UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Sistema de Apoyo a la Contabilidad Electrónica

INFORME

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO EN COMPUTACIÓN

P R E S E N T A :

IGNACIO GABINO JOSÉ LUIS

DIRECTOR(A): M.I NORMA ELVA CHÁVEZ RODRÍGUEZ

2015

Índice

Introducción

En este trabajo se presenta el desarrollo de un Sistema de apoyo a la contabilidad electrónica, que ayuda a reducir considerablemente el tiempo de captura y verificación de un CFDI por un contador.

El sistema está constituido por 5 partes: Subir archivos, Comprobación, Búsqueda, Recuperación del XML y Reportes de Gastos en Excel, las cuales serán desarrolladas en el transcurso de este documento.

El objetivo de este proyecto es poder validar los archivos XML ante el SAT, verificando que los CFDI emitidos existan en su base de datos (Verificar que no sean apócrifos) para esto también se valida que los archivos se encuentren vigentes. Además de reducir el tiempo en el que un contador tarda en validar y capturar un CFDI.

Y cumplir con la Resolución Miscelánea Fiscal del 2013, publicada en el Diario Oficial de la Federación, en su regla 1.2.7.1.1 y 1.2.8.3.1.11 señala que los CFDI deberán ser conservados en medios magnéticos, ópticos o de cualquier otra tecnología en su formato electrónico XML.

Capítulo 1 Generalidades

1.1. Objetivo de la empresa con respecto a la aplicación

Posicionarse en el mercado como una empresa confiable e innovadora en ofrecer servicios basados en las nuevas tecnologías de información y telecomunicaciones para ayudar a las PyMEs a automatizar ciertos procesos que permitan reducir costos y tiempo operativos.

1.2. Objetivo de la Aplicación

El principal objetivo de la aplicación es de reducir los tiempos de captura de un concentrado de facturas en Excel para cualquier persona con conocimientos básicos en contabilidad ya sea un contador, un despacho contable o el mismo contribuyente.

Además de poder validar ante el SAT, si los CFDI generados son válidos, con la flexibilidad de uno a varios archivos.

1.3. Planteamiento del Problema

El principal problema que al mismo tiempo representa una gran oportunidad para los objetivos de este trabajo es el de ofrecer una herramienta que reduzca los tiempos en el que un contador tarda en validar y capturar cada factura en su concentrado.

El tiempo aproximado en el que un contador tarda en buscar una factura, abrirla copiar cantidades a su Excel es de 2 a 4 minutos por cada una de las facturas.

Además de poder cumplir con la Resolución Miscelánea Fiscal del 2013, publicada en el Diario Oficial de la Federación, en su regla 1.2.7.1.1 y 1.2.8.3.1.11

1.4. Propuesta de Solución

Desarrollar un sistema de fácil manejo y sin complicaciones que cuente con protocolos de seguridad y encriptación para que su información se encuentre segura además de incrementar la productividad laboral reduciendo los tiempos de captura.

Con opción de recuperar sus archivos XML en cualquier momento por diferentes tipos de búsqueda, para su comodidad y así poder cumplir con la Resolución Miscelánea Fiscal del 2013, publicada en el Diario Oficial de la Federación.

Capítulo 2 Marco Teórico

2.1. Sistemas de Información

Es un conjunto de elementos que interactúan entre sí con el fin de apoyar las actividades de una empresa o negocio.

Es el conjunto total de procedimientos, operaciones, funciones y difusión de datos o información en una organización.

2.1.1. Componentes del Sistema de Información

Un Sistema de Información realiza cuatro actividades básicas: entrada de información, almacenamiento, procesamiento y salida de información.

- Entrada de información:

Es el proceso mediante el cual el Sistema de Información toma los datos que requiere para procesar la información. Las entradas pueden ser manuales o automáticas. Las **manuales** son aquellas que se proporcionan en forma directa por el usuario, mientras que las **automáticas** son datos o información que provienen o son tomados de otros sistemas o módulos. Esto último se denomina interfaces automáticas.

Las unidades típicas de entrada de datos a las computadoras son las terminales, las cintas magnéticas, las unidades de discos, los códigos de barras, los escáner, la voz, los monitores sensibles al tacto, el teclado y el ratón, entre otras cosas.

- Almacenamiento de Información:

Es una de las actividades o capacidades más importantes que tiene una computadora, ya que a través de esta propiedad el sistema puede recordar la información guardada en la sesión o proceso anterior. Esta información suele ser almacenada en estructuras de información denominadas ficheros. La unidad típica de almacenamiento son los discos magnéticos o duros, de estado sólido, los discos compactos (CD-ROM).

