Universidad de las Ciencias Informáticas

Facultad 10.

Título: Propuesta de arquitectura para una herramienta que apoye la toma de decisiones de los webmasters en el ámbito del Posicionamiento Web.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor(es): Yoandy Campos Díaz.

Tutor(es): Ing. Yanedi Abreu Bartolomeo.

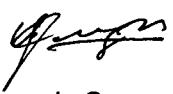
Co-tutor: Lic. Nelson Rodríguez Proenza.

Junio 2007.

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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



Yoandy Campos Díaz.

Ing. Yanedi Abreu Bartomeo.

DATOS DE CONTACTO

Yanedi Abreu Bartomeo: Graduado de Ingeniería en Ciencias Informáticas en la Universidad de Ciencias Informáticas "UCI ", de Ciudad de La Habana, en el año 2007. Actualmente se encuentra trabajando en la UCI, impartiendo la asignatura de Ingeniería de Software. Se desempeña como analista del proyecto Fábrica de Portales.

Nelson Rodríguez Proenza: Instructor Recién Graduado en Licenciatura en Ciencias de la Computación en la Universidad de La Habana en el año 2007. Actualmente es profesor de programación en la Facultad 6. Es jefe de la línea de trabajo de Posicionamiento Web en el Proyecto Operación Verdad.

AGRADECIMIENTOS

A mis padres por ser la guía y ejemplo, sus consejos y confianza que siempre me han dado y porque nadie más que ellos hacen realidad el hecho de verme realizado un ingeniero.

A mis hermanas Yere y Mile por el cariño y apoyo que siempre me han brindado.

A mis primos, tías, y en especial a mi abuelo porque de una forma u otra me han ayudado y apoyado.

A mi novia por ser la persona más especial que pude haber conocido durante estos cinco años.

A mis suegros, cuñado y su familia que me acogieron con gran cariño desde el principio de mi relación.

A todos mis compañeros, los del grupo 3, al Gordo, al Corzo, el Disan y el Lisandro.

A todos aquellos que han contribuido de una forma u otra en mi formación como estudiante y como persona.

A mi tutora y a mi cotutor que me ayudaron y dieron su paso al frente en la realización de este trabajo.

Al Comandante en Jefe Fidel Castro Ruz por haber hecho posible la realización de esta universidad y por haber hecho de esta revolución una obra justa.

DEDICATORIA

Este trabajo va dedicado primeramente a dos personas que desde que tengo uso de razón me han ayudado, me han guiado. Que han estado ahí siempre en los buenos y malos momentos. Gracias a ellos tuve la mejor formación que una persona puede pedir. Por las veces que hice mal las cosas y siempre estuvieron ahí para apoyarme y aconsejarme. Por sus sabios regaños que siempre me enseñaron a ser cada día mejor. A ustedes, mami y papi, gracias por darme la gran oportunidad de ser lo que soy hoy, porque los amo con todas mis fuerzas y siempre estarán ahí conmigo, en mi corazón. Vieja y Viejo, ustedes se merecen esto y mucho más.

A mi hermana Yerenia que es una de las personas que más quiero en este mundo. Gracias por tu cariño, por siempre estar ahí conmigo, por hacerme enojar a veces.

A mi hermana Milena por sus regaños y consejos, por el gran apoyo que me dio cuando estábamos juntos cumpliendo una misión internacionalista en la República de Venezuela. A pesar de no estar presente hoy aquí, te doy las gracias.

A mi novia, por estar a mi lado todo el tiempo, ayudándome con los seminarios hasta altas horas de la madrugada. Por ser mi amante, mi amiga y ser mi otra mitad.

A toda mi inmensa familia por apoyarme todos estos años, siempre les voy a estar agradecido.

A todos mis amigos de la universidad, tanto estudiantes como profesores, les doy las gracias por haber compartido estos cinco años.

A nuestra Revolución por ser autora de este sueño.

RESUMEN

Cuando se realiza una consulta en los buscadores internacionales sobre Cuba, la mayoría de la información no corresponde a sitios Web cubanos y mucha de ella es tergiversada en contra de la política cubana. Debido a las malas prácticas de los webmaster cubanos sobre las técnicas y estrategias de Posicionamiento Web, es que surge la idea de crear un software que realice el análisis de la competencia de los sitios Web en los buscadores, dando la posibilidad al webmaster de aplicar a su sitio ciertas técnicas a partir del análisis realizado por el software.

La presente investigación propone el diseño de la arquitectura del software antes mencionado, para lograr una mejor posición de los sitios Web cubanos en los buscadores internacionales. Se analizan varios estilos arquitectónicos que tienden a ser hoy en día los más aplicados a sistemas informáticos, seleccionando para el desarrollo del software el estilo en capas; así como la fundamentación de patrones arquitectónicos y de diseño. También se presentan las vistas de arquitectura según la metodología RUP para tener una visión de forma global del software.

La arquitectura propuesta permitirá a los desarrolladores una única visión sobre lo que se pretende implementar, además de presentar una descripción detallada y los métodos para la evaluación de la calidad de la arquitectura y por consiguiente del software que se pretende desarrollar.

PALABRAS CLAVE

Arquitectura, Posicionamiento Web.

INDICE

INTRODUCCIÓN

Desde hace varias décadas en el mundo se ha venido desarrollando con gran rapidez las tecnologías informáticas. Conjuntamente con ello la Red de redes se ha ido desarrollando cada vez más y ha aumentado el flujo de información que fluye a través de la misma.

En estos últimos tiempos debido a la gran cantidad de información que fluye en la Internet, se ha hecho más difícil para los webmasters posicionar sus sitios Web en los buscadores internacionales. Cuba ha pasado por una serie de manipulaciones y limitaciones para lograr pertenecer a la gran Red de redes. No ha sido nada fácil, ya que el bloqueo impuesto por más de 40 años y otros factores relacionados a esto han ocasionado para Cuba muchos impedimentos para poder obtener algo de Internet y a la misma vez tener sitios Web propios que aparezcan bien posicionados en los Buscadores.

La Universidad de las Ciencias Informáticas (UCI) y específicamente el proyecto Operación Verdad están trabajando en la ardua tarea de difundir la verdad sobre Cuba a través de la Internet. Para lograr este objetivo, los trabajos deben estar bien posicionados en los buscadores internacionales, para lograr que un mayor número de personas de otros países conozcan la verdad sobre Cuba.

Para que las páginas Web o los sitios Web sean visitados por la mayor cantidad de personas, estos tienen que estar bien posicionados, ya que las personas en Internet cuando realizan búsquedas, no revisan como promedio más de tres páginas de resultados. Por lo que el Posicionamiento Web se ha convertido en una de las prácticas más cotizada en los últimos tiempos para los webmasters.

En la actualidad cuando las personas que navegan en la Web realizan búsquedas sobre términos relacionados con Cuba, solamente encuentran información tergiversada o de páginas pertenecientes a otros países que no ofrecen la información legítima. Este fenómeno se debe, entre otros factores, a que en Cuba los webmasters no tienen el hábito de aplicarle a las páginas Web las estrategias de posicionamiento existentes.

Por lo anteriormente planteado se hace necesario el desarrollo de herramienta automatizada que ayude a los webmasters a tomar decisiones sobre la optimización de las páginas o los sitios Web para lanzarlas a las primeras posiciones en los buscadores. Este tipo de herramientas en el mundo se han desarrollado bajo licencias propietarias, por lo que adquirir un software de este tipo sería algo costoso y las versiones que son gratuitas ofrecen servicios restringidos en el tema. Además muchas de estas herramientas especulan sobre darse de alta en cientos de buscadores, convenciendo a los webmasters de que tendrán mejoras al poner links en directorios. La herramienta a la que se le aplicará esta arquitectura pretende proponer una simple caracterización de la competencia que ayude y le ahorre tiempo al webmaster.

Para desarrollar dicha herramienta se requiere una arquitectura que permita a los desarrolladores tener una única visión para que puedan organizar su trabajo. Sin una arquitectura para el desarrollo de un software no se tendrían en cuenta las principales tecnologías y herramientas adecuadas para revolucionar el sistema, trayendo como consecuencia factores negativos que de una forma u otra pueden frenar el desarrollo eficiente de la aplicación.

Analizando lo anteriormente expuesto es válido plantear el siguiente **problema**: ¿Cómo diseñar la arquitectura de una herramienta que apoye la toma de decisiones de los webmasters en el ámbito del Posicionamiento Web?

Tomando como **objeto de estudio** el desarrollo de arquitecturas para sistemas informáticos; definiendo el **campo de acción** el diseño arquitectónico de una herramienta automatizada que permita realizar estudios y procesos de análisis, que ayude a los webmaster a tomar decisiones para optimizar sus páginas y posicionarlas en los buscadores internacionales.

El **objetivo general** que persigue este trabajo es diseñar la arquitectura de una herramienta que apoye la toma de decisiones de los webmasters en el ámbito del Posicionamiento Web.

El presente trabajo se plantea la siguiente **idea a defender**: si se realiza un buen diseño arquitectónico, se logra: entender el sistema; una aplicación robusta, flexible y reusable. Además los desarrolladores logran organizar el trabajo y serán capaces de revolucionar el sistema.

Como objetivos específicos:

- Analizar aspectos fundamentales del Posicionamiento Web que ayuden a la realización del sistema a desarrollar.
- Definir componentes, vistas y funcionalidades reusables que permitan agilizar el desarrollo de la aplicación.
- Definir patrones de diseño útiles para el desarrollo de la aplicación.

Para darle cumplimiento a estos objetivos se han propuesto las siguientes **tareas**:

- Análisis de las estrategias del posicionamiento existentes.
- Estudio del funcionamiento de los buscadores.
- Comparación de las propuestas de arquitectura consultadas teniendo en cuenta, aspectos positivos y negativos.
- Consulta y análisis de herramientas para la Optimización para Motores de Búsqueda (SEO).
- Determinación y estudio de patrones de arquitectura y de diseño a utilizar.
- Argumentación de la utilización de los patrones, así como su aplicación.
- Definición de las herramientas y tecnologías a usar.
- Definición de los casos de uso más significativos.
- Análisis y definición de las vistas arquitectónicas.

El presente trabajo está compuesto por tres capítulos. El primer capítulo está compuesto por el estudio del estado del arte de la Arquitectura de Software para el sistema a desarrollar, realizándose un análisis de los principales conceptos relacionados con la aplicación a desarrollar, así como de la arquitectura de software y la tecnología a usar para la metodología y modelado.

El capítulo dos muestra la solución propuesta para resolver el problema planteado en el diseño teórico, determinada por las vistas arquitectónicas propuestas por la metodología RUP. Este capítulo está estructurado por dos acápites:

- Línea base de la Arquitectura.
- Descripción de la Arquitectura.

El tercer capítulo muestra un análisis de la solución propuesta, evaluándose el impacto de la arquitectura del sistema desarrollado en el proyecto.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción.

En este capítulo se definen los principales conceptos y definiciones relacionados con el mundo del Posicionamiento Web en los buscadores, para dar un conocimiento básico y una idea a los webmasters de lo que se desea desarrollar. Se abordan además, temas sobre la arquitectura de software, y estilos arquitectónicos. También se describen los principales modelos o vistas de la arquitectura de software para entender en que se basa. Se definen algunas de las herramientas y tecnologías utilizadas para llevar a cabo el desarrollo de la aplicación, teniendo en cuenta las necesidades a cumplir.

1.2 Aspectos fundamentales para entender el surgimiento del posicionamiento.

1.2.1 Internet

Desde el año 1969 DARPA (Defense Advanced Research Projects Agency) comenzó a realizar sus primeros estudios y aplicaciones para conectar computadoras en caso de una guerra atómica que dejara sin comunicación a los seres humanos, con fines principalmente de defensa. Luego en el año 1972 aparece públicamente la primera Red denominada ARPANET, financiada por DARPA. Mas tarde ARPANET cambia su protocolo NCP por el TCP/IP, el cual se usa hoy en día; y varios años más tarde ya existían pequeñas redes en todos los continentes. Estos fueron los inicios de **Internet**, la Red de redes.

Poco a poco esta gran Red se ha ido desarrollando y aumentando el número de usuarios y navegantes, conjuntamente con ello los servicios e información.

En el año 1990, en Ginebra un grupo de físicos crearon el primer cliente Web, llamado World Wide Web (WWW) y el primer servidor Web. Pero no se debe confundir Internet con WWW, ésta última es parte de la Internet, siendo uno de los tantos servicios que ofrece la Red de Redes. La Web es un sistema de información creado mucho más reciente y utiliza la Internet como medio de trasmisión.

Anteriormente Internet servía para un objetivo bastante claro, hoy en día también, pero con la diferencia de que una persona navegando en Internet puede perderse en el amplio espectro de información que ofrece la misma. Esta gran Red, ofrece tantas posibilidades de ideas diferentes, de personas distintas que para una mente humana es casi imposible este gran cúmulo de información. El aumento de personas en la Red ha llegado a formar una gran comunidad, donde todos sus miembros de una forma u otra están conectados entre sí. Tanto es así, que se han llegado a sacar conclusiones que cuando una persona tiene necesidad de conocimiento popular o de conocimiento no escrito en libros recurre a la fuente más fiable, más acorde a su necesidad y más accesible que le sea posible, siendo esta la Internet. Este hecho lejos de ser perjudicial para la especie humana, implica la existencia de un medio capaz de albergar soluciones para problemas que antes eran mucho más difíciles de resolver.

Para ninguna persona es una novedad el crecimiento excesivo de información en Internet, por lo que encontrar algo en la Web se hacía bastante difícil. Las fuentes de información de Internet poseen cinco características principales, que son:

- Extenso y omnipresente.
- Mayormente semiestructurada o sin estructura alguna.
- De diversa calidad.
- Dinámica
- Distribuida y autónoma.

Es por ello y por su crecimiento exponencial, que se estudió la necesidad de poner un orden, control o algún tipo de clasificación a las páginas Web, documentos, sitios y servidores de la Red. Este tipo de trabajo lo comenzaron a realizar los Robots de la Web, pero no todos estos robot son iguales, ni sus técnicas de búsqueda. Cada robot tiene una visión diferente de la Red. Así surgieron los buscadores.

1.2.2 Los Buscadores.

Los **buscadores** son uno de los principales componentes o elementos a tener en cuenta para posicionar las páginas o sitios Web. Esta palabra tiene muchas definiciones en los glosarios de Internet. Por lo tanto a continuación se muestran las definiciones que más se apegan a este trabajo, para que se entienda poco a poco a lo que se desea llegar. La aplicación a desarrollar optimizará las

páginas o sitios Web que el webmaster desea posicionar en un buscador o varios buscadores. Porque una buena forma de dar popularidad a los sitios es dándole de alta en varios buscadores.

¿Qué son los buscadores?:

Herramientas de búsqueda de la Red que permiten, mediante palabras o combinaciones de palabras, encontrar documentos alojados en páginas Web.(Muñiz)

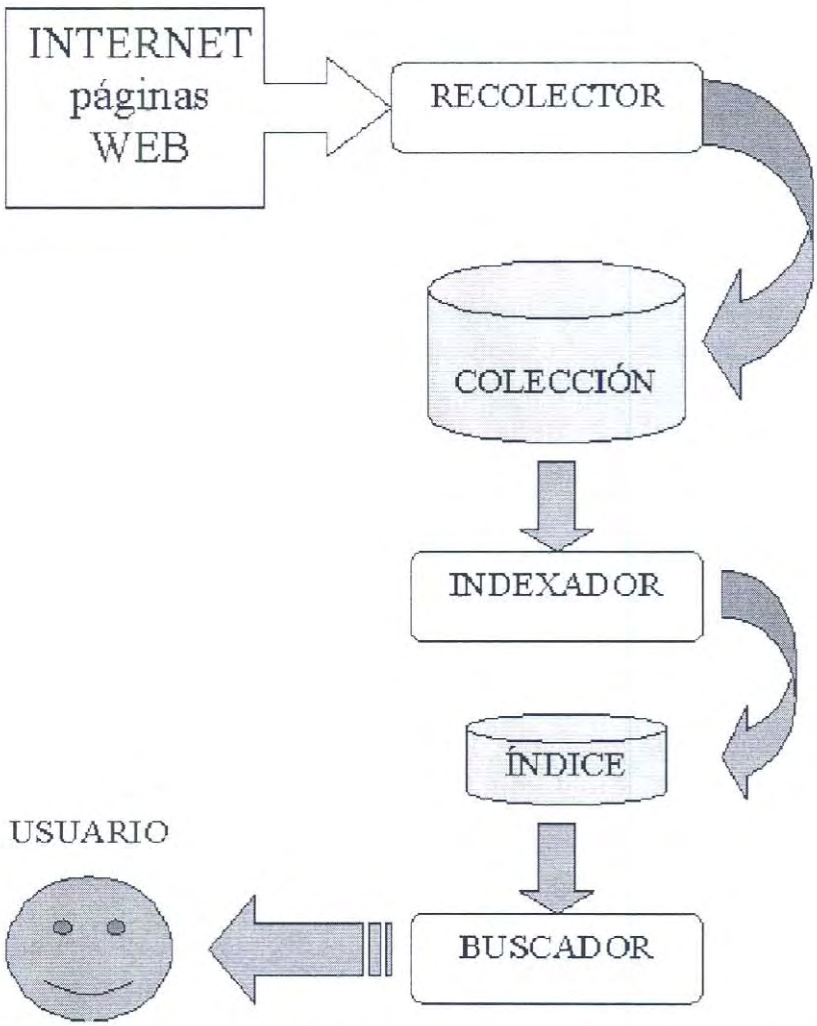


Fig. 1. Funcionamiento de un buscador.

Los buscadores están compuestos por:

- Motor de búsqueda: Realiza un proceso de selección y jerarquización del contenido almacenado en la base de datos en función de términos y criterios de búsquedas para dar los resultados.
- Araña o Spider: Navega constantemente la Web para indexar los sitios Web, los documentos y todo el contenido visitado en la Red para optimizar la búsqueda.
- Base de datos documental: Contiene todo el contenido para ser recuperado.

Las búsquedas se hacen con palabras clave o con árboles jerárquicos por temas; el resultado de la búsqueda es un listado de direcciones Web en los que se mencionan temas relacionados con las palabras clave buscadas. Se pueden clasificar en dos:

1. Índices temáticos: Son sistemas de búsqueda por temas o categorías jerarquizados (aunque también suelen incluir sistemas de búsqueda por palabras clave). Se trata de bases de datos de direcciones Web elaboradas "manualmente", es decir, hay personas que se encargan de asignar cada página Web a una categoría o tema determinado.

2. Motores de búsqueda: Selecciona y jerarquiza contenidos de la Base de Datos en función de términos y criterios de búsqueda para dar los resultados.

Como operan en forma automática, los motores de búsqueda contienen generalmente más información que los directorios. Sin embargo, estos últimos también han de construirse a partir de búsquedas (no automatizadas) o bien a partir de avisos dados por los creadores de páginas (lo cual puede ser muy limitativo). Los buenos directorios combinan ambos sistemas.

Estos motores de búsquedas les asignan de forma numérica una relevancia a los documentos y páginas Web para después de ser indexados en sus bases de datos, colocando sus direcciones en una base de datos confeccionada por el robot de búsqueda. Esta función es conocida como **popularidad**. El sistema pagerank es usado por uno de los motores de búsqueda más populares del Internet, que es el Google, para ayudarle a determinar la importancia o relevancia de una determinada página. La relevancia es calculada mediante un algoritmo basado en el número de links externos, internos, del texto de los enlaces, de la temática de las páginas enlazadas y del peso de las mismas, de las etiquetas meta, de la frecuencia relativa del término en la búsqueda en el texto, así hasta más de 100 variables, dando como resultado un valor numérico que oscila entre 0 y 10 llamado

popularidad, también existen valores negativos, que son utilizados por el buscador para penalizar las páginas Web.

Con este valor numérico es que el motor de búsqueda de Google ubica en una posición los documentos y páginas. Pero los directorios como Yahoo, no usan un robot de búsqueda para indexar y almacenar las direcciones de las páginas Web, estos directorios gestionan bases de datos confeccionada por humanos. Esta base de datos almacena y clasifican en categorías las URLs junto con sus comentarios.

1.2.3 Posicionamiento Web.

Pero desde el surgimiento de los buscadores y directorios ha existido la competencia entre los webmaster para posicionar las páginas y sitios Web. Surgiendo así un nuevo término de gran importancia: **Posicionamiento Web**.

Fig. 2. Posicionamiento Web.

El Posicionamiento Web tiene varias definiciones, haciendo un resumen de estas se puede decir que es un conjunto de procedimientos y técnicas que estudian las características que proporcionan a un sitio o una página Web la máxima visibilidad en Internet en un lugar óptimo entre los resultados proporcionados por un buscador(Codina.).

