CAPÍTULO I

RESUMEN

Las aplicaciones de escritorio ha venido evolucionando a grandes pasos, en la antigüedad los programas iniciales fueron simplemente con tarjeta perforadas que luego llegaron a ser procesados como datos binarios dando a los investigadores pautas para que el area de la programación siga dando muchas iniciativas para generar apliaciones que ayuden a sobre salir a las empresas que desean automatizar los procesos manuales que se llevan a cabo.

Las empresas que desarrollan aplicaciones en la actualidad son muchas y luchan por ser mejores ante tantas competencias, dando soluciones eficientes a tantos problemas causados por los procesos manuales o mejorando procesos automáticos que al inicio fueron eficientes. Los lenguajes de programación al inicio daban mucho que desear por el hecho de no satisfacer todas las necesidades de los usuarios dejando algunos procesos manuales sin automatizar, después de muchas décadas los lenguajes ha mejorado y logrado

realizar procesos que en la antigüedad eran imposibles de realizar, esto se puede decir de las aplicaciones de escritorio.

Las aplicaciones web ha venido siendo para los investigadores un tema por que preocuparse ya que los procesos mediante esta tecnología fueron muy imposibles de realizar, esto paso por muchos años y es hasta la actualidad que los investigadores han logrado poner a las aplicaciones web como otra opción para las automatizaciones de los procesos de las empresas que lo necesitan, es mas aplicaciones web en la actualidad se están implementando mas de lo que antes se lo realizaba.

Con esta idea la historia de las aplicaciones de escritorio y web ha logrado que el mundo actual pueda usar y automatizar todo tipo de procesos, dejando todo lo manual para la historia, dando a las emplesas eficiencia y eficacia, ahorrando tiempo y dinero a la hora de obtener la información necesaria de la empresa. Estas aplicaciones se lo puede realizar usando cualquier tipo de lenguaje de programación, cada uno de los lenguajes tienen difenete lógica de interpretar el código que el programador implementa, esto se puede medir en el tiempo de demora de la ejecución de la aplicación, es aquí donde las aplicaciones se miden por eficientes a la hora de procesar la información.

Historia de las aplicaciones de Escritorio

Conocer la historia del desarrollo de cualquier campo del saber humano es de gran importancia para aquellos que planeen desenvolverse en dicho campo, ya que tiene cuando menos dos beneficios tangibles: Conocer el estado promedio del arte y evitar cometer errores ya cometidos. Así es más fácil tomar las decisiones correctas y sobrevivir sin esfuerzo a muchas modas que estrictamente proclaman la reinvención de una nueva tecnología. Como dijo el filósofo español George Santayana "El que no conoce la historia está condenado a repetirla".

Es comúnmente aceptado que la historia de la computación moderna comienza con Charles Babbage, en 1822, sin embargo las primeras computadoras mecánicas y eléctricas no contaban con un lenguaje de programación tal como lo conocemos ahora, la mayoría de ellas era construida para llevar a cabo una

tarea específica; por lo tanto la forma de programarla era particular a cada computadora.

Fue en el año de 1945 cuando el matemático Jhon Von Neumann fascinado por las posibilidades de ENIAC, elaboró un estudio que demostró que una computadora podía tener una estructura física muy simple y fija, y aun así ser capaz de ejecutar cualquier tipo de programa mediante un control correcto, sin la necesidad de modificar la computadora para esto.

A partir de esta innovación de Von Neumann, que en conjunto se conoce como la técnica de programa almacenado es que se inicia la era de los lenguajes de programación quienes tendrán una revolución tecnológica brillante, y desde esos momentos a la actualidad el ámbito del desarrollo ha cambiado de forma drástica.

1. Introducción a las aplicaciones de escritorio

1.1. Historia de las aplicaciones de escritorio

Introducción

Conocer la historia del desarrollo de cualquier campo del saber humano es de gran importancia para aquellos que planeen desenvolverse en dicho campo, ya que tiene cuando menos dos beneficios tangibles: Conocer el estado promedio del arte y evitar cometer errores ya cometidos. Así es más fácil tomar las decisiones correctas y sobrevivir sin esfuerzo a muchas modas que estrictamente proclaman la reinvención de una nueva tecnología. Como dijo el filósofo español George Santayana "El que no conoce la historia está condenado a repetirla".

Es comúnmente aceptado que la historia de la computación moderna comienza con Charles Babbage, en 1822, sin embargo las primeras computadoras mecánicas y eléctricas no contaban con un lenguaje de programación tal como lo conocemos ahora, la mayoría de ellas era construida para llevar a cabo una tarea específica; por lo tanto la forma de programarla era particular a cada computadora.

Fue en el año de 1945 cuando el matemático Jhon Von Neumann fascinado por las posibilidades de ENIAC, elaboró un estudio que demostró que una computadora podía tener una estructura física muy simple y fija, y aun así ser capaz de ejecutar cualquier tipo de programa mediante un control correcto, sin la necesidad de modificar la computadora para esto.

A partir de esta innovación de Von Neumann, que en conjunto se conoce como la técnica de programa almacenado es que se inicia la era de los lenguajes de programación quienes tendrán una revolución tecnológica brillante, y desde esos momentos a la actualidad el ámbito del desarrollo ha cambiado de forma drástica.

Los primeros lenguajes

Shortcode.- En 1949, aparece el primer lenguaje que se usó en computadoras electrónicas: "***Shortcode***", requería que el programador convirtiera (compilara) su programa a 0's y 1's de manera manual.

A-0.- Fue hasta el año de 1951, que Grace Hopper trabajando para Remington Rand, comienza a desarrollar el primer compilador, lo que trajo consigo una programación más rápida.

Fortran.- Es en el año de 1957 cuando aparece el primero de los grandes lenguajes de programación de uso aún actualmente: FORTRAN, que proviene de ***FOR***mula ***TRAN***slating system.

Fue desarrollado por IBM para cómputo científico, el líder del proyecto fue John Backus, que después contribuiría en ALGOL.

El lenguaje original solo incluía FOR, DO y GOTO. También definió los tipos de datos básicos: TRUE, FALSE, integer, real, double presicion.

El lenguaje original era bueno manejando números, pero malo manejando entrada y salida, lo cual propicio la aparición de otros lenguajes orientados a negocios.

LISP.- En el año de 1958, el profesor John McCarthy del M.I.T. comenzó a desarrollar la teoría de un lenguaje de de procesamiento de listas. En 1959 aparece públicamente la primera implementación llamada LISP 1.5.

En 1960 McCarthy publica un histórico artículo acerca de los fundamentos de LISP que hizo por la programación lo mismo que hizo Euclides por la geometría, demostrar como con unos cuantos operadores y una notación para funciones es posible construir un lenguaje de programación completo.

Es importante hacer notar que McCarthy no solo marco un hito en la historia de los lenguajes de programación sino que creó un modelo de programación que ha demostrado ser superior, tanto que podemos decir que en la actualidad existen dos grandes modelos de programación el de C y el de Lisp, y podemos notar que los lenguajes del modelo C tratan de parecerse cada vez más a Lisp.

ALGOL.- En el año de 1958 un comité crea la especificación del lenguaje ALGOL, tenía la particularidad de no definir la manera de manejar entrada y salida, dejando esta parte libre a cada implementación.