- Procesamiento de Información:

Es la capacidad del Sistema de Información para efectuar cálculos de acuerdo con una secuencia de operaciones preestablecidas. Estos cálculos pueden efectuarse con datos introducidos recientemente en el sistema o bien con datos que están almacenados. Esta característica de los sistemas permite la transformación de datos fuente en información que puede ser utilizada para la toma de decisiones.

❖ Salida de Información:

La salida es la capacidad de un Sistema de Información para sacar la información procesada o bien datos de entrada al exterior. Las unidades típicas de salida son las impresoras, terminales, cintas magnéticas, la voz, los graficadores y los plotters, entre otros. Es importante aclarar que la salida de un Sistema de Información puede constituir la entrada a otro Sistema de Información o módulo. En este caso también existe una interface automática de salida.

2.1.2. Objetivos del Sistema de Información

Algunos de los principales objetivos de los Sistemas de Información, son:

- Proporcionar datos oportunos y exactos que permitan tomas de decisiones acertadas y mejorar la relación entre los recursos de la empresa.
- Garantizar información exacta y confiable, así como su almacenamiento de tal forma que esté disponible cuando se necesite.
- Servir como herramienta para que los gerentes realicen planeación, control y toma de decisiones en sus empresas.

2.1.3. Clasificación de los Sistemas de Información

❖ **Sistemas transaccionales:**

A través de éstos suelen lograrse ahorros significativos de mano de obra, debido a que automatizan tareas operativas de la organización.

Con frecuencia son el primer tipo de Sistemas de Información que se implanta en las organizaciones. Se empieza apoyando las tareas a nivel operativo de la organización para continuar con los mandos intermedios y posteriormente con la alta administración conforme evolucionan.

Son intensivos en entrada y salida de información; sus cálculos y procesos suelen ser simples y poco sofisticados. Estos sistemas requieren mucho manejo de datos para poder realizar sus operaciones y como resultado generan también grandes volúmenes de información.

Tienen la propiedad de ser recolectores de información, es decir, a través de estos sistemas se cargan las grandes bases de información para su explotación posterior. Estos sistemas son los encargados de integrar gran cantidad de la información que se maneja en la organización, la cual será utilizada posteriormente para apoyar a los mandos intermedios y altos.

Son fáciles de justificar ante la dirección general, ya que sus beneficios son visibles y palpables. El proceso de justificación puede realizarse enfrentando ingresos y costos.

Esto se debe a que en el corto plazo se pueden evaluar los resultados y las ventajas que se derivan del uso de este tipo de sistemas. Entre las ventajas que pueden medirse se encuentra el ahorro de trabajo manual.

Son fácilmente adaptables a paquetes de aplicación que se encuentran en el mercado, ya que automatizan los procesos básicos que por lo general son similares o iguales en otras organizaciones. Ejemplos de este tipo de sistemas son la facturación, nóminas, cuentas por cobrar, cuentas por pagar, contabilidad general, conciliaciones bancarias, inventarios, etcétera.

- **Sistemas de Apoyo a las decisiones**

Las principales características de estos sistemas son las siguientes:

Suelen introducirse después de haber implantado los Sistemas Transaccionales más relevantes de la empresa, ya que estos últimos constituyen su plataforma de información.

La información que generan sirve de apoyo a los mandos intermedios y a la alta administración en el proceso de toma de decisiones.

Suelen ser intensivos en cálculos y escasos en entradas y salidas de información. Así, por ejemplo, un modelo de planeación financiera requiere poca información de entrada, genera poca información como resultado, pero puede realizar muchos cálculos durante su proceso.

No suelen ahorrar mano de obra. Debido a ello, la justificación económica para el desarrollo de estos sistemas es difícil, ya que no se conocen los ingresos del proyecto de inversión.

Suelen ser Sistemas de Información interactivos y amigables, con altos estándares de diseño gráfico y visual, ya que están dirigidos al usuario final.
Apoyan la toma de decisiones que, por su misma naturaleza son repetitivas y de decisiones no estructuradas que no suelen repetirse. Por ejemplo, un Sistema de Compra de Materiales que indique cuándo debe hacerse un pedido al proveedor o un Sistema de Simulación de Negocios que apoye la decisión de introducir un nuevo producto al mercado.

Estos sistemas pueden ser desarrollados directamente por el usuario final sin la participación operativa de los analistas y programadores del área de Informática.

Este tipo de sistemas puede incluir la programación de la producción, compra de materiales, flujo de fondos, proyecciones financieras, modelos de simulación de negocios, modelos de inventarios, etcétera.