Existen dos tipos de posicionamiento, que son el natural y el planificado. El posicionamiento natural se conoce como SEO (Search Engine Optimization)(Ugarte 2002), término muy utilizado en la actualidad y significa "Optimización para Motores de Búsqueda". Es natural por la característica que tiene de optimizar las páginas Web y posicionarlas sin usar marketing. Cuando el motor de búsqueda visita estas páginas Web optimizadas y le asigna un ranking, la incorpora automáticamente en su base de datos, es decir, incorpora la URL de la página Web en la base de datos del buscador, siempre con la característica de que esta página no fue posicionada pagando dinero ni utilizando ningún tipo de marketing.

El otro tipo de posicionamiento que existe es el denominado SEM (Search Engine Marketing) (Ugarte 2002) no es más que el "Marketing de Buscadores", este tipo de búsqueda está basada en patrocinar las URL de las páginas Web en los motores de búsqueda. Como es lógico para este tipo de posicionamiento hay que invertir capital monetario para alcanzar éxito, de lo contrario nunca serviría seguir esta línea. Por lo que se recomienda el uso del posicionamiento SEO. Cuba no está en condiciones de pagar grandes sumas de dinero para que las páginas Web cubanas puedan ser visitadas.

Es por esto que el posicionamiento que se recomienda y en el que se basará la aplicación a desarrollar es en el SEO, siendo esto para los webmaster un reto, el posicionamiento natural. La técnica SEO tiene como objetivo principal aprovechar los criterios de valoración de los diferentes buscadores. Estas técnicas conllevan a cambiar el contenido de la página, esto es títulos, enlaces, metadatos; de manera que el motor de búsqueda a la hora de procesar la página Web para su indexación, le otorgue un valor alto a la página.
Según las funcionalidades de las herramientas SEO, estas se pueden categorizar en tres tipos:

- Análisis: Permiten comprobar distintos aspectos del funcionamiento y codificación de las páginas Web.
- Mejora: Analizan la página Web y ofrecen consejos sobre la estructura y contenido del sitio Web, con el objetivo de conseguir un mejor posicionamiento. Un ejemplo de herramientas que usan esta funcionalidad son WordTracker y Adwords(Iniesta 2006), estas aplicaciones son capaces de brindar palabras claves, sugieren elección de títulos a partir de conceptos y temas utilizados en la página Web.

- Monitorización: Permite realizar un control de la página Web, para comprobar su estado, evolución y cambios a lo largo del tiempo.

¿Qué elementos se deben tener en cuenta para lograr una buena optimización de las páginas Web?

- Cumplir estrictamente los requisitos de código HTML.
- Uso de los metatags con valor en la propiedad name=keyword.
- No abusar del uso de las palabras claves.
- Tener en cuenta las peculiaridades y diferencias de cada buscador.
- Relevancia de la palabra en el texto del sitio.
- Peso de la palabra en el sitio.
- Entropía de la palabra en el sitio.

1.2.4 Herramientas SEO existentes.

Promotor Web (*Dynamic Submission 7.0*).(Web)

Este software se dedica a promocionar los sitios Web a partir de un envío que realiza de un sitio Web a los buscadores de forma automática. Este brinda además la posibilidad de optimizar las páginas Web, dando sugerencia de cómo mejorarlas para un mejor posicionamiento en los buscadores.

Active Link Exchange.(Exchange 2006)

El Active Link Exchange realiza una serie de intercambios de enlaces de un sitio Web con los sitios mejor posicionados, siendo este uno de los factores para que los motores de búsquedas califiquen el sitio como bueno y le de un buen pagerank. Permitiendo así un buen posicionamiento.

Active WebTraffic.(Myrasoft)

Este software da de alta un sitio Web en más de un millón de buscadores. Tiene incluido también un generador de palabras claves. Además analiza la densidad de palabras claves, comparándolas con los sitios Web de su competencia. También comprueba la popularidad de los enlaces que apuntan a un sitio Web analizando como afecta su ranking.

Google Monitor Software.(evosdesigne)

Es un software gratuito que a partir de una URL y palabras claves entradas por un usuario ofrece un listado de sitios ordenados por ranking de todos los sitios que responden a esas palabras claves, posicionando la URL dentro de ese listado, dando una idea al webmaster de cómo está su sitio Web. Esta herramienta permite evaluar continuamente la posición de un sitio Web determinado. Es decir permite darle un seguimiento continuo a dicho sitio Web.

AgentWebRanking.(AgentWebRanking)
Imita manualmente el proceso de búsqueda para cada buscador. Crea informes de posiciones seguros y exactos. Brinda la posibilidad de emitir estos informes en varios idiomas, es capaz de realizar informes personalizados.

Todos estos software antes mencionado básicamente están centrados en dar de alta automáticamente en miles de buscadores de todo el mundo, lo que por supuesto facilitaría este trabajo que sería muy engorroso hacerlo manualmente.

Si bien esto es el primer paso de avance para lograr estar indexado en las bases de datos de los buscadores, no es en lo que esencialmente se deben centrar éstos software. Ya que con solo darse de alta en aquellos buscadores que lideran en el mundo como es el caso de Google y Yahoo entre otros más, tendremos el primer paso de avance resuelto para lograr el posicionamiento que deseamos, puesto que es a través de estos que se realiza el mayor número de consultas en el mundo actualmente.

Se deben centrar más, en brindar otras funcionalidades que sí dan un real aporte para lograr posicionar nuestras páginas, como por ejemplo: Listar los 10 mejores sitios posicionados, así como algunos parámetros a tener en cuenta de estos: si posee o no mapa de navegación para cada página, mostrar las palabras más predominantes, los enlaces rotos, determinar la consulta formada con las palabras claves que mejor recupera el sitio a posicionar, etc. Esto ayudaría a realizar un análisis de estos parámetros y enfocar nuestro sitio en ese sentido.

Lo que daría más provecho para lograr lo que realmente se busca, que no es más que nuestras páginas aparezcan en los primeros lugares de búsqueda para consultas determinadas, en los principales buscadores del mundo.

En el caso de aquellos softwares que brindan consejos de optimización a las páginas, a partir de datos de entrada como son: palabras claves, título, etc. presentan incoherencias e imprecisiones. Ofrecen títulos con la concatenación de las palabras claves que el usuario ha entrado que tiene la página que desea mejorar, algo sin ningún tipo de sentido.

Por otra parte, el mejor software hay que pagarlo para poder adquirirlo. Con la investigación y desarrollo de una primera versión de un software de este tipo que intente solucionar estos problemas, se podría garantizar de manera gratuita para nuestro país, además de que se realizaría un compendio de las más importantes funcionalidades que tienen estos y dando una mayor importancia al análisis lexicográfico del contenido Web.

1.3 Arquitectura de Software:

1.3.1 Breve reseña histórica.

Desde el surgimiento de la informática, la programación se consideraba un arte por lo compleja y complicada que era para la mayoría de las personas. Pero poco a poco se comenzaron a desarrollar metodologías para conseguir propósitos y metas. Y a todas estas técnicas se le llamó Arquitectura de Software (AS). Existen muchas definiciones y conceptos para la AS, los principales de ellos dados por grandes estudiosos de la materia, por ejemplo una definición reconocida es la de Paul Clements: "La AS es, a grandes rasgos, una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se la percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema. La vista arquitectónica es una vista abstracta, aportando el más alto nivel de comprensión y la supresión o diferimiento del detalle inherente a la mayor parte de las abstracciones."(Clements 1996)

Otra definición que no deja de ser importante es la definición adoptada por la IEEE Std 1471-2000: "La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución".(Reynoso 2006)

La AS es también conocida como Arquitectura Lógica, consiste en un conjunto de patrones y abstracciones relacionados entre sí que brindan el marco de referencia necesario para orientar y dirigir la línea de construcción del software para un sistema de información.

Esta a la vez plantea los fundamentos para que analistas, diseñadores y otros roles trabajen conjuntamente en una misma línea permitiendo alcanzar los objetivos del sistema, garantizando todas las necesidades.

Una AS se selecciona y diseña basada en objetivos y restricciones bien definidas. Hay arquitecturas que son más recomendables de implementar con ciertas tecnologías por las diferentes características de los distintos sistemas a desarrollar. La AS define de manera abstracta los componentes que llevan a cabo algunas tareas, sus interfaces y la comunicación entre ellos.

En términos más concretos, la AS abarca decisiones muy importantes sobre la organización del sistema de software, los elementos estructurales que componen el sistema y su interfaz, su comportamiento y el estilo que guía esta organización.

La AS no ha tenido todavía una historia aceptable ya que desde los 90´s desde que Mary Shaw y David Garlan reseñaron escuetamente la especialidad, los párrafos han sido reutilizados una y otra vez sin realizar más investigaciones acerca del tema.

Desde la década de los 60´s ya se hablaba de la terminología AS en los círculos de investigación como fue el ejemplo de Edsger Dijkstra, David Parnas y de Fred Brooks. Desde ese entonces se planteaba que debía existir una estructuración de los sistemas de software antes de comenzar a programar. Siguiendo pasos formales para descomponer problemas más complicados y organizarlos en capas. Para 1970(Reynoso 2006) aparecen el diseño estructurado y conjuntamente con ello los primeros modelos explícitos de desarrollo de software. Surgieron entonces las primeras investigaciones académicas sobre diseño de sistemas complejos, independizándose el diseño de la implementación. Pero la verdadera época de la AS se dice que fue a partir de 1990, período conocido como la "crisis del software". En este tiempo fue que surgieron los patrones, realizando trabajos más cerca de la implementación y el código, aunque la reutilización de patrones guarda estrecha relación con el diseño orientado a objetos. Tanto en los patrones como en la arquitectura la idea dominante es

la reutilización. En esta década es que la AS alcanza su madurez como concepto, se desarrolla la tipificación de los estilos arquitectónicos y se elaboran lenguajes de descripción de arquitectura (ADLs). También se consolidó la concepción de las vistas o modelos reconocidos en los framework.

En el siglo XXI, la AS aparece determinada por estrategias orientadas a líneas de productos y por establecer disimiles formas de análisis, diseño, refinamiento, recuperación, diseños basados en escenarios y estudios de casos; redefiniendo todas las metodologías ligadas al ciclo de vida en términos arquitectónicos.

1.3.2 Modelos o vistas de la Arquitectura de Software:

En la AS se deben describir diversos aspectos del software. Normalmente, cada uno de estos aspectos se describe de manera bastante comprensible si se utilizan distintos modelos o vistas. Es de gran relevancia destacar que cada uno de ellos constituye una descripción parcial de una misma arquitectura y es bueno que exista cierto solapamiento entre ellos. Debido a que todas las vistas deben ser coherentes entre si.

Como es lógico, cada desarrollo exige diferentes números y tipos de vistas o modelos para describir una arquitectura de un determinado sistema. Estos modelos se pueden resumir en los siguientes:

1) **Modelos estructurales:** Sostienen que la AS está compuesta por componentes, conexiones entre ellos y (usualmente) otros aspectos tales como configuración, estilo, restricciones, semántica, análisis, propiedades, racionalizaciones, requerimientos, necesidades de los participantes. El trabajo en esta área está caracterizada por el desarrollo de lenguajes de descripción arquitectónica (ADLs).

2) **Modelos de framework:** Son similares a la vista estructural, pero su énfasis primario radica en la (usualmente una sola) estructura coherente del sistema completo, en vez de concentrarse en su composición. Los modelos de framework a menudo se refieren a dominios o clases de problemas específicos. El trabajo que ejemplifica esta variante incluye arquitecturas de software específicas de dominios, como CORBA, o modelos basados en CORBA, o repositorios de componentes específicos, como PRISM.

3) **Modelos dinámicos:** Enfatizan la cualidad conductual de los sistemas. "Dinámico" puede referirse a los cambios en la configuración del sistema, o a la dinámica involucrada en el progreso de la computación, tales como valores cambiantes de datos.

4) **Modelos de proceso:** Se concentran en la construcción de la arquitectura, y en los pasos o procesos involucrados en esa construcción. En esta perspectiva, la arquitectura es el resultado de seguir un argumento (script) de proceso. Esta vista se ejemplifica con el actual trabajo sobre programación de procesos para derivar arquitecturas.

5) **Modelos funcionales:** Una minoría considera la arquitectura como un conjunto de componentes funcionales, organizados en capas que proporcionan servicios hacia arriba. Es tal vez útil pensar en esta visión como un framework particular. Ninguna de estas vistas excluye a las otras, ni representa un conflicto fundamental sobre lo que es o debe ser la AS. Por el contrario, representan un espectro en la comunidad de investigación sobre distintos énfasis que pueden aplicarse a la arquitectura: sobre sus partes constituyentes, su totalidad, la forma en que se comporta una vez construida, o el proceso de su construcción. Tomadas en su conjunto, destacan más bien un consenso.

Estos modelos anteriormente planteados se conocen también de la siguiente forma:

- La visión **estática**: Describe qué componentes tiene la arquitectura.
- La visión **funcional**: Describe qué hace cada componente.
- La visión **dinámica**: Describe cómo se comportan los componentes a lo largo del tiempo y como interactúan entre sí.

Las vistas o modelos de una arquitectura pueden expresarse mediante uno o varios lenguajes. El más obvio es el lenguaje natural, pero existen otros lenguajes tales como los diagramas de estado, los diagramas de flujos de datos, etc. Estos lenguajes son apropiados únicamente para un modelo o vista. Afortunadamente existe cierto consenso en adoptar UML (Unified Modeling Language, lenguaje unificado de modelado) como lenguaje único para todos los modelos o vistas. Sin embargo, un lenguaje generalista corre el peligro de no ser capaz de describir determinadas restricciones de un sistema de información (o expresarlas de manera incomprensible).

1.3.3 Estilos arquitectónicos.

Los estilos arquitectónicos tienen muchas definiciones y conceptos definidos por los grandes desarrolladores de la ingeniería de software. Por ejemplo se puede citar una definición hecha por Mary Shawn y David Garlan donde plantean que "un estilo arquitectónico define una familia de sistemas en términos de patrón de organización estructural. Específicamente, un estilo arquitectónico determina el vocabulario de componentes y conectores que puede ser usado así como un conjunto de restricciones de cómo pueden ser combinados".(David Garlan 1993)

Los estilos arquitectónicos están basados en los patrones de arquitecturas que se usen. Los estilos agrupan clases, englobando una serie de estilos arquitectónicos que comparten características. Generalmente los estilos proveen guías para crear una clase amplia de arquitectura, donde los patrones se enmarcan en darle solución a problemas más pequeños y más específicos dentro de un estilo dado.

A continuación se muestran los estilos arquitectónicos más recomendados para conformar la arquitectura que se propone.

1.3.3.1 Estilos de Llamada y Retorno.

- Modelo Vista Controlador: Conocido más bien como Model View Controler (por sus siglas en ingles, MVC). Este patrón separa el modelado del dominio, la presentación y las acciones basadas en la entrada de datos por el usuario en tres clases diferentes. El modelo está encargado de administrar el comportamiento y los datos del dominio de aplicación, respondiendo a requerimientos de información sobre su estado y a instrucciones de cambiar su estado (normalmente desde el controlador); la vista que controla la visualización de la información y el controlador que interpreta las acciones del ratón y del teclado, informando al modelo y/o a la vista para que cambien su estado según les sea factible.

- Cliente-Servidor: Es una arquitectura compuesta por dos componentes fundamentales, uno de estos elementos es el cliente que puede estar compuesto por una o varias computadoras conectadas en Red que solicitan servicio al otro componente, que es el servidor, que no es más que otra computadora en la misma Red.

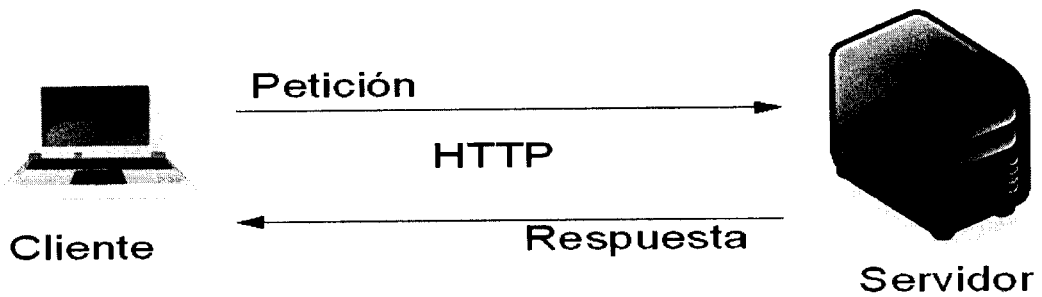


Fig.3. Modelo cliente-servidor.

Este tipo de arquitectura permite que los usuarios puedan acceder a los mismos datos al establecer conexión con el servidor, este brinda un acceso transparente a las aplicaciones que se ejecutan en el, controlando el acceso a los recursos. Como se puede observar en la figura 3 esta arquitectura se basa en procesos distribuidos, procesos de pedidos y respuestas. En la actualidad se hace uso de esta tecnología, por ejemplo la Internet en el amplio sentido de la palabra está basada en esta tecnología. Esta arquitectura es recomendable para aplicaciones Web.

- Arquitectura en Capas: Es una organización jerarquizada, donde una capa le brinda servicios única y exclusivamente a su capa superior y es provista de su capa inferior. Las interacciones entre capas generalmente son invocadas por métodos, donde normalmente las capas inferiores no pueden utilizar las funcionalidades de los niveles superiores. Este tipo de estilo permite desarrollar la aplicación a través de módulos, facilitando a la vez la corrección de errores y brindar un mejor soporte al sistema una vez finalizado.

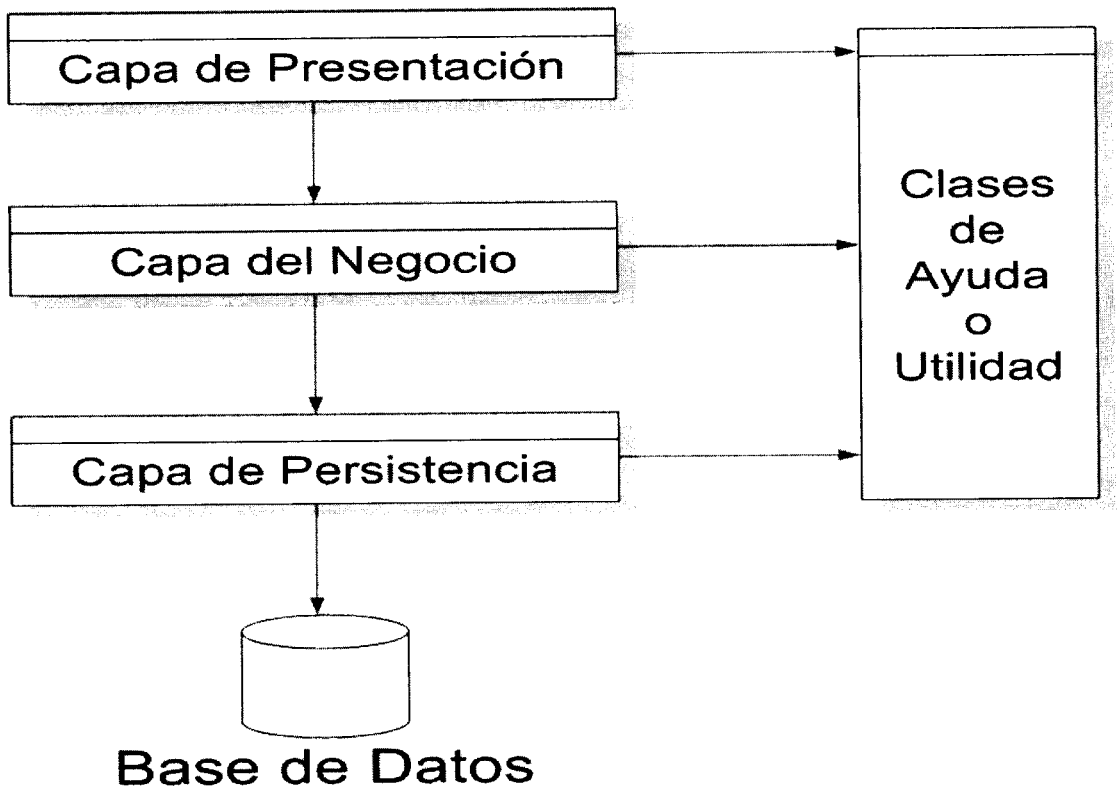


Fig. 4. Arquitectura en Capas.

Las capas normalmente están compuestas por subsistemas o paquetes, los cuales están formados por módulos de desarrollos. Esta arquitectura la podemos ver aplicada muy a menudo a sistemas de software porque es muy útil para organizar el sistema y además este estilo permite hacer uso de un gran número de patrones.

