libertades antes mencionadas e incluso incluir algunas restricciones a los redistribuidores. En ciertas condiciones algunas copias o modificaciones cesan de ser free software mientras que otras siguen siéndolo.

Si se lo desea, se puede obtener una copia que tenga esos términos de distribución y es libre. Sin embargo, hay versiones no libres también, y hay estaciones de trabajo populares y tarjetas gráficas para PC para las cuales versiones no libres son las únicas que funcionan.

1.2.6 LICENCIA PÚBLICA GENERAL GNU

La Licencia Pública General GNU (GNU General Public License o GPL) es una manera de plasmar legalmente el concepto de software bajo Copyleft.

La GPL es una licencia o contrato legal, un conjunto específico de términos de distribución para proteger con Copyleft a un programa. El software del Proyecto GNU, y muchos otros, se distribuyen protegidos por esta licencia.

1.2.7 LICENCIA PÚBLICA GENERAL PARA BIBLIOTECAS GNU

La Licencia Pública General para Bibliotecas GNU es una forma alternativa de la Licencia Pública General GNU que se aplica a algunas de las bibliotecas GNU. Esta licencia anteriormente se llamaba GPL Biblioteca ("Library GPL") pero fue cambiada de nombre porque el anterior invitaba a usar esta licencia más frecuentemente de lo que se debería usar.

1.2.8 EL SISTEMA GNU

Es un sistema operativo libre completo estilo Unix. El sistema GNU incluye todo el software GNU, así como muchos otros paquetes tales como el Sistema XWindow y TeX que no son software GNU.

Debido a que el propósito de GNU es ser libre, cada componente individual en el sistema GNU tiene que ser software libre. No todos tienen que estar protegidos con Copyleft, sin embargo; cualquier tipo de software libre es legalmente apto de incluirse si ayuda a alcanzar metas técnicas. Se puede hacer uso de software libre no protegido con Copyleft como el Sistema XWindow.

1.2.9 SOFTWARE GNU

Software GNU es software que es liberado bajo el auspicio del Proyecto GNU. La mayoría del software GNU está protegido con Copyleft, pero no todos. Algo de software GNU es escrito por el personal de la Fundación para el Software Libre, pero la mayoría del software GNU es aportada por voluntarios. Parte del software aportado está protegido con copyright por la Fundación para el Software Libre y otra parte está protegida con copyright por las personas que los escribieron.

1.2.10 SOFTWARE SEMI-LIBRE

El software semi-libre es software que no es libre, pero viene con autorización para particulares de usar, copiar, distribuir y modificar, incluyendo la distribución de versiones modificadas sin fines de lucro. PGP es un ejemplo de un programa semi-libre.

El software semi-libre es mucho mejor que el software propietario, pero aún plantea problemas y no podemos usarlo en un sistema operativo libre.

1.2.11 SOFTWARE PROPIETARIO

Software que no es ni libre ni semi-libre. El uso, modificación y redistribución está prohibido y se requiere que se adquiera un permiso.

1.2.12 SOFTWARE COMERCIAL

El software comercial es software que es desarrollado por una entidad que tiene la intención de hacer dinero del uso de dicho software. "Comercial" y "propietario" no son la misma cosa. La mayoría del software comercial es propietario, pero hay software libre comercial y hay software no libre no comercial. Por ejemplo, Ada de GNU siempre es distribuida bajo los términos de la Licencia Pública General de GNU y cada copia es software libre; pero los desarrolladores venden contratos de soporte.

1.2.13 FREEWARE

Los programas freeware son programas de Dominio Público, es decir, su autor los pone a disposición de cualquier persona de forma totalmente gratuita. El término “freeware" no tiene una definición clara aceptada, pero es usada comúnmente para

paquetes que permiten la redistribución pero no la modificación y su código fuente no está disponible. Estos paquetes no son software libre.

1.2.14 SHAREWARE

Es software que permite a los usuarios probar y realizar copias de los programas, pero requiere que todos aquellos que decidan usar el programa adquieran una licencia de uso. El shareware no es un tipo de programa sino una forma de distribución. El autor del programa no retiene los derechos de distribución, pero si los demás.

Generalmente el software shareware se distribuye sin fuentes, y no se permite ni modificarlo ni redistribuirlo salvo en los términos anteriores. Es software que viene con autorización para la gente de redistribuir copias, pero dice que quien continúe haciendo uso de una copia deberá pagar un cargo por licencia. El shareware no es software libre, ni siquiera semi-libre. Existen dos razones por las que no lo es:

- Para la mayoría del shareware, el código fuente no está disponible, de esta manera, no se puede modificar el programa en absoluto.
- El shareware no viene con autorización para hacer una copia e instalarlo sin pagar una cantidad por licencia, ni aún para particulares involucrados en actividades sin ánimo de lucro.

1.2.15 TRIAL O DEMO JUGABLE

Este es un tipo de software que en algunos aspectos se parece al shareware. Se lo puede ver mucho en juegos o en el caso de programas comerciales.

En el caso de los juegos, la técnica es ofrecer un capítulo o fase del juego para su distribución gratuita. Al resto de capítulos o fases sólo se podrá acceder cuando el usuario en cuestión se registre o adquiera la versión completa del juego.

En el caso de los software comerciales, la empresa distribuidora provee de una versión completa para ser descargada de su sitio personal, pero tiene un "plazo de vida" limitado. Esto quiere decir que pasado un tiempo de ejecución determinado para que el usuario pueda probarlo, el programa deja de funcionar, y sólo adquiriendo el programa se lo puede seguir usando.

1.2.16 MAILWARE

El concepto mailware implica que el autor cede su programa para ser evaluado por el usuario. Si el usuario decide que el programa es de su interés y que quiere utilizarlo por tiempo indefinido, debe enviar un mensaje por correo electrónico o convencional al autor y opcionalmente, enviarle una cantidad de dinero por el programa.

1.2.17 POSTALWARE O CARDWARE

En el postalware o cardware el autor cede su programa para ser evaluado por el usuario. Si el programa es del agrado de éste último, el usuario debe enviar una tarjeta postal de la ciudad donde viva el usuario al autor. Esto lo hace el autor principalmente por motivos estadísticos y el usuario suele mandar en la tarjeta algún tipo de comentario sobre el programa, alguna idea, un saludo, etc.

1.3 APLICACIONES WEB

1.3.1 INTRODUCCIÓN

Internet, la red de redes, nace a mediados de la década de los setenta, bajo los auspicios de DARPA, la Agencia de Proyectos Avanzados para la Defensa de

Estados Unidos. DARPA inició un programa de investigación de técnicas y tecnologías para unir diversas redes de conmutación de paquetes, permitiendo así a los ordenadores conectados a estas redes comunicarse entre sí de forma fácil y transparente.

De estos proyectos nació un protocolo de comunicaciones de datos, IP o Internet Protocol, que permitía a ordenadores diversos comunicarse a través de una red, Internet, formada por la interconexión de diversas redes.

A mediados de los noventa se inició el despunte de Internet. En esa época el número de proveedores de acceso privado se disparó, permitiendo a millones de personas acceder a Internet, que a partir de ese momento ya se empezó a conocer como la Red.

En estos momentos disponer de una dirección de correo electrónico, de acceso a la web, etc., ha dejado de ser una novedad para convertirse en algo normal en muchos países del mundo. Por eso las empresas, instituciones, administraciones y demás están migrando rápidamente todos sus servicios, aplicaciones, ofertas, etc., a un entorno web que permita a sus clientes y usuarios acceder a todo ello por Internet.

A pesar del ligero descenso experimentado en el ritmo de crecimiento, Internet está destinado a convertirse en un servicio universal de comunicaciones, permitiendo una comunicación universal.

1.3.2 HISTORIA DE LAS APLICACIONES WEB

Inicialmente la web era simplemente una colección de páginas estáticas, documentos, etc., que podían consultarse o descargarse. El siguiente paso en su evolución fue la inclusión de un método para crear páginas dinámicas que permitieran que lo mostrado fuese dinámico (generado o calculado a partir de los datos dela petición). Dicho método fue conocido como CGI (Common Gateway

interface) y definía un mecanismo mediante el cual podíamos pasar información entre el servidor HTTP y programas externos.

Los CGI siguen siendo muy utilizados, puesto que la mayoría de los servidores web los soportan debido a su sencillez. Además, proporcionan libertad a la hora de escoger el lenguaje de programación para desarrollarlos.

El esquema de funcionamiento de los CGI tenía un punto débil: cada vez que se recibe una petición, el servidor web lanzaba un proceso que ejecutaba el programa CGI. Como, por otro lado, la mayoría de CGI estaban escritos en algún lenguaje interpretado en algún lenguaje que requería run-time, esto implicaba una gran carga para la máquina del servidor.

Además, si la web tenía muchos accesos al CGI, esto suponía problemas graves. Por ello se empiezan a desarrollar alternativas a los CGI para solucionar este grave problema de rendimiento. Las soluciones vienen principalmente por dos vías. Por un lado se diseñan sistemas de ejecución de módulos más integrados con el servidor, que evitan que éste tenga que instanciar y ejecutar multitud de programas. La otra vía consiste en dotar al servidor de un intérprete de algún lenguaje de programación que nos permita incluirlas páginas en el código de manera que el servidor sea quien lo ejecute, reduciendo así el tiempo de respuesta.

A partir de este momento, se vive una explosión del número de arquitecturas y lenguajes de programación que nos permiten desarrollar aplicaciones web. Todas ellas siguen alguna de las dos vías ya mencionadas. De ellas, las más útiles y las que más se utilizan son aquellas que permiten mezclar los dos sistemas, es decir, un lenguaje de programación integrado que permita al servidor interpretar comandos que “incrustemos” en las páginas HTML y un sistema de ejecución de programas más enlazado con el servidor que no presente los problemas de rendimiento de los CGI.

1.3.3 FUNDAMENTOS DE LA WEB

El éxito de la web se basa en dos aspectos básicos y fundamentales:

- Protocolo HTTP
- Lenguaje HTML

1.3.3.1 Protocolo HTTP

El protocolo HTTP (HyperText Transfer Protocol) es el protocolo base de la WWW el cual permite una implementación simple y sencilla de un sistema de comunicaciones que nos permite enviar cualquier tipo de ficheros de una forma fácil, simplificando el funcionamiento del servidor y permitiendo que servidores poco potentes atiendan miles de peticiones y reduzcan los costes de despliegue.

Se trata de un protocolo simple, orientado a conexión y sin estado. La razón de que esté orientado a conexión es que emplea para su funcionamiento un protocolo de comunicaciones (TCP) de modo conectado, un protocolo que establece un canal de comunicaciones entre el cliente y el servidor, por el que pasa el flujo de bytes que constituyen los datos que hay que transferir.

El protocolo no mantiene estado, es decir, cada transferencia de datos es una conexión independiente de la anterior, sin relación alguna entre ellas, hasta el punto de que para transferir una página web tenemos que enviar el código HTML del texto, así como las imágenes que la componen.

1.3.3.2 Lenguaje HTML

El otro puntal del éxito del WWW ha sido el lenguaje HTML (HyperText Mark-up Language). Se trata de un lenguaje de marcas (se utiliza insertando marcas en el

interior del texto) que nos permite representar de forma rica el contenido y también referenciar otros recursos como imágenes, videos, etc., enlaces a otros documentos, mostrar formularios para posteriormente procesarlos, etc.

El lenguaje HTML actualmente empieza a proporcionar funcionalidades más avanzadas para crear páginas más ricas en contenido. Además se ha definido una especificación compatible con HTML, el XHTML (eXtensible HyperText Markup Language) que se suele definir como una versión XML validable de HTML, proporcionándonos un XML Schema contra el que se valida el documento para comprobar si está bien formado, etc.

1.3.4 DESARROLLO DE APLICACIONES WEB

Con el advenimiento de Internet y del Web, se han abierto infinidad de posibilidades en cuanto al acceso a la información desde casi cualquier sitio. Esto representa un desafío a los desarrolladores de aplicaciones, ya que los avances en tecnología demandan cada vez aplicaciones más rápidas, ligeras y robustas que permitan utilizar el Web.

En el mercado existen herramientas potentes para realizar esto, ya que han surgido nuevas tecnologías que permiten que el acceso a una base de datos desde el Web, por ejemplo, sea un mero trámite. El único problema es decidir entre el conjunto de posibilidades la correcta para cada situación.

El viejo CGI ha cumplido con el propósito de añadir interactividad a las páginas Web pero sus deficiencias en el desarrollo de aplicaciones y en la escalabilidad de las mismas ha conducido al desarrollo de APIs específicos de servidor como *Active Server Pages*, ASP, y PHP, que son más eficientes que su predecesor CGI.

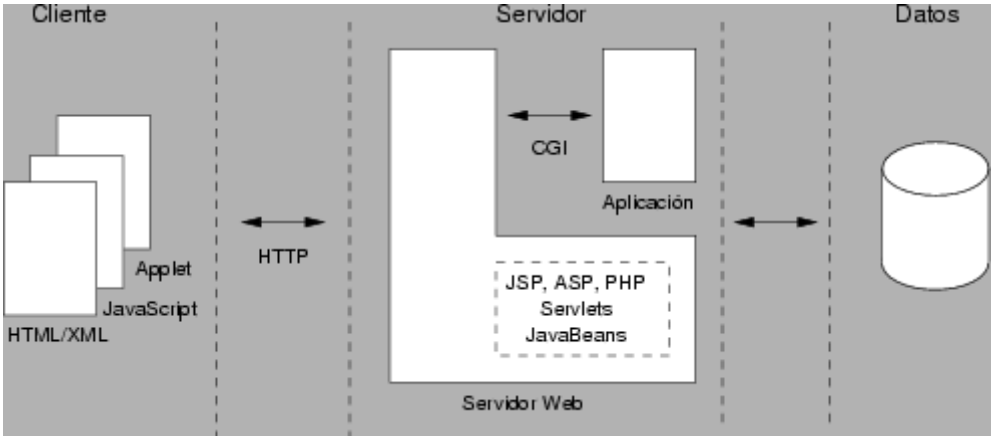
Para aprovechar el potencial de estas tecnologías y ofertar una solución de servidor más extensible y portable, Sun ha desarrollado la tecnología llamada *servlet*. Los

servlets Java son muy eficientes, debido al esquema de threads en el que se basan y al uso de una arquitectura estándar como la JVM, *Java Virtual Machine*.

Otra nueva tecnología viene a sumarse a las que extienden la funcionalidad de los servidores Web, llamada *JavaServerPages*, JSP. Los JSP permiten juntar HTML, aplicaciones Java, y componentes como las *JavaBeans* creando una página Web especial que el servidor Web compila dinámicamente en un servlet la primera vez que es llamada.

El esquema general donde se muestran cada tipo de tecnología involucrada en la generación e interacción de documentos Web se muestra en la figura 2.1.

Fig. 2.1 Esquema General de las Tecnologías Web



Fuente: Introducción a las Aplicaciones Web, Jesús Vegas[1]

1.3.5 HERRAMIENTAS DE SOFTWARE PARA DESARROLLO WEB

Existen una gran variedad de herramientas que permiten desarrollar aplicaciones Web con aspecto y prestaciones profesionales, aunque la mayoría de ellos requiere de una fase inicial de entrenamiento de duración variable según los casos.

[1] http://web1\Introducción a las Aplicaciones Web.htm

1.3.5.1 Características Básicas de un Servidor Web

Un servidor web es un programa que atiende y responde a las diversas peticiones de los navegadores, proporcionándoles los recursos que solicitan mediante el protocolo HTTP o el protocolo HTTPS (la versión segura, cifrada y autenticada de HTTP). Un servidor web básico tiene un esquema de funcionamiento muy sencillo, ejecutando de forma infinita el bucle siguiente:

1. Espera peticiones en el puerto TCP asignado (el estándar para HTTP es el 80).
2. Recibe una petición.
3. Busca el recurso en la cadena de petición.
4. Envía el recurso por la misma conexión por donde ha recibido la petición.
5. Vuelve al punto 2.

Un servidor web que pueda seguir el esquema anterior cumpliría los requisitos básicos de los servidores HTTP, aunque, eso sí, sólo podría servir ficheros estáticos.

A partir del esquema anterior se han diseñado y construido todos los programas servidores de HTTP que existen, variando sólo el tipo de peticiones (páginas estáticas, CGI, Servlets, etc.) que pueden atender, en función de que sean o no multi-proceso, multi-hilados, etc.

1.3.5.2 Servidor Apache

Apache es un servidor web de código libre robusto cuya implementación se realiza de forma colaborativa, con prestaciones y funcionalidades equivalentes a las de los servidores comerciales. El proyecto está dirigido y controlado por un grupo de voluntarios de todo el mundo que, usando Internet y la web para comunicarse, planifican y desarrollan el servidor y la documentación relacionada.

Estos voluntarios se conocen como el Apache Group. Además del Apache Group, cientos de personas han contribuido al proyecto con código, ideas y documentación.

1.3.5.2.1 Historia

En febrero de 1995, el servidor web más popular de Internet era un servidor de dominio público desarrollado en el NCSA (National Center for Supercomputing Applications en la Universidad de Illinois). No obstante, al dejar Rob McCool (el principal desarrollador del servidor) la NCSA en 1994, la evolución de dicho programa había quedado prácticamente reducida a la nada. El desarrollo pasó a manos de los responsables de sitios web, que progresivamente introdujeron mejoras en sus servidores. Un grupo de éstos, usando el correo electrónico como herramienta básica de coordinación, se puso de acuerdo en poner en común estas mejoras (en forma de “parches”). Dos de estos desarrolladores, Brian Behlendorf y Cliff Skolnick, pusieron en marcha una lista de correo, un espacio de información compartida y un servidor en California donde los desarrolladores principales pudiesen trabajar.

A principios del año siguiente, ocho programadores formaron lo que sería el Grupo Apache. Éstos, usando el servidor NCSA 1.3 como base de trabajo, añadieron todas las correcciones de errores publicadas y las mejoras más valiosas que encontraron y probaron el resultado en sus propios servidores. Posteriormente publicaron lo que sería la primera versión oficial del servidor Apache (la 0.6.2, de Abril de 1995). Casualmente, en esas mismas fechas, NCSA reemprendió el desarrollo del servidor NCSA.

En este momento el desarrollo de Apache siguió dos líneas paralelas. Por un lado, un grupo de los desarrolladores siguió trabajando sobre el Apache 0.6.2 para producir la serie 0.7, incorporando mejoras. Un segundo grupo reescribió por completo el código, creando una nueva arquitectura modular. En julio de 1995 se

migraron a esta nueva arquitectura las mejoras existentes para Apache 0.7, haciéndose público como Apache 0.8.

El 1 de diciembre de 1995, apareció Apache 1.0, que incluía documentación y muchas mejoras en forma de módulos incrustables. Poco después, Apache sobrepasó al servidor de NCSA como el más usado en Internet, posición que ha mantenido hasta nuestros días. En1999 los miembros del Grupo Apache fundaron la Apache Software Foundation, que provee soporte legal y financiero al desarrollo del servidor Apache y los proyectos laterales que han surgido de éste.

1.3.5.2.2 Características

Las características más destacadas del servidor Apache son:

- Soporte del protocolo HTTP 1.1: Es totalmente compatible con los estándares del protocolo HTTP
- Sencillo, con la configuración basada en un poderoso archivo: Se trata de un sencillo archivo de configuración llamado httpd. Conf que se puede utilizar para configurar Apache.
- Soporte para CGI (Common Gateway Interface): Apache soporta CGI utilizando los módulos modcgi y modcgid. Es compatible con CGI y aporta características extendidas como personalización de las variables de entorno y soporte de reparación de errores o debugging, que son difíciles de encontrar en otros servidores Web.
- Soporte de FastCGI: Apache tiene una solución para esto.
- Soporte de host virtuales: Apache soporta tanto host basados en IP como host virtuales.
- Soporte de autentificación HTTP: Apache soporta autentificación básica basada en la Web. Está también preparado para autentificación basada en la digestión de mensajes, que es algo que los navegadores Web populares ya

han implementado. Apache puede implementar autentificación básica utilizando tanto archivos estándar de contraseña como los DBM, llamadas a SQL o llamadas a programas externos de autentificación.

- Perl integrado: Perl se ha convertido en el estándar para la programación de scripts CGI.
- Soporte de scripts PHP: Apache tiene un amplio soporte de PHP utilizando el modulo modphp.
- Soporte de servlets de Java: Puede ejecutar servlets de Java utilizando el premiado entorno Tomcat con Apache.
- Servidor proxy integrado: Apache se lo puede configurar en un servidor proxy cache.
- Estado del servidor y adaptación de registros: Apache le da una gran cantidad de flexibilidad en el registro y la monitorización del estado del servidor. El estado del servidor puede monitorizarse mediante un navegador Web. Además, puede adaptar sus archivos de registro a su gusto.
- Soporte de Server Side Includes (SSI): Apache contiene un conjunto de Server Side Includes que añaden una gran cantidad de flexibilidad para el desarrollador del sitio Web.
- Soporte de Secured Sockect Layer (SSL): puede crear fácilmente un sitio Web SSL utilizando Open SSL y el modulo modssl de Apache.