Es en 1960 cuando aparece el lenguaje ALGOL 6.0 el primer lenguaje estructurado en bloques. Este lenguaje fue muy popular en el segundo lustro de los 60's.

Su principal contribución es ser la raíz del árbol que ha producido lenguajes tales como pascal, C, C++, y Java.

COBOL.- En 1959, ***Conference on Data Systems and Languages (CODASYL)*** crea COBOL, un lenguaje para negocios; que fuera fácil de aprender para gente que no tuviera formación en ciencias de la computación. Sus únicos tipos de datos fueron cadenas y números. Lo que le dio la característica de poder agruparlos en arreglos sencillos, de modo que los datos podían ser organizados y seguidos de una mejor manera. Las sentencias de COBOL se parecen mucho a las usadas por el idioma inglés, haciendo que fuera fácil de aprender. Todo esto con la finalidad de que los negocios promedio lo pudieran aprender y usar.

Desarrollo de los lenguajes de programación

Basados en los primeros lenguajes de programación han surgido muchos otros lenguajes que siempre tienen la intención de tomar lo mejor, desechar lo malo, y agregar alguna novedad, respecto a los existentes. Solo se hace referencia a los más conocidos.

Basic.- Inventado en 1964 por John George Kemeny y Thomas Eugene Kurtz en el Colegio Dartmounth.

Es un lenguaje muy limitado que fue diseñado para personas que no fueran del área de ciencias de la computación.

El nombre de BASIC, significa ***B****eginners* ***A****ll-purpose* ***S****ymbolic* ***I****nstruction* ***C****ode.*

Los siete principios de diseño de BASIC fueron:

Ser fácil de usar para los principiantes.

Ser un lenguaje de propósito general.

Permitir que los expertos añadieran características avanzadas, mientras que el lenguaje permanecía simple para los principiantes.

Ser interactivo.

Proveer mensajes de error claro y amigable.

Responder rápido a los programas pequeños.

No requerir un conocimiento del hardware de la computadora.

El lenguaje se baso en Fortran y Algol 6.0

Pascal.- Diseñado por Niklaus Wirth, como una herramienta de enseñanza de la programación. Sus desarrolladores se concentraron en desarrollar buenas herramientas que contribuyeran a la enseñanza, tal como un buen debugger, y un buen editor. Además tuvieron como meta el tener soporte para la mayoría de los microprocesadores populares en esa época en las instituciones de enseñanza.

Fue diseñado de una manera muy ordenada, reflejando la experiencia de su diseñador, tomó las mejores características de los lenguajes de su tiempo, COBOL, ALGOL, y FORTRAN, al mismo tiempo que busco evitar sus deficiencias, y hacerlo lo más claro posible. La combinación de sus características de entrada/salida, y sus solidas características matemáticas pronto lo convirtieron en un lenguaje muy exitoso. También implemento el tipo apuntador y agrego el CASE, e hizo uso de variables dinámicas. Sin embargo no implemento arreglos dinámicos ni agrupamiento de variables lo cual contribuyó a su pérdida de popularidad frente a nuevos lenguajes.

Delphi es una versión moderna y orientada a objetos de Pascal.

Prolog.- Diseñado en 1970 en la universidad de Aix-Marseille por los profesores Alain Colmerauer y Phillipe Roussel, su nombre proviene del francés ***Pro****gramation et* ***Lo****gique,* es popular en los círculos de interés en Inteligencia Artificial.

Inicialmente era un lenguaje interpretado, hasta que a mediados de la década de los 70's David Warren desarrolló un compilador, que convertía el programa en Prolog a un conjunto de instrucciones de una máquina abstracta denominada Warren Abstract Machine, o WAM.

Su sintaxis y semántica son consideradas simples y claras, su paradigma de programación es el declarativo y un programa en Prolog se compone de *cláusulas* que constituyen reglas del tipo modus ponens, es decir, "Si es verdad el *antecedente*, entonces es verdad el *consecuente*". No obstante, la forma de escribir las cláusulas es al contrario de lo habitual. Primero se escribe el consecuente y luego el antecedente.

Lenguaje C.- Fue diseñado en 1971, por Dennis Ritchie y Ken Thompson mientras trabajaban para los Laboratorios Bell, y se baso en los lenguajes de programación B y BCPL.

Se basa en el paradigma imperativo y desde su creación estuvo pensado para programación de sistemas operativos, se creó para usarse en UNIX, y creció de la mano del desarrollo de UNIX, lo que propicio la creación de características avanzadas tales como variables dinámicas, multitarea, manejo de interrupciones, forking y un poderoso manejo entrada/salida de bajo nivel. Debido a esto C es comúnmente usado para programación de nivel de sistema en UNIX, Linux y Mac.

Se trata de un lenguaje no fuertemente tipiado de medio nivel pero con muchas características de bajo nivel. Dispone de las estructuras típicas de los lenguajes de alto nivel pero, a su vez, dispone de construcciones del lenguaje que permiten un control a muy bajo nivel. Destaca su gran riqueza de operadores y expresiones.

C++.- A finales de los 70's y principio de los 80's un nuevo modelo de de programación fue desarrollado, la programación orientada a objetos, la idea básica es que los objetos son piezas de código auto contenidas y reusables. Bearne Stroustroup, también de los Laboratorios Bell, desarrolló un nuevo lenguaje basado en C que aplica los conceptos de la programación orientada a objetos, inicialmente se llamó C con clases, para posteriormente tomar su nombre definitivo C++ cuando fue publicado en 1983.

Las principales características del C++ son abstracción, el soporte para programación orientada a objetos y el soporte de plantillas o programación genérica. Por lo cual, se puede decir que C++ es un lenguaje multi paradigma que abarca tres paradigmas de la programación: La programación estructurada, la programación genérica y la programación orientada a objetos.

Actualmente cuenta con un estándar ISO y es muy popular en la programación de aplicaciones.

Perl.- Perl (**P**ractical **E**xtraction and **R**eport **L**anguage), fue desarrollado por Larry Wall, inicialmente motivado para cubrir las carencias y las limitaciones del shell, su primer versión fue anunciada en el Newsgroup el 18 de diciembre de 1987.

Actualmente Perl es un lenguaje de propósito general, usado para amplia gama de tareas que incluyen administración de sistemas, desarrollo web, programación en red, desarrollo GUI, si lo deseas, puedes programar orientado a objetos y mucho más.

Es uno de los lenguajes más flexibles y poderosos, junto con Lisp, permite desarrollar prácticamente cualquier cosa con él, es poderoso, flexible y expresivo. Perl es libre y es divertido, permite ser creativo y no te ata a las restricciones impuestas por el diseñador del lenguaje, en Perl siempre hay más de una manera de hacer las cosas.

Python.- Es un lenguaje interpretado e interactivo, creado por Guido Van Rossum en 1991. Es completamente tipiado dinámicamente, usa manejo automático de memoria, por lo cual es similar a Lisp, Perl, Ruby, Scheme y Tcl.

Es desarrollado como proyecto de software libre, manejado por la Python Software Foundation. Tomó varias de sus características de Lisp.