- Arquitectura Orientada a Objetos: En este estilo los componentes se basan en los principios orientados a objetos (OO). En una forma simple un diseño orientado a objetos permite diseñar sistemas para encapsular los datos y los objetos proveen explícitamente interfaces a otros objetos, las interfaces están separadas de las implementaciones. Pero la principal característica de este estilo es que se puede modificar la implementación de un objeto sin afectar la interfaz. Hay muchas variantes en este estilo ya que algunos sistemas permiten que los objetos sean tareas concurrentes, otros admiten que los objetos contengan diferentes interfaces.

- Arquitecturas basadas en componentes: En un estilo de este tipo los componentes son las unidades de diseño, modelado e implementación. Las interfaces están separadas de sus implementaciones y a la vez son el centro del diseño arquitectónico. Los componentes soportan una especie de régimen que permite que las funcionalidades y propiedades puedan ser descubiertas solamente en tiempo de ejecución de la aplicación. En cuanto a las restricciones se puede admitir que una interfaz sea implementada por múltiples componentes.

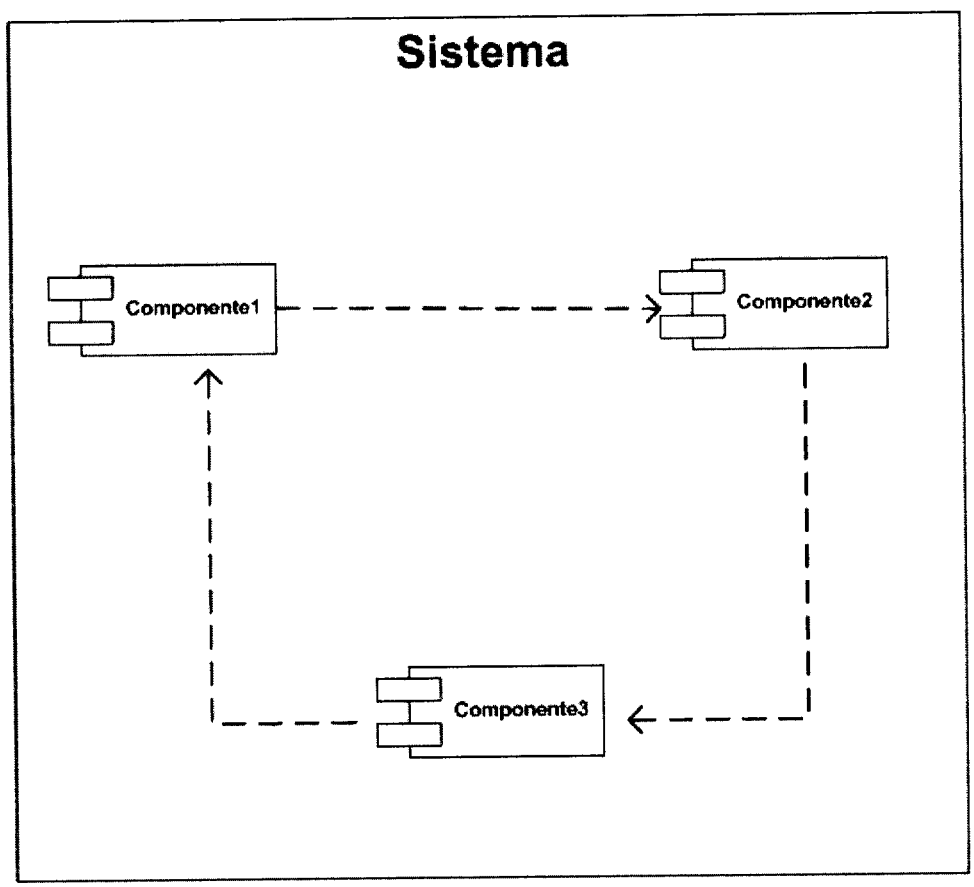


Fig. 5. Arquitectura basada en componentes.

Los componentes deben mantener una baja dependencia del resto de los componentes, los mismos pueden ser reemplazados y deben ser actualizados según cambien los requisitos del sistema. Tiene grandes ventajas este estilo, pues permite la incorporación de nuevos componentes bien documentados y comprobados.

1.3.3.2 Estilos de Código Móvil.

- Arquitectura de Máquinas Virtuales: En la nueva estrategia arquitectónica de Microsoft la máquina virtual por excelencia tiene gran relación con el Common Language Runtime (CLR), el CLR admite programación orientada a objetos (C#, F#, C++, Java, Phyton) y otros paradigmas puros y templados. Con una máquina virtual común el proceso evita la redundancia de compiladores compitiendo por recursos, unificando depurador y profilers.

1.3.3.3 Estilos heterogéneos.

- Sistemas de control de procesos: Se caracterizan por los tipos de componentes y por las relaciones que mantienen entre ellos. El objetivo de este tipo de sistema de control es mantener valores elementales en un rango específico de calibración, algo así como un termostato. La principal ventaja para este estilo es la elasticidad que tiene ante perturbaciones externas.

1.3.3.4 Estilo Orientado a Servicios.

- Una arquitectura muy novedosa es la orientada a servicios (Service Oriented Architecture SOA). Esta es una arquitectura en la que se pueden establecer conexiones para consumir servicios que provee un servidor determinado, estos servicios pueden ser consumidos por terceras personas ya que esta arquitectura permite brindar información en interfaces estándares. En esta arquitectura hay tres componentes que juegan un papel importante que son: el proveedor del servicio, el registro del servicio y el consumidor del servicio. Generalmente a estos servicios son conocidos como webservices (servicios web) usando como formato principal para el intercambio de datos el estándar XML.

  Este estilo presenta características muy importantes como es el bajo acoplamiento entre sus componentes, esto se debe en parte a que las funcionalidades de los servicios pueden ser cambiadas sin traer grandes cambios, ya que como se dijo anteriormente usa formato XML para el intercambio de información.

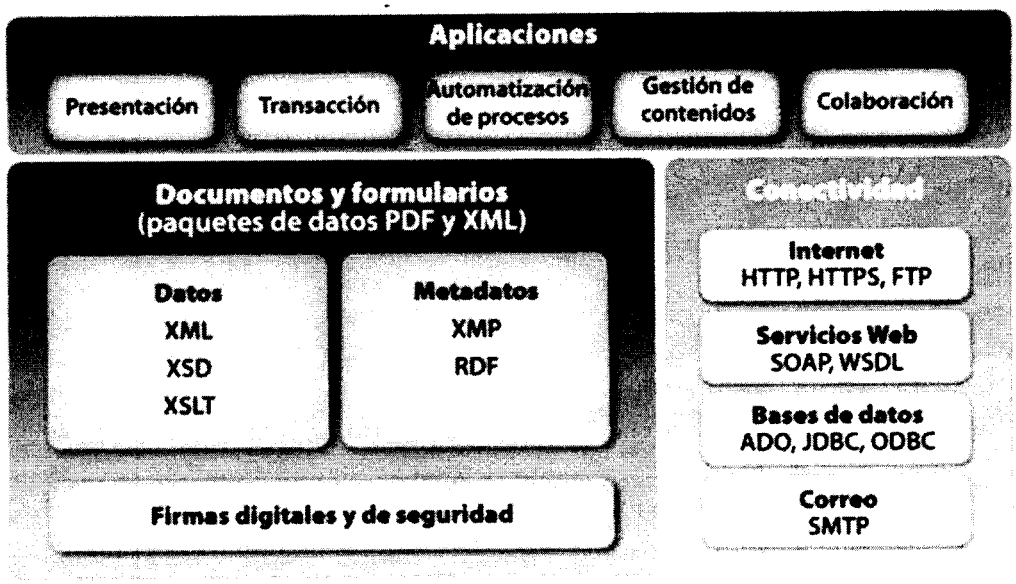


Fig. 6. Representación Arquitectura Orientada a Servicios de Adobe.

Las soluciones basadas en SOA son altamente flexibles y permiten la reutilización de los recursos tanto humanos como técnicos. Este tipo de arquitectura permite que todos los componentes del negocio sean reutilizables con una interfaz estándar y desacoplada. Se piensa que por la gran demanda de este estilo arquitectónico, se convierta en la base de todas las aplicaciones a desarrollar en un futuro.

1.4 Metodologías de Desarrollo.

1.4.1 Metodología Rational Unified Process (RUP).

El Proceso Unificado Racional (Rational Unified Process en inglés como se conoce comúnmente, RUP) es un proceso de desarrollo de software que unido al Lenguaje Unificado de Modelado (UML), constituye la metodología estándar más usada para el análisis, implementación y documentación de los sistemas orientados a objetos (OO). El RUP utiliza el UML para preparar todos los esquemas de un sistema software, de hecho el UML es una parte esencial en el RUP, ya que se desarrollaron paralelamente ambos modelos.

El RUP es bastante flexible, lo que lo convierte un sistema no esquematizado, adaptando una serie de metodologías adaptables al contexto y necesidades de cada situación. El RUP está basado en cinco principios claves:

- Adaptar el proceso.
- Balancear prioridades.
- Demostrar valor iterativamente.
- Elevar el nivel de abstracción.
- Enfocarse en la calidad.

RUP como todo sistema tiene ciclo de vida, utilizando la implementación del Desarrollo en espiral. Organizando las tareas en fases e iteraciones. Este ciclo está compuesto por fases de vital importancia para el desarrollo del sistema o de la aplicación.

Estas fases deben ser cumplidas en un determinado tiempo previamente planificado y son:

- Inicio: Se hace un plan de fases, se identifican los principales casos de uso y se identifican los riesgos
- Elaboración: Se hace un plan de proyecto, se completan los casos de uso y se eliminan los riesgos.
- Construcción: Se concentra en la elaboración de un producto totalmente operativo y eficiente y el manual de usuario.
- Transición: Se Instala el producto en el cliente y se entrena a los usuarios. Como consecuencia de esto suelen surgir nuevos requisitos a ser analizados.

Cada fase tiene un hito, que no es más que un punto de control de objetivo intermedio antes de que el proyecto finalice, los cuales son:

- Conceptualización o Inicio: Objetivos (visión).
- Elaboración: Arquitectura.
- Construcción: Funcionalidad operativa.
- Transición: Release del sistema.

1.4.2 Metodología Extreme Programing (XP).

Es una metodología ágil, muy orientada a la implementación y utilizada para proyectos cortos. Su trabajo está orientado directamente al objetivo, basado en las relaciones interpersonales y la velocidad de reacción. XP intenta minimizar el riesgo de fallo del proceso por medio de la disposición permanente de un representante competente del cliente a disposición del equipo de desarrollo.

¿En qué se diferencia entonces la XP de otras metodologías?
En la XP, la importancia de los test (pruebas para encontrar errores) y la experiencia del cliente, adquieren una mayor importancia, haciendo que se trabaje en ciclos de menor tiempo.

En la programación extrema se da por supuesto que es imposible, prever todo antes de empezar a codificar. Es imposible capturar todos los requisitos del sistema, saber qué es todo lo que tiene que hacer ni hacer un diseño correcto al principio. Es bastante normal hacer un diseño, ponerse a codificar, ver que hay faltantes o errores en el diseño, empezar a codificar fuera del diseño y al final el código y el diseño, o no se parecen, o hemos echado un montón de tiempo en cambiar la documentación de diseño para que se parezca al código.

En vez de tratar de luchar contra todo eso, lo asume y busca una forma de trabajar que se adapte fácilmente a esas circunstancias. Básicamente la idea de la programación extrema consiste en trabajar estrechamente con el cliente, haciéndole mini-versiones con mucha frecuencia (cada dos semanas). En cada mini-versión se debe hacer el mínimo de código y lo más simple posible para que funcione correctamente. El diseño se hace sobre la marcha, haciendo un mini-diseño para la primera mini-versión y luego modificándolo en las siguientes mini-versiones. Además, no hay que hacer una documentación para el diseño. El código, por tanto, también se modifica continuamente de mini-versión en mini-versión, añadiéndole funcionalidad y extrayendo sus partes comunes.

1.5 Lenguaje de Modelado.

1.5.1 Unified Modeling Lenguage (UML).

El Lenguaje Unificado de Modelado (Unified Modeling Lenguage por sus siglas en inglés, UML) es ante todo un lenguaje, que proporciona un vocabulario y unas reglas para permitir una comunicación. En este caso, se centra en la representación gráfica de un sistema.
Nos indica cómo leer los modelos, pero no dice cómo crearlos. Esto último es el objetivo de las metodologías de desarrollo. Sirve para el modelado completo de sistemas complejos, tanto en el diseño de los sistemas software como para la arquitectura hardware donde se ejecuten. UML detalla los artefactos en el sistema y puede modelar además de los sistemas de software, sistemas de hardware y organizaciones del mundo real.

1.6 Herramientas CASE

Las Herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero. Estas herramientas nos pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, documentación o detección de errores entre otras.

1.6.1 Rational Rose Enterprise.

Las herramientas de Rational brindan la libertad que contribuye a una mejor creatividad. Con esta herramienta, se puede elegir el nivel de abstracción que se adapte mejor a cada tarea, se puede utilizar el ambiente de desarrollo que más convenga al equipo y crear el ambiente de proyecto que se ajuste a sus necesidades específicas, siempre existe una opción de licenciamiento que se adapta a las necesidades de la empresa o proyecto. Sin importar cuál sea la plataforma de desarrollo o la función dentro del equipo, siempre hay una solución Rational que puede ayudar.

Con esta herramienta se pueden utilizar cuatro tipos de modelos para realizar el diseño del sistema, utilizando una vista estática y otra dinámica de dichos modelos, uno lógico y otro físico. Permite crear y refinar, logrando así un modelo completo que representa el dominio del problema y el sistema de software.

También es posible descomponer el modelo en unidades controladas e integrarlas con un sistema para realizar el control de proyectos que permite mantener la integridad de dichas unidades. Proporciona un soporte completo entre funciones a través de integraciones de productos y flujo de trabajo para optimizar el desarrollo incrustado y en tiempo real. Encuentra y elimina errores de ejecución, pérdidas de memoria y cuestiones de rendimiento. Acelera el desarrollo a través del modelado visual y las funciones de generación de código e ingeniería inversa.

Sin embargo la herramienta Rational Rose no es una herramienta multiplataforma hasta el momento, por lo que desde un punto de vista tecnológico pone en desventajas los avances del proyecto. Hoy en día se exigen herramientas de trabajo más completas, es decir, que cumpla con determinados requerimientos y con la calidad necesaria.

1.6.2 Visual Paradigm.

Visual Paradigm para UML es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML.

Lista de características:

- Soporte de UML versión 2.1.
- Diagramas de Procesos de Negocio - Proceso, Decisión, Actor de negocio, Documento.
- Modelado colaborativo con CVS y Subversión (nueva característica).
- Interoperabilidad con modelos UML2 (metamodelos UML 2.x para plataforma Eclipse) a través de XMI (nueva característica).
- Ingeniería de ida y vuelta.
- Ingeniería inversa - Código a modelo, código a diagrama.
- Ingeniería inversa Java, C++, Esquemas XML, XML, .NET exe/dll, CORBA IDL.
- Generación de código - Modelo a código, diagrama a código.

- Editor de Detalles de Casos de Uso - Entorno todo-en-uno para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de uso.
- Diagramas EJB - Visualización de sistemas EJB.
- Generación de código y despliegue de EJB´s - Generación de beans para el desarrollo y despliegue de aplicaciones.
- Diagramas de flujo de datos.
- Soporte ORM - Generación de objetos Java desde la base de datos.
- Generación de bases de datos - Transformación de diagramas de Entidad-Relación en tablas de base de datos.
- Ingeniería inversa de bases de datos - Desde Sistemas Gestores de Bases de Datos (DBMS) existentes a diagramas de Entidad-Relación.
- Generador de informes para generación de documentación.
- Distribución automática de diagramas - Reorganización de las figuras y conectores de los diagramas UML.
- Importación y exportación de ficheros XMI.
- Integración con Visio - Dibujo de diagramas UML con plantillas (stencils) de MS Visio.
- Editor de figuras.
- Otras herramientas y plugins de modelado UML que ofrece Visual Paradigm.
- Plataforma Java (Windows/Linux/Mac OS X).
  SDE para Eclipse.
  SDE para NetBeans.
  SDE para Sun ONE.
  SDE para Oracle JDeveloper.
  SDE para JBuilder.
  SDE para IntelliJ IDEA.
  SDE para WebLogic Workshop.

Por lo que se escoge esta herramienta para el desarrollo y modelado UML de todos los diagramas necesarios para el sistema. Teniendo en cuenta que la principal característica de esta herramienta es que es multiplataforma.

1.7 Lenguaje de programación para el desarrollo.

El lenguaje C Sharp (C#), es un nuevo lenguaje de programación orientado a objetos, este lenguaje tiene un propósito general diseñado por Microsoft para su plataforma .Net. Los principales creadores de este lenguaje de programación son Scott Wiltamuth y Anders Hejlsberg, este último creador también del Turbo Pascal y de la herramienta RAD (Rapid Aidded Design. Diseño Rápido Ayudado. Herramientas que facilitan la programación y el diseño de prototipos) de Delphi.

La sintaxis de C# se deriva de dos potentes lenguajes orientados a objetos, que son el C y el C++, ya que la idea de la Microsoft es facilitar la migración de estos lenguajes a C# y que su aprendizaje no fuera muy complejo. C# toma las mejores características de lenguajes como Visual Basic, Java y C++ combinándolas en uno solo. El hecho de ser relativamente reciente no implica que sea inmaduro, pues Microsoft ha escrito la mayor parte de la BCL (Basic Control Language, Control de Lenguaje Básico) usándolo, por lo que su compilador es el más depurado y optimizado de los incluidos en el .NET Framework SDK.

C# se selecciona para el desarrollo del sistema por ser un lenguaje de programación optimizado para el trabajo con la plataforma Desktop, permite crear código muy eficiente, en aquellos puntos de la aplicación que son críticos. Además, permite aprovechar la experiencia creciente por parte del equipo de desarrollo; el alto rendimiento y escalabilidad, la seguridad mejorada sobre otras tecnologías de programación existentes. Así como, la posibilidad de desplegar los sistemas desarrollados con esta tecnología, en ambientes tanto Windows como Linux. Esto no siempre se cumple, ya que las plataformas de ambos sistemas operativos son diferentes y hacen uso de Apis nativas diferentes.

1.8 Marco de Trabajo para el desarrollo de la aplicación.

Cada cierto tiempo las tecnologías empleadas para el desarrollo de aplicaciones sufren cambios. La Microsoft desarrolló una fuerte plataforma conocida por todos como .Net. Con la aparición de .NET (en varias versiones), el framework y la posibilidad de hacer aplicaciones Windows, distribuidas o no, da la oportunidad de estructurar un aplicación en .Net, de forma que sea mantenible y extensible.

El Framework.Net está compuesto por dos componentes fundamentales: Common Language Runtime (CLR) y la biblioteca de clases de Framework.NET. El CLR es un motor que administra el código en tiempo de ejecución y proporciona servicios centrales, como la administración de memoria, la administración de subprocesos y la interacción remota, al tiempo que aplica una seguridad estricta a los tipos y otras formas de especificación del código que fomentan su seguridad y solidez. La biblioteca de clases posee una colección de tipos reutilizables que están integrados estrechamente con el CLR. Esta es orientada a objetos, lo que proporciona tipos de los que su propio código administrado puede derivar funciones. Estos dos componentes hacen que el Framework.Net se pueda utilizar para desarrollar las siguientes aplicaciones y servicios:

- Aplicaciones de consola.
- Aplicaciones GUI de Windows (Windows Forms).
- Aplicaciones de Windows Presentation Foundation (WPF).
- Aplicaciones de ASP.Net.
- Servicios Web.
- Servicios de Windows.
- Aplicaciones orientadas a servicios utilizando Windows Communication Foundation (WCF).
- Aplicaciones habilitadas para el flujo de trabajo utilizando Windows Workflow Foundation (WF).

A partir de la organización de la arquitectura a través de los estilos definidos, se propone la utilización del Framework .NET 2.0. Se utilizará a todos los niveles y capas de la aplicación, reutilizando y redefiniendo de éste interfaces y componentes. Microsoft .NET Framework versión 2.0 instala los archivos asociados de .NET Framework necesarios para ejecutar aplicaciones desarrolladas sobre dicho framework. Mejora la escalabilidad y el rendimiento de aplicaciones gracias a características mejoradas como el almacenamiento en caché, el desarrollo de aplicaciones y la actualización con ClickOnce; además, es compatible con la gama más amplia de exploradores y dispositivos con servicios y controles ASP.NET 2.0.