2.2. Base de Datos

La manipulación directa de los datos entraña una complejidad importante, además de que, de esta forma, se corre el riesgo de realizar operaciones incorrectas o no deseadas con dichos datos. Por este motivo, se han desarrollado una serie de programas informáticos que tratan de aislar al usuario final de los ficheros de datos: son los llamados Sistemas Gestores de Bases de Datos (SGDB – DBMS, *Data Base Management Systems* -), programas que se encargaran de gestionar y controlar el Acceso a los datos ofreciendo una representación más sencilla de ellos.
Los Sistemas Gestores de Bases de Datos más conocidos y extendidos hoy en día son Access de Microsoft, base, FileMaker, y Paradox en el segmento de aplicaciones para particulares y pequeñas empresas y SQL Server de Microsoft, Oracle, DB2, de IBM, Informix y Sybase, en el segmento de las medianas y grandes bases de Datos.

Podemos definir una base de datos como “un conjunto estructurado de datos que se guardan en un sistema informático y sobre los cuales es posible efectuar una serie de operaciones básicas de consulta, modificación, inserción o eliminación”.
Con objeto de acelerar la búsqueda de información, se han ideado distintos métodos de acceso rápido a los ficheros, siendo el más extendido el basado en la utilización de “índices”. Básicamente, un índice se puede construir mediante un fichero auxiliar que permite localizar dónde se encuentra cada uno de los registros del fichero auxiliar de datos.

De esta forma, se dota de un contenido semántico a la estructura de la base de datos. Además, hay que tener en cuenta que el SGDB reserva para cada tipo de datos un determinado espacio físico en el dispositivo de almacenamiento.
Una característica adicional de los tipos de datos es que permiten al SGDB controlar la inserción o modificación de datos, de manera que este se va encargar de detectar e impedir que se introduzcan valores inapropiados en un determinado campo de un registro.

El “modelo relacional”, propuesto por E. F. Codd, es el más utilizado por las modernas bases de datos, ya que cuenta con una sólida base conceptual lógico-matemática y resulta muy sencillo y potente a la vez. En este modelo los ficheros de datos se representan de manera abstracta mediante tablas de valores.
El modelo relacional no se aplicó a la práctica hasta que, a finales de los años setenta, las minicomputadoras y las grandes computadoras empezaron a disponer de suficiente capacidad de procesamiento para el desarrollo de grandes bases de datos.

IBM fue pionero en estas experiencias y desarrolló un lenguaje de consulta denominado SQL (del inglés Structure Query Language) que, a la postre, ha llegado convertirse en el estándar de las bases de datos relacionales.

En un sistema relacional los datos se relacionan por sus valores y no mediante punteros, en el modelo relacional, por lo tanto, las filas de una determinada tabla constituyen los registros guardados en el fichero de datos y, a su vez, las columnas representan los campos o atributos de dichos registros.

La “Clave Candidata” es el atributo o conjunto de atributos que permite identificar de manera univoca a un registro de la tabla (en la tabla no pueden existir dos o más registros en los cuales ese atributo o conjunto de atributos tenga el mismo valor).

Por su parte, la “Clave Primaria” es la clave que se utiliza para identificar de manera univoca a un registro (es la escogida entre las posibles claves candidatas).
Con los datos de las tablas es posible realizar múltiples tipos de operaciones, conocidas por el nombre genérico de consultas.

La característica que confiere una mayor potencia y flexibilidad al modelo relacional es la posibilidad de establecer relaciones entre dos o más tablas por medio de determinados campos clave.

De este modo, es posible realizar operaciones de cruce de datos entre tablas, así como llevar a cabo “uniones de tablas” (joins), obteniendo nuevas tablas cuyos registros se forman al combinar los datos de las nuevas tablas de partida. La relación de los datos se establece mediante “Claves Foráneas”.

2.2.1. Comparativa de los SGBD más Utilizados

	Características	Ventajas	Desventajas
ORACLE	**Entorno Cliente/Servidor**	**Es muy usado a nivel mundial**	**Muy costoso**
	Gestión de grandes BD	**Puede ejecutarse en todas las plataformas**	**Mal configurado es extremadamente lento**
	Usuarios Concurrentes	**Permite el uso de particiones para mejorar su eficiencia**	
	Alto rendimiento	**Un aceptable soporte**	
	Gestión Segura		
	Portabilidad		
	Compatibilidad		
SQL SERVER	**Nuevas herramientas integradas**	**Soporte de transacciones**	**Utiliza mucha memoria RAM**
	Recuperación rápida	**Estabilidad, escalabilidad y seguridad.**	**No es gratuito**
	Mejoras en la recopilación	**Soporta procedimientos almacenados**	**No es muy útil a la hora de las prácticas referente a la versión express.**
	Aislamiento de imágenes	**Incluye un potente entorno gráfico**	
MYSQL	**Interioridad y portabilidad**	**Es de código abierto**	**El soporte para disparadores es muy básico**
	Escrito en C y C++	**Bajo costo**	**No soporta algunas conversiones de datos**
	Probado con un alto rango de compiladores diferentes	**Facilidad de configuración**	**Los privilegios de las tablas no se borran automáticamente**
	Funciona en diferentes plataformas	**Usa licencia GPL**	
	Proporciona un buen almacenamiento	**Velocidad al realizar las operaciones**	
	Relativamente sencillo		