Algunos lo consideran la "oposición leal" a Perl, lenguaje con el cual mantiene una rivalidad amistosa. Los usuarios de Python consideran a éste mucho más limpio y elegante para programar. Sin embargo esto es más un punto de vista de sus usuarios que una realidad.

Su facilidad de aprendizaje, su orientación a programadores promedio, su limpieza de código, hacen que sea uno de los lenguajes más exitosos al momento de escribir el presente trabajo.

Ruby.- Es un lenguaje de programación interpretado, orientado a objetos, con una sintaxis inspirada por Perl y Ada, que también tomó ideas de Lisp, Python, Dylan y CLU.

Fue creado por Yukihiro Matsumoto, quien comenzó su desarrollo en 1993, y lo publicó en 1995.

Entre sus características tenemos:

Lenguaje de guiones interpretado:

Posibilidad de realizar directamente llamadas al sistema operativo

Potentes operaciones sobre cadenas de caracteres y expresiones regulares

Retroalimentación inmediata durante el proceso de desarrollo Rápido y sencillo:

Son innecesarias las declaraciones de variables

Las variables son de tipo dinámico.

La sintaxis es simple y consistente

La gestión de la memoria es automática

Ansi Common Lisp.- En abril de 1981, después de una reunión patrocinada por DARPA, se unieron los esfuerzos de Symbolics, el proyecto SPICE, el proyecto NIL y el proyecto S-1 Lisp, para definir Common Lisp.

Common Lisp fue diseñado como una descripción de una familia de dialectos de Lisp. Common Lisp recibió alimentación de La Maquina Lisp, MacLisp, NIL, S-1 Lisp, Spice Lisp y Scheme. El libro Common Lisp: The language es la descripción de ese diseño.

Su semántica fue dejada sin especificar en aquellas partes donde una fuerte especificación pudiera ser un obstáculo en el uso e investigación de Lisp.

En 1986 se formo un grupo de trabajo para producir un draft para un estándar ANSI de Common Lisp, entre las especificaciones que presento se encontraban estandarizaciones para portabilidad, facilidades de iteración, manejo de grandes conjuntos de caracteres, un sistema de programación orientada a objetos, entre otras. El estándar ANSI Common Lisp fue publicado en 1994.

Lisp se consolido como el lenguaje de programación más poderosas y flexibles jamás inventadas. Muchos de los lenguajes modernos han copiado la mayoría de sus características, sin embrago no todas ya que entonces no podrían decir que son un nuevo lenguaje sino simplemente otro dialecto de Lisp.

Java.- Java es una plataforma de software desarrollada por Sun Microsystems, de tal manera que los programas creados en ella puedan ejecutarse sin cambios en diferentes tipos de arquitecturas y dispositivos computacionales.

La *plataforma Java* consta de las siguientes partes:

El lenguaje de programación, mismo.

La máquina virtual de Java o JRE, que permite la portabilidad en ejecución.

El API Java, una biblioteca estándar para el lenguaje

El lenguaje de programación fue desarrollado por James Gosling, y sus equipo en Sun Microsystems, entre 1990 y 1994, pensado originalmente como un reemplazo de C++, orientado a dispositivos embebidos, y a la televisión interactiva, posteriormente que para rescatarlo del fracaso y fue reorientado hacia su aplicación en la Web.

Las cinco metas del lenguaje Java son:

Orientado a objetos

Multiplataforma

Soporte integrado para redes de computadoras

Diseñado para ejecutar código de fuentes remotas de modo seguro

Fácil de usar

Desde sus inicios Java no fue bien recibido por su falta de aportaciones originales, su lentitud, y su rígida estructura, sin embargo en base a intensas campañas de marketing Sun ha logrado que en los ambientes empresariales Java sea uno de los estándares sobre todo para aplicaciones del lado del servidor.

Java también es presentado muchas veces como un lenguaje Libre lo cual no es completamente cierto.

Java Script.- Es un lenguaje de script basado en objetos, que se apoya en el modelo de prototipos. Es muy popular por su uso en sitios Web.

Fue desarrollado por Brendan Eich, en su trabajo para Netscape, quien lo publicó en diciembre de 1995.

Su sintaxis es parecida a la de C, y al igual que C utiliza el concepto de palabras reservadas, tiene soporte para expresiones regulares y UTF.

Cuando se usa en Web, se conecta a través de su interfaz DOM (Document Object Model) a las aplicaciones tanto del lado del cliente como del servidor. Gracias ha esto es usado para crear poderosas aplicaciones web dinámicas.

Desafortunadamente los diferentes navegadores exponen diferentes objetos y métodos por lo cual es necesario escribir versiones especificas de un programa en Java script para los diferentes tipos de navegadores.

Entre las principales tecnologías para interactuar con DOM que usan Java Script esta AJAX y DHTML.

C#.- Es un lenguaje orientado a objetos desarrollado por Microsoft, tomando ideas de C++ y Java, como parte de su estrategia comercial .Net.

Esta normalizado, por ECMA quien en el año 2001 publicó la ECMA-334 *C# Language Specification;* en el año 2003 se convirtió en un estándar ISO (ISO/IEC 23270)".

Además de la implementación de Microsoft existen implementaciones libres como Mono y dotGNU.

Gracias a la maquinaría comercial de Microsoft y la amplia base de escritorios Windows, el uso de C# es muy extendido en entornos corporativos y en el mundo Windows en general.

Los proyectos libres de C#, no son muy populares entre la comunidad del software libre, por la desconfianza hacia Microsoft y sus patentes.

1.2. Clasificación de las tecnologías usadas para el desarrollo de aplicaciones de escritorio

Las aplicaciones de escritorio y web comparten muchas tecnologías en común con la diferencia que las tecnologías que se apliquen al desarrollo web deben incorporarse la seguridad respectiva para que la información en el camino no sea alterada o robada para fines malévolos, caso contrario sucede en las aplicaciones de escritorio donde la seguridad de la información (robo, alteración de información) no es el punto de enfoque para poder ejecutar una aplicación.

Esto se puede simplificar en que las aplicaciones web son ejecutadas vía internet fuera de la compañía la cual necesariamente necesita una alta seguridad de la información que navega por lo largo del canal del internet, y las de escritorio solo se las puede usar vía intranet, dentro de una compañía donde solo se requiere que la información que navega desde un cliente al servidor sea coherente.

A partir de este conocimiento las aplicaciones se las puede aplicar tanto en redes con cableado físico como en redes inalámbricas tomando como referencia lo descrito anteriormente. Con esto podemos decir que lo que en tiempos pasados era imposible trabajar con PALM’s, PDA’s entre otros dispositivos inalámbricos, pues en la actualidad se usa estas tecnologías como parte del trabajo diario, como puede ser para la toma de inventarios físicos, toma de pedidos a clientes en sus locales, recepción de mercadería entre otros.

1.3. Análisis de los GUI'S de desarrollo de aplicaciones de escritorio

VISUAL BASIC.- B.A.S.I.C. significa: **B**eginner's **A**ll-purpose **S**ymbolic **I**nstruction **C**ode, traduciendo esto sería: Código de instrucciones de uso universal para principiantes. Basic es un lenguaje de programación que actúa como un intermediario entre el operador y el computador. Mediante el vocabulario y las reglas Basic, se le ordena al computador lo que se quiere que haga y el PC transforma estas instrucciones para poder ejecutarlas. Otros lenguajes de programación que pueden emplearse son el Pascal, Fortan y Ensamblador. BASIC es traducido a código binario, también se puede escribir programas en código binario pero es muy difícil por eso crearon el Basic.