La biblioteca de clases de .NET Framework es una colección de tipos reutilizables que se integran estrechamente con el CLR. La biblioteca de clases está orientada a objetos, lo que proporciona tipos de los que su propio código administrado puede derivar funciones. Esto ocasiona que los tipos de .NET Framework sean sencillos de utilizar y reduce el tiempo asociado con el aprendizaje de las nuevas características de este. Además, los componentes de terceros se pueden integrar sin

dificultades con las clases. Esta biblioteca está escrita en Lenguaje Intermedio de Microsoft (MSIL-Microsoft Intermediate Language) que es un conjunto de instrucciones independiente de la unidad central de procesamiento que se pueden convertir de forma eficaz en código nativo. Por tanto esta librería puede ser utilizada desde cualquier lenguaje cuyo compilador genere MSIL.

1.9 Common Language Runtime (CLR).

En la actualidad la mayoría de los compiladores y herramientas de desarrollo de software exponen sus funcionalidades en tiempo de ejecución, permitiendo escribir códigos fuente con ventajas que les ofrece este entorno de ejecución administrador. El código administrado no es más que el código desarrollado con un compilador de lenguaje orientado al tiempo de ejecución. Este tipo de código administrado se beneficia de características como:

- La integración de lenguajes.
- El control de excepciones entre lenguajes.
- La seguridad mejorada.
- La compatibilidad con la implementación y las versiones.
- Un modelo simplificado de interacción y servicios de creación de perfiles y depuración.

El CLR facilita el diseño de los componentes y de las aplicaciones cuyos objetos interactúan entre distintos lenguajes. Esto permite integrar el comportamiento de los objetos de forma precisa, por ejemplo, se puede definir una clase y a continuación, utilizar un lenguaje diferente para derivar de una clase de la clase original o llamar un método de la clase original. Esta integración entre varios lenguajes es posible porque los compiladores y herramientas de lenguajes orientados al motor en tiempo de ejecución utilizan un sistema de tipos común definido por el motor en tiempo de ejecución, y los lenguajes mantienen las reglas en tiempo de ejecución para definir nuevos tipos.
Como parte de los metadatos, todos los componentes administrados contienen información sobre los componentes y los recursos utilizados en su creación. El motor en tiempo de ejecución utiliza esta información para garantizar que el componente o la aplicación contengan las versiones especificadas de todo lo necesario, por lo que hay menos posibilidades de que la ejecución del código se interrumpa debido a una dependencia inadecuada. La información de registro y los datos de estado ya no se almacenan en el Registro, donde puede ser difícil establecer y mantener datos. En su lugar, la

información sobre tipos definidos por el usuario (y sus dependencias) se almacena con el código como metadatos y, de este modo, las tareas de réplica y eliminación de componentes es mucho menos complicada.

1.10 Visual Studio.

Visual Studio 2005 es una herramienta muy poderosa desarrollada y distribuida por la Microsoft bajo licencias propietarias, creada especialmente para la plataforma .Net. Esta herramienta ofrece acceso a un conjunto de herramientas de desarrollo necesarias para el desarrollo de aplicaciones tanto desktop como Web.

Con Visual Studio 2005, los desarrolladores pueden:

- Crear aplicaciones utilizando Visual Basic, C#, C++ y J#.
- Realizar aplicaciones para Windows, la Web y Dispositivos móviles desde un mismo entorno unificado de desarrollo.
- Construir aplicaciones cliente/servidor usando servicios Web e integrando herramientas de diseño para acceder a datos remotos.
- Usar SQL Reporting Services para resumir, tabular y mostrar los resultados.

Visual Studio 2005 proporciona varias maneras de desarrollar aplicaciones basadas en Windows que se ejecutan localmente en los equipos de usuarios. Con Visual Studio 2005, se puede crear aplicaciones basadas en Windows e interfaces de usuario (IU) utilizando formularios Windows Forms. También puede crear aplicaciones de Servicio de Windows con Visual Studio o el de Kit de desarrollo de software (SDK) Microsoft .NET Framework versión 2.0 además de crear aplicaciones Windows basadas en Win32 con el Asistente para proyectos de Visual Studio. Este editor aprovecha la eficacia de las clases System.Xml y System.Xml.Xsl de .NET Framework 2.0 y se ajusta a los estándares de XML.

1.11 Control de Versiones. Subversion.

Los sistemas de control de versiones gestionan archivos y directorios, donde su principal función es que almacena la historia de los cambios realizados sobre algún fichero o directorio. De esta forma el sistema nos permite recuperar alguna versión antigua de un fichero o directorio en caso de que el

actual haya sufrido algún error por accidente o algún cambio no deseado, incluso permite recuperar ficheros ya borrados.

Los sistemas de control de versiones son ampliamente utilizados en los proyectos de desarrollo de software, para mantener las versiones del código fuente. No obstante, su aplicación no está limitada a esta actividad, sino que permiten gestionar documentos, imágenes y ficheros de todo tipo.

En primer lugar, los sistemas de control de versiones utilizan para su funcionamiento algún mecanismo de almacenamiento de los datos y la información asociada (metadatos). Esta base de datos o "almacén", suele denominarse repositorio. Los sistemas de control de versiones más populares funcionan con un repositorio centralizado, es decir, aunque permiten el trabajo colaborativo entre varios puestos de trabajo, mantienen el repositorio centralizado en único ordenador, estando accesible para el resto de equipos a través de Red local o Internet.
Subversion (SVN), es un sistema de control de versiones que se ha popularizado en los últimos tiempos, especialmente en la comunidad de desarrolladores de software libre. Cabe destacar que se distribuye de forma libre bajo una licencia de tipo Apache.

1.12 TortoiseSVN.

El TortoiseSVN es un cliente para el Subversion, el cual maneja ficheros y directorios. Es una herramienta desarrollada bajo la Licencia Publica General GNU (GPL). Los ficheros gestionados por esta herramienta se almacenan en un repositorio central, permitiendo recuperar versiones antiguas de ficheros y directorios, examinando la historia de cuando, y quienes modificaron sus datos.
La interfaz del TortoiseSVN es totalmente amigable al usuario ya que se integra perfectamente al Shell de Windows, lo que permite que los usuarios no tengan que adiestrarse en nuevos conocimientos para su manejo, brindando la posibilidad de que los usuarios no tengan que cambiar de sistemas cada vez que necesiten trabajar con las funciones del control de versiones.

1.13 Conclusiones del Capítulo 1.

En este capítulo se trataron los temas introductorios para dar paso a los siguientes capítulos donde se mostrará de forma más detallada características y fundamentación de la arquitectura. Se trataron temas generales y de forma conceptualizada de los inicios del Internet, de los buscadores, de cómo

trabajan estos con sus motores de búsquedas y en los principales aspectos que se basan para posicionar páginas y sitios Web. También se muestran aspectos sobre la arquitectura de software, como sus antecedentes, principales modelos o vistas de arquitecturas, los estilos arquitectónicos usados para el desarrollo de la aplicación.

Para el desarrollo, control y planificación de la aplicación se hizo uso del Proceso Unificado de Modelado como proceso de desarrollo de software, por las facilidades y características que lo diferencian y hacen único.

Se define también las tecnologías a utilizar, tomándose como lenguaje de programación el csharp sobre la plataforma .Net, haciendo uso del marco de trabajo que proporciona la misma. Para el control de versiones se hace uso de dos potentes herramientas que son el TortoiseSVN y el Subversion.

CAPÍTULO 2: DISEÑO DE LA ARQUITECTURA.

2.1 Introducción.

Hasta ahora se han analizado solamente los elementos teóricos que son muy necesarios para dar comienzo a este capítulo. En este capítulo se analiza y se pone en práctica los patrones existentes asociados al campo de acción, las tecnologías y tendencias para conformar y dar solución al sistema a desarrollar. Se hace una propuesta de solución al problema planteado en el diseño teórico de la investigación.

La solución que se propone a continuación está dividida por dos subcapítulos que conforman los principales artefactos:

- Línea Base de la Arquitectura.
- Descripción de la Arquitectura.

Con el desarrollo de estos artefactos se define la metodología expuesta en el Capítulo 1. Se irá demostrando el uso de patrones bajo determinadas circunstancias, ya sean oportunas o necesarias. Es importante resaltar que el uso de patrones no sustituirá ninguna idea de los desarrolladores, ellos juegan un papel importante para desarrollar los patrones propuestos y llegar a soluciones factibles a través de estos patrones.

2.2 Requisitos en la Arquitectura.

2.2.1 Requerimientos funcionales.

R1. Analizar los mejores sitios posicionados.

R1.1. Brindar información si posee mapa de navegación para cada página.

R1.2. Informar la cantidad de tablas, imágenes y enlaces que posee cada página.

R1.3. Mostrar porciento de texto y código embebido que posee cada página.

R1.4 Mostrar idioma que posee cada página.

R1.5 Mostrar sitios que apuntan a cada página.

R1.6 Mostrar Page Rank para cada página.

R1.7 Mostrar correo de webmasters.

R1.8. Mostrar enlaces rotos en el sitio.

R1.9 Mostrar palabras que más predominan.

R1.9.1 Mostrar si están presentes en el párrafo o en el título.

R1.9.2 Mostrar la frecuencia, peso, relevancia y recuento.

R2. Determinar la consulta formada con las palabras claves que mejor recupera el sitio a posicionar.

R3 .Determinar las palabras más buscadas en uno o varios buscadores dado un término de búsqueda.

R4. Mostrar comparaciones entre páginas.

R5. Permitir al webmaster crear proyecto.

R6. Permitir al webmaster modificar proyecto.

R7. Permitir al webmaster eliminar proyecto.

2.2.2 Requerimientos del Hardware.

Estaciones de trabajo.

- Mouse y Teclado PS2 o USB.

- Tarjeta de red.
- RAM: 128 Mb.
- CPU Pentium II o superior.
- Almacenamiento en Disco Duro: 2 GB.
- Conexión a Internet.

2.2.3 Requerimientos del Software.

Estaciones de trabajo.

- Sistema Operativo Windows 98 o superior.
- Instalado Framework.Net 2.0.

2.3 Línea Base de la Arquitectura.

La Línea Base de la Arquitectura contiene los elementos más importantes para lograr una mejor abstracción en el diseño arquitectónico de la aplicación. Esta es una versión interna del sistema al final de la base de elaboración, contiene las versiones de todos los modelos que una aplicación finalizada debe tener al terminar la fase de construcción. En la misma se exponen los estilos arquitectónicos propuestos para dar solución al sistema.

Propósito.

El propósito de la Línea Base de la Arquitectura es brindar la información necesaria para estructurar y organizar el sistema desde un nivel de abstracción mayor.

Los usuarios que harán uso de la Línea Base de la Arquitectura son:

- Equipo de arquitectos del proyecto: Le sirve de quía para tomar decisiones arquitectónicas, además de ser los encargados de refinar la arquitectura.
- Equipo de desarrollo del proyecto: Hacen uso de la Línea Base para guiarse a la hora de implementar.
- Clientes: Con esta tienen una garantía de la calidad y del conocimiento en la cual se desarrolló el sistema.

Alcance.

La Línea Base de la Arquitectura describe de forma detallada el organigrama de la arquitectura basándose en los estilos arquitectónicos, los principales framework de desarrollo adaptándolos a la aplicación, se propone la utilización de un conjunto de patrones para eliminar los problemas que puedan presentarse en el desarrollo de la arquitectura.

2.3.1 Concepciones Generales.

El sistema se desarrollara haciendo uso de la metodología de desarrollo de software Proceso Unificado de Desarrollo (RUP). Esta metodología está basada en tres principios fundamentales que son:

- Guiado por Casos de Uso.
- Centrado en la Arquitectura.
- Iterativo e incremental.

El hecho de que el desarrollo de la solución sea guiado por los casos de uso permite dividir la aplicación en módulos de desarrollo, los cuales a la vez conforman los subsistemas. Definiendo los casos de uso más significativos teniendo en cuenta el nivel de complejidad y de prioridad escogiendo un conjunto de operaciones a desarrollar.

2.3.2 Organigrama de la Arquitectura.

Los organigramas de forma general describen la organización y jerarquización de un sistema, de una empresa, o de cualquier organización estructural que se desee organizar. Dejando bien definido los flujos de trabajo y las responsabilidades de los componentes que forman parte del organigrama.

La herramienta que se desea desarrollar debe mostrar un análisis a partir de un estudio que realizará automáticamente. Es muy importante la precisión de este análisis, ya que el mismo servirá de apoyo para que el webmaster sea capaz de tomar decisiones y aplicar las estrategias de posicionamiento a un sitio Web. A continuación se especifican las principales operaciones:

- Consultar los buscadores internacionales.

- Listar los enlaces del resultado obtenido.
- Mostrar las principales características de cada enlace.
- Guardar estos resultados.
- Mostrar estadísticas según los resultados.

Por las características que debe tener este software, hasta ahora no se usará una lógica de negocio complicada. Por lo que lo más conveniente y más eficiente es desarrollar la aplicación a través de módulos de desarrollo. Esto permite que si se desea realizar la ampliación de esta versión o el desarrollo de otras versiones, no proporcione impactos negativos para el sistema desarrollado.

Para hacer que el sistema funcione modularmente, lo más conveniente para la arquitectura de este software es hacer uso del estilo arquitectónico "Layers" (Capas). Se pueden observar las principales características de este estilo en la sección 1.3.3.

2.3.3 Estilo en Capas.

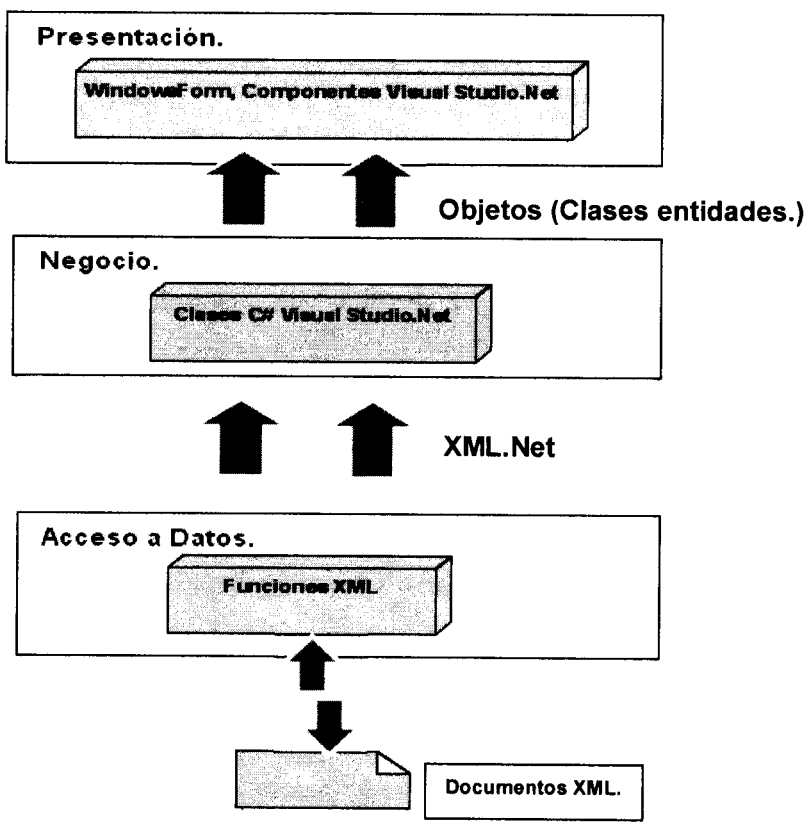


Fig. 7. Estilo Arquitectónico. Estilo en capas.

Principales componentes.

Los principales componentes o elementos que distinguen este tipo de estilo son las capas que lo conforman:

- Presentación.

  Esta capa está compuesta por las principales interfaces con las que interactúan los usuarios. Además de intercambiar toda la información necesaria para realizar todo el proceso de gestionar la información. Estas interfaces están conformadas por los componentes visuales del Visual Studio.Net.

  Los componentes de la interfaz de usuario administran la interacción con el usuario. Muestran los datos al usuario, obtienen los datos del mismo e interpretan los eventos generados por el usuario para actuar en esos datos, cambiar el estado de la interfaz o facilitar la tarea al usuario. Teniendo en cuenta que se hace uso de la interfaz de usuario incorporada en el WindowsForms.

- Negocio.

  En esta capa es donde se implementan todas las clases de la lógica del negocio. Es aquí donde se llevan a cabo todas las transacciones del software. Además de controlar la seguridad de cada método que se invoca, todo esto produce un intercambio entre objetos. A través de esta capa se accede a la capa de Acceso a Datos para realizar otras transferencias de datos.

  Los componentes de proceso de usuario en esta capa se implementan normalmente como clases .NET que exponen métodos a los cuales pueden llamar la interfaz de usuario. Cada método encapsula la lógica necesaria para realizar una acción específica en el proceso de usuario. La interfaz de usuario crea una instancia del componente del proceso de usuario y la utiliza para efectuar la transición a través de los pasos del proceso.

- Acceso a datos.

  Es la capa encargada de manejar todos los datos que se guardarán en documentos XML. Esta capa está compuesta por una serie de funciones predefinidas ya en las bibliotecas del .Net, además de crear otras funciones necesarias para la creación de objetos que contendrán los datos a transferir hacia otras capas.

- Conectores entre capas.

  Presentación-Negocio: La interacción que se genera entre estas dos capas es a través de la invocación de funciones a través de referencias realizadas por objetos.

  Negocio-Acceso a datos: Estas capas se relacionan a través de funciones XML, que son funciones internas y propias del .Net. Acceso a datos-Documentos XML: En el Acceso a datos se crean objetos a través de constructores, para poder acceder a documentos XML, en los cuales se guardarán los datos necesarios.

- Restricciones.

  La restricción más importante que se le aplica a estos componentes es una restricción básica de este estilo arquitectónico, que no es más que, los componentes ubicados en capas superiores solo pueden acceder a componentes que están en capas inferiores o que estén en su mismo nivel.

2.3.4 Patrones de Arquitectura a aplicar.

Un patrón de arquitectura de software describe un problema particular y recurrente del diseño que surge en un marco determinado y específico, y representa un esquema genérico y probado de solución. Los patrones son la abstracción de un problema y su solución, se ocupan de problemas recurrentes, identificando y especificando abstracciones de niveles más altos que el de las clases y sus componentes y como gran ventaja para su uso es que establecen un vocabulario y entendimiento común. El establecimiento de estos de los patrones que se proponen posibilita el aprovechamiento de la experiencia adquirida en el diseño de aplicaciones.

A continuación se proponen una serie de patrones para darle solución a la aplicación. Cada patrón propuesto brindara la justificación de por qué su uso.

1. Patrón de Capas.

La idea de este patrón es descomponer cada capa en subsistemas, ubicando en cada uno de ellos clases con funciones propias.

Los principales objetivos de este patrón es trabajar en la reutilización del Software basándose en la encapsulación y en la herencia, para lograr este objetivo es muy imprescindible no acoplar clases, creando una arquitectura apropiada para el sistema. Lo más importante es crear abstracciones de las diferentes funcionalidades o responsabilidades del sistema agrupándolas en capas ya que se obtendrán clases que serán totalmente independientes del resto, logrando con esto que se puedan reutilizar fácilmente.

Los beneficios que aporta la utilización de este patrón según Larman(Larman) son:

- Aislamiento de la lógica de la aplicación en componentes separados reutilizables en otra aplicación.
- Distribución de las capas en diferentes máquinas o procesos, lo cual puede mejorar el rendimiento, aumentando la coordinación y la compartición de la información.
- Dedicación de recursos en cada una de las capas y la posibilidad que brinda de desarrollarlas en paralelo.

Una desventaja sería el uso excesivo de niveles, es decir aplicar muchos niveles a la aplicación, esto corre el riesgo de que la aplicación sea ineficiente y un nivel muy bajo hace que la aplicación se vuelva un poco compleja e ineficiente. Además si se realiza un mal diseño esto podría traer como consecuencias que los cambios en la funcionalidad se pueden trasmitir de un nivel a otro.

Conjuntamente con los patrones de arquitectura se deben hacer uso de los patrones de diseño para lograr una eficiencia en el desarrollo. Estos patrones son la solución estándar para un problema común de programación, son técnicas para flexibilizar el código haciéndolo satisfacer ciertos criterios.

Algunos de estos patrones son conocidos por patrones **Grasp**, los cuales se usan para asignar responsabilidades.

2. Experto.

El patrón experto, es un patrón de diseño muy utilizado ya que el mismo asigna una responsabilidad al experto en información. Normalmente estas responsabilidades se le asignan a los objetos creados de las diferentes clases para cumplir con una tarea determinada. Esta asignación de responsabilidades es un principio básico en el diseño orientado a objetos.

Se conserva el encapsulamiento en los objetos, ya que estos se basan en la información que poseen para hacer lo que se les pide. Esto soporta un **bajo acoplamiento**, lo que favorece la creación de una aplicación robusta y de fácil mantenimiento.