1.3.5.3 PHP

PHP, cuyas siglas responden a un acrónimo recursivo (PHP: hipertexto preprocessor), es un lenguaje sencillo, de sintaxis cómoda y similar a la de otros lenguajes como Perl, C y C++. Es rápido, interpretado, orientado a objetos y multiplataforma. Para él se encuentra disponible una multitud de librerías.

PHP es un lenguaje ideal tanto para aprender a desarrollar aplicaciones web como para desarrollar aplicaciones web complejas. PHP añade a todo eso la ventaja de que el intérprete de PHP, los diversos módulos y gran cantidad de librerías

desarrolladas para PHP son de código libre, con lo que el programador de PHP dispone de una gran variedad de herramientas libres para desarrollar aplicaciones.

PHP suele ser utilizado conjuntamente con Perl, Apache, MySQL o PostgreSQL, formando una combinación barata ya que todos los componentes son de código libre, potente y versátil.

Apache, así como algunos otros servidores web, Roxen entre ellos, puede incorporar PHP como un módulo propio del servidor, lo cual permite que las aplicaciones escritas en PHP resulten mucho más rápidas que las aplicaciones CGI habituales.

1.3.5.3.1 Historia

PHP fue originalmente diseñado en Perl, seguidos por la escritura de un grupo de CGI binarios escritos en el lenguaje C por el programador Danés-Canadiense RasmusLerdorf en el año de 1994 para mostrar su currículum vitae y guardar ciertos datos, como la cantidad de tráfico que su página web recibía. En junio de 1995 fue publicado “Personal Home Page Tools” luego de que Lerdorf lo combinara con su propio FormInterpreter para crear PHP/FI.

Dos programadores israelíes de Technion, Zeev Suraski y Andi Gutmas, reescribieron el analizador gramatical (por ser en inglés) en el año de 1997 y crearon la base del PHP 3, cambiando el nombre del lenguaje a la forma actual. Experimentaciones públicas de PHP 3 comenzaron inmediatamente y fue lanzado oficialmente en junio de 1998.

Para 1999, Suraski y Gutmans reescribieron el código de PHP, produciendo lo que hoy se conoce como Zend Engine o motor Zend. También conformaron Zend Technologies en Ratmat Gan, Israel. En mayo del 2000 PHP 4 fue lanzado bajo el poder del motor Zend Engine 1.0. En julio del 2004, PHP 5 fue lanzado, utilizando el motor Zend Engine II (o Zend Engine 2). La versión mas reciente de PHP es la 5.1,

que aún se encuentra en estado beta, que incluye el novedoso PDO (Objetos de Información de PHP o PHP Data Objects) y mejoras utilizando las ventajas que provee el nuevo Zend Engine 2.

1.3.5.3.2 Características

Las características más destacadas de PHP son:

- Programación de páginas web dinámicas, habitualmente en combinación con el motor de base datos MySql, aunque cuenta con soporte nativo para otros motores, incluyendo el estándar ODBC, lo que amplia en gran medida sus posibilidades de conexión.
- Programación en consola, al estilo de Perl, en Linux, Windows y Macintosh.
- Creación de aplicaciones gráficas independientes del navegador, por medio de la combinación de PHP y GTK, que permite desarrollar aplicaciones de escritorio tanto para los sistemas operativos basados en Unix, como para Windows y Mac Os X.
- Capacidad de accesar la mayoría de las base de datos que se utilizan en la actualidad.
- Leer los datos desde diferentes fuentes, incluyendo datos que pueden meter los usuarios desde formas HTML y manipularlos de forma sencilla.
- Capacidad de expandir su potencial utilizando la enorme cantidad de módulos (llamados ext's o extensiones).
- Es Libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite las técnicas de Programación Orientada a Objetos.

1.3.5.4 MySQL

MySQL se disputa con PostgreSQL el puesto de SGBD más conocido y usado de código libre. MySQL es un SGBD desarrollado por la empresa MySQL AB, una

empresa de origen sueco que lo desarrolla bajo licencia de código libre (concretamente bajo GPL), aunque también, si se desea, puede ser adquirido con licencia comercial para ser incluido en proyectos no libres.

MySQL es un sistema gestor de base de datos extremadamente rápido. Aunque no ofrece las mismas capacidades y funcionalidades que otras muchas bases de datos, compensa esta pobreza de prestaciones con un rendimiento excelente que hace de ella la base de datos de elección en aquellas situaciones en las que necesitamos sólo unas Capacidades básicas.

1.3.5.4.1 Historia

MySQL surgió como un intento de conectar el gestor mSQL a las tablas propias de MySQL AB, usando sus propias rutinas a bajo nivel. Tras unas primeras pruebas, los desarrolladores vieron que mSQL no era lo bastante flexible para lo que necesitaban, por lo que tuvieron que desarrollar nuevas funciones. Esto resultó en una interfaz SQL a su base de datos, con una interfaz totalmente compatible a mSQL.

Dentro de la documentación de MySQL indica que no se sabe con certeza de donde proviene su nombre. Por un lado se explica que sus librerías han llevado el prefijo 'my' durante los diez últimos años. Por otro lado, la hija de uno de los desarrolladores se llama My. No se sabe cuál de estas dos causas (aunque bien podrían tratarse de la misma), han dado lugar al nombre de este conocido gestor de bases de datos.

1.3.5.4.2 Características de MySQL

Las características más destacadas de MySQL son:

- Soporte de transacciones (nuevo en MySQL 4.0 si lo usamos como motor de almacenamiento).

- Soporte de replicación (con un máster actualizando múltiples claves).
- Librería para uso embebido.
- Búsqueda por texto.
- Cache de búsquedas (para aumentar el rendimiento).
- Aprovecha la potencia de sistemas multiprocesador, gracias a su implementación multihilo.
- Gestión de usuarios y passwords, manteniendo un muy buen nivel de seguridad en los datos.

CAPÍTULO 2
SISTEMAS ADMINISTRADORES DE BASES DE DATOS

2.1 ANTECEDENTES

En esta sección se definirán los conceptos necesarios para el entendimiento del problema así como una comparación entre sistemas de gestión de bases de datos que se utilizarán para el desarrollo del presente trabajo.

2.2 CONCEPTOS BÁSICOS

Una base de datos es una herramienta para recopilar y organizar información. En las bases de datos, se puede almacenar información sobre personas, productos, o cualquier otro objeto.

Una tabla de una base de datos es similar en apariencia a una hoja de cálculo, en cuanto a que los datos se almacenan en filas y columnas. La principal diferencia entre almacenar los datos en una hoja de cálculo y hacerlo en una base de datos es la forma de organizarse los datos.

Cada fila de una tabla se denomina registro. En los registros es donde se almacena cada información individual. Cada registro consta de campos (al menos uno). Los campos corresponden a las columnas de la tabla. Los campos se deben configurar con un determinado tipo de datos, ya sea texto, fecha, hora, numérico, o cualquier otro tipo.

La clave principal es una columna que se utiliza para identificar inequívocamente cada fila, como Id. de producto o Id. de pedido.

Un sistema de gestión de bases de datos (DBMS), consiste en una colección de datos interrelacionados y un conjunto de programas para acceder a esos datos. Su objetivo principal es proporcionar una visión abstracta de los datos, es decir el sistema esconde algunos detalles de cómo se almacenan y mantienen los datos.

El diseño global de la base de datos se llama esquema de la base de datos. La capacidad para modificar una definición de esquema en un nivel sin afectar a una definición de esquema en el siguiente nivel superior se llama independencia de datos.

Un lenguaje de manipulación de datos (DML) permite a los usuarios acceder o manipular los datos. Existe dos tipos; procedurales, que requieren que el usuario especifique qué datos se necesitan y cómo obtenerlos, y no procedurales, que requieren que el usuario especifique qué datos se necesitan sin especificar cómo obtenerlos.

2.3 COMPARACIÓN DE SISTEMAS ADMINISTRADORES DE BASES DE DATOS

La rapidez, la efectividad en los procesos y el manejo de gran cantidad de información tienen prioridad al momento de optimizar servicios y productos. Lo que ha dado lugar al surgimiento de multitud de gestores de bases de datos, siendo estos programas los que permiten manejar la información de modo sencillo y que brindan servicios para el desarrollo y el manejo de bases de datos.

Con la salida al mercado de múltiples entornos de desarrollo la preocupación están en conocer las características, ventajas y desventajas de cada manejador de base de datos, y para el caso específico de este trabajo damos a conocer las características generales de los productos que más se destacan como son SQL Server, Oracle, MySql y PostGreSQL.

2.3.1 INFORMACIÓN GENERAL

En la Tabla 2.1 se presenta información general de las bases de datos mencionadas.

- **Creador** hace referencia a la empresa que desarrolló e implementó la base de datos
- **Fecha de la primera versión pública** es la fecha en la cual se dio a conocer la aplicación
- **Última versión estable** es la versión estable más reciente que salió al mercado
- **Licencia de software** es el tipo de contrato entre autor y cliente del programa informático, para utilizar el software cumpliendo una serie de términos y condiciones establecidas dentro de sus cláusulas

Tabla 2.1 Información general de las bases de datos

	Creador	Fecha de la primera versión pública	Última versión estable	Licencia de software
Microsoft SQL Server	Microsoft	1989	2008	Propietario
MySQL	MySQL AB	Noviembre de 1996	5.0	GPL o propietario
Oracle	Oracle Corporation	1977	11g Release 2	Propietario
PostgreSQL	PostgreSQL Global DevelopmentGroup	Junio de 1989	9.0	Licencia BSD

Elaborado por: Autor

2.3.2 SOPORTE DEL SISTEMA OPERATIVO

En la Tabla 2.2 se presenta el soporte del sistema operativo de las bases de datos mencionadas.

Tabla 2.2 Soporte del Sistema Operativo

	Windows	Mac OS X	Linux	BSD	Unix	z/OS
Microsoft SQL Server	Sí	No	No	No	No	No
MySQL	Sí	Sí	Sí	Sí	Sí	Quizá
Oracle	Sí	Sí	Sí	Sí	Sí	Sí
PostgreSQL	Sí	Sí	Sí	Sí	Sí	No

Elaborado por: Autor

2.3.3 CARACTERÍSTICAS FUNDAMENTALES

En la Tabla 2.3 se presentan características fundamentales de las bases de datos mencionadas.

- **ACID** conjunto de características necesarias para que una serie de instrucciones puedan ser consideradas como una transacción
- **Integridad referencial** garantiza que una entidad (fila o registro) siempre se relacione con otras entidades válidas, es decir, que existen en la base de datos
- **Transacciones** en un Sistema de Gestión de Bases de Datos (SGBD), es un conjunto de órdenes que se ejecutan formando una unidad, es decir, en forma indivisible o atómica. Un SGBD se dice transaccional, si es capaz de mantener la integridad de los datos, haciendo que estas transacciones no puedan finalizar en un estado intermedio. Cuando por alguna causa el sistema debe cancelar la transacción, empieza a deshacer las órdenes ejecutadas hasta dejar la base de datos en su estado inicial (llamado punto de integridad), como si la orden de la transacción nunca se hubiese realizado
- **Unicode** es un estándar de codificación de caracteres diseñado para facilitar el tratamiento informático, transmisión y visualización de textos de múltiples lenguajes y disciplinas

Tabla 2.3 Características Fundamentales de las bases de datos

	ACID	Integridad referencial	Transacciones	Unicode
Microsoft SQL Server	Sí	Sí	Sí	Sí
MySQL	Depende [1]	Depende [1]	Depende [1]	Sí
Oracle	Sí	Sí	Sí	Sí
PostgreSQL	Sí	Sí	Sí	Sí

Elaborado por: Autor

Nota 1. Para las transacciones y la integridad referencial, el tipo de tabla InnoDB debe ser usado

2.3.4 TABLAS TEMPORALES Y VISTAS

En la Tabla 2.4 se presenta la funcionalidad de tablas temporales y vistas de las bases de datos mencionadas.

- **Tabla Temporal** existen dos tipos de tablas temporales: locales y globales. Las tablas temporales locales son visibles sólo para sus creadores durante la misma conexión a una instancia de SQL indicando cuando se crearon o cuando se hizo referencia a ellas por primera vez. Las tablas temporales locales se eliminan cuando el usuario se desconecta de la instancia de SQL. Las tablas temporales globales están visibles para cualquier usuario y conexión una vez creadas, y se eliminan cuando todos los usuarios que hacen referencia a la tabla se desconectan de la instancia de SQL
- **Vista** es un resultado de una consulta SQL de una o varias tablas; también se le puede considerar una tabla virtual

Tabla 2.4 Tablas temporales y Vistas

	Tabla temporal	Vista materializada
Microsoft SQL Server	Sí	Similar [2]
MySQL	Sí	No
Oracle	Sí	Sí
PostgreSQL	Sí	No [3]

Elaborado por: Autor

Nota 2.: El servidor MS SQL provee vistas indexadas.

Nota 3. La vista realizada puede ser emulada con PL/PgSQL

2.3.5 ÍNDICES

En la Tabla 2.5 se presenta la funcionalidad de índices de las bases de datos mencionadas.

- **Índice** de una base de datos es una estructura de datos que mejora la velocidad de las operaciones, permitiendo un rápido acceso a los registros de una tabla en una base de datos. Al aumentar drásticamente la velocidad de acceso, se suelen usar sobre aquellos campos sobre los cuales se hacen frecuentes búsquedas
- **Arboles-R** son estructuras de datos de tipo árbol similares a los árboles-B, con la diferencia de que se utilizan para métodos de acceso espacial, es decir, para indexar información multidimensional
- **Hash** se refiere a una función o método para generar claves o llaves que representen de manera casi particular a un documento, registro, archivo, etc., resumir o identificar un dato a través de la probabilidad, utilizando una *función hash* o *algoritmo hash*. Un hash es el resultado de dicha función o algoritmo
- **Expresión** es una combinación de constantes, variables o funciones, que es interpretada (evaluada) de acuerdo a las normas particulares de precedencia y asociación para un lenguaje de programación en particular

Tabla 2.5 Índices de las bases de datos

	Árbol R-/R+	Hash	Expresión	Parcial	Reversa	Mapa de bits
Microsoft SQL Server	?	?	No	No	No	No
MySQL	Tablas MyISAM solamente	Tablas HEAP solamente	No	No	No	No
Oracle	Edición EE solamente	?	Sí	No	Sí	Sí
PostgreSQL	Sí	Sí	Sí	Sí	No	No

Elaborado por: Autor

2.3.6 OTROS OBJETOS

En la Tabla 2.6 se presenta la funcionalidad adicional de las bases de datos mencionadas.

- **Dominio** es un conjunto de ordenadores conectados en una red que confían a uno de los equipos de dicha red la administración de los usuarios y los privilegios que cada uno de los usuarios tiene en dicha red
- **Cursor** se refiere a una estructura de control utilizada para el recorrido y procesamiento de los registros del resultado de una consulta.
- **Trigger** (o disparador) en una Base de datos, es un procedimiento que se ejecuta cuando se cumple una condición establecida al realizar una operación
- **Función** y **procedimiento** se refieren a las rutinas internas escritas en SQL o lenguajes procedurales como PL/SQL.
- **Rutina externa** se refiere a la escritura en los lenguajes anfitriones como C, Java, Cobol, etc. "Procedimiento almacenado" es un término comúnmente usado para ese tipo de rutinas. Sin embargo, su definición varía entre diferentes vendedores de bases de datos

Tabla 2.6 Otros objetos de las Bases de Datos

	Dominio	Cursor	Trigger	Funciones	Procedimiento	Rutina externa
Microsoft SQL Server	No	Sí	Sí	Sí	Sí	Sí
MySQL	No	Sí	Sí	Sí	Sí	Sí
Oracle	Sí	Sí	Sí	Sí	Sí	Sí
PostgreSQL	Sí	Sí	Sí	Sí	Sí	Sí

Elaborado por: Autor

2.3.7 PARTICIONAMIENTO

En la Tabla 2.6 se presenta funcionalidad de particionamiento de las bases de datos mencionadas.

Una **partición** es una división de una base de datos lógica o sus elementos constituyentes en partes independientes. La partición de bases de datos se hace normalmente por razones de mantenimiento, rendimiento o manejo

Tabla 2.7 Particionamiento en las Bases de Datos

	Rango	Hash	Compuesto (Rango+Hash)	Lista
Microsoft SQL Server	Sí	No	No	No
MySQL	Sí	Sí	Sí	Sí
Oracle	Sí	Sí	Sí	Sí
PostgreSQL	Sí	No	No	Sí

Elaborado por: Autor

CAPÍTULO 3
DESARROLLO DEL SISTEMA

3.1 ANTECEDENTES DE LOS PATRONES

3.1.1 GENERALIDADES

El diseño, es un modelo del sistema, cumpliendo con una serie de principios y técnicas, que permite representar el sistema con el suficiente detalle como para ser implementado. Pero los principios y reglas no son suficientes, en el contexto de diseño podemos observar que los buenos desarrolladores tienen esquemas y estructuras de solución que usan numerosas veces en función de un determinado problema.

Estos esquemas y estructuras son conocimientos reusables los que permiten dar solución ante problemas similares. Christopher Alexander (creador del libro “A PatternLanguage: Towns/Building/Construction” en 1977) comenta que “Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, para describir después el núcleo de la solución a ese problema, de tal manera que esa solución pueda ser usada más de un millón de veces sin hacerlo siquiera dos veces de la misma forma”.

3.1.2 DEFINICIÓN

Un patrón conlleva una descripción general de una solución recurrente para un problema recurrente el cual contempla diferentes objetivos y condiciones. Pero un patrón hace algo más que identificar una solución, detalla el porqué se necesita la solución. Se debe tenerse presente que no cualquier solución, algoritmo, fórmula de diseño o heurística constituye un patrón. Sin embargo algo que parezca tener todos

los ingredientes necesarios para ser un patrón, no será considerado como tal hasta que verifique ser un problema recurrente.

Es un buen criterio no denominar a nada patrón hasta que no haya sido sometido a algún tipo de examen o revisión por parte de otras personas.

Para James Coplien un buen patrón debe cumplir los siguientes requisitos:

- Debe solucionar un problema: Los patrones capturan soluciones, no sólo principios abstractos o estrategias.
- Son conceptos probados: Los patrones capturan soluciones que han sido probadas, no teorías o especulaciones.
- La solución no es obvia: La mayoría de las técnicas de resolución de problemas (tales como métodos de diseño) intentan derivar soluciones partiendo de principios básicos. Los mejores patrones generan una solución para un problema indirectamente, una aproximación necesaria para los problemas de diseño más complejos.
- Describe una relación: Los patrones no deben describir módulos, sino que deben describir sistemas, estructuras o mecanismos más profundos.
- El patrón debe tener un componente humano importante: Todo software sirve para el confort humano o para la calidad de vida; los mejores patrones recurren explícitamente a la estética y a la utilidad.

3.1.3 ORIGEN DE LOS PATRONES DE SOFTWARE

El concepto de patrón de software se hizo popular dentro del mundo del desarrollo del software en 1995 con la publicación del libro DesignPatterns: Elements of Reusable Object-Oriented Software, normalmente conocido como GoF o Gang-of-Four debido a sus cuatro autores: Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides. Este libro presenta un conjunto de 23 patrones de diseño de

software, que suponen la aceptación definitiva del concepto de patrón de diseño para los desarrolladores de software.

De todas formas cabe destacar que ni los orígenes de los patrones, ni el comienzo de su uso en el mundo del software se encuentran en el famoso libro del GoF. Realmente el uso del término patrón, con el significado que actualmente se le da en la Ingeniería del Software, se deriva de los trabajos del arquitecto Christopher Alexander desde mediados de los años 60.

El trabajo de Alexander se resume en varios libros y artículos sobre planificación de urbanismo y arquitectura de edificios. Sin embargo, aún siendo sus publicaciones propias de arquitectura, sus ideas son aplicables a muchas otras disciplinas, entre las que cabe estacar su aplicación en el desarrollo de software.

Las ideas de Alexander fueron utilizadas por Ward Cunningham y Kent Beck para desarrollar un lenguaje de patrones (compuesto por cinco patrones) como guía de iniciación a la programación en Smalltalk.

3.1.4 COMPONENTES BÁSICOS DE UN PATRÓN DE SOFTWARE

Existen diferentes formatos para describir los patrones, sin embargo hay una serie de elementos que se consideran básicos y que deben contener todas las representaciones de los patrones. Así, en el libro de GoF se indica que un patrón debe contar con cuatro elementos esenciales:

- El nombre del patrón: una descripción manejable y entendible del problema. Debe ser corto (una palabra o dos).Amplía el vocabulario de diseño.
- El problema: describe cuando aplicar el patrón. Explica el problema y su contexto.
- La solución: describe los elementos que forman el diseño, sus relaciones, responsabilidades y colaboraciones. No describe un diseño o una

implementación concreta, sino que ofrece una descripción abstracta de un problema.

- Las consecuencias: son los resultados de aplicar un patrón. Son necesarias para evaluar alternativas de diseño, así como para evaluar los costes y beneficios de su aplicación.

3.1.5 CLASIFICACIÓN DE LOS PATRONES DE SOFTWARE

Según el libro de GoF los patrones se clasifican según el propósito para el cual han sido definidos.