BASIC es un lenguaje de programación desarrollado por los estadounidenses **John Kemeny** y **Thomas Kurtz** en el Dartmouth College. La versión original del lenguaje Basic fue creada en el año **1964**, ganándose una enorme popularidad gracias sobre todo a dos implementaciones, Tiny BASIC y Microsoft BASIC, que convirtieron a este lenguaje en la primera lengua franca de los microordenadores. Otras importantes implementaciones han sido CBASIC (BASIC Compilado), Integer y Applesoft BASIC (para el Apple II), GW-BASIC (para computadoras personales), Turbo BASIC (de Borland) y Microsoft QuickBASIC. El lenguaje ha cambiado en el transcurso de los años pues nació con el objetivo de servir como lenguaje para aquellas personas que deseaban introducirse por primera vez en el mundo de la programación, y luego fue sufriendo modificaciones, hasta que en 1978 se estableció el Basic estándar.

Aplicaciones Data-Aware

Mientras la adopción de Visual Basic en las corporaciones se expandía, también lo hacía la necesidad de una herramienta para el desarrollador que permitiera aplicaciones data-aware robustas. Visual Basic 3.0, anunciado solamente seis meses después de la salida al mercado de la versión 2.0, solucionaba esta necesidad combinando el motor de la base de datos de Microsoft Access 1.1 con un conjunto rico de controles data-aware. Por primera vez, los desarrolladores podían conectar fácilmente a las bases de

datos en un ambiente cliente/servidor usando un diseñador visual intuitivo. La complementación de estas características era los Data Access Object (Objetos de Acceso a Datos) (DAO), un completo paquete de los objetos que proporcionaban al acceso mediante código a la base de datos. Finalmente, Visual Basic 3.0 amplió la capacidad de la herramienta de desarrollo incluyendo los Crystal Reports, un motor para visualizar datos extraídos en una variedad de formatos personalizables.

En los años venideros, la industria informática comenzaría a abrazar el movimiento a la programación en 32-bits. La salida al mercado de Microsoft Windows 95 y de Microsoft Windows NT condujo a esta adopción y destacó la necesidad de herramientas de desarrollo más potentes que podrían soportar la nueva arquitectura. Fue entonces cuando la revista Windows Watcher señaló que Visual Basic estaba adoptado por más compañías (30 por ciento) que cualquier otro lenguaje de programación. Llevar una base instalada tan grande del desarrollo de aplicaciones en 16-bits a 32-bits sería una tarea de migración importante, pero aseguraría la existencia prolongada del lenguaje de programación Visual Basic y de su comunidad. La versión 32-bit de Visual Basic- versión 4.0-fue anunciada en septiembre de 1995 e incluía la edición estándar y profesional así como una nueva edición destinada al nivel empresarial y el desarrollo en equipo. La edición empresarial ofrecía nuevas capacidades tales como automatización remota, control de datos remoto, y una versión integrada de Microsoft Visual SourceSafe para la dirección de la configuración y realización de diferentes versiones.

Las versiones de Visual Basic 5.0 y 6.0-anunciadas en marzo de 1997 y en junio de 1998, respectivamente- representaron un paso importante hacia posibilitar a los desarrolladores en Visual Basic programar en los nuevos niveles del funcionamiento en el ambiente libre que representa Internet. Las características tales como el compilador del código nativo introdujeron aumentos del funcionamiento de hasta el 2.000 por ciento. El Webclass designer (diseñador de clases Web) simplificó la creación de las aplicaciones del Web proporcionando un modelo intuitivo del objeto para el servidor web. Y el diseñador de páginas DHTML permitió la creación de aplicaciones para

Microsoft Internet Explorer 4.0- que combinaron la riqueza de HTML dinámico (DHTML) con el funcionamiento y la productividad de Visual Basic. Finalmente, con la Control Creation Edition (Edición de Creación de Controles), los desarrolladores de Visual Basic podrían construir fácilmente los controles Microsoft ActiveX de alto rendimiento y con un amplio alcance que estaban destinados a Internet.

.NET .- Es la evolución de VISUAL BASIC, C++ y JAVA para los cuales se creó un solo paquete VISUAL STUDIO .NET el mismo que se lanzo en el año de 2002 donde incluía VISUAL BASIC .NET (proveniente de Visual Basic), C# .NET (Proveniente de C++), J# .NET (Proveniente de Java).
El manejo de estas herramientas estaban orientadas completamente a objetos, la construcción de clases, interfaces, controles hacían que el diseño y la programación sea cada vez más rápida y segura ya que una conexión a Base de Datos era posible con ciertas dificultades.

Pero claro todo lo que sean aplicaciones de escritorio era más fácil porque para la parte de diseño de las aplicaciones web todavía seguían siendo un problema ya era por la dificultad de manejar Bases de Datos con clientes ricos.

Así paso dos años más y la herramienta evolucionó obteniendo como resultado VISUAL STUDIO .NET 2003 donde el acceso a las bases de datos eran aun más sencillas, ya que la herramienta contenía dentro del mismo componente la arquitectura ADO de .NET lo cual realizar una conexión y manipulación de base de datos es tan sencillo, logrando realizar aplicaciones en menor tiempo. Pero en el ámbito de las aplicaciones web se incorporo lo que son clientes ricos manejando en las páginas java script de tal forma que las funciones que en ella corran se ejecuten en el cliente sin necesidad de viajar asía el servidor.

En esta versión las aplicaciones eran más solidas y fuertes tanto que la programación secuencial ya era parte del pasado, y con esto muchos lenguajes de programación tuvieron que ver una solución para no quedar obsoletos y se un competencia a la hora de desarrollar software de calidad.

Luego de varios años se .NET se fortalece y evoluciona sacando al mercado VISUAL STUDIO 2005 .NET con alguna mejoras en la parte de programación web, el poder de java script realizaba que las paginas dejen de ser tan pesadas como lo eran en las primeras aplicaciones. Con esto lograron sacarles parte de provecho a los clientes para que el peso de las aplicaciones web no caiga en su totalidad al servidor, sino que este solo sea el intérprete de resolver acceso a la base de datos.

Luego de unos años .NET evoluciona y da un gran giro en la parte de la programación de escritorio y web, la herramienta VISUAL STUDIO .NET 2008 logró lo que hasta hace poco era muy difícil de lograrlo que las aplicaciones web puedan correr dentro de un run-time propio de Microsoft, cosa que java ya tenía esta idea desarrollada pero no tan clara como lo es GWT su desarrollo es todavía complicado, la propuesta de GWT es programar todo en código java y al momento de compilar este lo convierte en código java script entendido por el cliente, de igual forma este corre en un run-time de java.

La versión de .NET 2008 incorpora WPF (Windows Presentation Fundation) una tecnología ambiciosa para la programación ya que supera los límites de programar por separado la parte de desarrollo y la de diseño grafico, es decir aquí se puede realizar las dos cosas a la vez una de las grandes ventajas para el desarrollo de aplicaciones, la desventaja de esta herramienta es que se necesita de otro GUI de desarrollo para la parte gráfica que es muy potente a la hora de diseñar pero muy débil a la hora de incorporar la programación en este GUI, por esta razón se necesita las dos herramientas para el desarrollo.