El comportamiento es distribuido entre las clases que cuentan con la información requerida, logrando con ello la declaración de clases más sencillas y bien cohesionadas entre ellas, que son más fáciles de entender y mantener. Brindando soporte al patrón **alta cohesión**.

3. Creador.

Este patrón se aplica a la hora de asignar una responsabilidad a una determinada clase para crear una instancia de otra clase. Sería algo así como:
Darle la tarea a la clase B la responsabilidad de crear una instancia de la clase A si alguna de las siguientes premisas es cierto:

- B agrega los objetos A.
- B contiene los objetos A.
- B registra la instancia de los objetos A.
- B utiliza específicamente los objetos A.
- B posee los datos de inicialización que serán trasmitidos a A cuando este objeto sea creado (así B se convierte en un **Experto** en la creación de A).

Es importante aclarar que el diseño bien asignado puede soportar un bajo acoplamiento, una mayor claridad, el encapsulamiento y la reutilizabilidad. Con la aplicación de este patrón se brinda soporte a un **bajo acoplamiento,** proporcionando un fácil mantenimiento y mejores oportunidades de reutilización. Esto hace la probabilidad de que el acoplamiento no aumente debido a que la clase creada tiende a ser visible a la clase creador, debido a las asociaciones existentes que conllevaron a elegirla como el parámetro adecuado.

4. Bajo Acoplamiento.

El objetivo principal de este patrón de diseño es lograr una mayor reutilización y una dependencia escasa. Para lograr este objetivo se le asigna la responsabilidad a una clase de no ser muy dependiente, es decir mientras más independientes sea la clase, menor será el acoplamiento.

Durante la aplicación de este patrón no debe considerarse un aislamiento de otros patrones como el creador, el experto y alta cohesión. Sin embargo es un patrón que se debe tener en cuenta cuando se intente mejorar un diseño.

En los lenguajes orientados a objetos se establecen formas comunes entre dos objetos uno *TipoX* y otro *TipoY*, los cuales se plantean a continuación:

- *TipoX* que se refiere a una instancia *TipoY* o al propio *TipoY.*
- *TipoX* tiene un método que siempre hace referencia a una instancia de *TipoY* o incluso al propio objeto *TipoY.*
- *TipoX* es una subclase directa o indirecta de *TipoY.*
- *TipoY* es una interfaz y *TipoX* la implementa.

La aplicación de este patrón soporta el diseño de clases más independientes, logrando reducir el impacto de los cambios que puedan realizarse, conllevando con ello una mejor eficiencia de la aplicación y una mayor productividad por parte del equipo de desarrollo. Una característica muy importante a la hora de usar el acoplamiento es que este patrón solamente es útil cuando se utiliza reutilización, mientras no cumple objetivo alguno hacer uso del mismo. No existe medida para determinar cuando se hace un uso excesivo del acoplamiento, lo importante es poder determinar en que grado se encuentra el acoplamiento y si pueden surgir problemas en caso de incrementarlo.

Generalmente se deben tener con un escaso acoplamiento las clases genéricas, y estas deben ser lo más reusables posible.

Beneficios: El uso de este patrón trae los siguientes beneficios para las clases:

- No se afectan por cambios de otros componentes.
- Son fáciles de entender individualmente.
- Son fáciles de reutilizar.

5. Alta Cohesión.

La alta cohesión es en muchas ocasiones es imposible mantener por lo complicada que es. Se plantea que las clases que no realizan tantas tareas mantienen una alta cohesión.
Las clases para mantener una alta cohesión deben ser:

- Fáciles de comprender.
- Deben permitir la reutilización.
- Tienen que conservarse.
- No pueden ser delicadas para que no les afecten constantemente los cambios.

Según expertos en la materia de patrones, como es el caso de Grady Booch, la alta cohesión se logra cuando todos los elementos de un componente (una clase, por ejemplo) trabajan en conjunto para definir un comportamiento determinado.

Lo ideal es mantener una alta cohesión, así se logra que las clases trabajen poco, con funciones importantes y comparten su tarea con otros componentes en caso de tener un trabajo fuerte. Este tipo de clases son fáciles de mantener, de entender y de reutilizarla, principios básicos en la arquitectura de software.

Los beneficios que brinda este patrón son que mejoran la claridad y facilidad con que se entiende el diseño, se logra un fácil mantenimiento y mejoras en la funcionalidad, es capaz de lograr un bajo acoplamiento y un aspecto muy importante en las funcionalidades es que soporta la reutilización.

6. Controlador.

Este patrón se encarga de identificar que objeto se encargará de manipular y controlar los eventos que se generan en el sistema definiendo su método de operación, ya sea un evento generado por un usuario o por el propio sistema. Los eventos de la aplicación o del sistema, pueden ser generados por factores externos (ejemplo, un actor); estos eventos son eventos de entrada, y los eventos que se generan para darle respuesta a estos eventos de entrada son eventos internos (propios del sistema) o eventos sistémicos.

La solución a este tipo de problema es encargarle a una clase la tarea de manejar un determinado mensaje de los eventos de un sistema. La mayoría de las aplicaciones reciben eventos de entradas, los cuales presentan una interfaz grafica para los usuarios, otros eventos pueden ser producidos por dispositivos externos, como por ejemplos, los decodificadores de código de barras, o cualquier dispositivo de control automático. En el caso del sistema que se pretende desarrollar, los eventos serán generados por un usuario que manipulará el software una vez terminado.

Normalmente los componentes controladores deben estar presentes en la capa del negocio, un aspecto importante a tener en cuenta al usar este tipo de patrón es que la capa de presentación no debe llevar a cabo la tarea de manejar los eventos de entrada. Es muy importante asegurarse de que las operaciones del sistema sigan un proceso consecuente y bien definido, es decir, que una actividad no se ejecute hasta que otra haya terminado por su dependencia sobre la otra, suponiendo que estas operaciones se realicen dentro de un mismo caso de uso. Un controlador es el más indicado para poder capturar el mensaje del segundo evento en caso de que no se cumpla la regla.

7. Fabricación pura.

Este patrón se usa para crear un objeto fabricación pura denominado factoría que maneje la creación. Es decir, su propósito es crear objetos que permitan al sistema identificar que clase debe instanciar en tiempo de ejecución. Es un patrón altamente cohesivo que asigna una serie de tareas a una clase creada solamente para brindar una alta cohesión, un bajo acoplamiento y una buena reutilización. Este patrón, cuando se usa para diseñar sistemas orientados a objetos, debe lograrse una reutilización eficiente de la nueva clase que se crea, en la cual se agrupan otras clases, otras clases que deben ser pequeñas y altamente cohesivas entre sí. A este proceso se le llama granularidad fina.

Con el uso de este patrón se da soporte a una alta cohesión, ya que se reparten responsabilidades entre clases con granularidad fina, centradas en una serie de tareas afines a ellas.

2.4 Descripción de la Arquitectura.

Propósito

La descripción de la arquitectura es un documento que se plantea varios propósitos:

- Mejor comprensión del sistema.
- Organizar el desarrollo.
- Fomentar la reutilización.
- Evolucionar el sistema.

Para lograr estos propósitos el documento ofrece una comprensión arquitectónica global del sistema haciendo uso de las diferentes vistas arquitectónicas propuestas para el desarrollo de la aplicación, mostrando las diferentes características del sistema. Otra función de este documento muy importante es que tiene que servir como documento para la capacitación de los desarrolladores, directivos, clientes y otros usuarios para entender el más mínimo detalle del sistema propuesto.

Alcance

La descripción de la arquitectura es un documento que influye en el funcionamiento correspondiente al sistema que apoyará la toma de decisiones de los webmasters en el ámbito del posicionamiento Web. Abarcando cada uno de los módulos propuestos para el desarrollo de la aplicación.

2.4.1 Representación Arquitectónica.

La representación arquitectónica da una visión de cómo estará estructurada la descripción de la arquitectura propuesta. Para ello se muestran las vistas arquitectónicas que se aplicarán para el desarrollo de este documento. Los modelos que estas vistas arquitectónicas proponen, serán modelados con la herramienta Visual Paradigm 3.0.

Las vistas que a continuación se presentan son las vistas arquitectónicas propuestas.

- Vista de Casos de Uso.
- Vista Lógica.
- Vista de Implementación.
- Vista de Despliegue.

2.4.2 Objetivos o metas y restricciones arquitectónicas.

Estas metas u objetivos que se proponen, son teniendo en cuenta desde el punto de vista del sistema, los requisitos mínimos y las restricciones que debe cumplir el mismo en cuanto a seguridad, portabilidad, escalabilidad y reusabilidad, siempre y cuando sean significativamente útiles para el sistema.

2.4.2.1 Redes.

La Red existente en las instalaciones donde se hará uso de la aplicación debe soportar transacciones de paquetes para las máquinas en las cuales se usará esta aplicación. Además que deben tener conexión a Internet, una conexión de al menos 56 Kbps.

2.4.2.2 Seguridad.

La seguridad en cualquier sistema, es un punto muy importante que no se debe dejar pasar por alto, por la importancia que tiene. Los sistemas nunca son 100% seguros, las fallas de seguridad se van descubriendo en las fases de prueba de cada ciclo, por lo que es muy importante aplicar medidas extremas de seguridad desde fases de desarrollo iniciales. Para ello se debe garantizar tratamiento de excepciones, además de que parte de la seguridad corre por parte del Framework.Net 2.0.

2.4.2.5 Portabilidad, escalabilidad y reusabilidad.

1. El sistema podrá ser utilizado sobre plataforma Windows.
2. Cada módulo se desarrollará, teniendo en cuenta posibles cambios y ampliaciones del sistema, por lo que podrán ser reutilizados.

3. El sistema debe hacer uso racional de los recursos las máquinas donde se haga uso del mismo.
4. La aplicación se desarrollará haciendo uso de estándares definidos por el equipo de proyecto, teniendo en cuenta los estándares internacionales, para facilitar ampliaciones e integraciones futuras garantizando un mantenimiento fácil.
5. La documentación de la arquitectura debe estar disponible para integrarla como parte del producto final.

2.4.2.6 Restricciones de acuerdo a la estrategia de diseño.

El diseño de la aplicación debe ser desarrollado utilizando la Programación Orientada a Objetos (POO), haciendo uso de las facilidades tecnológicas que ofrece el Framework.Net definidas para cada una de las capas de la aplicación, para la capa de presentación se usará el WindowsForms para aplicación de escritorio; para la capa de negocio se usarán los objetos de esta capa, y en la capa de acceso a datos se usarán funciones ya definidas por la biblioteca del Framewrok.
Cada PC donde se ejecute la aplicación debe tener instalado el Framework.Net en su versión 2.0, este framework está accesible desde la instalación del Visual Studio 2005 o simplemente se puede instalar solo sin la necesidad de instalar el Visual Studio.

2.4.2.7 Estructura del equipo de desarrollo.

Todo proyecto bien organizado y bien planificado siempre se compone por una jerarquía que sirve de ayuda al desarrollo y avance del mismo. En este caso, el proyecto está compuesto por varias líneas de desarrollo, la línea de Posicionamiento Web es un pequeño grupo de proyecto compuesto por:

- Jefe de Proyecto.
- Analista principal.
- Arquitecto.
- Equipo de desarrollo.
- Grupo de Investigación.
- Grupo para Control de la Calidad y Prueba del Sistema.
- 4 módulos de desarrollo.

Por cada Módulo de desarrollo se realiza la siguiente distribución:

- Jefe de Proyecto.
- Analista Principal.
- Equipo de desarrollo.

El desarrollo de la aplicación está compuesta por 4 módulos lógicos. Un módulo es una división conceptual del sistema que puede ser visto como una agrupación de funciones que tengan alguna relación entre ellas y, por lo tanto, puede presentar un servicio completo al exterior una vez se ha desarrollado.

Representación:

Un módulo se representa con el elemento Paquete de UML.

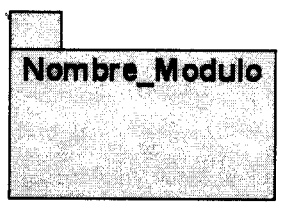


Fig. 8. Representación en UML de un paquete.

Para la primera versión se desarrollaran los siguientes módulos:

- 1er Módulo ----- Consulta.
- 2do Módulo ----- Análisis.
- 3er Módulo ----- Acceso a Datos.
- 4to Módulo ----- Estadístico.

2.4.2.7.1 Configuración de los puestos de trabajo por roles.

Analista.

- PC con mouse y teclado, 256 Mb RAM.
- Visual Paradigm v6.0.
- Microsoft Office 2007.
- **Desarrolladores.**
- PC con mouse y teclado, 256 Mb RAM.

- Visual Studio 2005.
- Framework.Net v2.0.

2.4.3 Vista Casos de Uso.

La Vista de Casos de Uso o el Escenario, es la vista que dirige las restantes vistas. En esta vista se definen los casos de uso más significativos para el sistema desde el punto de vista arquitectónico. Estos casos de uso tienen gran relevancia en cada iteración del ciclo de desarrollo, además de encapsular la funcionalidad del sistema. En la siguiente figura se muestra el diagrama de casos de uso del sistema, en el cual se muestran los principales casos de uso, es decir, los de más relevancia para la arquitectura:

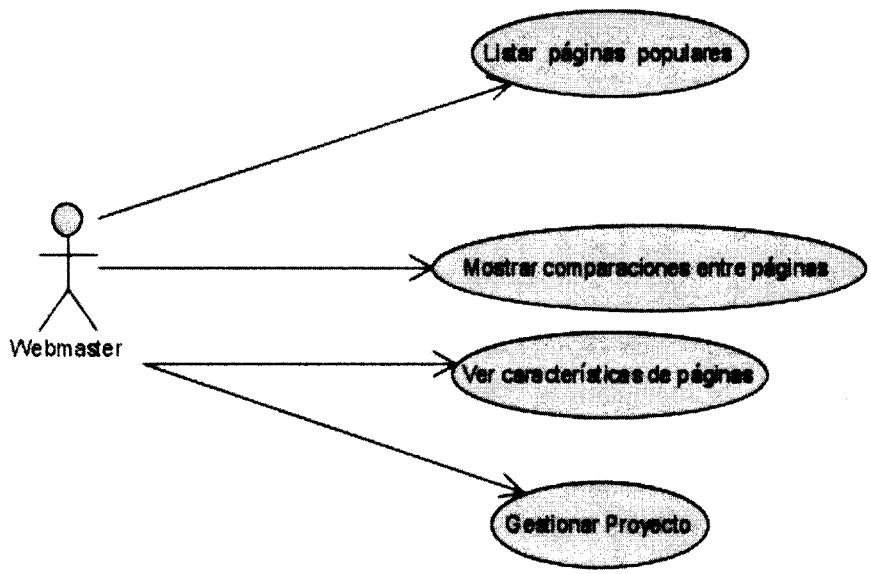


Fig. 8. Diagrama de Casos de Uso del Sistema.
Casos de Uso significativos arquitectónicamente.

2.4.3.1 Módulo Consulta.

Caso de Uso:	**Listar páginas populares.**
Actores:	Webmaster.
Resumen:	El Caso de Uso se inicia cuando al webmaster se le da la opción de hacer la búsqueda por palabras claves o por términos de búsquedas, introduce los datos según haya seleccionado, especifica en que buscador desea realizar el trabajo y entra los datos de conexión, luego el sistema muestra el resultado al usuario.
Precondiciones:	Entrada de palabras claves del sitio o el término de búsqueda del usuario. Selección del buscador. Entrada de datos de conexión.
Referencias	**R1**

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1 El webmaster selecciona la opción de Listar páginas más populares.	2 El sistema brinda la posibilidad de entrar las palabras claves del sitio del webmaster o de entrar términos de búsquedas del usuario, especificar un buscador determinado y entrar los datos de conexión.
3 El webmaster entra las palabras claves de su página o entra los términos de búsqueda del usuario, selecciona el buscador y entra los datos de conexión.	4 El sistema muestra las páginas.
Flujos Alternos	

Acción del Actor	Respuesta del Sistema
3 El webmaster no entra las palabras claves o los términos de búsquedas, no selecciona buscador o no entra datos de conexión.	4 El sistema muestra mensaje de que debe entrar las palabras claves o términos de búsqueda, o que debe seleccionar el buscador, o que debe entra los datos de conexión.
	5 El sistema presenta problemas de conexión y muestra un mensaje de error que no se ha podido conectar.
Poscondiciones	Se muestran las páginas populares.

2.4.3.2 Módulo Análisis.

Caso de Uso:	**Ver características de páginas.**
Actores:	Webmaster.
Resumen:	El Caso de Uso se inicia cuando el webmaster selecciona la opción de ver características de una página de las que se mostraron en el caso de uso (Listar páginas populares), luego el sistema procesa los datos de dicha página y al final muestra las características.
Precondiciones:	Realización del caso de uso: Listar páginas populares. Selección de una página.
Referencias	**R1**
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**

1 El webmaster escoge una de las páginas y selecciona la opción de ver características.	2 El sistema brinda las siguientes características de cada una de las páginas : .Brindar información si posee mapa de navegación para cada página. .Informar la cantidad de tablas, imágenes y enlaces que posee cada página. .Mostrar porciento de texto y código embebido que posee cada página. .Mostrar idioma que posee cada página. .Mostrar sitios que apuntan a cada página. .Mostrar Page Rank para cada página. .Mostrar correo de webmaster. .Mostrar palabras que más predominan. .Mostrar si está posicionado en párrafo o en título. .Mostrar la frecuencia, entropía, peso, relevancia y recuento. .Mostrar enlaces rotos.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
1 El webmaster no escoge una de las páginas.	2 El sistema muestra mensaje de que debe seleccionar una de las páginas.
Poscondiciones	Se muestran las características.

2.4.3.3 Módulo Estadístico.

Caso de Uso:	**Mostrar comparaciones entre páginas.**
Actores:	Webmaster.
Resumen:	El caso de uso se inicia cuando el webmaster escoge la opción de la comparación de las páginas, luego el sistema le da la posibilidad de hacer la comparación por característica o general. Después el sistema brinda un resumen gráfico del comportamiento de las páginas a comparar.
Precondiciones:	Realización del caso de uso: Ver características de páginas. Selección de las páginas.
Referencias	**R4**

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1 El webmaster selecciona la opción de comparación entre páginas.	2 El sistema muestra la posibilidad de escoger las páginas a comparar y por qué categoría comparar.
3 El webmaster escoge las páginas y la forma en que va a comparar.	4 El sistema analiza cada una de las páginas y muestra las gráficas de resumen.

Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
3 El webmaster no escoge las páginas o no	4 El sistema muestra mensaje de que debe

especifica la forma en que va a comparar.	escoger las páginas o de que tiene que especificar la forma de comparación.
Poscondiciones	Se muestra la gráfica de comparación.

2.4.3.4 Módulo Acceso a Datos.

Caso de Uso:	**Gestionar Proyecto**
Actores:	Webmaster.
Resumen:	El caso de uso se inicia cuando el webmaster escoge la opción de Gestionar proyecto, luego escoge la acción a realizar e introduce los datos necesarios. Con esto el sistema realiza la acción seleccionada y termina el caso de uso.
Referencias	**R5, R6, R7.**
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1 El webmaster selecciona la opción de Gestionar proyecto.	2 El sistema brinda las opciones de: a) "Crear Proyecto". b) "Modificar Proyecto". c) "Eliminar Proyecto".
Sección "Crear Proyecto"	
Acción del Actor	**Acción del Sistema**
1 El webmaster selecciona la opción de Crear Proyecto.	2 El sistema pide los datos del nombre del proyecto y la ubicación donde se guardará.

3 El webmaster introduce los datos necesarios.	4 El sistema verifica la información y crea el proyecto.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
3 El webmaster no introduce el nombre o la ubicación.	4 El sistema muestra mensaje de que debe introducir los datos de nombre y ubicación.
Flujo Normal de Eventos	
Sección "Modificar Proyecto"	
Acción del actor	**Respuesta del sistema**
1 El webmaster selecciona la opción de Modificar Proyecto.	2 El sistema muestra la opción de escoger la ubicación donde se encuentra el proyecto, seleccionarlo y cargarlo.
3 El webmaster selecciona el proyecto y lo carga.	4 El sistema muestra los datos referentes a dicho proyecto.
5 El webmaster modifica los datos necesarios.	6 El sistema verifica los datos y luego modifica el respectivo proyecto.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	6 El sistema comprueba que existe algún error en los datos entrados y emite un mensaje de error, para que se introduzcan correctamente.
Flujo Normal de Eventos	

Sección "Eliminar Proyecto"	
Acción del actor	**Respuesta del Sistema**
1 El webmaster selecciona la opción de Eliminar Proyecto.	2 El sistema muestra la opción de escoger el proyecto a eliminar.
3 El webmaster escoge el proyecto a eliminar y ejecuta la acción de eliminar el proyecto seleccionado.	4 El sistema le muestra los datos del proyecto y pide verificación de eliminación.
5 El webmaster confirma que desea eliminar los datos.	6 El sistema elimina el proyecto seleccionado.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
5 El webmaster escoge la opción de no eliminar el proyecto.	6 Culmina el caso de uso sin ejecutar ninguna acción.
Poscondiciones	Se crea el proyecto. Se modifica el proyecto. Se elimina el proyecto.