Tabla 3.8 Clasificación de los patrones de software

	CREACIÓN	ESTRUCTURAL	DE CONDUCTA
CLASE	Método de Fabricación	Adaptador (clases)	Interprete Plantilla
OBJETO	Fabrica, Constructor, Prototipo, Singleton	Adaptador(objetos), Puente, Composición, Decorador, Fachada, Flyweight	Cadena de Responsabilidad, Comando(orden), Iterador, Intermediario, Observador, Estado, Estrategia, Visitante, Memoria

Elaborado por: Autor

3.1.6 PATRÓN DE DESARROLLO MVC

El patrón de software que será utilizado en el presente trabajo es el Patrón MVC Modelo Vista Controlador. Es esta sección se detalla sus características.

3.1.6.1 RAZONES PARA UTILIZAR PATRONES

La clave para la reutilización es anticiparse a los nuevos requisitos y cambios, de modo que los sistemas evolucionen de forma adecuada.

Cada patrón permite que algunos aspectos de la estructura del sistema puedan cambiar independientemente de otros aspectos. Facilitan la reusabilidad, extensibilidad y mantenimiento.

Un patrón es un esquema o micro arquitectura que supone una solución a problemas (dominios de aplicación) semejantes.

También conviene distinguir entre un patrón y una arquitectura global del sistema. Por decirlo en breve, es la misma distancia que hay entre el diseño de un componente (o módulo) y el análisis del sistema. Es la diferencia que hay entre el aspecto micro y el macro, por ello, en ocasiones se denomina a los patrones como "micro arquitecturas".

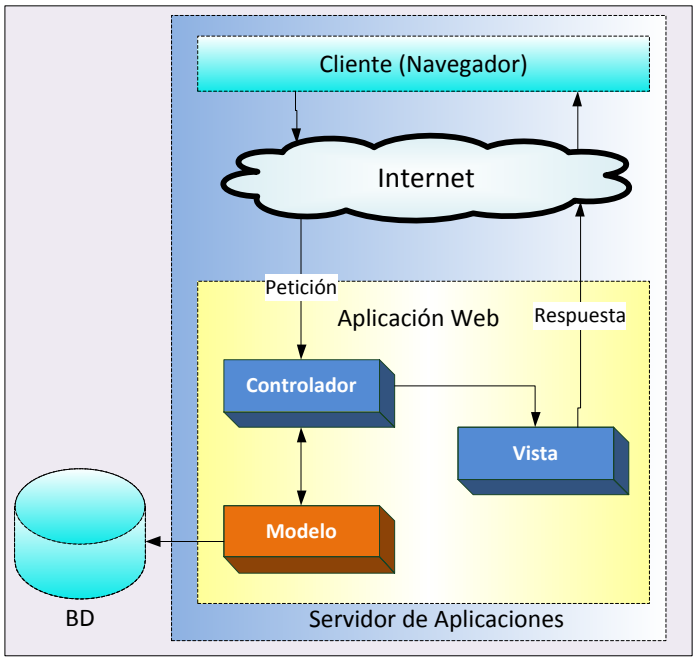
3.1.6.2 MODELO VISTA CONTROLADOR MVC

Modelo Vista Controlador (MVC) es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

El patrón MVC se ve frecuentemente en aplicaciones web en donde la lógica de un interfaz de usuario cambia con más frecuencia que los almacenes de datos y la lógica de negocio. Si realizamos un diseño que mezcle los componentes de interfaz y de negocio, entonces la consecuencia será que, cuando necesitemos cambiar el interfaz, tendremos que modificar laboriosamente los componentes de negocio. Se trata de realizar un diseño que desacople la vista del modelo, con la finalidad de mejorar la reusabilidad. De esta forma las modificaciones en las vistas impactan en menor medida en la lógica de negocio o de datos.

El modelo es el Sistema de Gestión de Base de Datos y la Lógica de negocio, y el controlador es el responsable de recibir los eventos de entrada desde la vista.

Fig. 3.2 Esquema del patrón MVC



Elaborado por: Autor

3.1.6.2.1 Historia

La arquitectura MVC fue introducida como parte de la versión Smalltalk-80 del lenguaje de programación Smalltalk. Fue diseñada para reducir el esfuerzo de programación necesario en la implementación de sistemas múltiples y sincronizados de los mismos datos. Inicialmente diseñado para aplicaciones de escritorio cuya idea principal fue separar la presentación del modelo de dominio.

En esas épocas la vista se encargaba del dibujo de mapas de bits, el controlador atendía acciones del mouse y teclado, y el modelo guardaba el estado de la ventana y la información mostrada por ella. Cada vista estaba asociada con un controlador y un modelo de su creación. Cualquier modelo podía ser asociado débilmente a cualquier visa ya que un modelo podía relacionarse con varias vistas. El controlador no se encargaba de manejar la lógica del negocio en la aplicación, su propósito era manejar la interacción del mouse y el teclado, y actualizar la vista a la cual estaba asociada, lo cual no era una tarea muy fácil.

3.1.6.2.1 Componentes

3.1.6.2.1.1 El Modelo

El modelo es un conjunto de clases que representan la información del mundo real que el sistema debe procesar (encapsula los datos y las funcionalidades), sin tomar en cuenta ni la forma en la que esa información va a ser mostrada ni los mecanismos que hacen que esos datos estén dentro del modelo, es decir, sin tener relación con ninguna otra entidad dentro de la aplicación es decir el modelo es independiente de cualquier representación de salida y/o comportamiento de entrada.

3.1.6.2.1.2 Las Vistas

Las vistas son el conjunto de clases que se encargan de mostrar al usuario la información contenida en el modelo. Una vista esta asociada a un modelo, pudiendo existir varias vistas asociadas al mismo modelo.

Una vista obtiene del modelo solamente la información que necesita para desplegar y se actualiza cada vez que el modelo del dominio cambia por medio de notificaciones generadas por el componente controlador.

3.1.6.2.1.3 El Controlador

El controlador es un objeto que se encarga de dirigir el flujo del control de la aplicación debido a mensajes externos, como datos introducidos por el usuario u opciones del menú seleccionadas por él. A partir de estos mensajes, el controlador se encarga de modificar el modelo o de abrir y cerrar vistas. El controlador tiene acceso al modelo y a las vistas, pero las vistas y el modelo no conocen de la existencia del controlador.

3.1.6.3 FLUJO DEL PATRÓN MVC

El flujo que sigue el patrón generalmente es el siguiente:

1. El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, enlace).
2. El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos (handler) o callback.
3. El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario. Los controladores complejos están a menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.
4. El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se refleja los cambios en el modelo (si se utiliza la variante 2 descrita anteriormente, de lo contrario lo obtiene a través del Controlador). El modelo no debe tener conocimiento directo sobre la vista. Sin embargo, el patrón de observador puede ser utilizado para proveer cierta indirección entre el modelo y la vista, permitiendo al modelo notificar a los interesados de cualquier cambio. Un objeto vista puede registrarse con el modelo y esperar a los cambios, pero aun así el modelo en sí mismo sigue sin saber nada de la vista. El controlador no pasa objetos de dominio (el modelo) a la vista aunque puede dar la orden a la vista para que se actualice. Nota: En algunas implementaciones la vista no tiene acceso directo al modelo, dejando que el controlador envíe los datos del modelo a la vista
5. La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

3.1.6.4 VENTAJAS

Desarrollar una aplicación siguiendo este patrón de diseño tiene muchas ventajas:

- La aplicación esta implementada modularmente
- Separación del Modelo de la Vista, es decir separa los datos de la representación visual de los mismos.
- Es más sencillo agregar múltiples representaciones de los mismos datos o información.
- Facilita agregar nuevos tipos de datos según sea requerido por la aplicación ya que son independientes del funcionamiento de las otras capas.
- Crea independencia de funcionamiento.
- Facilita el mantenimiento de caso de errores
- Ofrece maneras más sencillas para probar el correcto funcionamiento del sistema.
- Permite el escalamiento de la aplicación en caso de ser requerido.

3.1.6.5 DESVENTAJAS

Las desventajas de seguir el patrón de diseño MVC son:

- La separación de conceptos en capas agrega complejidad al sistema.
- La cantidad de archivos a mantener y desarrollar aumenta considerablemente
- La curva de aprendizaje del patrón de diseño es más alta que usando otros modelos más sencillos.

3.2 ANÁLISIS

3.2.1 VISIÓN GENERAL

En esta sección se presenta una visión general del enfoque de desarrollo utilizado, el cual es orientado por la práctica de desarrollo iterativo y manejo del riesgo. Los marcos de referencia aquí mencionados definen qué debe hacerse y en algunos casos cómo hacerlo.

Existe un conjunto de modelos de ciclo de vida que han evolucionado desde el tradicional modelo en cascada, pasando por algunas propuestas de mejoramiento como son el caso de cascadas con fases solapadas o cascada con subproyectos. Estos modelos no permiten una identificación temprana de los riesgos, los cuales pueden aparecer en las etapas finales del desarrollo o implantación, momento en que un cambio en el diseño del producto, en la arquitectura del sistema o en la infraestructura del hardware o software puede llevar al fracaso completo del proyecto.

Estos modelos son muy prácticos para proyectos pequeños y con muy bajos niveles de riesgos, tales como proyectos de nuevas versiones de algún software existente, donde haya requerimientos claros y cuya arquitectura e infraestructura de software y hardware no van a cambiar mucho respecto a la versión anterior.

Como alternativa al problema presentado por los modelos anteriores, los ciclos de vida evolucionaron y se han presentado propuestas como el modelo de entrega por etapas y el de entrega evolutiva.

Cada uno de ellos adiciona mayores niveles de complejidad a la administración, pero aseguran poseer un marco de trabajo más sólido y ajustado para el desarrollo de proyectos con niveles moderados de riesgo. Se requiere de todas formas un modelo

de ciclo de vida de proyectos que trabaje adecuadamente con altos niveles de riesgo, así que se desarrolló el modelo en espiral.

Este modelo tiene como objetivos la identificación de los riesgos para determinar la viabilidad del proyecto y definir planes de manejo para garantizar desde las fases iniciales la eliminación o mitigación de los riesgos donde es menos costoso y además la entrega desde las fases iniciales de productos probados, lo que permite un proceso continuo de pruebas y retroalimentación.

3.2.2 CLAVES DEL PROCESO DE DESARROLLO DE SOFTWARE

3.2.2.1 Desarrollo iterativo

El desarrollo iterativo es un método de construcción de productos de software cuyo ciclo de vida está compuesto por un conjunto de iteraciones, las cuales tienen como objetivo entregar versiones del software.

Cada iteración se considera un subproyecto que genera productos de software y no sólo documentación, permitiendo al usuario tener puntos de verificación y control más rápidos y provocando un proceso continuo de pruebas y de integración desde las primeras iteraciones.

Las iteraciones están compuestas por el conjunto de normas o actividades ya conocidas en el proceso de desarrollo de software. Estas son la especificación de requerimientos, el análisis y diseño, las pruebas, la administración de la configuración y el proceso de gerencia de proyectos.

3.2.2.2 Orientación al manejo del riesgo

Cada proyecto tiene asociado específicamente un conjunto de riesgos que requieren un plan de manejo claramente establecido, documentado y con una implementación

eficaz. De esta manera se pretende evitar posibles retrasos en los tiempos de entrega, problemas de calidad en el producto o en el peor de los casos, que puedan afectar la culminación del proyecto.

En las primeras etapas se implementan las funcionalidades con mayor exposición al riesgo y las de mayor complejidad, mejorando la posibilidad de éxito del proyecto. En la fase inicial del proyecto, el nivel de exposición al riesgo en ambos modelos es casi igual, pero en las fases siguientes es completamente diferente para cada modelo.

Se procede a la fase de elaboración donde se implementan aquellos casos de uso que atacan los riesgos de más alta prioridad, lo cual se denomina período de resolución de riesgos. Al final de esta fase se debe tener definida la arquitectura del sistema, así como la infraestructura en la que se soportará.

3.2.2.3 Orientación al cliente

Cuando se inicia un proyecto de desarrollo de software se conoce la importancia de la participación del cliente para lograr su terminación exitosa, pero usualmente cometemos el error de olvidar esta norma básica, lo que implica que la participación del cliente se restringe al inicio y finalización del proyecto, lo que en la mayoría de los casos produce un alto grado de insatisfacción en el usuario, al no obtener el producto con las especificaciones esperadas por los tanto es necesario contar con su participación en el proceso de planificación de las fases y de las iteraciones.

Posteriormente se requiere su participación en cada iteración para proveer retroalimentación temprana, garantizando el cumplimiento de las expectativas que tiene, además de ofrecerle una visión permanente del estado del proyecto, asegurando su compromiso para terminarlo exitosamente. Se debe tener en cuenta que el cliente no se interesa por los aspectos técnicos de alta complejidad y riesgo, razón por la cual se debe combinar esta práctica con una orientación al manejo del riesgo.

3.2.2.4 Desarrollo evolutivo

Cuando se trabaja con una especificación de requerimientos monolítica, se cae en el error de creer que se comprende completamente el concepto del producto sin haberlo validado con el cliente con algo más que documentos y modelos abstractos. Este proceso inicia con un concepto poco claro del producto a construir, y sólo se tiene claridad en la medida que se vaya desarrollando y verificando el producto con el cliente. Este tipo de proyectos se asemejan más al patrón que siguen los proyectos de investigación y desarrollo de nuevos productos.

3.2.3 METODOLOGIA DE DESARROLLO

Para implementar la solución propuesta se ha escogido una metodología que permita el desarrollo de manera ágil y que incluya las especificaciones y documentación necesarias para modelar los requerimientos obtenidos y desarrollar los modelos de análisis y diseño de la solución.

Para tener un modelo de solución ágil en el sentido de establecer los requerimientos, modelar la solución, e implementarlo de acuerdo a dicho modelamiento, se optó por la metodología AUP (Agile Unified Process).

Esta metodología es una versión simplificada de la metodología RUP (Rational Unified Process), que tiene como principios desarrollar una aplicación utilizando técnicas de modelamiento agiles y documentación necesarias y suficientemente buenas para el entendimiento del problema y el desarrollo de la solución.

3.2.4 ETAPAS DEL DESARROLLO AGIL DE SOFTWARE

3.2.4.1 Etapa de ingeniería

Esta etapa agrupa las fases de concepción y de elaboración, lo que básicamente le da por objetivos la conceptualización del sistema y el diseño inicial de la solución del problema.

Se inicia el proceso de administración de los requerimientos con la identificación y especificación de casos de usos, así como el proceso de aseguramiento de la calidad a través de los casos de prueba.

Se identifican los riesgos y se establece su plan de manejo, se ajusta ese plan según la tabla de priorización de riesgos y la de casos de usos vs. riesgos, para determinar en qué orden y en qué iteraciones se desarrollarán los artefactos de software que son la solución a los casos de uso.

Se identifican los recursos necesarios, tanto económicos como humanos, acordes con las necesidades del proyecto. Se da comienzo al proceso de estimación y planificación inicial a un nivel macro para todo el proyecto y posteriormente se realiza una estimación detallada de tiempos y recursos de las fases de concepción y elaboración.

3.2.4.1.1 Fase de concepción

Esta fase tiene como propósito definir y acordar el alcance del proyecto con los patrocinadores, identificarlos riesgos asociados al proyecto, proponer una visión muy general de la arquitectura de software y producir el plan de las fases y el de iteraciones.

3.2.4.1.1.1 Planeación de las fases y de las iteraciones

A partir del modelo de casos de uso y de la lista de riesgos, se puede determinar qué casos de uso deben implementarse primero para atacar los riesgos de mayor exposición.

Con base en la información previa se realiza el proceso de planificación general y un plan de trabajo detallado para la siguiente fase, así como el plan para la siguiente iteración.

Se debe establecer una relación clara y directa entre los casos de uso y los casos de prueba para facilitar que el proceso de aseguramiento de la calidad del software se ejecute adecuadamente.

El plan de pruebas debe planearse en esta fase, ejecutarse desde la primera iteración de la fase de elaboración y refinarse sucesivamente durante el ciclo de vida del proyecto.

3.2.4.1.2 Fase de elaboración

Los casos de uso seleccionados para desarrollarse en esta fase permiten definir la arquitectura del sistema, se realiza la especificación de los casos de uso seleccionados y el primer análisis del dominio del problema, se diseña la solución preliminar del problema y comienza la ejecución del plan de manejo de riesgos, según las prioridades definidas en él.

Al final de la fase se determina la viabilidad de continuar el proyecto y si se decide proseguir, dado que la mayor parte de los riesgos han sido mitigados, se escriben los planes de trabajo de las etapas de construcción y transición y se detalla el plan de trabajo de la primera iteración de la fase de construcción.

3.2.4.2 Etapa de producción

En esta etapa se realiza un proceso de refinamiento de las estimaciones de tiempos y recursos para las fases de construcción y transición, se define un plan de mantenimiento para los productos entregados en la etapa de ingeniería, se implementan los casos de uso pendientes y se entrega el producto al cliente, garantizando la capacitación y el soporte adecuados.

3.2.4.2.1 Fase de construcción

El propósito de esta fase es completar la funcionalidad del sistema, para ello se deben clarificar los requerimientos pendientes, administrar el cambio de los artefactos construidos, ejecutar el plan de administración de recursos y mejoras en el proceso de desarrollo para el proyecto.

3.2.4.2.2 Fase de transición

El propósito de esta fase es asegurar que el software esté disponible para los usuarios finales, ajustar los errores y defectos encontrados, capacitar a los usuarios y proveer el soporte técnico necesario. Se debe verificar que el producto cumpla con las especificaciones entregadas por las personas involucradas en el proyecto al inicio del mismo.

3.2.5 MODELADO DEL NEGOCIO Y REQUERIMIENTOS

3.2.5.1 PRESENTACIÓN GENERAL

Con el advenimiento de nuevas aplicaciones orientadas a la Web se está ganando cada vez más espacio en el ámbito del desarrollo y estas se han convertido en la mejor manera de darse a conocer e informar acerca de las actividades de todo

tipo de empresas las cuales tienen la oportunidad de ofrecer sus servicios a través de cualquier máquina conectada a Internet.

Gracias al Internet se hace posible que las personas puedan tener una participación directa con los servicios e información que presta cualquier empresa o institución por medio de una página.

En este entorno muchas empresas del país no poseen un sistema que permita dar a conocer información concerniente a la misma, ni permitir al personal interactuar tanto con la información almacenada en sus bases de datos. Entonces surge la necesidad de construir un sistema en línea para poder apoyar la labor de los administradores de bases de datos y usuarios finales en las tareas de obtener información de varias bases de datos, tomando en cuenta el ahorro en tiempo y recursos que implica la generación de la información pertinente.

3.2.5.2 CLIENTES

Los principales beneficiarios de esta solución son:

- Los administradores de bases de datos ya que pueden contar con una herramienta que les permita tener acceso a su información en una sola aplicación.
- El usuario final podrá acceder a la información de acuerdo a un nivel de seguridad.
- La parte gerencial contaran con información en menor tiempo.
- Además tendrán la posibilidad de verificar desde cualquier parte mediante Internet utilizando como herramienta de consulta la pantalla de un navegador.

3.2.5.3 METAS

En términos generales, la meta es la automatización del la consulta de información almacenada en varias bases de datos, dar soporte a servicios más ágiles, eficientes, eficaces, más baratos y mejores, los cuales esta dirigidos a los procesos de las instituciones o empresas.

Para ser más concretos, la meta incluye:

- Servicio más eficiente a los clientes
- Consulta de línea de la información generada
- Análisis rápido y exacto de los datos

3.2.5.4 FUNCIONES DEL SISTEMA

Las funciones del sistema describen lo que éste habrá de hacer; hay que listarlas en grupos cohesivos y lógicos.

3.2.5.4.1 Categorías de las funciones

Las funciones deben ser clasificadas con el fin de establecer las prioridades entre ellas e identificar las que son prioritarias para el desarrollo del sistema. Las categorías son:

Evidente.- Debe realizarse, el usuario debe saber que se ha realizado.
Oculta.- Debe realizarse, no es visible para el usuario pero debe realizarse.
Superflua.- No repercute significativamente en otras funciones del sistema.

3.2.5.4.2 Clasificación de las funciones

La clasificación de las funciones del sistema es la siguiente:

- GESTIÓN GENERAL DEL SISTEMA

Tabla 3.9 Funciones de la Gestión General del Sistema

REF #	FUNCIÓN	CATEGORÍA
R0.1	Ingreso de información de usuarios del sistema.	Evidente
R0.2	Ofrecer un mecanismo de almacenamiento persistente	Oculta
R0.3	Ofrece mecanismos de comunicación entre procesos y sistemas	Oculta

Elaborado por: Autor

3.2.5.5 ATRIBUTOS DEL SISTEMA

Los atributos del sistema son sus características o dimensiones (no funcionales) las cuales pueden abarcar todas las funciones o ser específicas de una función o grupo de funciones.

Estos atributos pueden tener un conjunto de *detalles de atributos* e indican que pueden llegar a tener valores discretos, confusos o simbólicos. Además, algunos atributos del sistema pueden tener *restricciones de frontera* que generalmente puede ser un rango numérico de valores.