2.3. Lenguajes y Entornos de Programación

Un lenguaje de programación está constituido por una serie de instrucciones, de operadores y de reglas que permiten controlar el hardware de un equipo informático para que realice determinado proceso o función. Se trata de la herramienta básica para el desarrollo del software, entendiendo como tal el conjunto de aplicaciones y programas que se van ejecutar en un sistema informático.

A lo largo de la historia, se pueden distinguir varias generaciones de lenguajes de programación, en función a su nivel de abstracción y de su dependencia de la arquitectura hardware de la máquina sobre la cual se van a ejecutar las aplicaciones desarrolladas.

El primer tipo de lenguaje de programación es el código máquina, totalmente dependiente del conjunto de instrucciones del ordenador en que se va a ejecutar y en el que la programación se debe realizar en el sistema binario, codificado mediante ceros y unos las instrucciones y los datos a procesar. Se trata de la primera generación de lenguajes, muy difícil y engorrosa de utilizar y totalmente dependiente de la arquitectura del equipo (hardware).

El segundo tipo de lenguaje de programación es el conocido como Lenguaje Ensamblador, que representó en su día un importante avance al utilizar instrucciones y etiquetas que eran posteriormente traducidas por las correspondientes secuencias de ceros y unos (Código Máquina), que es lo que, en definitiva, entiende el ordenador.

Con la aparición de los lenguajes de Alto nivel, surge una nueva generación, más orientada a la resolución general de operaciones, con independencia de la máquina en la que se realice tal operación, y que incorporan un nuevo nivel de abstracción. Con ello, se facilita enormemente la programación, al utilizar un conjunto de expresiones y operaciones más amigables y próximos al lenguaje natural, que equivalen a varias decenas de instrucciones básicas en código máquina.

De este modo, se mejora la legibilidad del código fuente de los programas y se reduce su tamaño, con la ventaja añadida que supone su portabilidad, es decir, que con un mismo programa de alto nivel pueda ser traducido al código máquina de distintos ordenadores. En estos lenguajes procedimentales de tercera generación, como Basic, C, Fortran, Cobol, Pascal, Etc., el programador debe indicar el algoritmo con la secuencia de comandos que debe ejecutar el ordenador para cumplir con las distintas funcionalidades de la aplicación.

Los entornos de programación cuentan con herramientas que posibilitan y facilitan la labor de programación, entre las cuales podemos citar:

- **Editor:** es la herramienta que permite escribir y revisar el código fuente, de acuerdo con el léxico (conjunto de instrucciones) y las reglas sintácticas del lenguaje de programación utilizado.
- **Depurador:** se trata de una herramienta que analiza el código generado para detectar errores lógicos o de sintaxis, así como posibles fallos en la aplicación, para eliminar errores en la codificación del programa.
- **Compilador:** los "lenguajes interpretados" van traduciendo las instrucciones a código máquina a medida que se va ejecutando la aplicación. A su vez, los "lenguajes compilados" son traducidos a código máquina una sola vez (al finalizar la creación del programa), obteniendo así un programa ejecutable directamente en la máquina. Esta labor es realizada por el compilador y proporciona una aplicación más rápida y eficiente, por cuanto no tiene que ser traducida cada vez que se va a ejecutar en el ordenador.
- **Enlazador:** es el encargado de enlazar el código máquina obtenido del compilador con las distintas librerías del sistema operativo. Estas librerías facilitan una serie de funciones básicas sobre las que se puede apoyar el programador para simplificar la creación de sus aplicaciones.

2.3.1. C Sharp

Microsoft C# es un nuevo lenguaje de programación diseñado para crear un amplio número de aplicaciones empresariales que se ejecutan en .NET Framework. Supone una evolución de Microsoft C y Microsoft C++; es sencillo, moderno, proporciona seguridad de tipos y está orientado a objetos. El código creado mediante C# se compila como código administrado, lo cual significa que se beneficia de los servicios de Common Language Runtime. Estos servicios incluyen interoperabilidad entre lenguajes, recolección de elementos no utilizados, mejora de la seguridad y mayor compatibilidad entre versiones.