WPF está orientado al diseño grafico, tanto para las aplicaciones de escritorio como la web, esta tendencia se implemento ya que la programación en Flash era posible logrando aplicaciones atractivas para los usuarios y fáciles de utilizar de igual forma que WPF este funcionaba en un run-time propio de la herramienta. Esto despertó el interés de Microsoft implementar este tipo de aplicaciones para no perder mercado.

Desde esta versión ya se puede implementar aplicaciones de escritorio que corran en la web, o viceversa. La idea que se propones es que el código que

se maneje sea reutilizable de tal forma si una aplicación se desea llamar de alguna aplicación web solo tiene que agregarse como referencia a la aplicación y poder acceder a los métodos y funciones que esta tenga permitida acceder.

Con esta nueva idea de programación pasaron dos años y se lanzó VISUAL STUDIO .NET 2010 el mismo que incorporó otra herramienta de desarrollo que corre en el run-time de .NET el mismo que se llama Silverligth este diseño de aplicaciones web más livianas que la de WPF que se ejecutan en el cliente hicieron que la programación de aplicaciones web ya no sea complicado como lo eran antes. La conexión a base de datos en la web es muy sencilla y rápida de implementar con una idea de arquitectura MVC.

Otra namespace que incorpora Microsoft desde .NET 2008 es LINQ (Language Integrated Query) es un nuevo conjunto de herramientas diseñado para reducir la complejidad del acceso a Base de Datos, a través de extensiones para C++ y Visual Basic así como para Microsoft .NET Framework. Permite filtrar, enumerar, y crear proyecciones de muchos tipos y colecciones de datos utilizando toda la misma sintaxis, prescindiendo del uso de lenguajes especializados como SQL o XPath.

NETBEANS.- NetBeans comenzó como un proyecto estudiantil en República Checa (originalmente llamado Xelfi), en 1996 bajo la tutoría de la Facultad de Matemáticas y Física en la Universidad Carolina en Praga. La meta era escribir un entorno de desarrollo integrado (IDE) para Java parecida a la de Delphi. Xelfi fue el primer entorno de desarrollo integrado escrito en Java, con su primer pre-release en 1997.

Xelfi fue un proyecto divertido para trabajar, ya que las IDEs escritas en Java eran un territorio desconocido en esa época. El proyecto atrajo suficiente interés, por lo que los estudiantes, después de graduarse, decidieron que lo podían convertir en un proyecto comercial. Prestando espacios web de amigos y familiares, formaron una compañía alrededor de esto. Casi todos ellos siguen

El plan original era desarrollar unos componentes JavaBeans para redes. Jarda Tulach, quien diseñó la arquitectura básica de la IDE, propuso la idea de llamarlo NetBeans, a fin de describir este propósito. Cuando las especificaciones de los Enterprise JavaBeans salieron, decidieron trabajar con este estándar, ya que no tenía sentido competir contra él, sin embargo permaneció el nombre de NetBeans.

En la primavera de 1999, Netbeans DeveloperX2 fue lanzado, soportando Swing. Las mejoras de rendimiento que llegaron con el JDK 1.3, lanzado en otoño de 1999, hicieron de NetBeans una alternativa realmente viable para el desarrollo de herramientas. En el verano de 1999, el equipo trabajó duro para rediseñar DeveloperX2 en un NetBeans más modular, lo que lo convirtió en la base de NetBeans hoy en día.

Sun adquirió otra compañía de herramientas al mismo tiempo, Forté, y decidió renombrar NetBeans a Forté for Java. El nombre de NetBeans desapareció por un tiempo.

Seis meses después, se tomó la decisión de hacer a NetBeans open source. Mientras que Sun había contribuido considerablemente con líneas de código en varios proyectos de código abierto a través de los años, NetBeans se convirtió en el primer proyecto de código abierto patrocinado por ellos. En Junio del 2000 NetBeans.org fue lanzado.

1.4. Aplicaciones de escritorio

1.4.1. Estudio del uso de aplicaciones de escritorio

Para poder usar una aplicación de escritorio se debe hacer un estudio pos desarrollo analizando el alcance que va a tener esta aplicación al momento de correr en un cliente, si esta aplicación va a ser usada fuera de la compañía es recomendable usar una aplicación web su facilidad de instalación, si esta va a funcionar dentro de una compañía u corporación lo ideal sería desarrollar una aplicación de escritorio. Como hemos revisado anterior mente en .NET esto ya no importa porque ya sea una aplicación de escritorio esta puede correr en un

entorno web solo hay que tener en cuenta sus requerimientos de los clientes para que las aplicación es corran de forma correcta y eficiente.
Otro de los estudios a realizar es el acceso a datos, es decir la forma como se va a comunicar el cliente con el servidos de aplicaciones y este como se comunica al servidor de base de datos, debe ser una forma tal que no exista saturación ni cuellos de botella a la hora de obtener la información, aunque todo esto depende del estudio de la arquitectura que se le implemente. Las dos arquitecturas que hasta el momento han dado buenos resultados a la hora de acceder a los datos son los conocidos web services también se los incluye aquí WCF (Windows Communication Fundation), y los conocidos COM+.

¿Qué es Windows Communication Fundation?

La aceptación global de servicios Web que incluye los protocolos estándar para la comunicación de aplicación a aplicación, ha cambiado el desarrollo de software. Por ejemplo, las funciones que proporcionan los servicios Web ahora incluyen seguridad, coordinación de transacciones distribuidas y una comunicación fiable. Las ventajas de los cambios en servicios Web se deberían reflejar en las herramientas y tecnologías que los programadores utilizan. Windows Communication Foundation (WCF) está diseñado para ofrecer un enfoque manejable a la informática distribuida, interoperabilidad ancha y asistencia directa para la orientación sobre el servicio.

WCF simplifica el desarrollo de aplicaciones conectadas a través de un nuevo modelo de programación orientado a servicios. WCF admite muchos estilos de desarrollo de aplicaciones distribuidas proporcionando una arquitectura superpuesta. En su base, la arquitectura de canal de WCF proporciona primitivos asíncronos de paso de aprobación de mensajes sin tipo. Generados sobre esta base están las funciones de protocolos para un intercambio de datos de transacción seguro y fiable, así como una amplia variedad de opciones de codificación y transporte.

El modelo de programación tipificada (llamado *modelo de servicio*) está diseñado para facilitar el desarrollo de aplicaciones distribuidas y proporcionar a los desarrolladores pericia en servicios Web ASP.NET, comunicación remota

.NET Framework y Enterprise Services, así como a aquellos que llegan a WCF con cierta experiencia en desarrollo. El modelo de servicio presenta una asignación sencilla de conceptos de servicios Web para aquellos de Common Language Runtime (CLR) .NET Framework, incluyendo la asignación ampliable y flexible de mensajes para la implementación de servicios en lenguajes como Visual C# o Visual Basic. Incluye funciones de serialización que habilitan el acoplamiento separado y el control de versiones y proporciona integración e interoperabilidad con sistemas distribuidos .NET Framework existentes, como Message Queue Server (MSMQ), COM+, servicios Web ASP.NET, Mejoras de servicios Web (WSE) y varias funciones más.