Tabla 3.10 Atributos del Sistema

REF #	FUNCIÓN	CAT.	ATRIBUTO	DETALLES Y RESTRICCIONES	CAT.
R0.1	Ingreso de información de usuarios del sistema	Evidente	Tiempo de respuesta	El tiempo de respuesta a las peticiones del usuario no debe ser mayor a	**Obligatorio**

				los cinco segundos	
			Metáfora de interfaz	Ventanas orientadas a la metáfora de una forma y cuadros de diálogo. Ingreso de datos encriptados Colorido	**Obligatorio** **Obligatorio** **Opcional**
R0.2	Ofrecer un mecanismo de consulta persistente	Oculta	Tolerancia a fallas	La base de datos se encargara de controlar cuando se produzcan fallas de energía o del equipo	**Obligatorio**
			Tiempo de respuesta	El tiempo de respuesta a las peticiones del usuario no debe ser mayor a los cinco segundos	**Obligatorio**
R0.3	Ofrece mecanismos de comunicación entre procesos y sistemas	Oculta	Tolerancia a fallas	La base de datos se encargara de controlar cuando se produzcan fallas de energía o del equipo	**Obligatorio**
			Tiempo de respuesta	El tiempo de respuesta a las peticiones del usuario no debe ser mayor a los cinco segundos	**Obligatorio**
R1.1	Consultas de registros	Evidente	Tiempo de respuesta	El tiempo de respuesta a las peticiones del usuario no debe ser mayor a los cinco segundos	**Obligatorio**
			Metáfora de	Ventanas orientadas	**Obligatorio**

			interfaz	a la metáfora de una forma y cuadros de diálogo. Colorido	**Opcional**

Elaborado por: Autor

3.2.5.6 PLAN DE RIESGOS

En el presente documento se presenta un listado de riesgos que pueden afectar al normal desenvolvimiento del proyecto y que deben ser debidamente tratados a tiempo y durante todo el desarrollo del proyecto para irlos atenuando y que no se vayan a convertir en factores categóricos para el fracaso del proyecto.

3.2.5.6.1 Identificación de Riesgos

Vamos a clasificar los riesgos en diferentes categorías para tener un mejor control de los mismos y estas son:

- Riesgos de tecnologías
- Riesgos de personas
- Riesgos de requerimientos
- Riesgos de estimación.

En el cuadro siguiente detallamos cada uno de los riesgos que podemos tener en este proyecto debidamente clasificado.

Tabla 3.11 Identificación de Riesgos Posibles

Tipo de riesgo	Riesgos Posibles
Tecnología	Utilizar componentes de software reutilizables con la probabilidad que no se ajusten a la funcionalidad del sistema. Imposibilidad de uso de MySql para el almacenamiento de los

	datos específicos de la aplicación. Rendimiento inferior al previsto de la base de datos. Imposibilidad de utilizar la herramienta de desarrollo, o rendimiento menor al previsto. Desconocimiento de las herramientas por parte del responsable del proyecto.
Personas	Limitación de tiempo por estar dedicado a labores propias del trabajo cotidiano. No ser experto en las herramientas de desarrollo. Imposibilidad de realización de pruebas en una determinada institución.
Requerimientos	Imposibilidad de reunirse con personas expertas en el tema de aplicación.
Estimación	El tamaño del software es subestimado. Subestimación de tiempos para cumplir con cada una de las etapas del proceso ágil de desarrollo. No se dispone del tiempo necesario para la culminación del proyecto.

Elaborado por: Autor

3.2.5.6.2 Análisis de los Riesgos

Los efectos del riesgo pueden ser:

- Catastróficos
- Serios
- Tolerables

- Insignificantes

Los cuales dependen de su incidencia, pero los hemos clasificado también por la probabilidad de que ocurran, para tratarlos a tiempo y tenerlos debidamente identificados para su posterior tratamiento:

Tabla 3.12 Identificación de Probabilidad y Efectos de Riesgos

Riesgo	Probabilidad	Efectos
Utilizar componentes de software reutilizables con la probabilidad que no se ajusten a la funcionalidad del sistema.	Moderada	Serio
Imposibilidad de uso de MySql, para el almacenamiento de los datos específicos de la aplicación.	Muy Baja	Serio
Rendimiento inferior al previsto de la base de datos.	Moderada	Serio
Imposibilidad de utilizar la herramienta de desarrollo, o rendimiento menor al previsto.	Baja	Tolerable
Desconocimiento de las herramientas por parte del responsable del proyecto.	Moderada	Catastrófico
Limitación de tiempo por estar dedicado a labores propias del trabajo cotidiano.	Alta	Tolerable
No ser experto en las herramientas de desarrollo.	Alta	Tolerable
Imposibilidad de realización de pruebas en una determinada institución.	Baja	Serio
Imposibilidad de reunirse con personas expertas en el tema de aplicación.	Moderada	Serio
El tamaño del software es subestimado.	Alta	Tolerable
Subestimación de tiempos para cumplir con cada una de las etapas del proceso unificado	Alta	Serio
No se dispone del tiempo necesario para la	Baja	Catastrófico

culminación del proyecto.		

Elaborado por: Autor

3.2.5.6.3 Manejo de Riesgos

A continuación tomamos en cuenta cada riesgo previamente analizado y desarrollamos una estrategia para manejar el mismo, para lo cual nos valemos de estrategias evasivas considerando que la probabilidad de que el riesgo surja es reducida, estrategias de minimización considerando que el impacto y el riesgo tendrán sobre el proyecto o el producto es reducida y los respectivos planes de contingencia si el riesgo surge efectivamente.

Tabla 3.13 Estrategias para el Manejo de Riesgos

Riesgo	Estrategia
Utilizar componentes de software reutilizables con la probabilidad que no se ajusten a la funcionalidad del sistema.	Reemplazar componentes incompatibles con componentes pre-elaborados de confiabilidad conocida
Imposibilidad de uso de MySql para el almacenamiento de los datos específicos de la aplicación.	Analizar la posibilidad de utilizar PostgreSQL que es una base de datos que brinda las mismas prestaciones y también es software libre.
Rendimiento inferior al previsto de la base de datos.	Utilizar PostgreSQL como una alternativa ante el pobre rendimiento que podría presentar MySql.
Imposibilidad de utilizar la herramienta de desarrollo, o rendimiento menor al previsto por tratarse de herramientas de software libre.	Hay muchas herramientas de software libre para desarrollo que se podrían utilizar en forma alternativa de ser necesario.
Desconocimiento de las herramientas	Realizar cursos intensivos de las

por parte del responsable del proyecto.	herramientas seleccionadas para el desarrollo de este proyecto, existen muchas opciones de capacitación en la actualidad.

Elaborador por: Autor

3.2.6 IDENTIFICACION DE ACTORES

Un actor es una entidad externa del sistema que de alguna manera participa en el caso de uso y que por lo regular estimula al sistema con eventos de entrada o recibe algo de él. Los actores están representados por el papel que desempeña en el caso.

Estos son los actores que intervienen en el desarrollo del presente trabajo

Tabla 3.14 Actores del Sistema

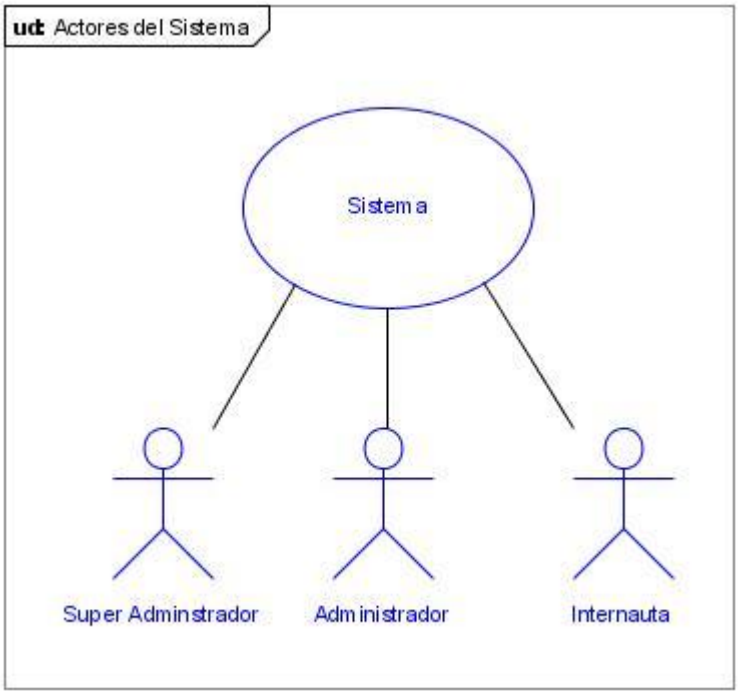
#	Actor	Descripción
1	Súper Administrador	Al administrador se le ha establecido una página exclusiva, se identificará por medio de un usuario y una clave establecida, con la que accederá a la aplicación como tal, además, tiene acceso a todos los servicios del sistema, inclusive al mantenimiento de la aplicación, perfiles y usuarios.
2	Administrador	Se identificará por medio de un usuario y una clave establecida. Este usuario tiene acceso sólo a los servicios de definición de opciones de menú y submenú y definición de formularios por submenú; los cuales permitirán la estructuración de la información y la estructuración de los contenidos de las opciones de submenú.

3	Internauta	Se identificará por medio de un usuario y una clave establecida y podrá acceder a las partes públicas del sistema.

Elaborado por: Autor

En un caso de uso hay un actor iniciador que produce la estimulación inicial y, posiblemente, otros actores participantes; tal vez convenga indicar quién es el iniciador, en la figura 3.2 veremos los actores del sistema.

Fig. 3.3 Actores del Sistema



Elaborado por: Autor

3.2.6.1 Identificación de Casos de Uso

Para poder entender en parte los requerimientos del sistema se debe conocer los procesos del dominio de estudio y los factores externos que participan en los procesos, y estos procesos los podemos expresar en Casos de Uso los cuales describimos a continuación:

Tabla 3.15 Casos de Uso

#	Nombre	Descripción
1	Ingresar Usuarios	Consiste en el ingreso de los usuarios al sistema con sus respectivas claves de acceso y privilegios.
2	Actualizar Usuarios	Consiste en la modificación de la información del usuario del sistema de sus respectivas claves de acceso y privilegios.
3	Borrar Usuarios	Consiste en la eliminación de la información de usuarios del sistema.
4	Control de Acceso	Ingreso al sistema de los usuarios a través de una autentificación.
5	Crear Consultas	Consiste en la creación de un consulta con los datos requeridos.
6	Actualizar Consultas	Consiste en la modificación de la información de la consulta que se va a presentar
7	Borrar Consultas	Consiste en la eliminación de la consulta previamente creada
8	Editar Consulta	Consiste en mostrar como esta construida la consulta y de donde provienen los datos.

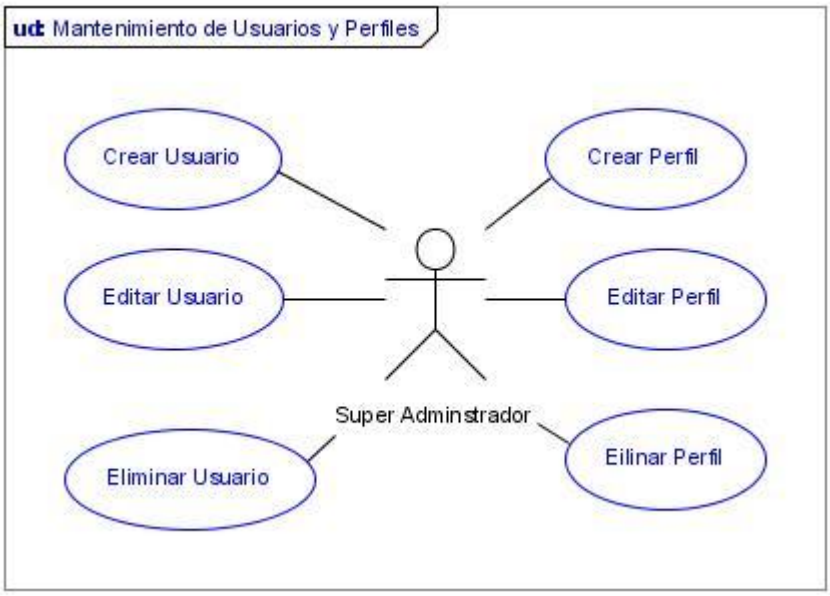
Elaborador por: Autor

3.2.6.2 Modelo de Casos de Uso

El modelo de casos de uso tiene por objeto ofrecer un tipo de diagrama contextual que nos permite conocer rápidamente los actores externos de un sistema y las formas básicas en que lo utilizan, a continuación se presentan los casos de uso del sistema.

La Fig. 3.4 ilustra el Modelo de los Casos de Uso del Sistema Mantenimiento de Usuarios y Perfiles.

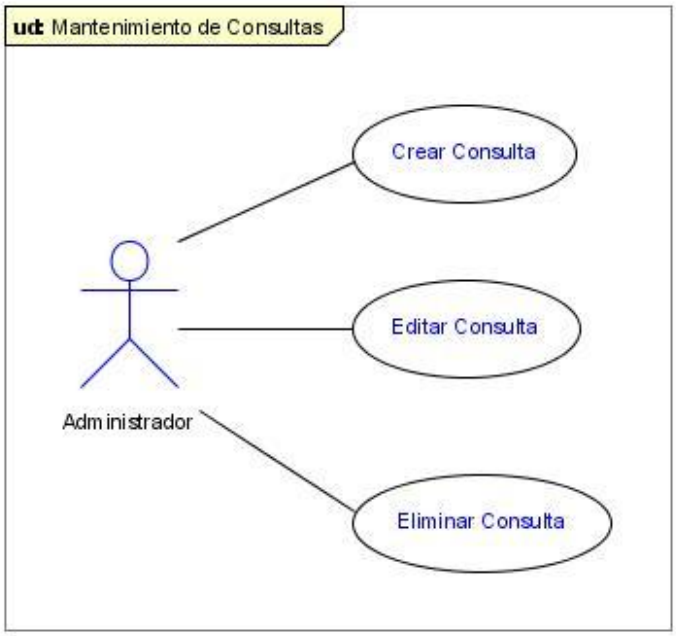
Fig. 3.4 Modelo de los Casos de Uso Mantenimiento de Usuarios y Perfiles



Elaborador por: Autor

La Fig. 3.5 ilustra el Modelo de los Casos de Uso del Sistema Mantenimiento de Consultas.

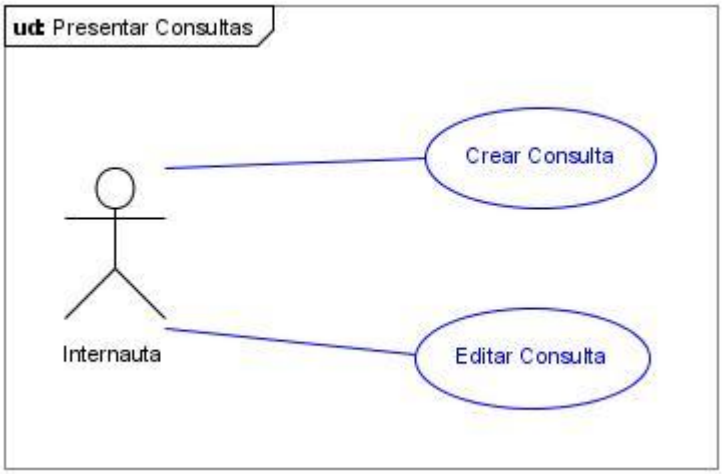
Fig. 3.5 Modelo de los Casos de Uso Mantenimiento de Consultas



Elaborado por: Autor

La Fig. 3.6 ilustra el Modelo de los Casos de Uso para Presentar Consultas

Fig. 3.6 Modelo de los Casos de Uso Presentar Consultas



Elaborado por: Autor

3.2.6.3 Especificación de Casos de Uso

Para poder entender los requerimientos se necesita, en parte, conocer los procesos del dominio y los factores externos que participan en los procesos, estos procesos de dominio se los expresan en los casos de uso.

Los casos de uso se limitan a describir los procesos los cuales constituyen un paso muy útil porque reseñan las especificaciones del sistema.

En las tablas siguientes se describen detalladamente la especificación de cada caso de uso del sistema:

Tabla 3.16 Caso de Uso Crear Usuarios

Nombre del caso de uso	Crear Usuarios
Actor Principal	Súper Administrador

Versión	Final
Actores involucrados	Administrador
Objetivos	Registrar información de los usuarios en la base de datos
Precondición	No estar registrado en la bases de datos de usuarios
Secuencia normal	1.A El administrador realiza el control de acceso 2.S El sistema muestra la página de menú del administrador 3.A El administrador selecciona la opción usuarios 4.S El sistema presenta un formulario para el ingreso de usuarios en blanco 5.A El administrador llena la información del usuario y selecciona la opción de guardar 6. S El sistema valida que la información esta bien ingresada, guarda los datos en la base y presenta el formulario en blanco.
Excepciones	6.1.S Si se a ingresado información errónea el sistema presenta un mensaje de error y permite que se vuelva a corregir los datos
Frecuencia Esperada	Cada vez que se requiera ingresar un nuevo usuario al sistema

Elaborado por: Autor

Tabla 3.17 Caso de Uso Editar Usuarios

Nombre del caso de uso	**Editar Usuarios**
Actor Principal	Súper Administrador
Versión	Final
Actores involucrados	Administrador
Objetivos	Modificar la información de los usuarios en la base de datos
Precondición	El usuario debe estar registrado en la bases de datos de usuarios
Secuencia normal	1.A El administrador realiza el control de acceso 2.S El sistema muestra la página de menú del administrador

	3.A El administrador selecciona la opción usuarios 4.S El sistema presenta un formulario para el ingreso de usuarios 5.A El administrador selecciona la opción buscar en la barra de herramientas 6.S El sistema presenta un dialogo para ingresar el criterio de búsqueda del usuario 7.A El administrador ingresa el criterio de búsqueda y selecciona la opción aceptar 8.S El sistema presenta un dialogo con el listado de todos los usuarios que coinciden con el criterio de búsqueda ingresado 9.A El administrador selecciona el usuario que busca del listado 10.S El sistema presenta los datos del usuario seleccionado en el formulario de ingreso de usuarios 11.A El administrador modifica la información del usuario y selecciona la opción grabar 12.S El sistema valida que la información esta bien ingresada, guarda los datos en la base y presenta el formulario en blanco.
Excepciones	8.1.S Si el criterio de búsqueda no encuentra ninguna coincidencia, aparece el dialogo de usuario en blanco
Frecuencia Esperada	Cada vez que se requiera modificar la información de un usuario del sistema

Elaborado por: Autor

Tabla 3.18 Caso de Uso Eliminar Usuarios

Nombre del caso de uso	**Eliminar Usuarios**
Actor Principal	Súper Administrador
Versión	Final

Actores involucrados	Administrador
Objetivos	Borrar la información de los usuarios en la base de datos
Precondición	El usuario debe estar registrado en la bases de datos de usuarios
Secuencia normal	1.A El administrador realiza el control de acceso 2.S El sistema muestra la página de menú del administrador 3.A El administrador selecciona la opción usuarios 4.S El sistema presenta un formulario para el ingreso de usuarios 5.A El administrador selecciona la opción buscar en la barra de herramientas 6.S El sistema presenta un dialogo para ingresar el criterio de búsqueda del usuario 7.A El administrador ingresa el criterio de búsqueda y selecciona la opción aceptar 8.S El sistema presenta un dialogo con el listado de todos los usuarios que coinciden con el criterio de búsqueda ingresado 9.A El administrador selecciona el usuario que busca del listado 10.S El sistema presenta los datos del usuario seleccionado en el formulario de ingreso de usuarios 11.A El administrador selecciona la opción borrar en la barra de herramientas 12. S El sistema borra la información en la base de datos y presenta el formulario en blanco.
Excepciones	8.1.S Si el criterio de búsqueda no encuentra ninguna coincidencia, aparece el dialogo de usuario en blanco
Frecuencia Esperada	Cada vez que se requiera borrar la información de un usuario del sistema

Elaborado por: Autor

Tabla 3.19 Caso de Uso Control de Acceso

Nombre del caso de uso	Control de Acceso
Actor Principal	Administrador
Versión	Final
Actores involucrados	Internauta
Objetivos	Restringir el acceso al sistema a los usuarios y que pueden acceder a los diferentes módulos del sistema
Precondición	El usuario debe estar registrado en la bases de datos de usuarios con su respectivo nombre, clave de acceso y privilegio
Secuencia normal	1.A Los usuarios activan la página de inicio del sistema y seleccionan la opción del menú principal 2.S El sistema muestra un formulario donde se solicita el nombre de usuario y clave de acceso 3.A El usuario ingresa su nombre de usuario y clave de acceso 4.S El sistema verifica que sea una cuenta de usuario valida 5.A Si es una cuenta de valida se presenta la página de menú personalizada para el usuario
Excepciones	5.1.S Si no es una cuenta valida el sistema presenta un mensaje de acceso no valido
Frecuencia Esperada	Cada vez que un usuario acceda al sistema

Elaborado por: Autor

Tabla 3.20 Caso de Uso Crear Consultas

Nombre del caso de uso	Crear Consultas
Actor Principal	Administrador
Versión	Final
Actores involucrados	Internauta