- ❖ Características

Es un lenguaje de programación orientado a objetos.

Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma.NET el cual es similar al de Java aunque incluye mejoras derivadas de otros lenguajes.

Es un lenguaje de programación simple pero eficaz, diseñado para escribir aplicaciones empresariales.

Utiliza muchas de las características de C++ en las áreas de instrucciones, expresiones y operadores.

Presenta considerables mejoras e innovaciones en áreas como seguridad de tipos, control de versiones, eventos y recolección de elementos no utilizados (liberación de memoria).

Proporciona acceso a los tipos de API más comunes: .NET Framework, COM, Automatización y estilo C.

Asimismo, admite el modo **unsafe**, en el que se pueden utilizar punteros para manipular memoria que no se encuentra bajo el control del recolector de elementos no utilizados.

2.3.2. ASP.NET

ASP.NET es un modelo de desarrollo Web unificado que incluye los servicios necesarios para crear aplicaciones Web empresariales con el código mínimo. ASP.NET forma parte de .NET Framework y al codificar las aplicaciones ASP.NET tiene acceso a las clases en .NET Framework. El código de las aplicaciones puede escribirse en cualquier lenguaje compatible con el Common Language Runtime (CLR), entre ellos Microsoft Visual Basic, C#, JScript .NET y J#. Estos lenguajes permiten desarrollar aplicaciones ASP.NET que se benefician del Common Language Runtime, seguridad de tipos, herencia.

- Características

Páginas: Las páginas de ASP.NET, conocidas oficialmente como "*web forms*" (formularios web), son el principal medio de construcción para el desarrollo de aplicaciones web. Los 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.

Modelo Code-behind: Microsoft recomienda que para realizar programación dinámica se use el modelo **code-behind**, o de respaldo, que coloca el código en un archivo separado o en una etiqueta de script especialmente diseñada.
Los nombres de los archivos *code-behind* están basados en el nombre del archivo ASPX tales como *MiPagina.aspx.cs* o *MiPagina.aspx.vb* (esta práctica se realiza automáticamente en Microsoft Visual Studio y otros entornos de desarrollo).

Cuando se usa este estilo de programación, el desarrollador escribe el código correspondiente a diferentes eventos, como la carga de la página, o el clic en un control, en vez de un recorrido lineal a través del documento.

El modelo *code-behind* de ASP.NET marca la separación del ASP clásico y alienta a los desarrolladores a construir aplicaciones con la idea de presentación y contenido separados en mente. En teoría, esto permite a un diseñador web, por ejemplo, enfocarse en la creación del diseño con menos posibilidades de alterar el código de programación mientras lo hace. Esto es similar a la separación en el Modelo Vista Controlador.

Controles de usuario: ASP.NET permite la creación de componentes reutilizables a través de la creación de Controles de Usuario (User Controls).
Un control de usuario sigue la misma estructura que un formulario web, excepto que los controles derivan de la clase System.Web.UI.UserControl, y son almacenados en archivos ASCX. Como los archivos ASPX, un ASCX contiene etiquetas HTML o XHTML, además de etiquetas para definir controles web y otros controles de usuario. También pueden usar el modelo code-behind.

- ❖ Estructura de los directorios

En general, la estructura de directorios de ASP.NET puede ser determinada por las preferencias del desarrollador. Aparte de unos pocos nombres de directorios reservados, el sitio puede expandirse a cualquier número de directorios. La estructura es típicamente reflejada directamente en las urls.

Los nombres de directorios especiales (a partir de ASP.NET 2.0 son):

App_Browsers: Contiene archivos de definición específicos para navegadores.

App_Code: Es un directorio para códigos. El servidor ASP.NET automáticamente compilará los archivos (y subdirectorios) en esta carpeta en un ensamblado que es accesible desde cualquier página del sitio.

Es típicamente usada para código de acceso a datos, código de modelo o código de negocios. También cualquier manejador http específico para el sitio e implementación de módulos y servicios web van este directorio.

App_Data: Directorio por defecto para las base de datos, tales como archivos mdb de Microsoft Access y archivos mdf de Microsoft SQL Server. Este directorio es usualmente el único con permisos de escritura en la aplicación.

App_LocalResources: Contiene archivos de recursos localizados para páginas individuales del sitio.

App_GlobalResources: Contiene archivos **resx** con recursos localizados disponibles para cada página del sitio. Este es donde el desarrollador ASP.NET típicamente almacenara mensajes que serán usados en más de una página.