2.4.4 Vista Lógica.

Esta vista representa un subconjunto del artefacto Modelo de diseño, la cual representa los elementos del diseño más importantes para la arquitectura del sistema. Este describe las clases más importantes, su organización en paquetes y subsistemas. También describe las realizaciones de casos de uso más importantes, como por ejemplo, las que describen aspectos dinámicos del sistema.

Una de las mejores formas de representar la estructura y organización modular del sistema, es a través de un diagrama UML que muestre los módulos y su relación. En la siguiente figura se muestra dicho diagrama para darle solución a la propuesta arquitectónica desde esta vista:

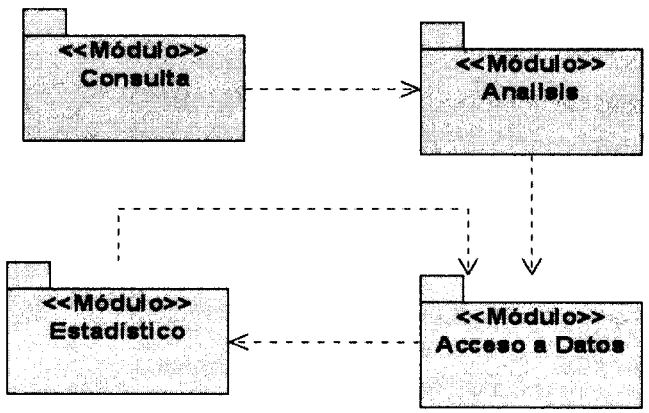


Fig. 9. Representación de los Módulos para el Sistema.

En la figura se pretende mostrar la división del sistema en módulos, con el objetivo de promover la reusabilidad y facilitar el mantenimiento del sistema ante cambios en los procesos de negocio, garantizando que solo se modifique el módulo al que pertenece la funcionalidad afectada por el cambio. También se facilita el trabajo del equipo de desarrollo ya que permite que se puedan desarrollar módulos paralelos, con sus dependencias.

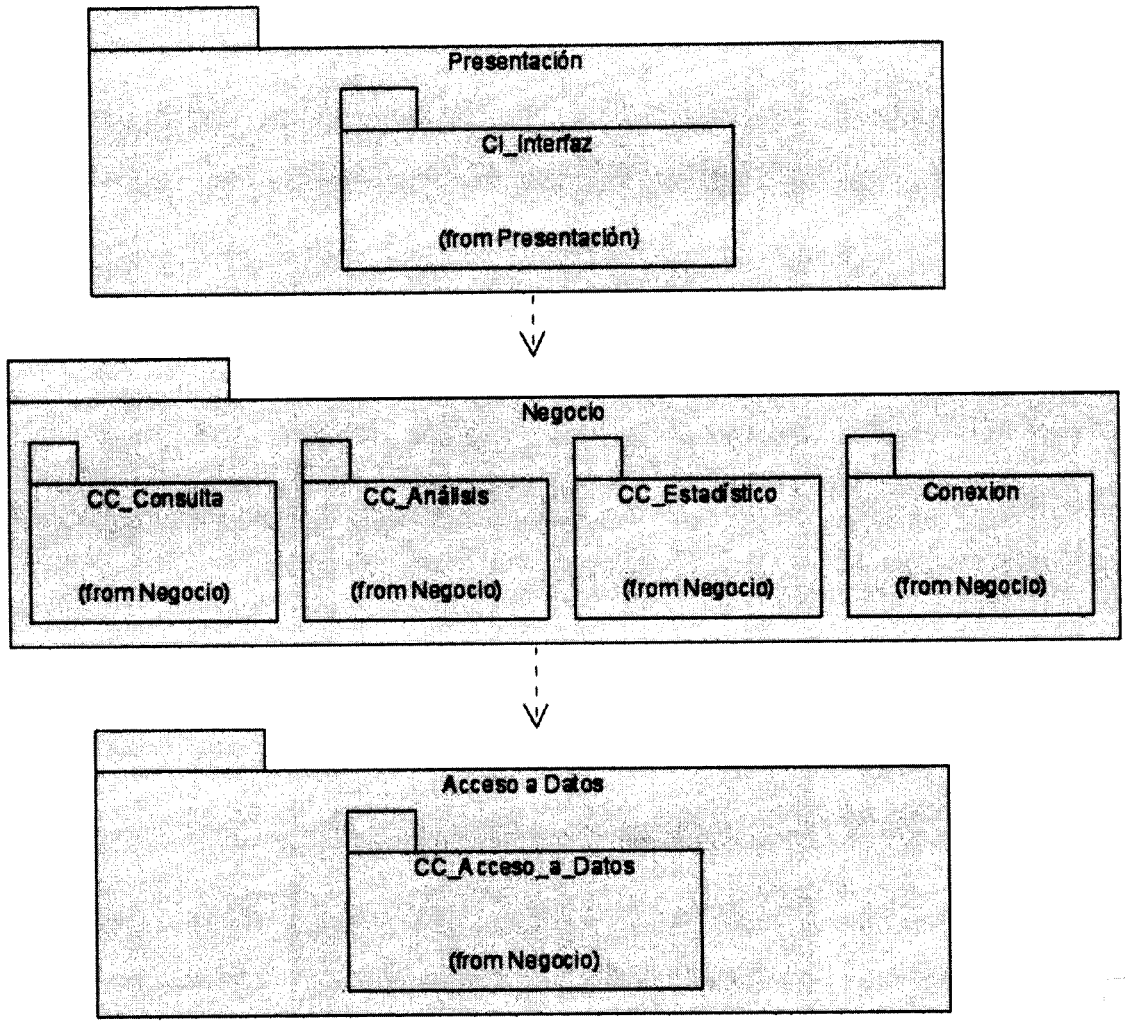


Fig. 10. Distribución del sistema por Paquetes más significativos.

La figura 10 no es más que una representación y alineamiento de los paquetes en las capas del sistema, esto da una visión lógica de cómo estará distribuido el sistema en paquetes de clases, dándole solución a los casos de uso más significativos.

En la capa de Presentación se encuentra un paquete de clases (Interfaz), en el cual se agrupan todas las clases de interfaz de usuario de la herramienta. Como es un pequeño sistema y es desktop, no lleva un gran número de clases de Interfaz.

Sin embargo para la Lógica del Negocio se recogen una serie de paquetes los cuales se encuentran en cada uno de los pequeños módulos del sistema. Estos paquetes son los más significativos para la arquitectura y a través de ellos se les da solución a los casos de uso con sus respectivas clases. Se hace referencia solamente a los paquetes CC_Consulta y Conexión del módulo Consulta.

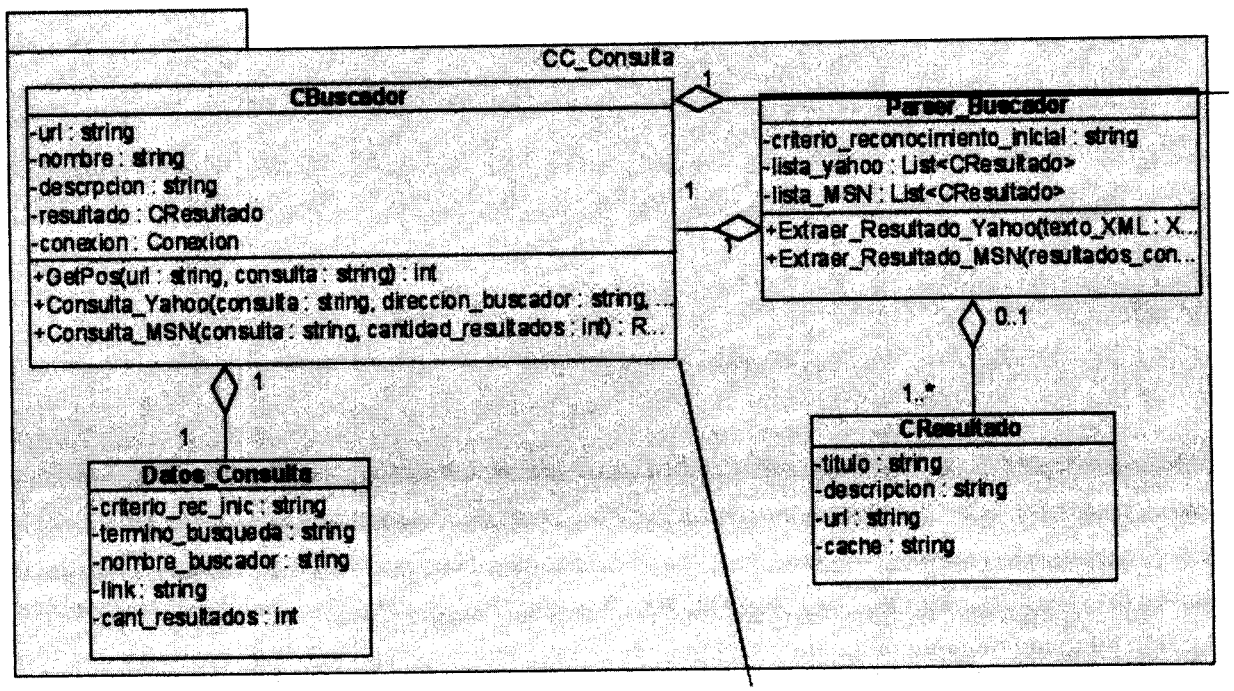


Fig. 11. Paquete CC_Consulta del Caso Uso ***Listar páginas populares.***

Este paquete da solución al caso de uso Listar páginas populares, precisamente en la clase Parser_Buscador, la cual recibe de la clase CBuscador los arreglos que contienen la información requerida para ser parseada y extraída en los métodos Extraer_Resultado_Yahoo y Extraer_Resultado_MSN. Este paquete depende del paquete Conexión para las credenciales de Red. La clase CBuscador, cuenta con dos métodos importantes que son Consulta_Yahoo y Consulta_MSN, los cuales realizan consultas a los buscadores internacionales correspondientes consumiendo servicios Web, en el caso de Yahoo utiliza consumo por la tecnología REST y en el caso del MSN a través de tecnología WSDL.

Con la tecnología REST para Yahoo se consume una página XML que devuelve la consulta que se le realiza al buscador a través de la dirección:
http://search.yahooapis.com/WebSearchService/V1/contextSearch?
Esta URL hay que ir completándola con atributos que permiten obtener una consulta. Un ejemplo sería:
http://search.yahooapis.com/WebSearchService/V1/webSearch?appid=YahooDemo&query=Cuba&results=2
Donde:

- appid es un ID que se obtiene y es una llave de acceso para poder consumir este servicio.
- query es la consulta que se desea realizar.
- results es la cantidad de resultados que se desea obtener.

Existen otros atributos de cómo se puede consumir este servicio, visibles en http://developer.yahoo.com/search/Web/V1/webSearch.html.

Un ejemplo de código de cómo se podría realizar una pequeña aplicación sería:

```
   static void Main(string[] args)
{
// Captura la consulta desde la consola.
      Console.WriteLine("Su consulta para www.yahoo.com's Web Search es:");
      string query = Console.ReadLine();
      // Si consulta es vacia, finaliza.
      if (query == "") return;
  foreach (XmlNode node in GetImageSearch(query))
      {
           Console.WriteLine("{0} @ {1} ",
     //Se estrae titulo y URL de cada resultado
       node["Title"].InnerText,
     node["URL"].InnerText);
  }

 }
  static XmlNodeList GetImageSearch(string query)
  {

   // Crea la respuesta Web
      Uri address = new Uri("http://search.yahooapis.com/WebSearchService/V1/contextSearch?");
       string appId = "YahooDemo";

       HttpWebRequest request = WebRequest.Create(address) as HttpWebRequest;
```

```
        // Metodo de envio es por post
        request.Method = "POST";
        request.ContentType = "application/x-www-form-URLencoded";
        // creando los datos para autocompletar la URL
        StringBuilder data = new StringBuilder();

        data.Append("appid=" + appId);
        data.Append("&query=" + query);
        // Creando el array de los datos a enviar
        byte[] byteData = UTF8Encoding.UTF8.GetBytes(data.ToString());
        // Tamaño del contenido enviado
        request.ContentLength = byteData.Length;

 // Escribiendo datos
     using (Stream postStream = request.GetRequestStream())
     {
          postStream.Write(byteData, 0, byteData.Length);
}
 //

string DireccionDelProxy = "10.0.0.1"; //La URL o dirección IP del servidor proxy
int PuertoDelProxy = 8080; //Puerto por el que escucha el proxy
//Se crea una instancia del objeto proxy
System.Net.WebProxy miProxy = new System.Net.WebProxy(DireccionDelProxy, PuertoDelProxy);
//y se asigna al envoltorio del servicio Web (o similar)
request.Proxy = miProxy;
request.Credentials = new System.Net.NetworkCredential("user", "password", "dominio");

     using (HttpWebResponse response = request.GetResponse() as HttpWebResponse)

     {
          // Get the response stream
         StreamReader reader = new StreamReader(response.GetResponseStream());
         XmlDocument xmlOutput = new XmlDocument();
         xmlOutput.LoadXml(reader.ReadToEnd());
         XmlNodeList resultsNodes = xmlOutput.GetElementsByTagName("Result");
         return resultsNodes;
}


    }
```

Pero para consumir servicios Web de MSN se necesita otra tecnología y es a través de la SOAP. Para ello se necesita descargar a través de las referencias Web del Visual Studio el fichero wsdl que permite consumir dicho servicio. Este fichero está disponible en:
http://soap.search.msn.com/webservices.asmx?wsdl
En este fichero se encuentran todas las funciones necesarias para realizar el trabajo que se desea desarrollar con este buscador.

Google cuenta con sus servicios también. En estos últimos tiempos ha desarrollado muchos servicios Web que pueden ser consumidos desde varias formas. También cuenta con un fichero wsdl que permite consumir funciones necesarias para nuestro trabajo. Este fichero está disponible en:
http://api.google.com/GoogleSearch.wsdl

Pero tiene un inconveniente hacer el uso de estas funciones brindadas, que desde el año 2005 Google no permite crear nuevas claves de acceso para consumir este servicio. Por lo que se recomienda seguir estudiando otras formas de consumir servicios Web para Google. Aunque permite la opción de crear un Api Key para consumir este servicio en:
http://code.google.com/apis/ajaxsearch/signup.html

Para más información de este servicio se puede consultar la página:
http://code.google.com/apis/soapsearch/reference.html

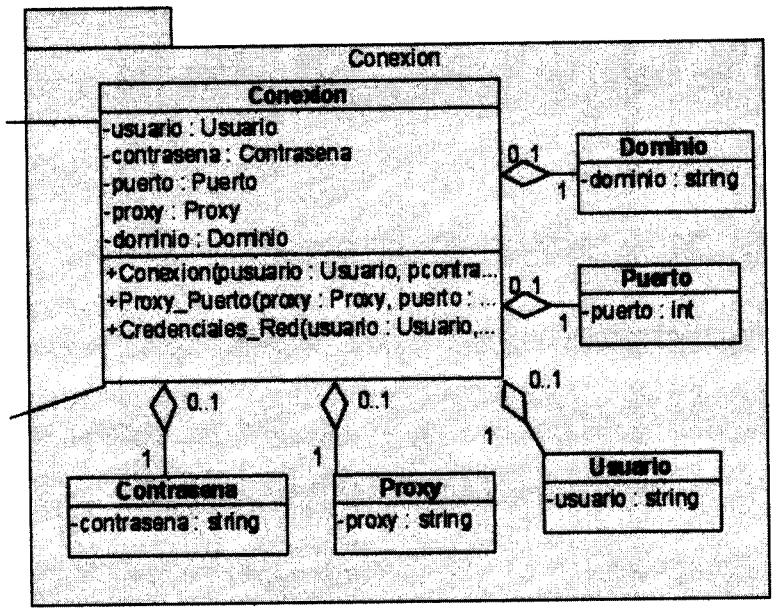


Fig. 12. Paquete Conexión del Caso Uso ***Listar páginas populares.***

El paquete Conexión representado en la figura 12, cuenta con la información necesaria para realizar todas las conexiones a través de proxy, brindando a los demás paquetes las credenciales de Red para poder realizar las consultas y conexiones a Internet cada vez que se requiera. El método Proxy_Puerto devuelve un tipo de dato de tipo WebProxy el cual es necesario para identificar el proxy y el método Credenciales_Red devuelve un tipo de dato de tipo NetWorkCredentials el cual contiene usuario, contraseña y dominio. Este paquete es muestra de cuan escalable es la herramienta ya que es un paquete que generaliza y gestiona las credenciales de la Red para cualquier conexión, este paquete es usado también en otros paquetes para poder realizar conexiones a Internet.

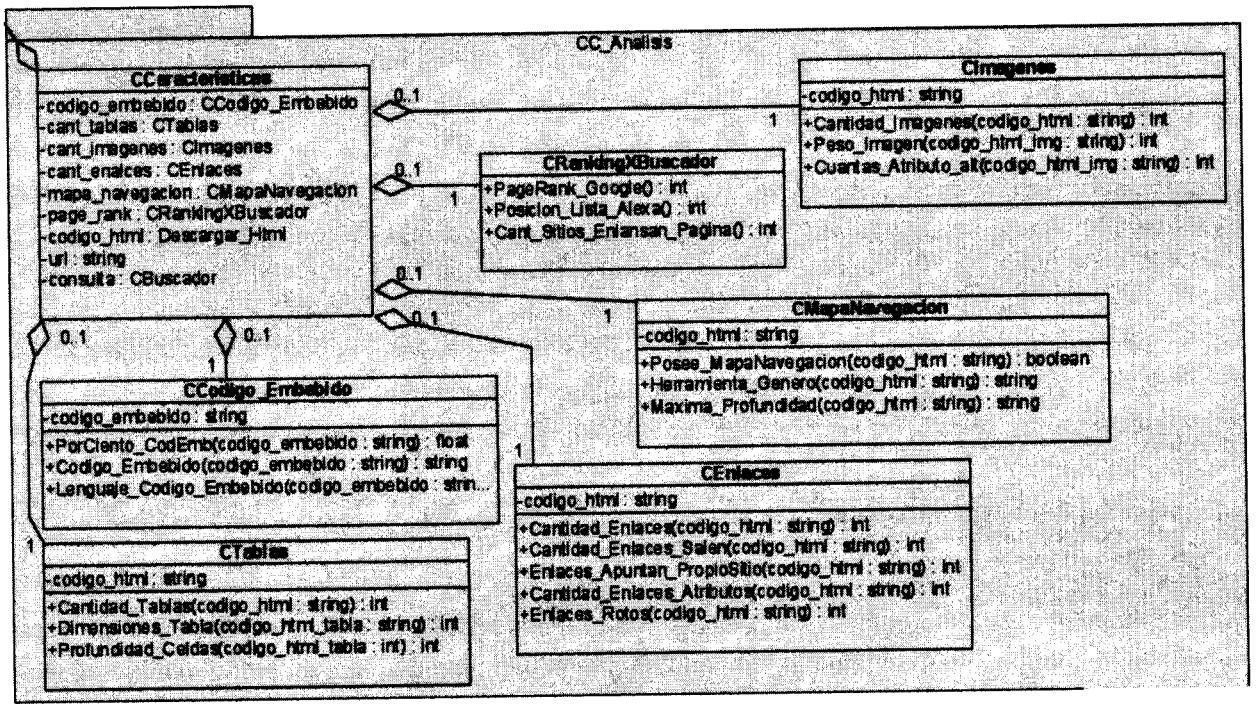


Fig. 13. Paquete CC_Analisis del Caso de Uso ***Ver características de páginas****.*

El paquete de la figura 13 representa las clases del diseño que le dan solución al caso de uso Ver características de páginas. La clase principal en este paquete es la clase CCaracterísticas el cual contiene todas las demás clases del paquete. El resto de las clases contienen funciones importantes para brindar al webmaster, mucha de esta información será utilizada en el paquete CC_Estadistico. Este paquete recibe la lista de URL que se obtiene en el paquete CC_Consulta y a cada una de esas URL, es a la que se le va a extraer información ya sea cantidad de imágenes que posee, cantidad de tablas, si contiene código embebido y si posee mapa de navegación entre otras informaciones importantes.