Objetivos	Crear las consultas necesarias que requiera la institución
Precondición	Tener las conexiones a las bases de datos necesarias para las consultas
Secuencia normal	1.A El administrador realiza el control de acceso 2.S El sistema muestra la página de menú personalizado para el administrador 3.A El administrador selecciona la opción de consultas 4.S El sistema presenta un panel para crear una sentencia SQL para seleccionar los datos a mostrar 5.A El administrador llena la información de la sentencia SQL y selecciona la opción de guardar 6. S El sistema valida que la información esta bien ingresada, guarda los datos en la base y presenta el formulario en blanco.
Excepciones	6.1.S Si se a ingresado información errónea el sistema presenta un mensaje de error y permite que se vuelva a corregir los datos
Frecuencia Esperada	Cada vez que se requiera generar una nueva consulta al sistema

Elaborado por: Autor

Tabla 3.21 Caso de Uso Editar Consultas

Nombre del caso de uso	**Editar Consultas**
Actor Principal	Administrador
Versión	Final
Actores involucrados	Internauta
Objetivos	Actualizar las consultas necesarias que requiera la institución
Precondición	La consulta debe estar registrado en la bases de datos de consultas
Secuencia normal	1.A El administrador realiza el control de acceso

	2.S El sistema muestra la página de menú personalizado para el administrador 3.A El administrador selecciona la opción de consultas 4.S El sistema presenta un panel para crear una sentencia SQL para seleccionar los datos a mostrar 5.A El administrador selecciona la consulta a ser actualizada y selecciona el botón de editar 6.S El sistema presenta los datos de la consulta seleccionada 7.A El administrador modifica la información de la sentencia SQL y selecciona la opción grabar 8. S El sistema valida que la información este bien ingresada, guarda los datos en la base.
Excepciones	5.1.S Si el criterio de búsqueda no encuentra ninguna coincidencia, aparece el dialogo de consulta en blanco
Frecuencia Esperada	Cada vez que se requiera modificar la información de una consulta en el sistema

Elaborado por: Autor

Tabla 3.22 Caso de Uso Eliminar Consulta

Nombre del caso de uso	**Eliminar Consultas**
Actor Principal	Administrador
Versión	Final
Actores involucrados	Internauta
Objetivos	Eliminar la información de las consultas que se generaron
Precondición	La consulta debe estar registrado en la bases de datos de consultas
Secuencia normal	1.A El administrador realiza el control de acceso 2.S El sistema muestra la página de menú del administrador 3.A El administrador selecciona la opción de consultas

	4.S El sistema presenta un panel para el ingreso de consultas y un listado de las consulta ya ingresadas 5.A El administrador selecciona la consulta que va a ser eliminada del listado y elige el botón borrar 6.S El sistema presenta un mensaje de alerta al usuario 7. S El sistema borra la información en la base de datos y presenta el menú inicial.
Excepciones	5.1.S Si el criterio de selección no encuentra ninguna coincidencia, aparece el dialogo en blanco
Frecuencia Esperada	Cada vez que se quiera eliminar los datos de una consulta

Elaborado por: Autor

Tabla 3.23 Caso de Uso Presentar Consulta

Nombre del caso de uso	**Presentar Consulta**
Actor Principal	Internauta
Versión	Final
Actores involucrados	Administrador
Objetivos	Verificar el contenido de la información de las consultas necesarias que requiera la institución
Precondición	La consulta debe estar registrado en la bases de datos de consultas
Secuencia normal	1.A El internauta realiza el control de acceso 2.S El sistema muestra la página de menú del internauta 3.A El internauta selecciona la opción consultas 4.S El sistema presenta un panel con un listado de las consultas ya creadas 5.A El internauta selecciona la consulta que va a ser generada del listado y elige el botón ver contenido 6.S El sistema presenta un mensaje de alerta al usuario

	7. S El sistema presenta el contenido de la información de la consulta.
Excepciones	5.1.S Si el criterio de selección no encuentra ninguna coincidencia, aparece el dialogo en blanco
Frecuencia Esperada	Cada vez que se quiera verificar el contenido de las consultas

Elaborado por: Autor

3.3 DISEÑO

3.3.1 ARQUITECTURA DEL SISTEMA

Tomando en cuenta las recomendaciones para el desarrollo de aplicaciones web, se ha dividido el sistema en n capas, solución que se ve apoyada por la infraestructura elegida para la aplicación:

- La capa de sesión/presentación incluirá subsistemas de control de peticiones (control de accesos, gestión de información enviada y distribución de peticiones).
- La capa del negocio incluirá un subsistema de servicio para la comunicación entre otros subsistemas (carga y recuperación de información de la base de datos)
- La capa de servicios un subsistema por cada servicio de usuario que se añada (generación de reportes y seguridad).
- La capa de acceso a datos estará compuesta por el subsistema de acceso a datos y los recursos de datos.

Entre las razones por las cuales se recurre a la arquitectura multicapas se cuentan las siguientes:

- El aislamiento de la lógica de aplicaciones en componentes independientes susceptibles de reutilización en otros sistemas.
- Distribución de las capas en varios nodos físicos de cómputo y en varios procesos.
- Las diferentes capas pueden ser diseñadas en forma independiente obteniendo la capacidad de realizar actividades simultáneas.

En la figura 3.7 podemos ver la arquitectura lógica de la aplicación Web que se va a implementar.

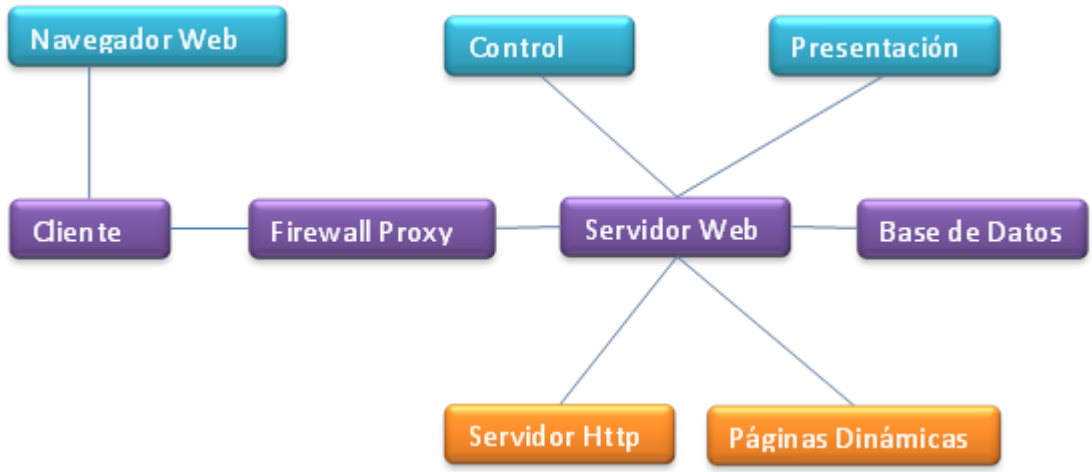
Fig. 3.7 Arquitectura lógica de la aplicación Web

Elaborado por: Autor

Para poder representar la topología de computadores y dispositivos sobre los que se ejecuta la aplicación que se está desarrollando, se va a utilizar el diagrama de despliegue (Ver figura 3.8). Con este diagrama se pretende detallar 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.

La siguiente figura muestra el diagrama de despliegue como una vista física de la arquitectura del sistema elaborado:

Fig. 3.8 Diagrama de Despliegue de la aplicación Web



Elaborado por: Autor

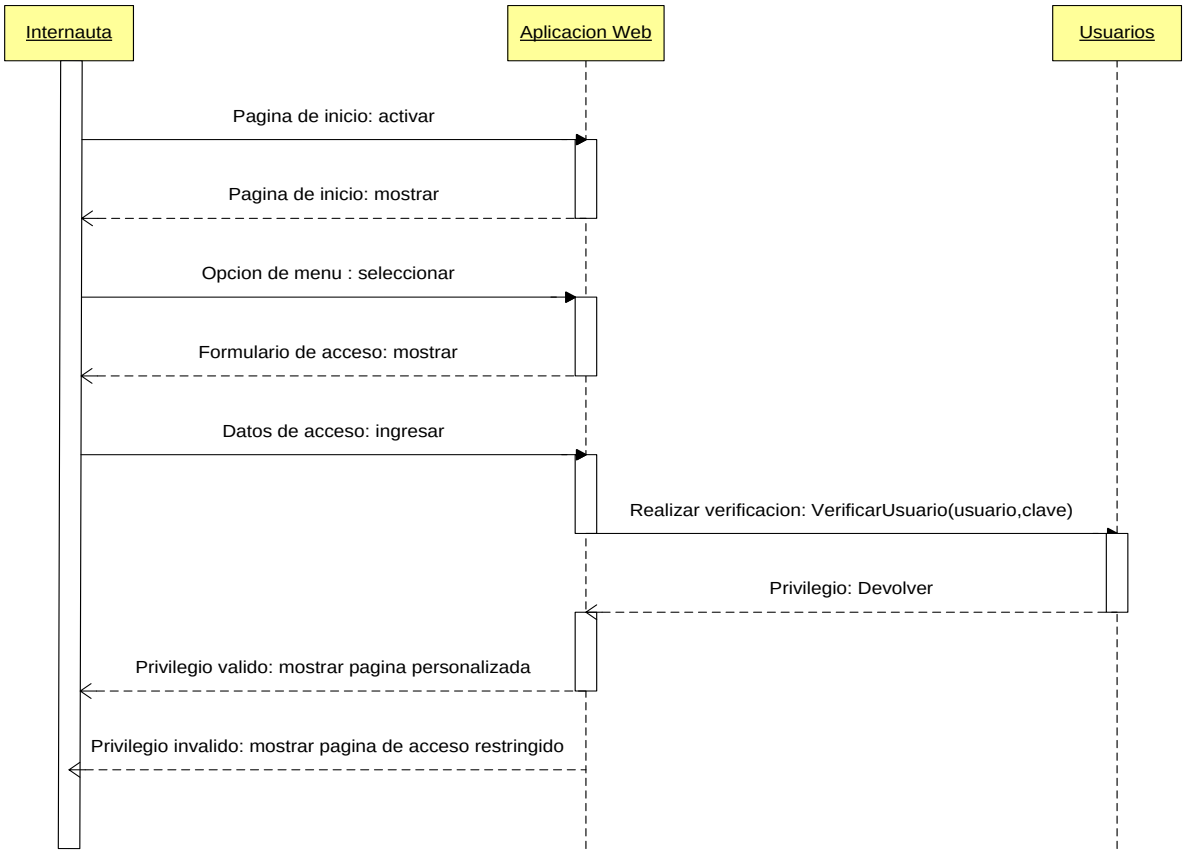
3.3.2 DIAGRAMAS DE SECUENCIA DEL SISTEMA

Los casos de uso indican cómo los actores interactúan con el sistema que es lo que en realidad deseamos crear. Durante la interacción un actor genera eventos dirigidos a un sistema, solicitando alguna operación de cambio, con el evento de esa petición se inicia una operación del sistema.

Es conveniente aislar y explicar gráficamente las operaciones que un actor solicita a un sistema, ya que de esta manera contribuye de manera importante a entender el comportamiento del sistema.

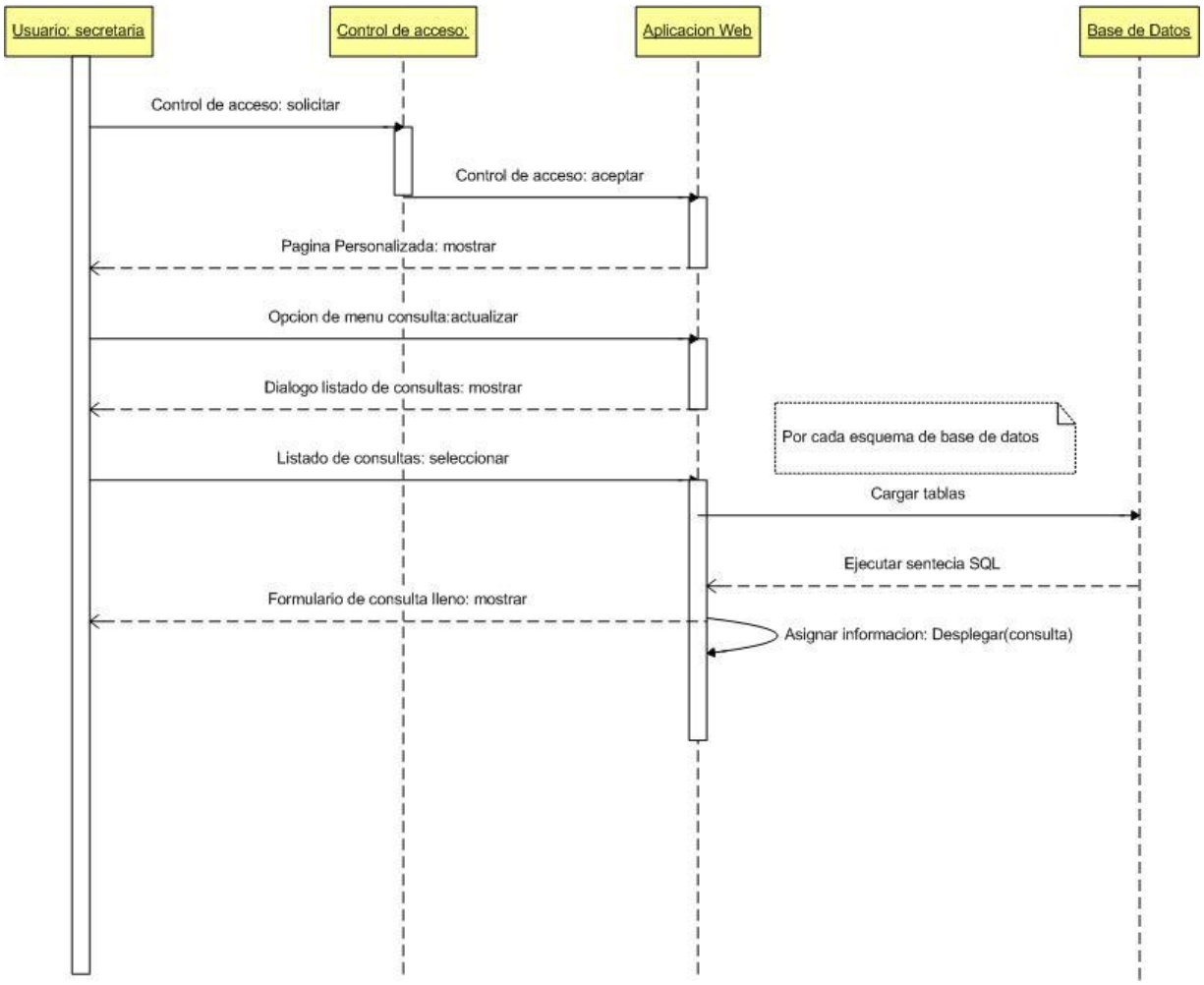
Los diagramas de secuencia son una representación que muestran, en determinado escenario de un caso de uso, los eventos generados por actores externos, su orden y los eventos internos del sistema