Ejemplo del problema

El siguiente ejemplo muestra algunos de los problemas que WCF resuelve. Una compañía de alquiler de coches decide crear una nueva aplicación para reservar los coches.

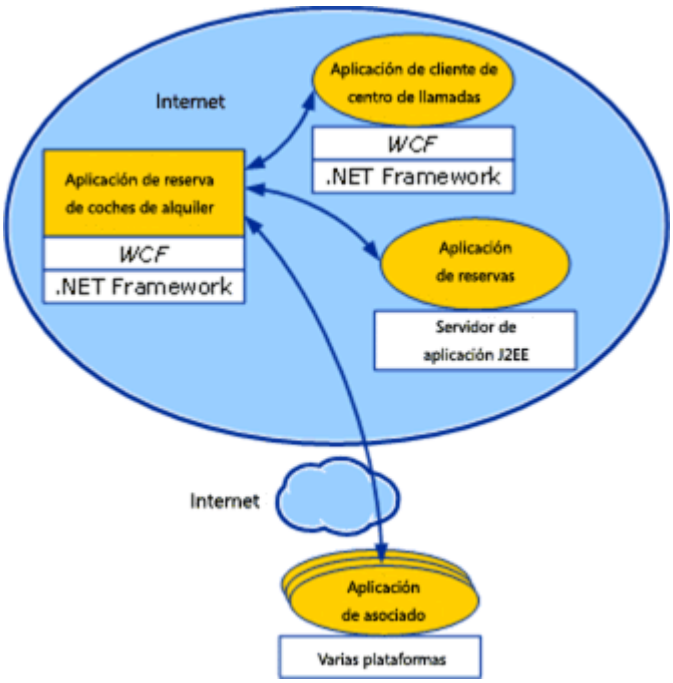

Figura 1.1 Ejemplo del Problema

Los creadores de esta aplicación de reserva de coches de alquiler saben que la lógica empresarial que implementa debe ser accesible por otro software que se ejecute dentro y fuera de la compañía. De acuerdo con la Figura 1.1,

deciden integrarlo en un estilo orientado a servicios, con la lógica de la aplicación expuesta a otro software a través de un conjunto bien determinado de servicios. Para implementar estos servicios y así comunica con otro software, la nueva aplicación utilizará WCF.

1.4.2. Que contiene una aplicación de escritorio.

Las aplicación es de escritorio han venido desarrollándose desde varias décadas atrás dando grandes beneficios a las empresas que lo usan, llevando sus datos de forma ordenada y coherente.
Existen varias aplicaciones de escritorio que enumeraremos a continuación:

Aplicaciones de monitoreo.- Son aquellas aplicaciones que se usan para controlar los accesos a la red, uso de misma, uso de internet, historial de navegaciones. Este tipo de aplicaciones no necesariamente necesitan base de datos por poder funcionar, la mayoría de estas aplicaciones usan archivo planos, tal como se usaban en la antigüedad.

Aplicaciones de multimedia.- Este tipo de aplicación son las usadas para ayudara al aprendizaje a los niños o personas que lo necesiten, como lo pueden ser para personas con defectos físicos de oídos, ojos, boca. Estos casi nunca usan una base de datos para su funcionamiento, salvo el caso que este implementado con inteligencia artificial donde el sistema vaya alimentándose de la información de la persona que lo usa tal como lo fuese un robot.

Aplicaciones de consola (Juegos).- Aquí se incluye también lo que es multimedia de tal forma que la aplicación se atractiva y entretenida a la vista del usuario, este tipo de aplicaciones casi nunca usan un almacenamiento de base de datos solido, simplemente usan archivos planos o estructurados como los son los XML.

Aplicaciones con Acceso a BDD.- Con este tipo de aplicaciones podemos lograr que las personas que lo usen tengan su información salvaguardada de tal forma que si lo desean averiguar en un futuro esta les pueda brindar dicha

información. Este tipo de aplicaciones son usadas por pequeñas y grandes empresas que desean tener su información ya sea contable o de producción de forma segura y poder proyectar su información para obtener un flujo de ventas o información que sea de suma importancia para la empresa adquiriente.

No debemos olvidar que las aplicaciones en la actualidad han ido evolucionando que el aspecto visual impacta mucho a los usuarios de las aplicaciones, no es lo mismo una aplicación de C++ en consola que una aplicación den 3D de .NET y WPF.

1.4.3. Multimedia en una aplicación de escritorio

Para poder revisar el aspecto de la multimedia dentro de una aplicación de escritorio mencionaremos los controles que se pueden usar en una aplicación de escritorio de FORM's y un aplicación de escritorio de WPF.

SONIDO.- Este control es aplicable para los dos tipos de aplicaciones, desde un archivo .midi hasta un archivo .mp3.

IMAGEN.- Las imágenes en Windows forms se lo puede representar en el plano, mientras que en WPF se puede representar con contrastes, transparencias y hasta en 3D.

VIDEO.- Aplicables en las dos formas, con mayor utilidad en WPF ya que se puede controlar y editar el video dentro de la misma herramienta. Aquí se puede usar cualquier extensión que se encuentre hábil en el cliente.

ANIMACIONES.- Este control solo es aplicable en WPF ya que en Windows forms no contiene este tipo de controles, el uso de estilos como si fuese un sistema web hace que facilite el diseño de la aplicación en lo que es WPF.

WPF usa todos estos controles que unificados se obtienen controles atractivos los mismo que podrán ser usados en un formulario de WPF o en formularios web desarrollados en WPF.

1.5. Aplicaciones Web

Historia.- Las aplicaciones web comenzó como unas simples paginas que se usaban para navegar entre documentos esto era usado para crear documentación con enlaces a diferentes documentos ya sea en la misma página como de otros documentos. El paso inmediatamente posterior en su evolución fue la inclusión de un método para elaborar páginas dinámicas que permitieran que lo mostrado tuviese carácter dinámico (es decir, generado a partir de los datos de la petición). Este método fue conocido como CGI ("Common Gateway Interface") y definía un mecanismo mediante el que se podía pasar información entre el servidor y ciertos programas externos. Los CGIs siguen utilizándose ampliamente; la mayoría de los servidores web permiten su uso debido a su sencillez. Además, dan total libertad para elegir el lenguaje de programación que se desea emplear.

El funcionamiento de los CGIs tenía un punto débil: cada vez que se recibía una petición, el servidor debía lanzar un proceso para ejecutar el programa CGI. Como la mayoría de CGIs estaban escritos en lenguajes interpretados, como Perl o Python, o en lenguajes que requerían "run-time environment", como Java o VisualBasic, el servidor se veía sometido a una gran carga. La concurrencia de múltiples accesos al CGI podía comportar problemas graves.

Por eso se empiezan a desarrollar alternativas a los CGIs que solucionaran el problema del rendimiento. Las soluciones llegan básicamente por 2 vías: 1) se diseñan sistemas de ejecución de módulos mejor integrados con el servidor, que evitan la instanciación y ejecución de varios programas, y 2) se dota a los servidores un intérprete de algún lenguaje de programación que permita incluir el código en las páginas de forma que lo ejecute el servidor, reduciendo el intervalo de respuesta.