App_Themes: usado para temas alternativos del sitio.

App_WebReferences: Usado para archivos de descubrimiento y archivos WSDL para referencias a servicios web para ser consumidos en el sitio.

Bin: Contiene código compilado (archivos .dll) para controles, componentes y otro código que pueda ser referenciado por la aplicación. Cualquier clase representada por código en la carpeta Bin es automáticamente referenciada en la aplicación. Son archivos o librerías que tienen como principal acción ejecutar una función cuando estas son llamadas o se invocan.

2.4. La Arquitectura Cliente Servidor

En el ámbito de TCP/IP las comunicaciones entre computadoras se rigen básicamente por lo que se llama modelo Cliente-Servidor, éste es un modelo que intenta proveer usabilidad, flexibilidad, interoperabilidad y escalabilidad en las comunicaciones. El término Cliente/Servidor fue usado por primera vez en 1980 para referirse a PC’s en red.

Su funcionamiento es sencillo: se tiene una máquina cliente, que requiere un servicio de una máquina servidor, y éste realiza la función para la que está programado.

En el modelo cliente servidor, el cliente envía un mensaje solicitando un determinado servicio a un servidor (hace una petición), y este envía uno o varios mensajes con la respuesta (provee el servicio) En un sistema distribuido cada máquina puede cumplir el rol de servidor para algunas tareas y el rol de cliente para otras.

De esta manera lo que plantea es que una computadora pueda realizar múltiples tareas por si sola.

2.4.1. Cliente

El cliente es el proceso que permite al usuario formular las solicitudes y pasarlos al servidor, se le conoce con el término front-end [1].

El cliente normalmente maneja todas las funciones relacionadas con la manipulación y despliegue de datos, por lo que están desarrollados sobre plataformas que permiten construir interfaces gráficas de usuario (GUI), además de acceder a los servicios distribuidos en cualquier parte de una red.

Las funciones que lleva a cabo el proceso cliente se resumen en los siguientes puntos:

- Administrar la interfaz de usuario
- Interactuar con el usuario
- Procesar la lógica de la aplicación y hacer validaciones locales
- Generar requerimientos de bases de datos
- Recibir resultados del servidor
- Formatear resultados

2.4.2. Servidor

Es el proceso encargado de atender múltiples clientes que hacen peticiones de algún recurso administrado por él. Al proceso servidor se le conoce con el término back-end [1].

El servidor normalmente maneja todas las funciones relacionadas con la mayoría de las reglas del negocio y los recursos de datos.
Las funciones que lleva a cabo el proceso servidor se resumen en los siguientes puntos:

- Aceptar los requerimientos de bases de datos que hacen los clientes.
- Procesar requerimientos de bases de datos.
- Formatear datos para transmitirlos a los clientes.
- Procesar la lógica de la aplicación y realizar validaciones a nivel de bases de datos.

2.4.2.1. Tipos de Servidores

Servidor de acceso a base datos: En un ambiente cliente/servidor, el proceso de aplicación se divide entre los sistemas cliente y servidor. Los servidores pueden ejecutar una parte de la lógica de negocio (Business logic) y la lógica de base de datos. Al igual que en los servidores de ficheros, los servidores de bases de datos ofrecen a los clientes el acceso a los datos que residen en el servidor. Sin embargo, los sistemas de gestión de bases de datos son más sofisticados que los métodos de acceso de E / S del fichero de básico.

Servidor web: Provee de contenidos estáticos a los navegadores. Este le envía los ficheros que carga por medio de la red al navegador del usuario. Los ficheros pueden ser imágenes, escrituras, documentos HTML y cualquier otro material web.

Servidor de aplicaciones: Proporciona servicios que soportan la ejecución y disponibilidad de las aplicaciones desplegadas. Es el corazón de un gran sistema distribuido.

2.5. Arquitectura de una aplicación web

La arquitectura de las aplicaciones web suele presentar un esquema de tres niveles:

- El primer nivel consiste en la capa de presentación que incluye no sólo el navegador, sino también el servidor web que es el responsable de presentar los datos un formato adecuado.
- El segundo nivel está referido habitualmente a algún tipo de programa o script.
- Finalmente, el tercer nivel proporciona al segundo los datos necesarios para su ejecución.
- Una aplicación Web típica recogerá datos del usuario (primer nivel), los enviará al servidor, que ejecutará un programa (segundo y tercer nivel) y cuyo resultado será formateado y presentado al usuario en el navegador (primer nivel otra vez).