Fig. 3.9 Diagrama de secuencia Control de Acceso



Elaborado por: Autor

Fig. 3.10 Diagrama de secuencia Crear Consulta



Elaborado por: Autor

3.3.3 DIAGRAMAS DE ACTIVIDADES DEL SISTEMA

Fig. 3.11 Diagrama de Actividades Mantenimiento de Usuarios y Perfiles

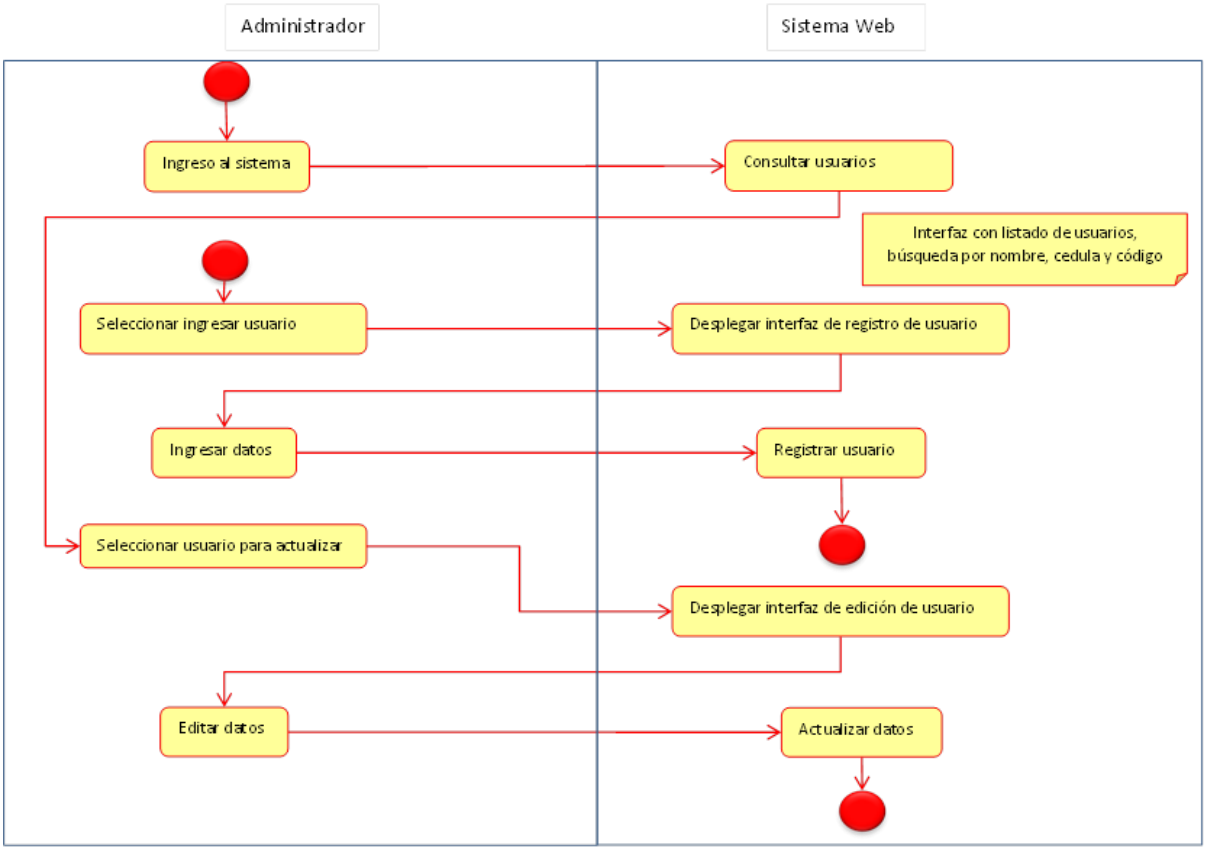


Fig. 3.122 Diagrama de Actividades iniciar sesión

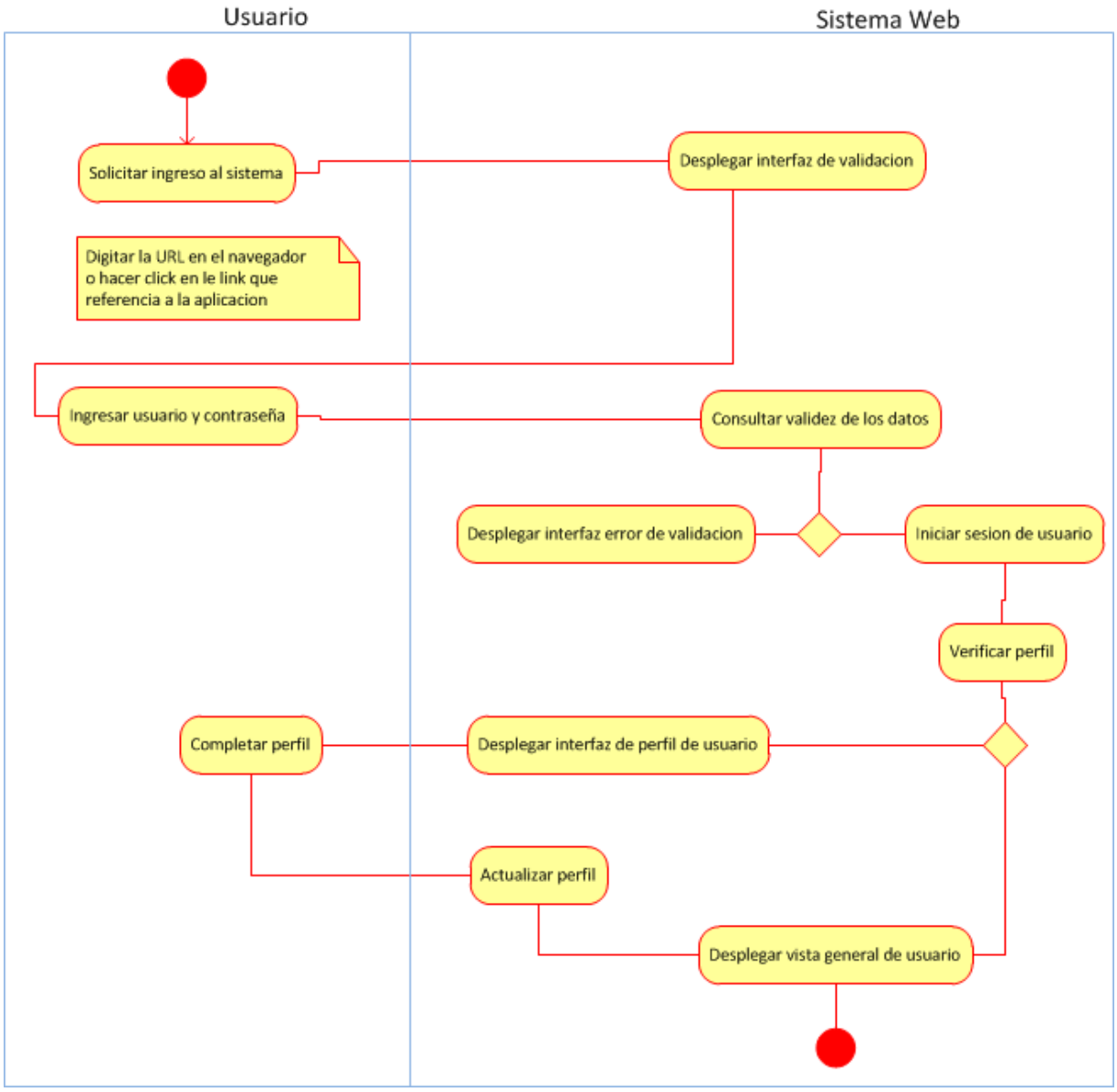


Fig. 3.133 Diagrama de Actividades terminar sesión

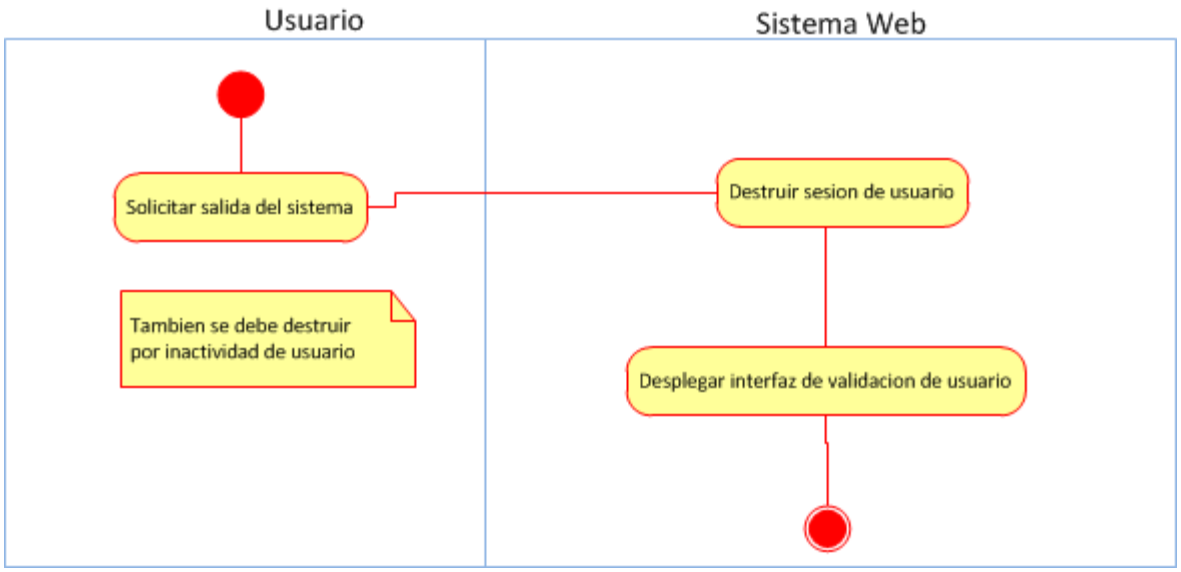


Fig. 3.144 Diagrama de Actividades cambiar clave

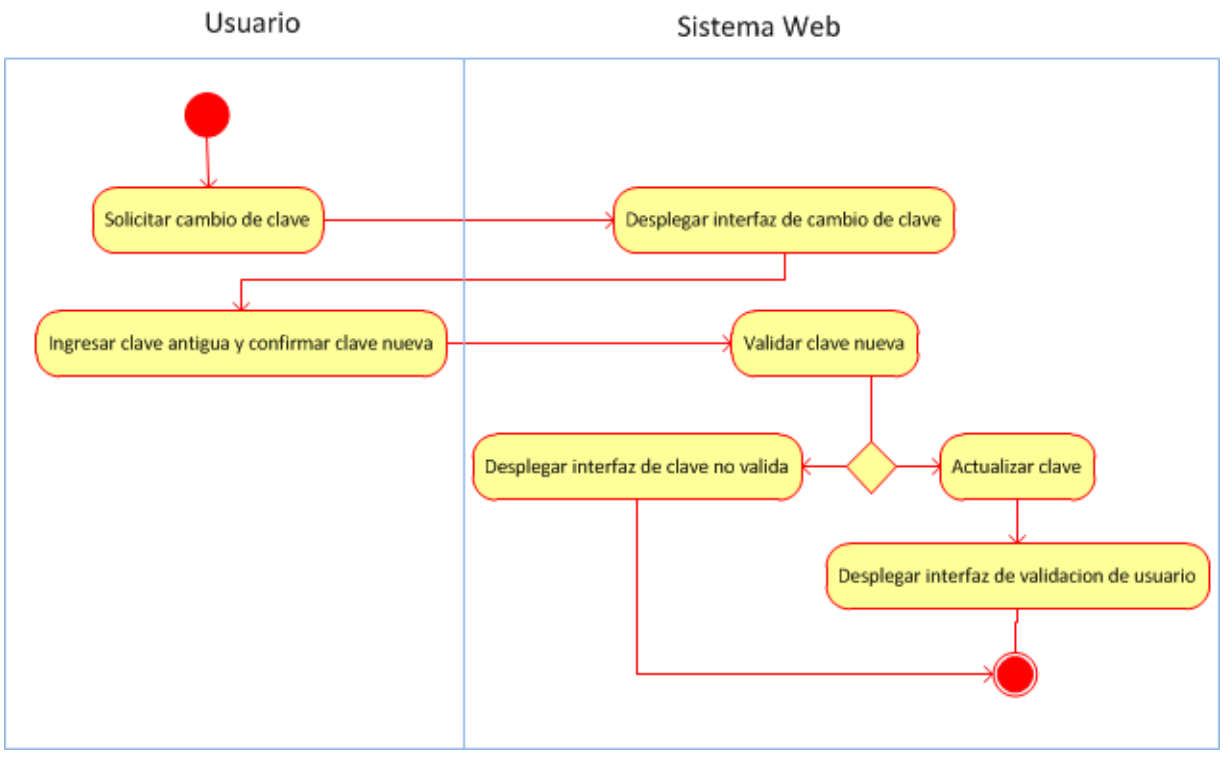


Fig. 3.155 Diagrama de Actividades realizar consulta

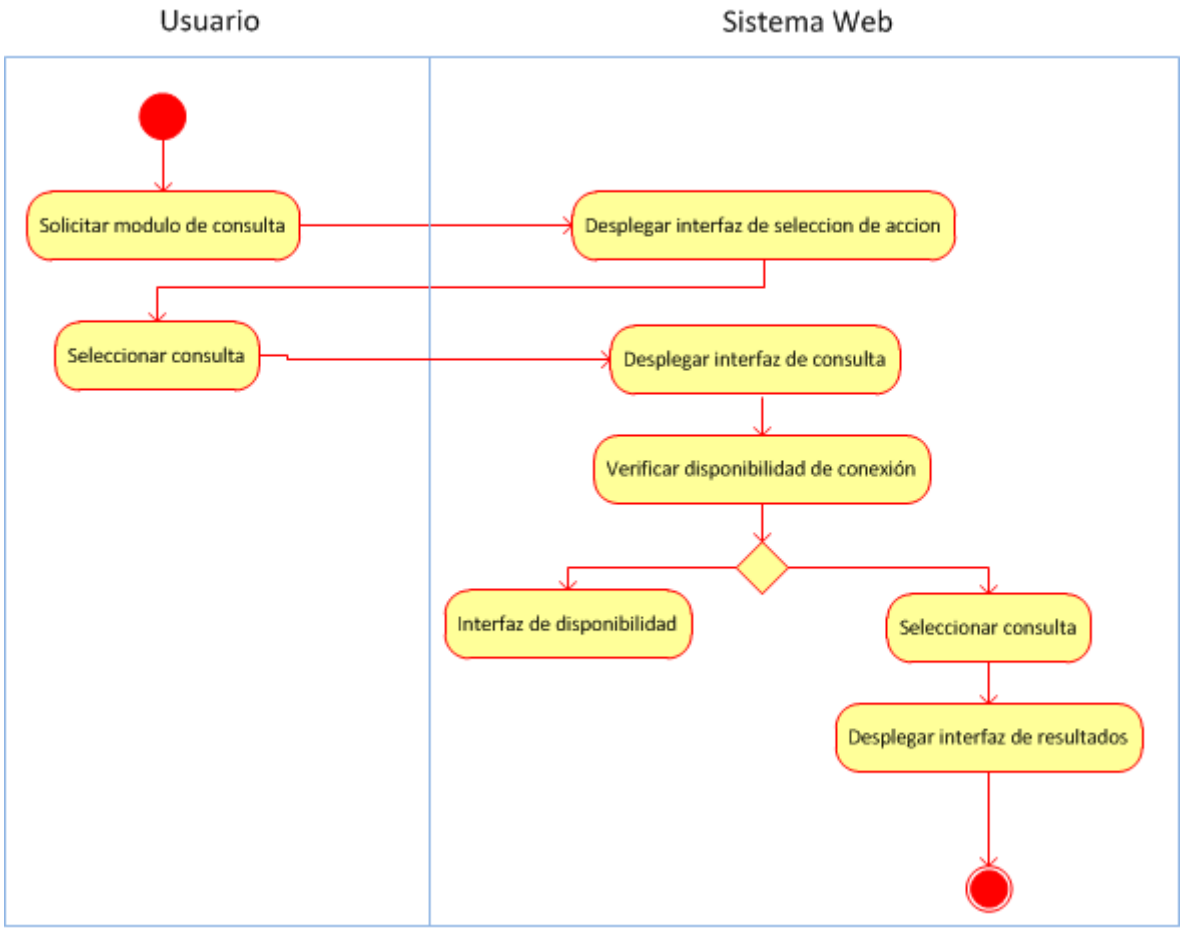
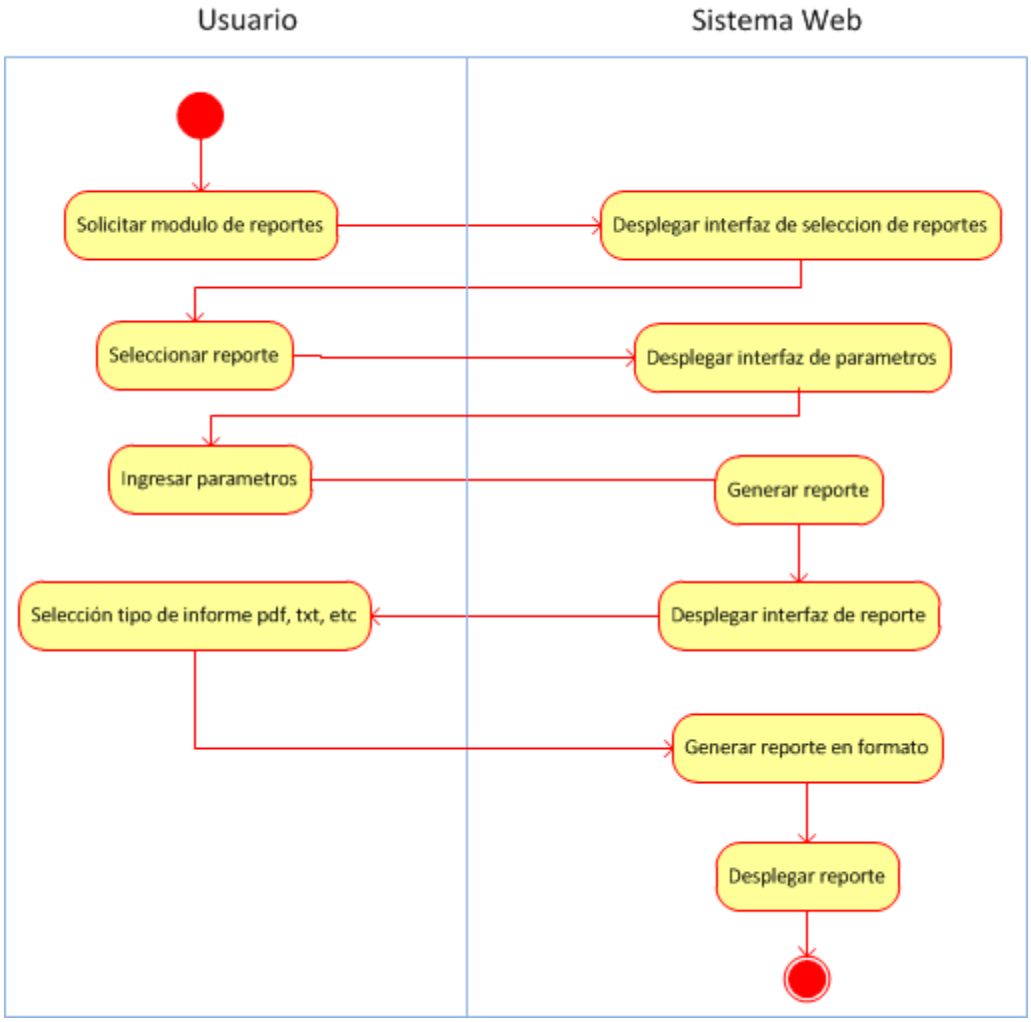


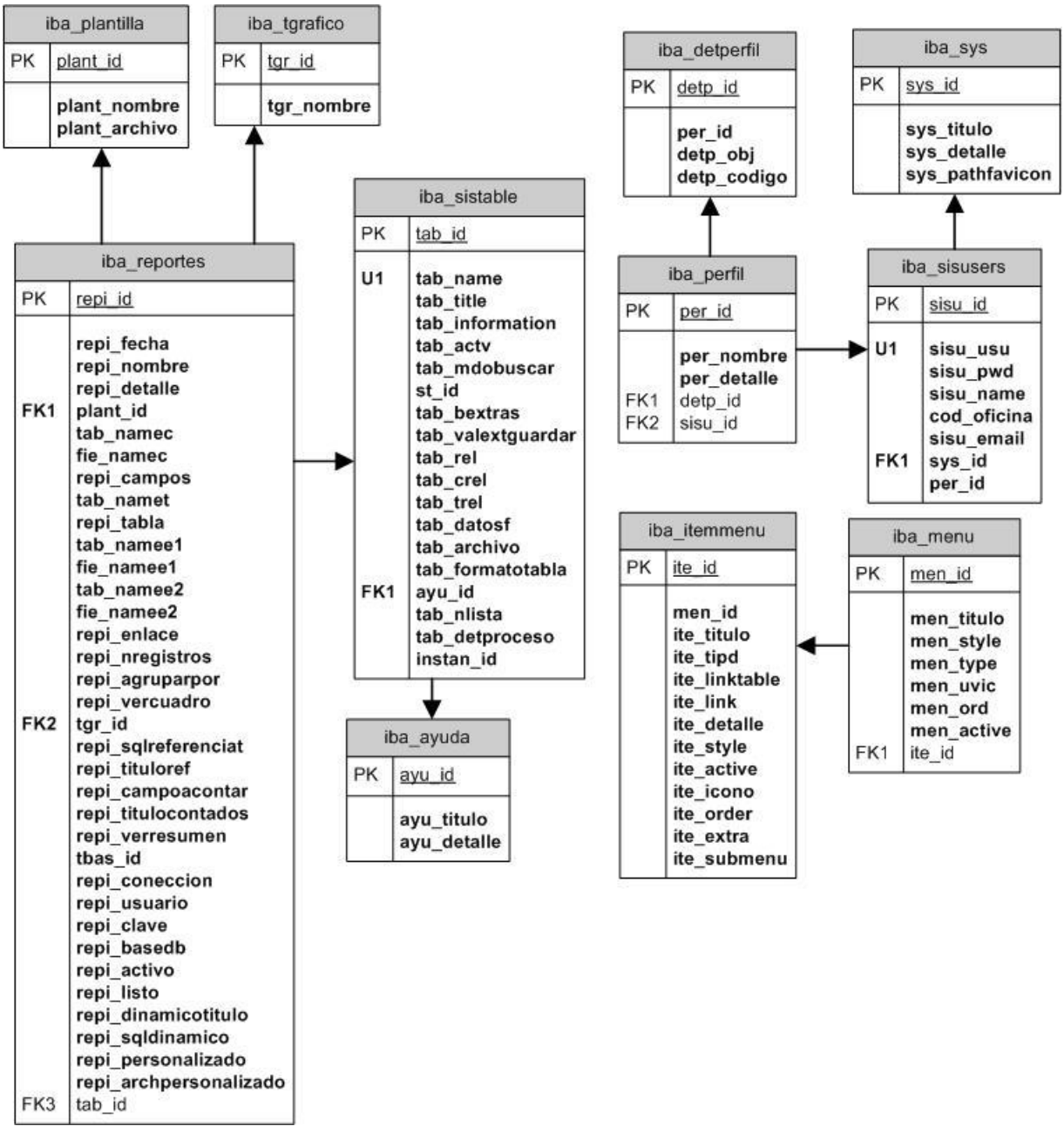
Fig. 3.166 Diagrama de Actividades generar reporte



3.3.4 DIAGRAMA DE CLASES DEL SISTEMA

Los diagramas de clases representan gráficamente las especificaciones de las clases de software y de las interfaces en la aplicación, es decir contiene las definiciones de las entidades del software que representan conceptos del mundo real.

Fig. 3.177 Diagrama de Clases



Elaborado por: Autor

3.4 IMPLEMENTACIÓN

3.4.1 HERRAMIENTAS DE IMPLEMENTACIÓN

Una vez que se a podido establecer el análisis y diseño de la aplicación se seleccionan las herramientas que se van a utilizar para la programación del mismo.

3.4.1.1 HTML

El HTML (Hyper Text MarkupLanguage) es un lenguaje de programación para crear documentos interactivos, es un lenguaje de marcas para la creación de hipertextos. El lenguaje HTML está basado en comandos que se insertan en lugares estratégicos en un texto ASCII.

Básicamente, el lenguaje HTML consta de una serie de órdenes, directivas o comandos, que indican al navegador que estamos utilizando, la forma de representar los elementos (texto, gráficos, etc.) que contenga el documento.

3.4.1.2 JavaScript

JavaScript es un lenguaje de programación para la creación de Scripts que se usa con HTML. JavaScript permite la interactividad con el usuario final y le da dinamismo a las páginas Web basadas en HTML. Este lenguaje se ejecuta desde el cliente de modo que no es necesario enviar datos al servidor para validarlos y después devolverlos al cliente.

Con JavaScript se puede crear botones de alerta, botones de mensajes, formularios, menús despleglables, etc.

3.4.1.3 DreamWeaver

Es un programa especializado en la edición y creación de páginas para Internet de sus características generales destacamos su facilidad en la administración y mantenimiento de Web site, como editor de documentos HTML, además permite adicionar una serie de elementos como mapa de imágenes, barras de navegación,

animaciones, Layers (manejo de capas), líneas de programación (scripts), menús de navegación, e-mail links.

Gracias a su sencillez de su entorno grafico y su familiaridad con otros editores HTML existentes en el mercado éste facilita la elaboración de páginas Web sin saber programar html, con mucha facilidad y con un aspecto profesional.

3.4.1.4 Apache

"El Servidor Apache HTTP es un servidor Web de tecnología Open Source sólido y para uso comercial, desarrollado por la Apache Software Foundation.

Apache es un sistema muy utilizado (actualmente es el servidor más utilizado en Internet). Normalmente se utiliza bajo un sistema Unix o Linux, pero existe un emulador para Windows, aunque este emulador no se le considera tan robusto como el apache de Unix.

La instalación es muy sencilla, lo que a menudo no suele ser tan sencillo es su configuración. La configuración del apache se hace desde el archivo "httpd.conf", que se encuentra en una subcarpeta dentro del directorio apache."[2]

3.4.1.5 PHP 5.x

PHP es un lenguaje que permite la generación dinámica de contenidos en un servidor web. Contiene muchos conceptos de C, Perl y Java, su sintaxis es muy similar a la de estos lenguajes, haciendo muy sencillo su aprendizaje. PHP esta embebido en documentos HTML de manera que es muy fácil incorporar información

[2] http://www.ciberteca.net/articulos/webmaster/phpwindows/apache.asp

actualizada en un sitio web, dispone de librerías de conexión con la mayoría de los sistemas de gestión de bases de datos.

3.4.1.6 MySql

Gestor de base de datos muy utilizado dentro de los entornos de desarrollo de sistemas Web con PHP, disponible bajo el estilo se software libre, accesibilidad al código fuente, funciona en las principales plataformas, rendimiento espectacular bajo determinadas condiciones, etc.