Entonces se experimenta un aumento del número de arquitecturas y lenguajes que permiten desarrollar aplicaciones web. Todas siguen alguna de estas vías. Las más útiles y las más utilizadas son las que permiten mezclar los 2 sistemas: un lenguaje integrado que permita al servidor interpretar comandos "incrustados" en las páginas HTML y, además, un sistema de ejecución de programas mejor enlazado con el servidor, que no implique los problemas de rendimiento propios de los CGIs.

Una de las más potentes es la seguida por Sun Microsystems con su Java, integrado por 2 componentes; un lenguaje que permite la incrustación de código en las páginas HTML que el servidor convierte en programas ejecutables, JSP ("Java Server Pages" o "Páginas de Servidor de Java"), y un método de programación muy ligado al servidor, con un rendimiento superior a los CGIs, denominado "Java Servlet".

Otra tecnología de éxito y una de las más utilizadas es el lenguaje PHP. Se trata de un lenguaje interpretado que permite la incrustación de HTML en los programas, con una sintaxis derivada de C y Perl. El hecho de ser sencillo y potente ha contribuido a hacer de PHP una herramienta muy apropiada para determinados desarrollos.

1.5.1. Estudio del uso de aplicaciones Web

En tiempos pasados no se tenía la idea de usar una aplicación web como un sistema que controle los datos de alguna empresa, las conexiones a datos ni se las imaginaban, pero vemos en la actualidad que ya existen varios sistemas que controlan los datos de las empresas de forma eficiente y segura, siendo esta vista como un punto ambicioso para las empresas desarrolladoras de software, tanto que en la actualidad muchas GUI de desarrollo se han implementado.
A continuación mencionaremos algunas de las empresas que vieron una visión tentadora y desarrollaron aplicaciones para diseñar aplicaciones web.

ASP .NET.- es un framework para aplicaciones web desarrollado y comercializado por Microsoft. Es usado por programadores para construir sitios web dinámicos, aplicaciones web y servicios web XML. Apareció en enero de

2002 con la versión 1.0 del .NET Framework, y es la tecnología sucesora de la tecnología Active Server Pages (ASP). ASP.NET está construido sobre el Common Language Runtime, permitiendo a los programadores escribir código ASP.NET usando cualquier lenguaje admitido por el .NET Framework.

Cualquier persona que está familiarizada con el desarrollo de aplicaciones web sabrá que el desarrollo web no es una tarea simple. Ya que mientras que un modelo de programación para aplicaciones de uso común está muy bien establecido y soportado por un gran número de lenguajes, herramientas de desarrollo, la programación web es una mezcla de varios lenguajes de etiquetas, un gran uso de lenguajes de script y plataformas de servidor. Por desgracia para el programador de nivel intermedio, el conocimiento y habilidades que se necesitan para desarrollar aplicaciones web tienen muy poco en común con las que son necesarias en el desarrollo tradicional de aplicaciones. Las páginas de ASP.NET, conocidas oficialmente como "web forms" (formularios web), son el principal medio de construcción para el desarrollo de aplicaciones de formularios web están contenidos en archivos con una extensión ASPX; en jerga de programación, estos archivos típicamente contienen etiquetas HTML o XHTML estático, y también etiquetas definiendo Controles Web que se procesan del lado del servidor y Controles de Usuario donde los desarrolladores colocan todo el código estático y dinámico requerido por la página web. Adicionalmente, el código dinámico que se ejecuta en el servidor puede ser colocado en una página dentro de un bloque <% -- código dinámico -- %> que es muy similar a otras tecnologías de desarrollo como PHP, JSP y ASP, pero esta práctica es, generalmente, desaconsejada excepto para propósitos de enlace de datos pues requiere más llamadas cuando se genera la página.

Este es una ejemplo que utiliza código "en línea", opuesto al código independiente (Code-behind).

```
<%@ Page Language="C#" %>
<! DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<script runat="server">
   protected void Page_Load(object sender, EventArgs e)
   {
      Label1.Text = DateTime.Now.ToLongDateString();
   }
</script>
<html xmlns="http://www.w3.org/1999/xhtml" >
```

```
<head runat="server">
   <title>Página de Ejemplo</title>
</head>
<body>
   <form id="form1" runat="server">
   <div>
      <asp:Label runat="server" id="Label1" />
   </div>
   </form>
</body>
</html>
```

JAVA (JSP).- JSP fue desarrollado inicialmente por un grupo de trabajo, el cual se encontraba bajo la supervisión de Sun Microsystems. El proyecto comenzó a finales de 1997 (bajo su nombre actual: JSP) y su desarrollo fue anunciado en la conferencia de JavaOne en marzo de 1998. La versión 0.91 de JSP (terminada en junio de 1998) fue la primera en ser puesta en práctica sin problemas. Un año más tarde, la versión 1.0 de JSP fue la primera versión que fue abierta al público en general.

JSP (**JavaServer Pages**).- es la tecnología de Sun para incluir contenido HTML dinámico generado con Java en páginas HTML estáticas; algo así como el equivalente Java a ASP y PHP.

Los JSP corren dentro de un producto software llamado "contenedor web" que les proporciona una serie de servicios y que está formado por un contenedor JSP y un contenedor de **servlets**. Un ejemplo de esta clase de aplicación podría ser Tomcat.

Los JSP están muy relacionados con los servlets, que no son más que clases que derivan de la clase GenericServlet y que cuentan con métodos en los que incluir código a ejecutar cuando les llegue una petición HTTP (doPost si es POST o doGet si es GET). Esta estrecha relación viene dada porque, en realidad, los JSP son transformados en servlets cuando accedemos a ellos por el motor JSP del contenedor web en el corren (el motor JSP de Tomcat se llama Jasper).

Características de JSP

- Separación de la presentación y el contenido.

Con JSP se programa dentro de las páginas con código de JAVA, pero encerrando el código en un conjunto de marcas que sólo se interpretan en el servidor, al momento de ejecutar la aplicación.

Este sistema hace que quede bien delimitado dónde empieza el trabajo de los desarrolladores y dónde termina. El diseñador no se preocupa por ese contenido sino sólo por el diseño.

- Re-uso de componentes.

El modelo de uso de JSP se basa en la reutilización de componentes Java Beans. El uso de los mismos hace que se optimice considerablemente la utilización de recursos en el servidor. La consigna es: en las páginas se codifica dentro de tags o marcas y para resolver tareas complejas se accede a componentes beans reutilizables por todos.

- Uso de XML dentro de los scripts.

Es una realidad que los programadores de páginas web, no siempre están familiarizados con todos los lenguajes de programación. Por tanto, este nuevo acercamiento les brinda la posibilidad de embeber lenguaje de marcas más complejos como XML para acceder a diferentes componentes o para levantar applets en los clientes.