2.5.1. Web Services

Es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de ordenadores como Internet. La interoperabilidad se consigue mediante la adopción de estándares abiertos.

Un Web Service puede ser registrado para poder dejarlo a disposición de otros usuarios y para que los mismos puedan localizarlo. Un mecanismo para registrar estos servicios es por medio de UDDI, sigla que corresponde a **Universal Description Discovery and Integration**, un “repositorio de Web Services”.

El mecanismo utilizado por un Web Service para especificar de qué forma hay que proporcionarle los datos, de manera tal que cualquiera pueda interaccionar con el mismo, es por medio de lenguaje **XML**. Esta información se almacena en un archivo llamado WSDL (**Web Services Description Language**), el cual contiene un documento XML junto con la descripción de ciertos mensajes SOAP y cómo deben intercambiarse, así como también dónde está el recurso del servicio y con qué protocolo debe dialogar quien lo consume.

El protocolo de comunicación utilizado es el **SOAP** generalmente, el cual es relativamente sencillo de utilizar.

Los Web Services utilizan protocolos comúnmente conocidos y difundidos tales como el formato XML, TCP/IP como protocolo de transporte y HTTP como protocolo de transferencia de hipertexto.

SOAP es un protocolo que define el formato XML para los mensajes de intercambio en el uso de un Web Service. Para aquellos programadores que solían utilizar llamadas del tipo RPC, SOAP también las soporta. Adicionalmente, es posible mediante SOAP definir un mensaje HTTP y este punto es de especial interés puesto que el protocolo imprescindible para Internet es HTTP.

Capítulo 3 Diseño de la BD

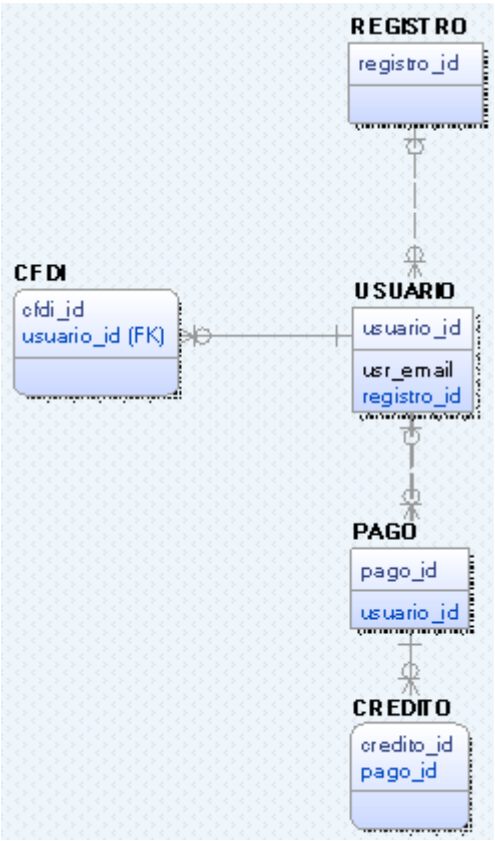
3.1. Modelo Entidad Relación

Es un modelo de datos basado en una percepción del mundo real que consiste en un conjunto de objetos básicos llamados entidades y relaciones entre estos objetos, implementándose en forma gráfica.

El modelo Entidad-Relación consta de:

- Tablas o Entidades: Es cualquier objeto, real o abstracto, que existe en un contexto determinado o puede llegar a existir y del cual deseamos guardar información.
- Atributos o Campos: Son características o propiedades asociadas a la entidad que toman valor en una instancia particular.
- Relación: Describe cierta dependencia entre entidades o permite la asociación de las mismas.

Modelo Entidad-Relación con las principales claves y entidades del negocio.



3.1.1. Descripción de las tablas

Entidad Registro: Esta entidad guarda la información básica de un nuevo registro con los atributos domicilio, nombre, etc.

Campo	Primary Key	Foreign Key	Constraint
registro_id	si	no	no

Entidad Usuario: En esta entidad se almacenan las claves de acceso para ingresar al sistema.

Campo	Primary Key	Foreign Key	Constraint
usuario_id	si	no	no
registro_id	no	si	no
usr_email	no	no	si

Entidad Crédito: En esta entidad se guardan la cantidad de créditos que un usuario va comprando, para poder seguir almacenando facturas. Un crédito es equivalente a una sola factura.

Campo	Primary Key	Foreign Key	Constraint
credito_id	si	no	no
pago _id	si	no	no

Entidad Pago: Esta entidad maneja la compra de créditos por un usuario, así como el tiempo en que un crédito tarda en expirar.