En la capa de persistencia de datos o Acceso a Datos se encuentra un solo paquete (Acceso_a_Datos) que recoge las clases que se encargarán de los datos. La herramienta no usará ningún gestor de Bases de Datos para almacenar los datos. Los datos serán guardados en documentos XML los cuales permitirán la gestión de todos los datos de cada paquete de la Lógica del Negocio.

2.4.5 Vista de Implementación.

La Vista de Implementación es un proceso que describe la descomposición del sistema o de la aplicación que se desea desarrollar en capas y subsistemas de implementación. También ofrece una vista de la trazabilidad de los componentes del diseño de la vista lógica, pero aplicada a la vista de implementación. Es decir muestra el software desde un punto de vista de los elementos físicos que lo componen, ya sean librerías, subsistemas, componentes, ficheros, etc.

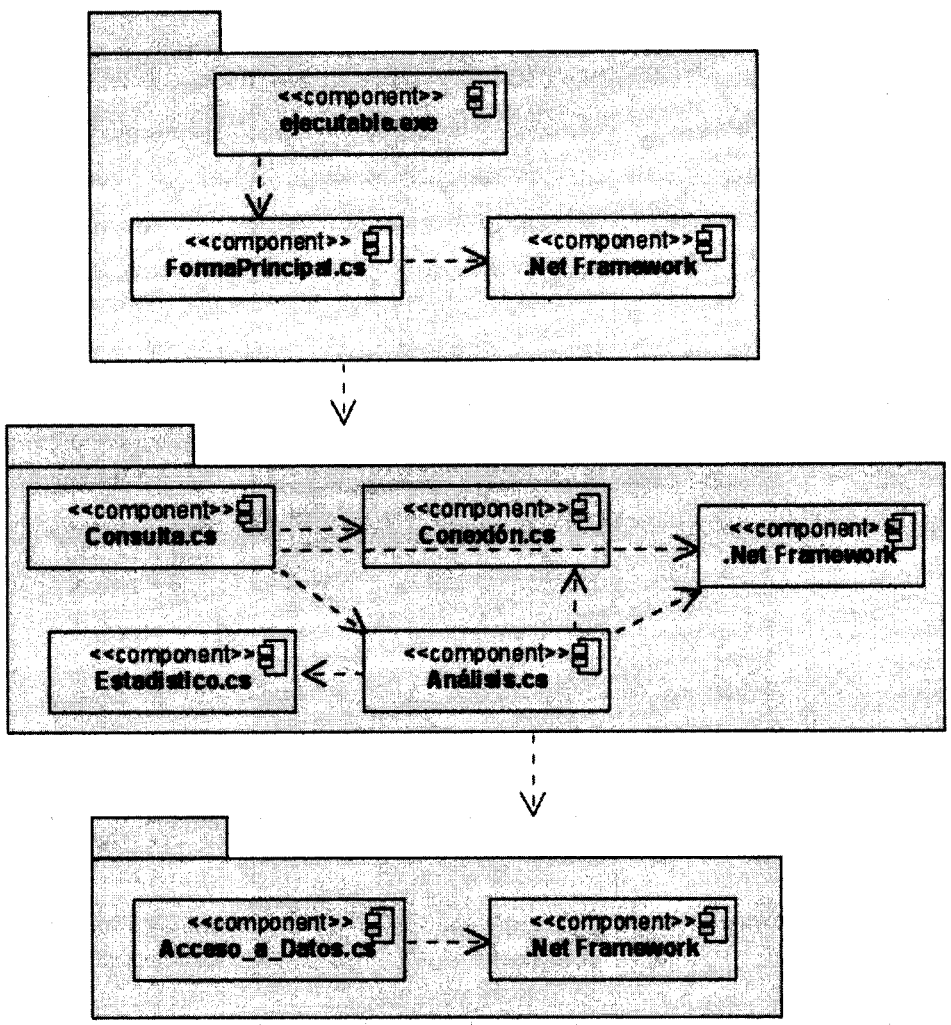


Fig. 14. Vista General del Modelo de Implementación para la herramienta.

En la figura 14 se muestra una representación general de la herramienta con sus principales componentes. Los componentes seleccionados se corresponden con la etapa actual del proyecto que son Análisis y Diseño.

Los componentes que se identificaron fueron siete, los cuales tienen una prioridad máxima para el desarrollo e implementación del software y a interés de los líderes del proyecto. En el primer paquete se represento el componente FormaPrincipal.cs que no es más que la interfaz de usuario que verá el cliente cuando interactúe con la herramienta. La mayor cantidad de componentes se encuentran distribuidos en el segundo paquete. Como se había planteado anteriormente "Consulta", "Estadístico" y "Análisis" son módulos a desarrollar por lo tanto se tomaron como componentes importantes de la herramienta, se representa otro componente Conexión.cs ya que éste forma parte de la herramienta del cual dependen otros dos componentes. En el tercer paquete se observa claramente otro componente, que dicho anteriormente representa un módulo. Y un componente muy importante es el que brinda el IDE Visual Studio .Net, el cual facilita el trabajo para los programadores y del cual dependen todos los paquetes por la cantidad de bibliotecas de clases que contiene este framework.

Entre los componentes existen relaciones de dependencia, las cuales significan que si se modificar un componente del cual depende otro, este otro hay que modificarlo. Por lo que se trató de en todo momento de no hacer tanta dependencia para que el sistema sea lo más coherente y reutilizable posible.

2.4.6 Vista de Despliegue.

Esta vista suministra una base para la comprensión de la distribución física de un sistema a través de nodos. Suele utilizarse cuando el sistema está distribuido. Y hay una traza directa del modelo de implementación, puesto que cada componente físico debe estar almacenado en un nodo, esto incluye también la asignación de tareas provenientes de la vista de procesos en los nodos. También en esta vista se le da parte de la solución a los requisitos no funcionales, es decir, los requerimientos de hardware y software.

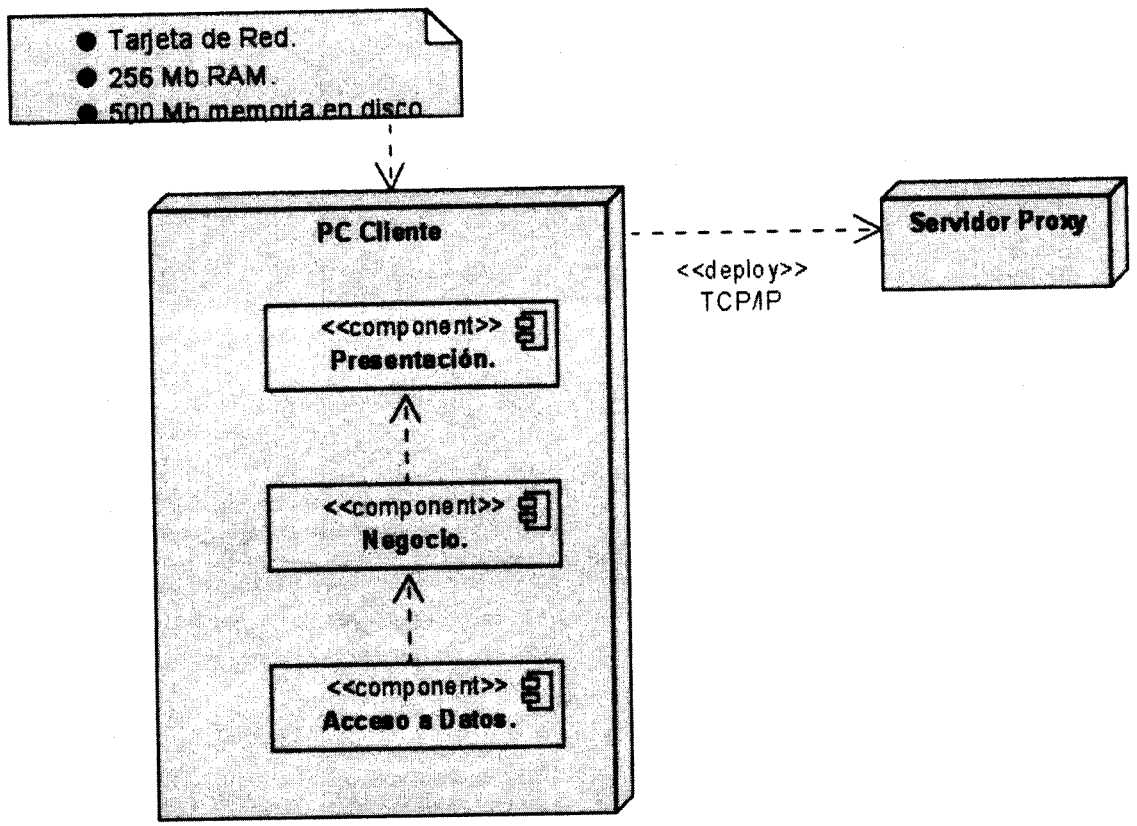


Fig. 15. Diagrama de despliegue para el Sistema.

En el nodo cliente (PC Cliente) es en el cual se instalará la aplicación una vez se desee trabajar con ella, en este nodo se nota la presencia de los tres componentes importantes para ejecutar la aplicación, como es la Presentación, el Negocio para desarrollar toda la lógica del dominio y el Acceso a Datos, estos ficheros XML se guardaran en la PC Cliente, disponible para los usuarios finales.

Este nodo cliente realiza una conexión a Internet por un segundo nodo (Servidor Proxy), a través de su tarjeta de red, usando el protocolo TCP/IP teniendo en cuenta las restricciones de Red descrita anteriormente.

En la nota que se observa en la figura 14, se muestran algunas especificaciones de software y hardware para ejecutar la aplicación final.

2.4.6. Vista de Datos.

La persistencia de datos determinada para la herramienta a desarrollar fue almacenar los datos en documentos XML. Esto se tomó a consideración y se determinó así porque la herramienta no tiene que realizar grandes procesamientos de datos que conlleve a la creación de una Base de Datos (BD). Además de que al almacenar los datos en este tipo de documentos permite para versiones siguientes un manejo fácil de estos ficheros, ya que si se desea en un futuro ampliar el sistema y brindar servicios Web, se pueden usar esos mismos ficheros para la transferencia sin tener que realizar tantos cambios para el protocolo y la estandarización de los datos.

Los documentos XML se estructuran de forma jerárquica, como si fuera un árbol. Estos documentos comienzan siempre con una entidad documento, que es la raíz del mismo. También forman parte de estos documentos declaraciones, elementos, comentarios, referencias a caracteres e instrucciones de procesamiento.

Los documentos XML se dividen en dos grupos, documentos bien formados(Reino 2000) y documentos válidos(Reino 2000).

- Bien formados: Son todos los que cumplen las especificaciones del lenguaje respecto a las reglas sintácticas sin estar sujetos a unos elementos fijados en un DTD. De hecho los documentos XML deben tener una estructura jerárquica muy estricta y los documentos bien formados deben cumplirla.
- Válidos: Además de estar bien formados, siguen una estructura y una semántica determinada por un DTD: sus elementos y sobre todo la estructura jerárquica que define el DTD, además de los atributos, deben ajustarse a lo que el DTD dicte.

¿Qué es DTD?

Siglas en inglés de Document Type Definition. La definición de tipo de documento (DTD) es una descripción de estructura y sintaxis de un documento XML o SGML. Su función básica es la descripción del formato de datos, para usar un formato común y mantener la consistencia entre todos los documentos que utilicen la misma DTD. De esta forma, dichos documentos, pueden ser validados, conocen la estructura de los elementos y la descripción de los datos que trae consigo cada

documento, y pueden además compartir la misma descripción y forma de validación dentro de un grupo de trabajo que usa el mismo tipo de información.

Los documentos XML son estándares ya que permiten el intercambio de información estructurada entre varias plataformas, estos tienen sus reglas de estructuración, que algunas pueden omitirse, pero otras no. Los documentos XML siempre deben comenzar con una línea de código que especifique la versión XML que usa ese documento, por ejemplo:

<? Xml version="1.0" encoding=" UTF-8" standalone= "yes "?>
Como se puede observar, en esa línea de código, se observan algunos atributos:

- Versión: Indica la versión de XML usada en el documento. Es obligatorio ponerlo, a no ser que sea un documento externo a otro que ya lo incluía.
- Encoding: La forma en que se ha codificado el documento. Se puede poner cualquiera, y depende del parser el entender o no la codificación. Por defecto es UTF-8, aunque podrían ponerse otras, como UTF-16, US-ASCII, ISO-8859-1, etc. No es obligatorio salvo que sea un documento externo a otro principal.
- Standalone: Indica si el documento va acompañado de un DTD ("no"), o no lo necesita ("yes"); en principio no hay por qué ponerlo, porque luego se indica el DTD si se necesita.

La propuesta del documento que se desea generar para almacenar los datos es el siguiente:

```
<?xml version="1.0" encoding="utf-8" ?>
<Proyecto>
 <fecha> </fecha>
 <Buscador>
  <Consulta>
  <nombrebuscador></nombrebuscador>
  <consulta></consulta>
  <cantidaURL></cantidaURL>
   </Consulta>
   <ListadoURL>
    <direccion_URL></direccion_URL>
    <URL>
     <CaracteristicaURL>
      <mapnavegac>
      </mapnavegac>
      <canttablas>
```

```
        </canttablas>
        <cantimg>
        </cantimg>
        <cantlink>
        </cantlink>
        <codigoemb>
        </codigoemb>
        <idioma>
        </idioma>
        <caracttitulo>
        </caracttitulo>
        <pagerank>
        </pagerank>
        <emailwebmaster>
        </emailwebmaster>
        <Predominiopalabra>
          <recuento>
          </recuento>
          <frecuencia>
          </frecuencia>
          <entropia>
          </entropia>
          <relevancia>
          </relevancia>
          <peso>
          </peso>
        </Predominiopalabra>
        <badlink>
        </badlink>
      </CaracteristicaURL>
    </URL>
  </ListadoURL>
  <Estadistica>
  </Estadistica>
 </Buscador >

</Proyecto>
```

Esquema del documento XML para la persistencia.

El documento cuenta con varios nodos y una raíz que es <Proyecto>. Para diferenciar los datos unos de otros es la fecha en que se almaceno dicho documento. Como se trata de almacenar datos de varios buscadores, de cada uno de ellos se almacena el nombre del buscador (<nombrebuscador>), consulta realizada a ese buscador (<consulta>) y cantidad de URL que se quieran analizar (<cantidadURL>). Pero de cada una de estas URL se puede almacenar las características

(<CaracteristicaURL>) de ellas, ya sea cantidad de imágenes, si posee mapa de navegación o no, entre otras que aparecen descritas en los requerimientos funcionales.

2.5. Conclusiones del Capítulo 2.

Este capítulo es el más importante para este documento ya que en el se recogen todas las características, restricciones, vistas y elementos más significativos para la arquitectura de la herramienta.

En la línea base de la arquitectura se muestra los requerimientos funcionales y no funcionales del sistema, además de que estos a la vez son restricciones que se tienen que cumplir estrictamente para evitar posibles fallos.

En la descripción de la arquitectura se muestran las diferentes vistas arquitectónicas, mostrando los elementos más importantes para la arquitectura. Hay que aclarar que a los diseños de clases representados les falta lo que diríamos un poco de condimento, que se irá retroalimentando según se valla implementando y desarrollando dicho software. También se muestran los principales patrones de diseño y se muestra el patrón arquitectónico a aplicar, patrón en Capas, en los cuales se da una breve descripción para tener idea de cómo aplicarlos.

CAPÍTULO 3: EVALUACION DE LOS RESULTADOS.

3.1 Introducción al Capítulo.

El documento de la arquitectura es un artefacto muy importante que comienza a hacer sus primeros textos desde la fase de elaboración. En esta fase se obtiene como artefacto fundamental una línea base de la arquitectura. Es decir, desde esta fase se define, se valida y establece la arquitectura del sistema. Además en esta fase la arquitectura propuesta demostrará que soportará la visión con un costo asequible en un determinado tiempo. Con la elaboración de la línea base de la arquitectura se logra la escalabilidad, capturando la mayoría de los requerimientos y reduciendo los riesgos más importantes. También se identifican los casos de usos más significativos para el sistema.

Ya desde esta fase queda definida la arquitectura base del sistema de la herramienta que hará los análisis correspondientes para apoyar las decisiones de los webmasters, dando la posibilidad de crear algunos elementos funcionales del sistema, como la realización de consultas a los buscadores internacionales, mostrar el resultado de estas consultas y extraer algunas de las características de estos enlaces obtenidos en las consultas. Es muy obvio que no todos los requisitos funcionales se desarrollan en esta fase, pero ya lo que se ha ido desarrollando se va poniendo a prueba, permitiendo identificar los principales problemas y riesgos e ir mitigándolos con el proceso de eliminación de errores. De esta forma se cumple con algunos de los requisitos no funcionales, como es la calidad de la herramienta.

Son muy importantes estos puntos que hemos abordado aquí, que son los requisitos funcionales y la calidad del software. Un sistema para cumplir con la exigencia de los clientes debe satisfacer de forma eficiente a los mismos, demostrando una seguridad en su funcionalidad, y sin calidad no existe la eficiencia, por lo que esta juega un papel muy importante para evaluar un sistema. Los requisitos funcionales son uno de los puntos de gran relevancia que se tienen en cuenta para evaluar la calidad del sistema, además de otros requisitos no funcionales que tienen igual relevancia.

3.2 Métodos de evaluación de la Arquitectura.

Existen disimiles métodos de evaluación de la arquitectura de un sistema. Hay que tener en cuenta que el éxito o fracaso de un sistema de software puede estar determinado por su arquitectura, y existen métodos que permiten saber e identificar si un equipo de proyecto eligió la arquitectura

correcta para el desarrollo de un sistema. Es muy importante realizar una evaluación de la arquitectura, siendo esto una forma bastante económica de evitar desastres, encontrando debilidades del sistema.

La evaluación de la arquitectura según el tiempo de realización se puede clasificar en Evaluación Temprana o Evaluación Tardía. Normalmente los que realizan las evaluaciones de arquitecturas son los equipos de evaluación del proyecto o los probadores (Stackeholders).

Las cualidades que pueden ser evaluadas en la arquitectura son:

- Presentación.
- Disponibilidad.
- Seguridad.
- Facilidad de Modificación.
- Los atributos de calidad tienden a ser muy imprecisos para el análisis, ya que el sistema deben ser robusto, seguro, modificable y debe tener una presentación aceptable.

Al final del análisis de la evaluación de la arquitectura se debe tener definido un listado de los atributos de la calidad requeridos para evaluar la arquitectura que está siendo evaluada, también se debe presentar los posibles riesgos y no riesgos que puedan entorpecer el desarrollo del sistema a partir de la arquitectura.

Los costos y beneficios que puede ofrecer una evaluación de la arquitectura son que reúne a los probadores, crea una articulación en las metas específicas de la calidad y deja una explicación clara y precisa de la arquitectura.

Un primer paso para comenzar a evaluar la arquitectura, es revisando activamente el diseño del sistema. Pero oficialmente existen métodos de evaluación de la arquitectura de sistemas, algunos de estos métodos son:

- Software Architecture Analysis Method (SAAM): Fue creado inicialmente para predecir la facilidad de modificación del sistema. Los propósitos de este método son: (1) que es basado en escenarios, (2) crea un foco de modificabilidad y (3) permite evaluar una arquitectura o

comparar y evaluar varias arquitecturas. Pero presenta debilidades ya que la generación de escenarios está basada en la visión de los interesados (clientes), no ofrece una métrica evaluable que sea clara para aplicársela a la calidad y el equipo de evaluación se basa en la confianza que ponen sobre la experiencia de los arquitectos para proponer arquitecturas candidatas. Este método es recomendable usarlo cuando el atributo de calidad Modificabilidad es el de más interés a evaluar.

- Architecture TradeOff Analysis Method (ATAM): Es un método de evaluación que deja claro cuando una arquitectura satisface las metas de calidad, también ofrece ideas de cómo esas metas interactúan entre ellas, a eso es lo que se le llama TradeOff. Es un método de evaluación que presenta nueve pasos divididos en cuatro grupos, encontrando los puntos sensitivos y tradeoff entre los atributos de calidad. Para aplicar este método de evaluación hay que tener en cuenta de que es un método que profundiza en la evaluación de aspectos más relacionados con la arquitectura, como la Presentación y la Confiabilidad.

- An Evaluation Method for Partial Architectures (ARID): Es un método de evaluación que se recomienda aplicar en etapas tempranas del desarrollo del sistema, para evaluar los diseños de estas etapas. Este método cuenta con Revisiones de Diseños Activas (ADR), las cuales se utilizan para la evaluación de diseños detallados de unidades del software, ya sean componentes o módulos. Este método evalúa mejor la factibilidad de la arquitectura.