PHP (Hypertext Pre-processor).- Es un lenguaje creado por una gran comunidad de personas. El sistema fue desarrollado originalmente en el año 1994 por Rasmus Lerdorf como un CGI escrito en C que permitía la interpretación de un número limitado de comandos. El sistema fue denominado Personal Home Page Tools y adquirió relativo éxito gracias a que otras personas pidieron a Rasmus que les permitiese utilizar sus programas en sus propias páginas. Dada la aceptación del primer PHP y de manera adicional, su creador diseñó un sistema para procesar formularios al que le atribuyó el nombre de FI (Form Interpreter) y el conjunto de estas dos herramientas, sería la primera versión compacta del lenguaje: PHP/FI.
Es usado principalmente en interpretación del lado del servidor (server-side scripting) pero actualmente puede ser utilizado desde una interfaz de línea de

comandos o en la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica usando las bibliotecas Qt o GTK+.

El gran parecido que posee PHP con los lenguajes más comunes de programación estructurada, como C y Perl, permiten a la mayoría de los programadores crear aplicaciones complejas con una curva de aprendizaje muy corta. También les permite involucrarse con aplicaciones de contenido dinámico sin tener que aprender todo un nuevo grupo de funciones.

Aunque todo en su diseño está orientado a facilitar la creación de página web, es posible crear aplicaciones con una interfaz gráfica para el usuario, utilizando la extensión PHP-Qt o PHP-GTK. También puede ser usado desde la línea de órdenes, de la misma manera como Perl o Python pueden hacerlo, a esta versión de PHP se la llama PHP CLI (Command Line Interface).

Cuando el cliente hace una petición al servidor para que le envíe una página web, el servidor ejecuta el intérprete de PHP. Éste procesa el script solicitado que generará el contenido de manera dinámica (por ejemplo obteniendo información de una base de datos). El resultado es enviado por el intérprete al servidor, quien a su vez se lo envía al cliente. Mediante extensiones es también posible la generación de archivos PDF, Flash, así como imágenes en diferentes formatos.

Permite la conexión a diferentes tipos de servidores de bases de datos tales como MySQL, Postgres, Oracle, ODBC, DB2, Microsoft SQL Server, Firebird y SQLite.

PHP también tiene la capacidad de ser ejecutado en la mayoría de los sistemas operativos, tales como UNIX (y de ese tipo, como Linux o Mac OS X) y Windows, y puede interactuar con los servidores de web más populares ya que existe en versión CGI, módulo para Apache, e ISAPI.

Luego de una breve explicación de los principales competidores de aplicaciones diremos que las los sistemas que se generan mediante esta tecnología será usada de forma rápida sin la necesidad de ser instalado en el cliente, lo que se debe tomar en cuenta es los requerimientos del cliente

satisfaga las necesidades de la aplicación, si hablamos de Microsoft diremos que lo prescindible para que una aplicación web corra sin problemas se debe tener instalado el framework si es posible la 4.0 que es la última versión que existe en la actualidad, si hablamos de JAVA diremos que es necesario instalar SUN y el servidor de aplicaciones que puede ser Tomcat.

No debemos olvidar que el riesgo de usar una aplicación web es la seguridad de la información que viaja por el canal del internet, la misma que va a ser sometida a terribles hackers que trataran de acceder a la información y hacer provecho de esa información capturada. Los bancos son las principales empresas que pueden estar en riesgos si no se maneja una adecuada seguridad para la información.

1.5.2. Que contiene una aplicación Web

Una aplicación web a mas de brindar la manipulación de los datos es necesario que se atractivo para la vista de los usuarios así como también debe brindar un área de información para el usuario como puede ser sus tareas pendientes, procesos ejecutados, procesos en ejecución en fin un sin número de adicionales que se pueden incluir en la aplicación para que este sea aprovechada a un 100%. La parte del diseño gráfico toma un papel muy importante a la hora de entregar un sistema web, la parte de imágenes, sonido en el caso de que lo requieran.

Como las aplicaciones de escritorio estas están divididas en varias áreas como pueden ser sistemas de control de transferencias, consulta de saldos en el caso de los bancos, el uso de sonido y videos en el caso de portales que venden la imagen y los productos que en ella se incorporan este es el caso de un sistema de venta de vehículos, venta de hospedaje, entre otros.

1.5.3. Multimedia en una aplicación Web

En todo este capítulo hemos la historia de la web así como su funcionamiento sus complejidades su contenido, sus competencias y sobre todo lo imprescindible que es el aspecto o la forma que se le dé al sistema en el

aspecto visual de tal forma que este se atractivo y cumpla todas las reglas de un sitio web óptimo, claro que para esto es un modulo muy aparte como la construcción misma de un sitio web.

Ahora veremos lo importante que es que una aplicación web contenga controles multimedia.

Si un sitio web no contuviese ninguna imagen, color o sonido esta dejaría de ser atractiva para el usuario y sería rechazada a la primera vista, muchos estudios certifican que las aplicaciones sin calidad de multimedia dan mucho que hablar a la hora de elegir entre otras opciones.

Imaginémonos un portal sin imágenes y animaciones, las visitas al él serian escasas y nadie realizaría compras si ese fuese el caso, el portal debe ser la carta de presentación de la organización, corporación o compañía que tenga sistemas web, es por esta y mucha más razones que los controles multimedia son importantes en una aplicación web. A lo contrario de las aplicaciones de escritorio ya que a él solo van a tener acceso personas de la misma empresa o compañía y la web podrán ser visitadas por personas que pertenezcan o no a la misma empresa o compañía dejando como primera impresión la imagen del sistema web.

1.6. Es posible manejar aplicaciones de escritorio y web con un mismo GUI?

Antes de existir las herramientas que en la actualidad existen diríamos que manejar aplicaciones de escritorio y web dentro de un mismo aplicativo era imposible, y que si lo había eran los únicos applets de java que eran aplicaciones de escritorio que podrían verse en una aplicación web. Desde allí pasaron varios años para que la competencia incorpore técnicas y arquitecturas que puedan manejar esta forma de diseñar aplicaciones y sistemas para empresas y compañías que lo requieran.

Para el estudio de la tesis en proceso realizaremos la investigación de WPF y Silverligth que son GUIS de desarrollo que manejan este tipo de alternativa ambiciosa para las personas que si bien desean usarla como escritorio o si desean usarla como web no tengan ninguna restricción. A continuación

mostraremos cómo es posible realizar que aplicaciones de escritorio y web se manejen en un mismo GUI. Lo que debemos saber es que lo que se desarrolle sea reutilizado, eso es el caso de aplicaciones de escritorio que sea usado en aplicaciones web ya sea ventanas de búsqueda de artículos, oficinas, entre otras. Las mismas que serán manejados como objetos llamada a métodos, paso de parámetros, etc. Es así como funciona esta nueva arquitectura diseñada por Microsoft realizando que aplicación es de escritorio ya en funcionamiento sean usadas en un sitio web.

El requerimiento que debe cumplir una aplicación de escritorio que vaya ser usada en un sitio web es que la arquitectura de conexión a base de datos debe ser construida con la arquitectura Web Service o WCF (Windows Presentation Fundation) que son accesos al servidor web y este se conecte al servidor de base de datos para devolver las peticiones del cliente, claro está si este sistema correrá en el internet. De ser el caso que la aplicación corra dentro del área local o dentro de la compañía el uso de ADO .NET o COM+ son las opciones que se les pueden agregar para la construcción de sistemas. Con esta explicación se da respuesta a la pregunta realizada, si es posible en la actualidad construir aplicaciones web y de escritorio dentro de un mismo GUI.