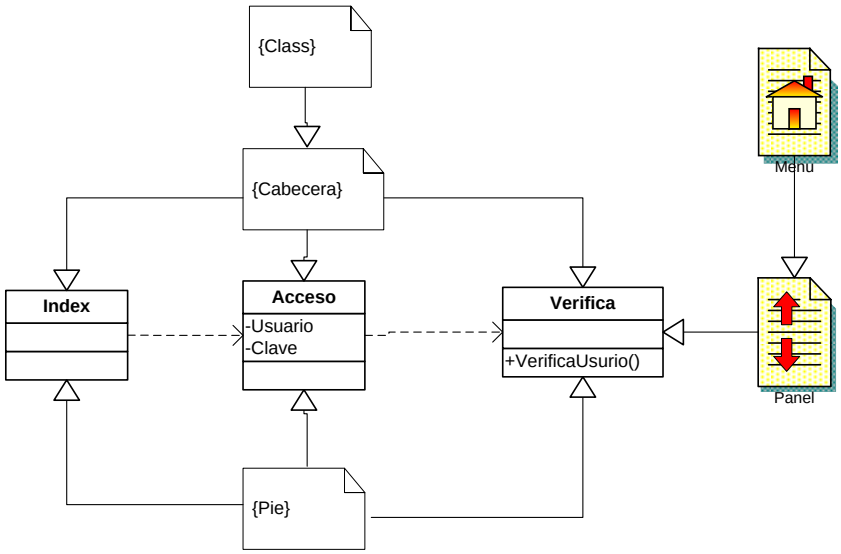
3.4.1.7 Poseidon

Adicionalmente a las herramientas de implementación se ha utilizado Poseidón for UML que es una herramienta general para modelar cualquier clase de sistema que precise programación orientada a objetos o incluso sistemas que no tengan nada que ver con software.

3.4.2 DIAGRAMA DE COMPONENTES

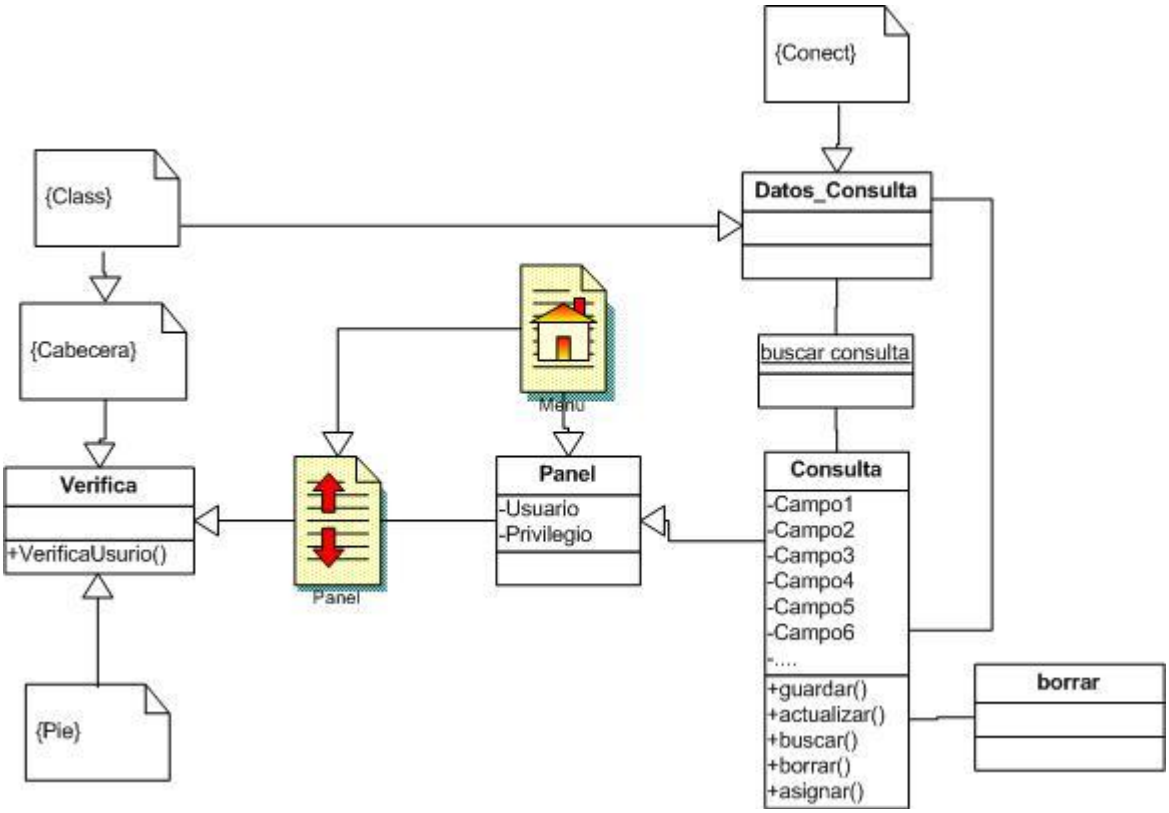
A través de un diagrama de componentes se describen los elementos y relaciones que aparecen en la página web. Con la ayuda de los caso de uso utilizando los elementos de la extensión para Aplicaciones Web de UML se muestran a continuación los diferentes de diagramas de componentes del sistema

Fig. 3.188 Diagrama de componentes Control de Acceso



Elaborado por: Autor

Fig. 3.199 Diagrama de componentes Crear Consulta



Elaborado por: Autor

3.4.3 CONVERSIÓN DE LAS BASES DE DATOS

3.4.3.1 Implementación de las Clases del Sistema

Una vez que se ha podido determinar la secuencia de las operaciones necesarias para la realización de los casos de uso, se procede a la implementación de las clases.

Las clases encontradas con sus respectivos atributos y métodos debidamente codificados las podemos encontrar en el Anexo No. 1, como una muestra de lo que son las otras clases.

3.4.3.2 Conexión con la Base de Datos

Para cubrir el acceso a la base de datos se utiliza una función denominada conexión, especialmente diseñada para conectar con MySql, que es la solución de PHP para la conexión con este tipo de bases de datos, la cual se la detalla a continuación.

```
//Funcion para establecer conexion
functionconexion()
{
    if (!($link = mysql_connect("localhost","root")))
            {
                echo "Error de conexión a la base de datos";
                        exit();
             }
             if (!mysql_select_db("cdarwin",$link))
            {
                 echo "Error en la base de datos";
                        exit();
             }
            return $link;



  }
```

3.5 PRUEBAS Y EVALUACION

El resultado principal del trabajo de pruebas es un modelo que describe qué hay que probar en el sistema, cómo se realizará el proceso de prueba y los mecanismos de pruebas automatizadas.

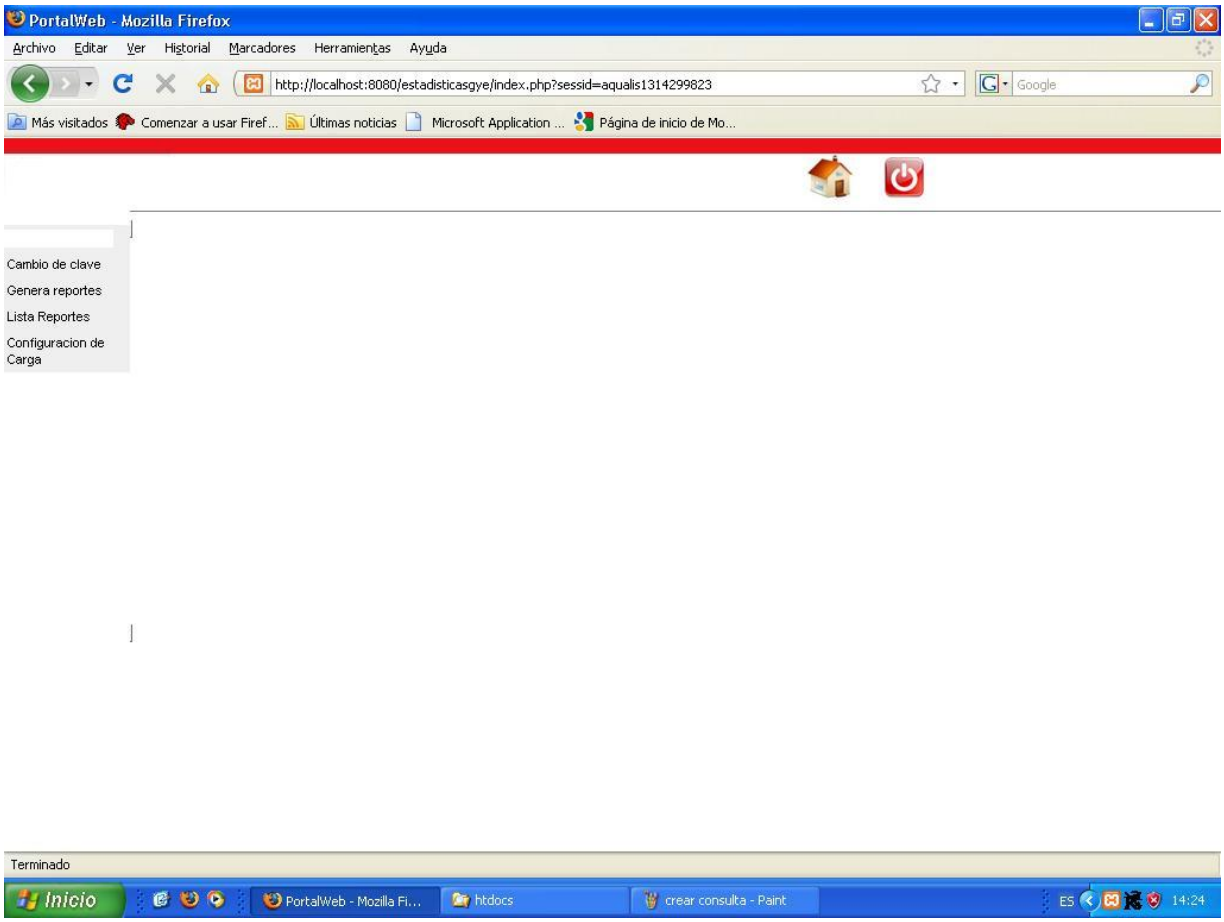
3.5.1 PRUEBAS DE UNIDAD

Las pruebas de unidad nos ayudan a validar las páginas implementadas como unidades individuales, con el objetivo de detectar todo posible mal funcionamiento mientras se va implementado la aplicación. En las siguientes figuras se hace mención al diseño de las mismas.

Fig. 3.20 Ingreso al Sistema

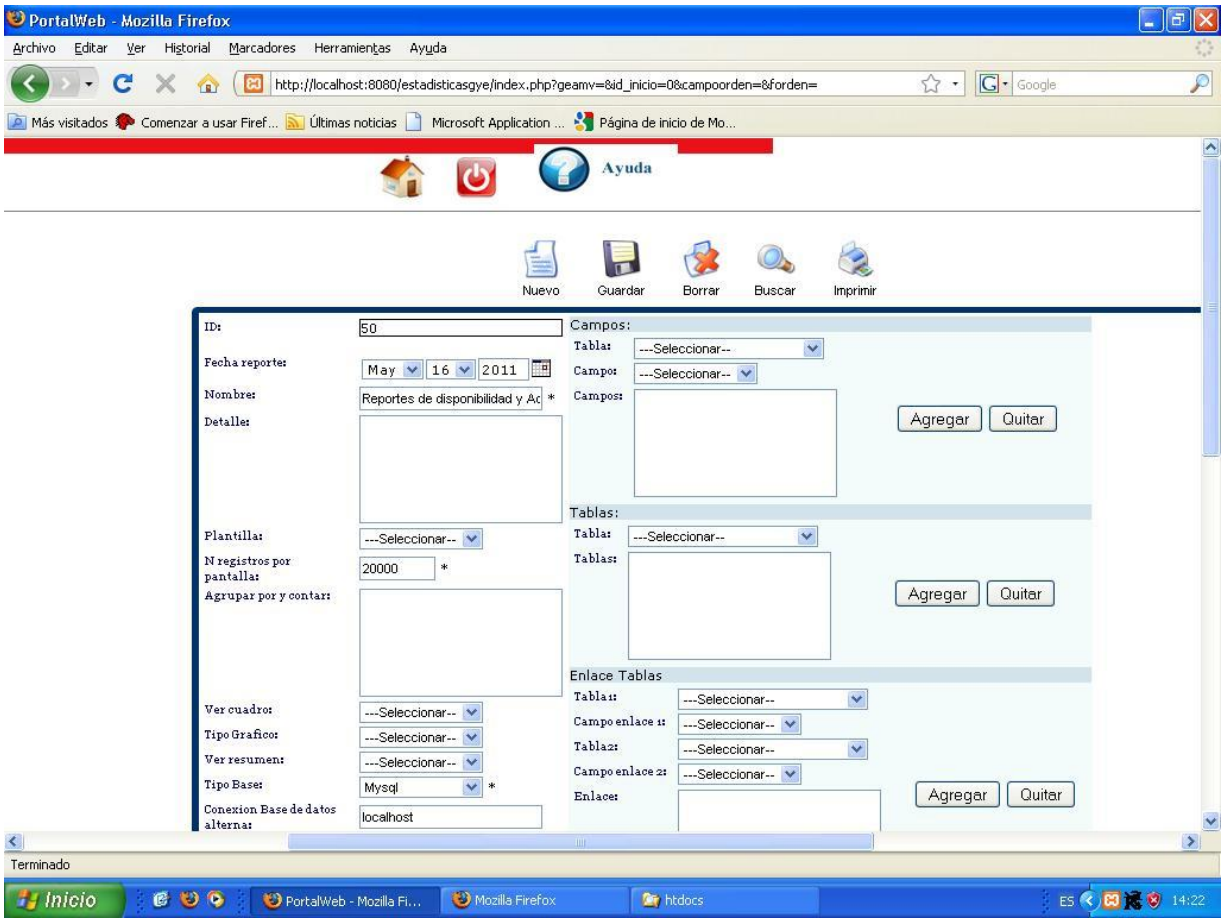
Elaborado por: Autor

Fig. 3.20 Menú General del Sistema



Elaborado por: Autor

Fig. 3.2120 Búsqueda de Consultas



Elaborado por: Autor

3.5.2 PRUEBAS DE INTEGRACIÓN

Pruebas de Integración (casos de uso). Se especificarán escenarios de prueba, consistentes en caso de uso a probar, entradas que se deben proporcionar al sistema y salidas esperadas. En el caso concreto de aplicaciones web, es muy importante comprobar que la navegación funciona correctamente, es decir, no existen enlaces “rotos” y los componentes de servidor se invocan correctamente desde las páginas cliente.

Tabla 3.244 Pruebas de Integración

Caso de Prueba	Entrada	Salida	Resultado
Control de Acceso	Usuario y clave del administrador	Privilegio 1	Activa menú del administrador. No se detectan errores
Control de Acceso	Usuario y/o clave incorrecta	Privilegio 0	Acceso denegado. No se detectan errores
Control de Acceso	Usuario y clave de usuario	Privilegio 2	Activa menú del usuario. No se detectan errores
Ingresar Usuarios	Privilegio 1 Información del usuario	Se guardan los datos	Formulario en blanco Validaciones: • Debe estar lleno el campo usuario • Deber tener lleno el campo de nombre • Debe tener lleno el campo de password • Debe tener lleno el campo de privilegio • No acepta valores duplicados
Actualizar Usuarios	Privilegio 1 Selección de usuario	Formulario con datos recuperados del usuario seleccionado Se guardan los datos	Formulario en blanco Validaciones: • Debe seleccionar el usuario a actualizar • Debe estar lleno el campo usuario • Deber tener lleno el campo de nombre • Debe tener lleno el campo de password • Debe tener lleno el campo de privilegio • No acepta valores duplicados
Borrar Usuarios	Privilegio 1. Selección de materia	Formulario con datos recuperados del usuario seleccionado Se borran los datos	Formulario en blanco Validaciones: • Debe seleccionar el usuario a ser borrado
Crear Consulta	Usuario y clave de usuario.	Se guardan los datos	Formulario en blanco Validaciones: • Debe estar lleno el código

	Datos generales de la consulta		de la consulta • No acepta valores duplicados
Editar Consulta	Usuario y clave de usuario. Selección de la consulta	Formulario con datos recuperados de la consulta seleccionada Se guardan los datos	Formulario en blanco Validaciones: • Debe seleccionar la consulta a ser editada • No acepta valores duplicados
Eliminar Consulta	Usuario y clave de usuario. Selección de la consulta a eliminar	Formulario con datos recuperados de la consulta seleccionada Se borran los datos	Formulario en blanco Validaciones: • Debe seleccionar la consulta a ser borrado • No valida que tenga una consulta asociada
Presentar Consulta	Usuario y clave de usuario. Lista general de consultas	Formulario con datos recuperados de consulta seleccionada Se pueden consultar los datos	Formulario en blanco Validaciones: • Debe seleccionar la consulta a ser presentada

Elaborado por: Autor

3.5.3 PRUEBAS DEL SISTEMA

Se debe comprobar que el sistema está correctamente desplegado en el entorno donde va a ser utilizado y el tiempo que tarda en llegar la petición del usuario, el procesamiento de la misma y el tiempo consumido para hacerle llegar la respuesta. Es por ello que se debe tipificar el ancho de banda a que tienen acceso el grueso de los usuarios, y en base a esto una interfaz que vaya acorde,

Tabla 3.255 Pruebas del Sistema

Caso	Tipo de conexión			
	Local	**Red**	**ADSL**	**Dial Up**
Ingresar Usuarios	**0,19339**	**0,28041**	**2,56133**	**2,43327**
Actualizar Usuarios	**0,16088**	**0,23328**	**2,62847**	**2,49704**
Borrar Usuarios	**0,16634**	**0,24120**	**2,69013**	**2,55563**
Control de Acceso	**0,17215**	**0,24962**	**4,77209**	**4,53348**

Colsulta a Mysql	**0,16808**	**0,24371**	**4,94657**	**4,69924**
Consulta a Sqlserver	**0,15161**	**0,21984**	**4,88518**	**4,64092**
Consulta a Oracle	**0,16802**	**0,24363**	**2,82400**	**2,68280**
Consulta a Postgress	**0,16422**	**0,23812**	**2,37517**	**2,25641**
Consulta Mysql-Sqlserver	**0,19738**	**0,28620**	**2,18404**	**2,07484**
Consulta Mysql-Oracle	**0,17866**	**0,25905**	**5,18295**	**4,92381**
Consulta Mysql-Postgress	**0,15323**	**0,22219**	**1,62860**	**1,54717**
Consulta Sqlserver-Oracle	**0,19039**	**0,27607**	**2,75740**	**2,61953**
Consulta Sqlserver-Postgress	**0,19800**	**0,28710**	**4,33388**	**4,11719**
Consulta Oracle-Postgres	**0,15205**	**0,22047**	**4,19353**	**3,98385**
Consulta Mysql-Sqlserver-Oracle	**0,15810**	**0,22924**	**3,73772**	**3,55083**
Consulta Mysql-Sqlserver-Postgress	**0,18775**	**0,27223**	**4,95802**	**4,71012**
Consulta Sqlserver-Oracle-Postgress	**0,17391**	**0,25217**	**1,24197**	**1,17987**

Elaborado por: Autor

CAPÍTULO 4
CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUSIONES

Para la realización de este proyecto se trabajo con la definición y aplicación del Proceso Ágil de Desarrollo para aplicaciones Web que se enmarcan en un conjunto de procesos de desarrollo que tiene a UML como notación. Cabe indicar que con este proceso se ha mostrado una guía fácil y clara de utilizar para el desarrollo de aplicaciones para el Web.

En la actualidad, muchas instituciones se han dado cuenta de la importancia que el Web y el software libre tiene en el desarrollo de sus potencialidades, ya que con ello pueden lograr una mejor comunicación con personas o instituciones situadas en cualquier lugar del mundo.

Una de las ventajas de utilizar el Web y el software libre, es que no hay restricciones en el sistema operativo que se debe usar, permitiendo la conexión entre si, de las páginas Web desplegadas en un navegador del Web que funciona en una plataforma, con servidores de bases de datos alojados en otra plataforma. Además, no hay necesidad de cambiar el formato o estructura de la información dentro de las bases de datos.

Toda institución sea pública o privada tiene como objetivo fundamental obtener el mayor rendimiento de sus operaciones con un uso adecuado de sus recursos disponibles; con la utilización del servidor Web Apache, el lenguaje PHP y la base de datos MySql es posible conseguirlo porque a más de ser lo suficientemente robustas son económicas ya que no cuestan nada.

La consecución de este proyecto tiene como propósito proporcionar un servicio alternativo a los administradores de bases de datos, los cuales obtienen una gran ventaja al obtener información de forma oportuna de las diferentes bases de datos y de esta manera poder realizar un mejor seguimiento y control de los datos almacenados.

4.2 RECOMENDACIONES

Se recomienda la utilización de software libre en nuevas aplicaciones Web ya que proporcionan una reducción considerable en costos de desarrollo tanto en hardware como software y por la gran versatilidad que estas ofrecen al ser utilizadas.

Se invita a poner mayor énfasis en la enseñanza del Proceso Ágil de Desarrollo ya que esta ofrece una metodología muy interesante en el desarrollo de las aplicaciones; tiene una filosofía de trabajo en grupo a través de roles, funciones y una planificación bien estructurada.