3.3 Atributos para la evaluación de la calidad de la arquitectura.

Se han mencionado algunos atributos de la calidad, los cuales son los criterios que se utilizan para evaluar las arquitecturas. Para la evaluación de este documento, se seleccionarán atributos de la calidad establecidos en el estándar ISO 9126 para medir la calidad del software. Este estándar propone seis características que debe tener la calidad de un producto. En la siguiente tabla se presentan estas características.

Características	Subcaracterísticas	Definiciones
Funcionabilidad	Adecuado	Capacidad del software para brindar un conjunto de funciones para tareas de los usuarios específicas.

	Exacto.	Capacidad del producto de software para proporcionar los resultados o efectos acordados.
	Interoperabilidad	Capacidad del producto de software para habilitar la interacción con uno o más sistemas especificados.
	Cumplimiento	Capacidad del producto de software para tomar normas, convenciones o regulaciones en leyes y prescripciones similares.
	Seguridad	Atributo del software que protege los datos y la información de manera que los usuarios no autorizados no puedan modificar o leerlos, ya sea accidentalmente o intencional.
Fiabilidad	Madurez	Atributo del software para evitar fallos frecuentes en el software.
	Tolerancia a Fallos	Atributo del software que mantiene un nivel especifico de presentación en caso de un fallo del software o en caso de infringir sus interfaces específicas.
	Recuperación	Capacidad del software de recuperar los datos en caso de haber un fallo en el mismo.
Usabilidad	Entendimiento	Este atributo le permite al usuario entender si el software es adecuado y como puede ser usado para algunas tareas.
	Capacidad de ser aprendido	Atributo que permite a los usuarios aprender sobre la aplicación.
	Operatibilidad	Este atributo les permite a los usuarios controlar y operar el software.
Eficiencia	Comportamiento temporal	Atributos que se encargan de los tiempos de procesamiento.
	Comportamiento de los recursos	Capacidad del software para controlar las cantidades de un determinado recurso cuando el software esté realizando una función bajo determinadas condiciones.

Mantenibilidad	Capacidad para ser analizado	Atributo que se encarga de analizar los fallos o deficiencias del sistema, identificando las partes que deben ser modificadas.
	Capacidad para ser cambiado	Aquí se muestra la capacidad del software para ser modificado, permitiendo que cualquier modificación pueda ser implementada.
	Estabilidad	El software debe ser capaz de evitar efectos inesperados producidos por las modificaciones.
	Capacidad para ser probado	Atributo que permite mostrar que las modificaciones que se le realicen al software sean validadas.
Portabilidad	Capacidad de ser adaptado	Atributo encargado de verificar que el software pueda adaptarse a diferentes entornos, sin aplicar acciones o mecanismos distintos de aquellos proporcionados.
	Instalable	Atributo responsable de que el software se instale en un entorno especifico.
	Coexistencia	Este atributo comprueba que el software coexista con otro software independiente y compartan recursos comunes.
	Reemplazar	Capacidad del software de ser usado en lugar de otro software para cumplir un mismo propósito, en el mismo entorno.

Tabla 1. Atributos para la calidad del software. ISO 9126.

Estos atributos anteriormente mencionados, con sus características, son aplicados y usados en las pruebas que se le realizan al producto final. Pero no se pueden obviar en el desarrollo de la arquitectura, de hecho, esta debe cumplir estrictamente cada una de estas características, para lograr un buen desarrollo de la herramienta. El principal análisis del presente capítulo está dirigido a demostrar y verificar que la arquitectura propuesta cumple con los atributos de la calidad, aplicando el método de evaluación ARID.

Funcionalidad

La arquitectura de un software debe ser funcional. Esto se basa en la definición de los casos de usos más significativos para el sistema, por los cuales se van a regir y dirigir todos los procesos del desarrollo del software, es decir, estar en el punto de vista de los usuarios para poder satisfacerlos. La selección de los casos de uso significativos ofrece una arquitectura inicial que va creciendo a medida que se hacen las siguientes iteraciones, implementándose nuevos casos de uso, garantizando siempre el cumplimiento de los requerimientos funcionales.

El atributo "adecuado" es la capacidad que tiene el producto de software para brindar un conjunto de funciones para tareas de los usuarios específicas. En la arquitectura propuesta se da solución a las tareas y propuestas hecha por los usuarios, teniendo en cuenta todos los requisitos funcionales y no funcionales que debe cumplir el software. La arquitectura desarrollada demuestra una adecuación correcta, debido a que la misma es capaz de soportar todo lo necesario para desarrollar herramientas que ayuden a los webmaster a tomar decisiones importantes en el ámbito del posicionamiento Web a partir de un análisis. La consulta a buscadores internacionales a partir de términos específicos de búsquedas, listar las URL de los sitios mejor posicionados en un determinado buscador, presentar las características de cada una de estas URL y mostrar una comparación estadística de esas URL, son requisitos funcionales soportados en la arquitectura que se propone y da como consecuencia, la capacidad de adecuación del producto.

La "exactitud" es otro atributo que se cumple en toda la extensión de la palabra en la arquitectura propuesta. Los detalles juegan un papel fundamental, por lo que un pequeño error en un resultado dado por el software, puede traer consecuencias desagradables para los usuarios, ya que las decisiones que toman los mismos, son a partir de los resultados que muestre la herramienta.

La arquitectura también cumple con el atributo "interoperabilidad", o sea, la capacidad del software de poder compartir, enviar y manipular información con otros sistemas. Este parámetro se ve bastante claro en la forma en que se guarda la información del software. Esos datos generados por el sistema se guardan en ficheros XML, un lenguaje de etiquetado estándar, compatible con varias plataformas de desarrollo.

La seguridad de acceso es otro de los atributos de la calidad que se tiene en cuenta en la arquitectura. La herramienta debe mantener una integridad de sus datos, debe ser capaz de que los datos sean modificados y usados por las personas y sistemas autorizados.

Fiabilidad

El sistema ante cambios externos o internos puede reaccionar de maneras diferentes, en ocasiones asimila estos cambios pero en otras ocasiones no, por lo que la tolerancia a fallos es un factor muy importante, por lo que la arquitectura toma en cuenta componentes que no permiten entradas de datos no validas para el sistema o que pueda dañar la estructura de la herramienta. En caso de que el usuario trate de valerse de otros medios para perjudicar el sistema, éste le mostrará errores muy explícitos y la información no será procesada hasta que los datos sean correctos.

Para llegar a este punto el sistema debe tener una cierta madurez, evitando fallos frecuentes, para ello la arquitectura hace uso del Framework de desarrollo, el producto puede ser validado a través de excepciones, lo que llamamos comúnmente tratamiento de errores, para que este pueda tolerar cualquier fallo.

Usabilidad

La usabilidad es otros de los atributos de la calidad del software que la arquitectura debe tener en cuenta, a pesar de que muchas personas piensen que a esta se le da solución en la capa de presentación. Muchos software son capaces de encapsular todas las facilidades de uso en la capa de presentación, otros por la complejidad de los datos manejan algunas formas de uso desde otras capas como la de Lógica de Negocio y en otras ocasiones en la de Acceso a Datos. La herramienta que se desea desarrollar no presenta ninguna complejidad en cuanto a sus datos, solamente en el módulo Estadístico es donde se trabajarán cálculos sin ningún nivel de complejidad y estos se tratarán en la capa de Lógica de Negocio, de forma interna, ya que la arquitectura al tratar el diseño de clases, mantuvo la capacidad de que el usuario no tenga que hacer contacto con datos engorrosos como los Porcientos, o Cotas Mínimas o Máximas. Esto evita errores humanos y da una mayor confianza al usuario.

La estrategia de tratamiento de errores comentada anteriormente, es parte de darle información al usuario, ésta puede estar compuesta también por una ayuda o un manual de indicaciones y soporte para la operabilidad con la herramienta.

Eficiencia

El presente documento en subíndices anteriores definió algunas restricciones que debe mantener la herramienta una vez este en uso. Hay que tener en cuenta estas restricciones para tratar de no llevar al traste el uso de la aplicación. Lo más preocupante para la arquitectura es las conexiones de red que existan en los ordenadores donde se haga uso de la misma. Por otro lado, como es una pequeña herramienta desktop no consume mucha memoria, ni carga tanto el procesador y las únicas características que se tienen que tener en cuenta con respecto a esto son las restricciones que impone el Framework. Además, la herramienta no realiza grandes procesamientos de datos ni cálculos de algoritmos complejos, por lo que no consumen grandes cantidades de tiempos reloj. El diseño de la arquitectura trató de que muchos de los procesos se realicen de forma que compartan recursos, para así mantener un nivel bajo de consumo de CPU.

Portabilidad

Toda arquitectura debe ser portable, y así mismo permitirá que el software a desarrollar lo sea también. Con la decisión del uso de tecnologías para el desarrollo como fue el del Framework, este permite compilar y ejecutar la aplicación desde cualquier ambiente que lo tenga instalado sobre Windows. Muchas veces pensamos que las tecnologías .Net no son compatibles con Linux, hoy por hoy sobre este sistema operativo se han compilado y ejecutado muchas aplicaciones desarrolladas en Windows sobre la plataforma .Net. Sin embargo la plataforma Mono es la que permite esta compatibilidad, claro siempre y cuando las herramientas desarrolladas no dependan en gran medida del uso de Windows, como las Apis nativas.

Mantenibilidad

La robustez y flexibilidad del producto pueden ser medidas con este atributo de la calidad. El presente documento al definir patrones y estilos arquitectónicos, da la posibilidad de que la herramienta pueda ser modificada sin que produzca impactos negativos sobre el sistema. En caso de que se le desee agregar un nuevo módulo o un nuevo paquete de clases, esto no será un problema para la herramienta, solamente se toman las medidas pertinentes para evitar cambios negativos. Por ejemplo, si se deseara analizar páginas de otro buscador que no sean los que se predefinen, con analizar las características de funcionamiento y forma en que comparte los datos este buscador, se podría insertar esta funcionalidad a la herramienta sin modificar su funcionamiento original. Otro ejemplo muy sencillo

es que si se deseara almacenar los datos en una Base de Datos, con solo modificar el componente Acceso_a_Datos ya se lograría el objetivo, claro todo esto lleva un previo análisis.

3.4 Conclusiones del Capítulo 3.

En el presente capítulo se abordaron las variables o atributos que definidos por la ISO para determinar y evaluar la calidad de la arquitectura. Los elementos que se evalúan, son teniendo en cuenta que los desarrolladores, conjuntamente con los que realizan las pruebas al software, tengan presente lo expuesto ya que permitirá una eficiencia mayor en el desarrollo e implementación.

También se analizaron algunos de los atributos de la calidad. Este análisis se realizó tomando en consideración el trabajo que realicen los desarrolladores. Es muy importante que estos elementos se tomen con seriedad ya que de ellos dependerá la calidad de la herramienta.

CONCLUSIONES

El diseño de la arquitectura de un software es uno de los pasos más importantes que respalda el desarrollo del software. Es de vital importancia ya que se trata de abstraer en un mayor grado lo que se quiere del software, brindando una mejor propuesta de solución que cumpla con los requerimientos funcionales y no funcionales, teniendo en cuenta las restricciones que puedan acarrear y con el máximo de eficiencia y calidad.

En el presente trabajo se le da cumplimiento a los objetivos propuestos por el mismo. Se realizó una investigación profunda acerca del funcionamiento de los softwares existentes para el posicionamiento en buscadores de forma natural, para un posterior análisis de seleccionar la arquitectura para el desarrollo del software. También se realizó un estudio acerca del funcionamiento de los buscadores para tratar de hacer lo más real posible el funcionamiento de nuestro software.

Después de un estudio de los principales estilos arquitectónicos que se tienden a aplicar hoy en día al desarrollo de software, se definió el estilo en Capas para el desarrollo de nuestro software ya que este permite la escalabilidad, portabilidad, mantenimiento y soporte tomando el desarrollo de cada capa individualmente. Además de definir los elementos que pueden ser reusables con la aplicación de este estilo.

Se determinó, después de un amplio análisis de los patrones de diseño, cuáles de estos son los más apropiados para el diseño de clases, para lograr un bajo acoplamiento y una alta cohesión del sistema.

Con el desarrollo de este trabajo se logró diseñar y modelar una arquitectura capaz de cumplir con los requerimientos funcionales y no funcionales, convirtiéndola en una arquitectura robusta flexible y reusable, dando una solución informática al software que se pretende desarrollar. También se le da solución a los requerimientos funcionales a través de las diferentes vistas arquitectónicas, reflejando en cada una de ellas los componentes y elementos más importantes para la arquitectura del sistema.

RECOMENDACIONES

1. Se recomienda realizar un refinamiento de la arquitectura durante el ciclo de desarrollo del software.
2. Realizar una evaluación de la arquitectura a través de los métodos de evaluación existentes y planteados en el Capítulo 3 del presente trabajo para poder identificar cualquier debilidad que pueda presentar la arquitectura.
3. Se recomienda migrar la herramienta hacia plataformas libres para lograr un software multiplataforma y así eliminar el uso de herramientas específicas que restrinjan el desarrollo y soporte del mismo. Con esto se lograría una mejor portabilidad.
4. Para futuros desarrollos dentro de la temática, tener en cuenta la propuesta de arquitectura presentada en el presente trabajo.

BIBLIOGRAFÍA REFERENCIADA

AgentWebRanking. "Search engine ranking software for SEO Experts." Retrieved 26 abril del 2008, 2008, from http://www.agentwebranking.com/.

Clements, P. (1996). Coming attractions in Software Architecture. Alemania.

Codina., L. "Posicionamiento Web: Conceptos y Ciclo de Vida." Retrieved 26 de abril del 2008, 2008, from http://www.hipertext.net/Web/pag216_print.htm.

David Garlan, M. S. (1993). An Introduction to Software Architecture. New Jersey, World Scientific Publishing Company.

evosdesigne. Retrieved 26 abril del 2008, 2008, from http://www.evosdesign.com/.

Exchange, L. (2006). "Link Building is what we do!" Retrieved 26 abril del 2008, 2008, from http://www.linkexchangeexperts.com.

Iniesta, V. (2006, 5 de Abril de 2006). "Recuperacion y Organizacion de la Informacion." Retrieved 10 de Diciembre, 2007, from http://www.telefonica.net/web2/herramientas-seo/herramientasmejora.html.

Larman, C. "UML y Patrones." 507.

Muñiz, R. Marketing en el Siglo XXI.

Myrasoft. "Active WebTraffic." Retrieved 26 abril del 2008, 2008, from http://www.myrasoft.com/activewebtraffic/activewebtrafficsp.htm.

Reino, A. (2000). "Introduccion a Xml en Castellano."

Reynoso, C. B. (2006, 10 diciembre 2007). "Introducción a la Arquitectura de Software." from http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/intro.mspx.

Ugarte, D. d. (2002). El libro del posicionamiento en buscadores. Madrid.

Web, P. "Promotor Web - Software para alta y posicionamiento en buscadores en español
".

BIBLIOGRAFÍA CONSULTADA

"Promotor Web - Software para alta y posicionamiento en buscadores en español".

"Vocabulario SEO y Terminos Google." Retrieved 28 de noviembre., 2007, from http://www.xeoweb.com/buscadores/terminos-seo.php.

(1991). "ISO 9126: The Standard of Reference."

(2006). "Arquitectura de aplicaciones de .NET: Diseño de aplicaciones y servicios."

(2007). "Introducción a Microsoft .NET Framework 3.0 ".

(2008). "Fase de Elaboración."

abansys. "Posicionamiento Web ¿En que consiste?" Retrieved Abril 26 del 2008., 2008, from http://www.abansys.com/posicionamiento_Web_en_que_consiste.html.

AgentWebRanking. "Search engine ranking software for SEO Experts." Retrieved 26 abril del 2008, 2008, from http://www.agentwebranking.com/.

Clements, P. (1996). Coming attractions in Software Architecture. Alemania.

Clements, P. (1996). A Survey of Architecture Description Languages. Alemania.

Codina., L. "Posicionamiento Web: Conceptos y Ciclo de Vida." Retrieved 26 de abril del 2008, 2008, from http://www.hipertext.net/Web/pag216_print.htm.

David Garlan, M. S. (1993). An Introduction to Software Architecture. New Jersey, World Scientific Publishing Company.

evosdesigne. Retrieved 26 abril del 2008, 2008, from http://www.evosdesign.com/.

Exchange, L. (2006). "Link Building is what we do!" Retrieved 26 abril del 2008, 2008, from http://www.linkexchangeexperts.com.

Iniesta, V. (2006, 5 de Abril de 2006). "Recuperación y Organización de la Información." Retrieved 10 de Diciembre, 2007, from http://www.telefonica.net/web2/herramientas-seo/herramientasmejora.html.

Kiccillof, C. R.-N. (2004). Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.

Larman, C. "UML y Patrones." 507.

Muñiz, R. Marketing en el Siglo XXI.

Myrasoft. "Active WebTraffic." Retrieved 26 abril del 2008, 2008, from http://www.myrasoft.com/activewebtraffic/activewebtrafficsp.htm.

R. Kazman, M. K., P. Clements (1998). Evaluating Software Architectures for Real-Time System.

Real., U. d. C.-L. M. E. d. I. d. C. Arquitectura Multicapa. Prácticas Ingenieras del Software 3.

Reino, A. (2000). "Introducción a Xml en Castellano."

Reynoso, C. B. (2006, 10 diciembre 2007). "Introducción a la Arquitectura de Software." from http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/intro.mspx.

Rodríguez, A. "Diseño de Aplicaciones Three Tier." Retrieved 3 de diciembre, 2007, from http://www.fpress.com/revista/Num9711/Nov97.htm.

Seco, J. A. G. (2001). "El lenguaje de programación C#."

SEOguru. (2007). "Uso de los Backlinks en el Web." Retrieved 28 de noviembre, 2007, from http://www.pagoclick.com.mx/blog/2007/10/24/uso-de-los-backlinks-en-el-Web/.

SUBMISSION, D. "Search Engine Submission Software ", from http://www.dynamicsubmission.com.

Welicki., L. (2005). "Patrones y Antipatrones: una Introducción - Parte II."

GLOSARIO DE TÉRMINOS

Arquitectura de Software: La arquitectura de software de un sistema de programa o computación es la estructura de las estructuras del sistema, la cual comprende los componentes del software, las propiedades de esos componentes visibles extremamente, y las relaciones entre ellos.

Estilo en Capas: Es uno de los estilos más usados para cualquier tipo de aplicaciones, este estilo se basa en dividir los elementos de diseño en paquetes de Interfaz de Usuario, Lógica de Negocio y Acceso a Datos.

IDE: (Integrated Development Environment); Entorno Integrado de Desarrollo, un entorno desde el que se pueden editar programas, compilarlos y depurarlos

ISO: Organización Internacional para la Estandarización (Internacional Organization for Standardization). Es una organización que ha definido un conjunto de protocolos diferentes, llamados protocolos ISO/OSI. Esta organización de carácter voluntario fue fundada en 1946 y es responsable de la creación de estándares internacionales en muchas áreas, incluyendo la informática, las ecológicas y las comunicaciones. Está formada por las organizaciones de normalización de sus 89 países miembros.

RUP: (Rational Unified Process) Metodología de desarrollo que divide el proceso de desarrollo de un software en cuatro fases.

Posicionamiento en Web: Consiste en aplicar diversas técnicas tendentes a lograr que los buscadores de Internet encuadren una página Web en una posición y categoría deseada dentro de su página de resultados para determinados conceptos clave de búsqueda.

URL: Acrónimo de Universal Resource Locator (Localizador Universal de Recursos /Identificador Universal de Recursos). Sistema unificado de identificación de recursos en la red. Es el modo estándar de proporcionar la dirección de cualquier recurso en Internet.

Vista de Casos de Uso: Es la vista arquitectónica según RUP, donde se muestra como principal artefacto el diagrama de casos de uso del sistema. Esta vista es un reflejo de cómo los usuarios, analistas y encargados de las pruebas perciben el software.

Vista de Despliegue: Vista que muestra la topología de hardware de los nodos del sistema una vez esté ejecutándose.

Vista de Implementación: Esta vista incluye todos los componentes; ya sean bibliotecas, ejecutables y archivos sobre el sistema físico.

Vista Lógica: Vista fundamental en la fase de diseño ya que comprende las clases, interfaces y colaboraciones que interpretan el problema desde una perspectiva fácil para los desarrolladores.

Webmaster: Persona responsable de la gestión y mantenimiento de un servidor Web, principalmente desde el punto de vista técnico; por lo que no debe ser confundido con un editor de Web.

XML: (eXtensible Markup Language). Formato estándar para el intercambio de datos basado en archivos de texto plano con una estructura de etiquetas.