El patrón Modelo Vista Controlador por su gran versatilidad se lo puede utilizar para el desarrollo de cualquier tipo de proyecto y en particular en aquellos que manejan gran cantidad de datos y transacciones complejas.

GLOSARIO DE TÉRMINOS

Apache

Servidor web de libre distribución creado bajo una licencia Open Source.

Apache Software Foundation

Fundación encargada de velar por el desarrollo y la promoción del servidor web Apache y de otros proyectos generalmente relacionados con tecnologías web.

API

AplicationProgram Interface –La interfaz de programas de aplicación describe a detalle los métodos de un sistema operativo o aplicación que pueden ser usadas por un programador y realizar llamadas a ellos

ASP

AplicationServiceProvider –Proveedor de Servicios de Aplicación. Giro comercial donde una empresa ofrece una aplicación a un conjunto de clientes y su uso a través de Internet.

CGI

Common Gateway Interface –Interfaz Común de Puerta de Enlace. Estándar para la transmisión de información entre el servidor y otras aplicaciones independientes del servidor.

DBMS

DataBase Management System –Sistema de Manejo de Base de Datos. Es un programa de computadora que permite a uno o mas usuarios crear y acceder datos en una base de datos

DLL

Dynamic Link Library –Biblioteca de Vínculos Dinámicos. Es un archivo que contiene funciones que se pueden llamar desde aplicaciones u otras DLL.

DNS

DomainName Server –Servidor de Nombres de Dominio. Protocolo tanto para la resolución de nombre de dominio como su viceversa.

Extranet

Red privada de una empresa o institución que utiliza los protocolos de Internet y un sistema de telecomunicaciones público para compartir, de manera segura, parte de la información de negocios con clientes, proveedores, socios, u otros negocios.

GNU

Acrónimo recursivo de GNU'sNot Unix. Proyecto lanzado por la Free SoftwareFoundation con el objetivo de conseguir un sistema operativo similar a Unix, pero totalmente libre.

GPL

General PublicLicense. Es la licencia que utiliza la Free SoftwareFoundation para proteger el software distribuido por el proyecto GNU, así como otros programas que no forman parte del proyecto GNU, pero que sus desarrolladores lo han adoptado.

GUI

GraphicUser Interface –Interfaz gráfica del usuario

Hipertexto

Es la organización de unidades de información en asociaciones conectadas que un usuario puede escoger. Una instancia de esa asociación es llamada vínculo o hipervínculo.

HTTP

HyperTextTransferProtocol –Protocolo de Transferencia de Hipertexto. Es un conjunto de reglas para intercambiar archivos (texto, gráficos, imágenes, sonido, video y otros archivos multimedia) en la WWW.

Internet

Es una red de redes de alcance mundial que utiliza los protocolos TCP/IP, también se la conoce como "La Red"

LDAP

LightweightDirectoryAccesProtocol –Protocolo Ligero de Acceso a Directorios. Es un protocolo que permite ubicar organizaciones, individuos y otros recursos tales como archivos y dispositivos en una red, ya sea Internet o una intranet corporativa.

LGPL

GNU Library General PublicLicense

MIME

Multiporpuse Internet Mail Extensions –Extensiones Multipropósito para el Correo de Internet.

MySql

Es un sistema de gestión de base de datos relacional. Es el servidor de base de datos open source más popular del mundo.

Open Source

Código Abierto. Es una modalidad en el desarrollo de software en la cual los desarrolladores hacen disponible el código fuente del mismo a terceras persona.

Perl

PracticalExtration and ReportLanguage –Lenguaje práctico de extracción y reportes.

PHP

HyperTextPreprocessor. Lenguaje de scripting que permite la generación dinámica de contenidos en un servidor web.

PostgreSQL

Manejador de base de datos Objeto-Relacionales.

Private Key

Llave Primaria. Es la llave utilizada para descifrar los datos cifrados con la llave pública correspondiente.

Public Key

Llave Pública. Es un valor provisto por una autoridad certificadora que puede usarse para cifrar mensajes.

Software

Componente intangible en la informática. Generalmente se trata de una serie de instrucciones elaboradas por humanos en lenguaje de programación de alto nivel (código fuente) que luego son traducidas por un compilador a código de máquina (unos y ceros comprendidos por las máquinas).

SSL

Secure Socket Layer –Capa de Sockets Seguros. Protocolo posterior al TCP/IP que cifra la información transmitida.

TCP/IP

Transmisión Control Protocol/ Internet Protocol –Protocolo para el Control de Transmisiones / Protocolo de Internet. Identifica a un conjunto de protocolos que operan en Internet.

Vinculo

Es una conexión entre una palabra, imagen u objeto de información a otro documento.

WWW (Word Wide Web)

La telaraña de cobertura mundial, también conocido como WWW o Web, es el conjunto de recursos y usuarios en la Internet que hacen uso del protocolo de transferencia de hipertexto (http). Según la W3C: “El Word Wide Web es el universo de información accesible por red, es una representación del conocimiento humano”.

XML

Extensible MarkupLanguage –Lenguaje de Marcado Extensible. Es un lenguaje para definir el formato de la información a compartir a través de una red.

REFERENCIAS BIBLIOGRÁFICAS

Libros y Publicaciones

- Booch Grady, Jacobson Ivar and Rumbaugh James. “El Proceso Unificado de Desarrollo de Software”. Prentice-Hall 2000.
- Booch, G. Rumbaugh, J. y Jacobson,I. “El Lenguaje Unificado de Modelado”. Ed. Addison-Wesley 1999.
- Booch, G. Rumbaugh, J. y Jacobson,I. “Utilización de UML”. Ed. Addison-Wesley 1999.
- Larman,C.”UML y Patrones. Introducción al Análisis y Diseño Orientado a Objetos”. Prentice-Hall, Inc. México 2000.
- Javier Gil, Jorge Tejedor, Agustín Yague, Santiago Alonso, Santiago & Abraham Gutiérrez. “Creación de sitios web con PHP4”. Primera Edición 2005

Referencias de Internet

- Apache http://www.apache.org
- Free Software Foundation http://www.fsf.org/. http:/www.gnu.org./philosophy/free-sw.es.html
- Grupo sobre software libre http://gsyc.escet.urjc.es/sobre/novatica-mono/
- Desarrollo de aplicaciones web . http://glud.udistrital.edu.co/glud/areas/doc/articulos/1_articulo_ws/aplicaciones-web.html;
- MySqlhttp://www.mysql.com
- PHP http://www.php.net

ANEXOS

ANEXO No. 1

CÓDIGO DE LAS CLASES

```
//Clases para el sistema
 include($director."libreria/dbcc.php");
 include($director."libreria/formulario.php");
 include($director."libreria/menu.php");
 include($director."libreria/acces.php");
 include($director."libreria/session.php");
 include($director."libreria/varsend.php");
 include($director."libreria/template.php");
 include($director."libreria/templateform.php");
 include($director."libreria/estadisticas.php");
// include($director."libreria/class.paginaZ.php");
 include($director."libreria/botones.php");
 include($director."libreria/parametros.php");
 include($director."libreria/ayuda.php");
 include($director."libreria/formatoxml.php");
 include($director."fckeditor.php") ;
 include($director."libreria/lista.php");
 include($director."libreria/errores.php");
 include($director."libreria/calculo.php");
 include($director."libreria/listareporte.php");

 //Utilizando Librerias
  //Base de datos
  $objBd = new  conecc();
  //Formulario
  $objformulario = new  formulario();
  $objformulario->ptaeditor=$ptaeditor;
  //Menu General
  $objmenu = new  menu();
$objmenu->submenuact=$smenu;
  //Conectando a la base de datos
  $objBd->conectardb($basededatos,$host,$userdb,$passwdb);
  //Sistemas
  $objacceso_system = new  acceso_system();
  //Sessiones
  $objacceso_session = new  session_system();
    //Template General
  $objtemplate = new  template();
  $objtemplate->select_template();

  //Templates Formularios de tablas
  $objtableform= new templateform();
```

```
  if ($table)
  {
  $objtableform->select_templateform($table);
  }
  //Botones
  $objbotones= new boton_aqualis();
  //Parametros generales
  $objparametros= new parametros_generales();
  $objparametros->parametros();

  //Ayuda
  $objayuda=new ayuda_aqualis();
  $objayuda->desplegar_ayuda($table);

  //XML
  $objxml=new formatoxml();

  //Lista reportes
  $objgridtabla=new listadogrid();

  //Errores
  $objerror=new manejo_errores();

   //Calculo
  $objcalculo=new funciones_calculo();

   //Lista reportes configurable
  $objgridtablareporte=new listadoreportegrid();

?>
```

ANEXO No. 2

ESQUEMA DE CREACIÓN DE LA BASE DE DATOS

Estructura de tabla para la tabla iba_aplication

Campo	Tipo	Nulo	Predeterminado
ap_id	int(11)	Sí	NULL
ap_nombre	varchar(100)	Sí	
ap_detalle	text	Sí	NULL
ap_creador	varchar(200)	Sí	
ap_path	varchar(200)	Sí	
ap_activo	int(2)	Sí	0
ap_protec	int(2)	Sí	0
ap_logo	varchar(250)	Sí	NULL

Estructura de tabla para la tabla iba_aplicationadm

Campo	Tipo	Nulo	Predeterminado
ap_id	int(11)	Sí	NULL
ap_nombre	varchar(100)	Sí	
ap_detalle	text	Sí	NULL
ap_creador	varchar(200)	Sí	
ap_path	varchar(200)	Sí	
ap_activo	int(2)	Sí	0
ap_protec	int(2)	Sí	0

Estructura de tabla para la tabla iba_areausuarios

Campo	Tipo	Nulo	Predeterminado
accw_id	int(11)	Sí	NULL
sys_id	int(11)	Sí	NULL
tab_id	int(11)	Sí	NULL
accw_cusuario	varchar(40)	Sí	NULL
accw_cnombre	varchar(20)	Sí	NULL
accw_cclave	varchar(40)	Sí	NULL
accw_cemail	varchar(40)	Sí	NULL
accw_tituloemail	varchar(200)	Sí	NULL
accw_replyto	varchar(250)	Sí	NULL
accw_paginaweb	varchar(200)	Sí	NULL
accw_codigo	varchar(20)	Sí	NULL
accw_cidtabla	varchar(20)	Sí	NULL
accw_rclave	int(11)	Sí	NULL
accw_rregistro	int(11)	Sí	NULL

accw_activo	int(11)	Sí	NULL

Estructura de tabla para la tabla iba_ayuda

Campo	Tipo	Nulo	Predeterminado
ayu_id	int(11)	Sí	NULL
ayu_titulo	varchar(200)	Sí	NULL
ayu_detalle	text	Sí	NULL

Estructura de tabla para la tabla iba_contenido

Campo	Tipo	Nulo	Predeterminado
con_id	int(11)	Sí	NULL
sec_id	int(11)	Sí	0
con_titulo	varchar(120)	Sí	
con_detalle	text	Sí	NULL
con_fechai	date	Sí	0000-00-00
con_fechaf	date	Sí	0000-00-00
con_tema	varchar(5)	Sí	0
con_pag	int(2)	Sí	0
codigob	mediumtext	Sí	NULL
con_contenido	text	Sí	NULL
con_menu	int(11)	Sí	0
foto_peq	varchar(250)	Sí	
foto_gran	varchar(250)	Sí	
con_activo	int(11)	Sí	1
catf_id	int(11)	Sí	NULL

Estructura de tabla para la tabla iba_datosg

Campo	Tipo	Nulo	Predeterminado
dat_id	int(11)	Sí	NULL
dat_titulo	varchar(120)	Sí	NULL
dat_timp	text	Sí	NULL
dat_pimp	text	Sí	NULL

Estructura de tabla para la tabla iba_detperfil

Campo	Tipo	Nulo	Predeterminado
detp_id	int(11)	Sí	NULL
per_id	int(11)	Sí	0
detp_obj	varchar(20)	Sí	
detp_codigo	text	Sí	NULL

Estructura de tabla para la tabla iba_instancia

Campo	Tipo	Nulo	Predeterminado
instan_id	int(11)	Sí	NULL
instan_nombre	varchar(200)	Sí	NULL
instan_detalle	text	Sí	NULL

Estructura de tabla para la tabla iba_itemmenu

Campo	Tipo	Nulo	Predeterminado
ite_id	int(11)	Sí	NULL
men_id	int(11)	Sí	0
ite_titulo	varchar(70)	Sí	
ite_tipd	int(11)	Sí	1
ite_linktable	varchar(250)	Sí	
ite_link	varchar(250)	Sí	
ite_detalle	text	Sí	NULL
ite_style	varchar(70)	Sí	
ite_active	int(11)	Sí	0
ite_icono	varchar(150)	Sí	
ite_order	int(11)	Sí	0
ite_extra	text	Sí	NULL
ite_submenu	int(11)	Sí	NULL

Estructura de tabla para la tabla iba_menu

Campo	Tipo	Nulo	Predeterminado
men_id	int(11)	Sí	NULL
men_titulo	varchar(70)	Sí	
men_style	varchar(70)	Sí	
men_type	varchar(70)	Sí	
men_uvic	varchar(20)	Sí	
men_ord	int(11)	Sí	0
men_active	int(11)	Sí	0

Estructura de tabla para la tabla iba_menudesp

Campo	Tipo	Nulo	Predeterminado
dsp_id	int(11)	Sí	NULL
sys_id	int(11)	Sí	0
dsp_nombre	varchar(200)	Sí	
dsp_style	varchar(70)	Sí	
dsp_type	int(11)	Sí	0
dsp_active	int(11)	Sí	0

Estructura de tabla para la tabla iba_objetos

Campo	Tipo	Nulo	Predeterminado
ob_id	int(11)	Sí	NULL
ob_value	varchar(20)	Sí	
ob_etiqueta	varchar(20)	Sí	

Estructura de tabla para la tabla iba_objtable

Campo	Tipo	Nulo	Predeterminado
ot_id	int(11)	Sí	NULL

ot_etiqueta varchar(40) Sí

Estructura de tabla para la tabla iba_perfil

Campo	Tipo	Nulo	Predeterminado
per_id	int(11)	Sí	NULL
per_nombre	varchar(100)	Sí	
per_detalle	text	Sí	NULL

Estructura de tabla para la tabla iba_pitemmenu

Campo	Tipo	Nulo	Predeterminado
itep_id	int(11)	Sí	NULL
menp_id	int(11)	Sí	0
secp_id	int(11)	Sí	0
itep_ltype	int(11)	Sí	0
con_id	int(11)	Sí	0
ap_id	int(11)	Sí	0
itep_parametro	varchar(200)	Sí	NULL
itep_titulo	varchar(70)	Sí	
itep_link	varchar(250)	Sí	
itep_detalle	text	Sí	NULL
itep_style	varchar(70)	Sí	
itep_active	int(11)	Sí	0
itep_icono	varchar(150)	Sí	
itep_order	int(11)	Sí	0
itep_extra	text	Sí	NULL
itep_menu	int(11)	Sí	NULL

Estructura de tabla para la tabla iba_plantilla

Campo	Tipo	Nulo	Predeterminado
plant_id	int(11)	Sí	NULL
plant_nombre	varchar(250)	Sí	NULL
plant_archivo	varchar(250)	Sí	NULL

Estructura de tabla para la tabla iba_pmenu

Campo	Tipo	Nulo	Predeterminado
menp_id	int(11)	Sí	NULL
sys_id	int(11)	Sí	0
menp_titulo	varchar(70)	Sí	
menp_style	varchar(70)	Sí	
menp_separador	varchar(70)	Sí	NULL
menp_uvic	varchar(20)	Sí	
menp_active	int(11)	Sí	0

Estructura de tabla para la tabla iba_ptemplate

Campo	Tipo	Nulo	Predeterminado

temp_id	int(11)	Sí	NULL
sys_id	int(11)	Sí	0
temp_nombre	varchar(150)	Sí	
temp_autor	varchar(150)	Sí	
temp_detalle	varchar(250)	Sí	
temp_url	varchar(250)	Sí	
temp_path	varchar(250)	Sí	
temp_active	int(11)	Sí	0

Estructura de tabla para la tabla iba_reportes

Campo	**Tipo**	**Nulo**	**Predeterminado**
repi_id	int(11)	Sí	NULL
repi_fecha	date	Sí	NULL
repi_nombre	varchar(200)	Sí	NULL
repi_detalle	text	Sí	NULL
plant_id	int(11)	Sí	NULL
tab_namec	varchar(60)	Sí	NULL
fie_namec	varchar(60)	Sí	NULL
repi_campos	text	Sí	NULL
tab_namet	varchar(60)	Sí	NULL
repi_tabla	text	Sí	NULL
tab_namee1	varchar(60)	Sí	NULL
fie_namee1	varchar(60)	Sí	NULL
tab_namee2	varchar(60)	Sí	NULL
fie_namee2	varchar(60)	Sí	NULL
repi_enlace	text	Sí	NULL
repi_nregistros	int(11)	Sí	NULL
repi_agruparpor	text	Sí	NULL
repi_vercuadro	int(11)	Sí	NULL
tgr_id	int(11)	Sí	NULL
repi_sqlreferenciat	text	Sí	NULL
repi_tituloref	varchar(200)	Sí	NULL
repi_campoacontar	text	Sí	NULL
repi_titulocontados	varchar(200)	Sí	NULL
repi_verresumen	int(11)	Sí	NULL
tbas_id	int(11)	Sí	NULL
repi_coneccion	varchar(200)	Sí	NULL
repi_usuario	varchar(200)	Sí	NULL
repi_clave	varchar(200)	Sí	NULL
repi_basedb	varchar(200)	Sí	NULL

repi_activo	int(11)	Sí	NULL
repi_listo	int(11)	Sí	NULL
repi_dinamicotitulo	text	Sí	NULL
repi_sqldinamico	text	Sí	NULL
repi_personalizado	int(11)	Sí	NULL
repi_archpersonalizado	varchar(200)	Sí	NULL

Estructura de tabla para la tabla iba_sess

Campo	Tipo	Nulo	Predeterminado
sess_id	varchar(200)	Sí	
sess_usu	varchar(60)	Sí	
sess_pwd	varchar(60)	Sí	
sess_name	varchar(60)	Sí	
sess_perid	int(11)	Sí	0
idg	int(11)	Sí	0
sess_ci	varchar(30)	Sí	NULL
sess_pin	int(11)	Sí	NULL
sess_fechahora	datetime	Sí	NULL

Estructura de tabla para la tabla iba_sistable

Campo	Tipo	Nulo	Predeterminado
tab_id	int(11)	Sí	NULL
tab_name	varchar(60)	Sí	
tab_title	varchar(100)	Sí	
tab_information	text	Sí	NULL
tab_actv	int(11)	Sí	1
tab_mdobuscar	varchar(200)	Sí	
st_id	int(11)	Sí	1
tab_bextras	varchar(250)	Sí	
tab_valextguardar	text	Sí	NULL
tab_rel	varchar(60)	Sí	
tab_crel	varchar(60)	Sí	
tab_trel	int(11)	Sí	0
tab_datosf	int(11)	Sí	NULL
tab_archivo	int(11)	Sí	NULL
tab_formatotabla	int(11)	Sí	1
ayu_id	int(11)	Sí	NULL
tab_nlista	int(11)	Sí	10
tab_detproceso	int(11)	Sí	NULL
instan_id	int(11)	Sí	NULL

Estructura de tabla para la tabla iba_sisusers

Campo	Tipo	Nulo	Predeterminado

sisu_id	int(11)	Sí	NULL
sisu_usu	varchar(60)	Sí	
sisu_pwd	varchar(200)	Sí	
sisu_name	varchar(150)	Sí	
cod_oficina	int(11)	Sí	NULL
sisu_email	varchar(200)	Sí	
sys_id	int(11)	Sí	0
per_id	int(11)	Sí	0

Estructura de tabla para la tabla iba_styletable

Campo	Tipo	Nulo	Predeterminado
st_id	int(11)	Sí	NULL
st_nombre	varchar(100)	Sí	
st_path	varchar(250)	Sí	
st_pathweb	varchar(250)	Sí	
st_timp	text	Sí	NULL
st_pimp	text	Sí	NULL

Estructura de tabla para la tabla iba_sys

Campo	Tipo	Nulo	Predeterminado
sys_id	int(11)	Sí	NULL
sys_titulo	varchar(250)	Sí	NULL
sys_detalle	text	Sí	NULL
sys_pathfavicon	varchar(250)	Sí	NULL

Estructura de tabla para la tabla iba_template

Campo	Tipo	Nulo	Predeterminado
tem_id	int(11)	Sí	NULL
tem_nombre	varchar(150)	Sí	
tem_autor	varchar(150)	Sí	
tem_detalle	varchar(250)	Sí	
tem_url	varchar(250)	Sí	
tem_path	varchar(250)	Sí	
tem_active	int(11)	Sí	0

Estructura de tabla para la tabla iba_typemenu

Campo	Tipo	Nulo	Predeterminado
tmen_id	int(11)	Sí	NULL
tmen_value	int(11)	Sí	0
tmen_title	varchar(20)	Sí	