Campo	Primary Key	Foreign Key	Constraint
pago_id	si	no	no
credito_id	no	si	no

Entidad CFDI: Esta entidad es la más importante del negocio, ya que almacena los CFDI de cada usuario.

Campo	Primary Key	Foreign Key	Constraint
cfdi_id	si	no	no
usuario_id	si	no	no

3.1.2. Tipos de Datos

En SQL Server, cada columna, variable local, expresión y parámetro tiene un tipo de datos relacionado. Un tipo de datos es un atributo que especifica el tipo de datos que el objeto puede contener: datos de enteros, datos de caracteres, datos de moneda, datos de fecha y hora, cadenas binarias, etc.

Categorías de tipos de dato

Números Exactos:

- Bigint
- Numeric
- Bit
- Smallint
- Decimal
- Smallmoney
- Int
- Tyint
- Money

Números Aproximados:

- Float
- Real

Cadenas de caracteres:

- Char
- Varchar
- Text

Fecha y Hora:

- Date
- Datetime2
- Datetime
- Time
- Smalldatetime
- Datetimeoffset

Cadenas de Caracteres Unicode:

- Nchar
- Nvarchar
- Ntext

Cadenas Binarias:

- Binary
- Varbinary
- Image

Otros tipos de datos:

- cursor,
- timestamp,
- hierarchyid,
- uniqueidentifier,
- sql_variant,
- xml,
- tabla,
- tipos espaciales

En SQL Server, basado en sus características de almacenamiento, algunos tipos de datos están designados como pertenecientes a los siguientes grupos:

- Tipos de datos de valores grandes: varchar(max), nvarchar(max) y varbinary(max)
- Tipos de datos de objetos grandes: text, ntext, image, varchar(max), nvarchar(max), varbinary(max) y xml

3.1.3. Tipos de Encriptación

Encriptación en SQL Server 2012

El cifrado es el proceso consistente en ofuscar los datos mediante el uso de una clave o contraseña. Esto puede hacer que los datos sean inútiles sin la clave o contraseña de descifrado correspondiente. El cifrado no resuelve los problemas de control de acceso. Sin embargo, mejora la seguridad debido a que limita la pérdida de datos, incluso si se superan los controles de acceso. Por ejemplo, si el equipo host de base de datos no está configurado correctamente y un usuario malintencionado obtiene datos confidenciales, esa información robada podría resultar inservible si está cifrada.
Se puede utilizar el cifrado para las conexiones, los datos y los procedimientos almacenados.

Algoritmo de cifrado

Los algoritmos de cifrado definen transformaciones de datos que los usuarios no autorizados no pueden revertir con facilidad. SQL Server permite a los administradores y los desarrolladores de software elegir entre varios algoritmos, incluidos DES, Triple DES, TRIPLE_DES_3KEY, RC2, RC4, RC4 de 128 bits, DESX, AES de 128 bits, AES de 192 bits y AES de 256 bits.

Principios Generales

- El cifrado seguro suele consumir más recursos de la CPU que un cifrado menos seguro.
- Las claves largas suelen producir un cifrado más seguro que las claves cortas.
- El cifrado asimétrico es menos seguro que el simétrico con la misma longitud de clave, pero es relativamente lento.
- Los cifrados en bloque con claves largas son más seguros que los cifrados de flujo.
- Las contraseñas largas y complejas son más seguras que las contraseñas cortas.
- Si cifra una gran cantidad de datos, debe cifrar los datos con una clave simétrica y cifrar la clave simétrica con una clave asimétrica.
- Los datos cifrados no se pueden comprimir, pero los datos comprimidos se pueden cifrar. Si usa compresión, debe comprimir los datos antes de cifrarlos.

Tipos de Cifrado

- ❖ Cifrar mediante una clave simétrica:

La encriptación a partir de llaves simétricas tiene como principio que para encriptar y des encriptar la información se necesita la misma llave. Es decir que si nuestra llave llega a ser pública y accesible para todos los usuarios, estos podrían ver la información sensible que tenemos encriptada en la base de datos.

Cuando se crea una clave simétrica, se debe cifrar mediante uno de los siguientes métodos: certificado, contraseña, clave simétrica, clave asimétrica o PROVIDER.

La clave puede tener más de un cifrado de cada tipo. En otras palabras, una misma clave simétrica puede cifrarse con varios certificados, contraseñas, claves simétricas y claves asimétricas a la vez.

Las claves simétricas creadas con ALGORITHM = TRIPLE_DES_3KEY utilizan TRIPLE DES con una clave de 192 bits.

Las claves simétricas creadas con ALGORITHM = TRIPLE_DES utilizan TRIPLE DES con una clave de 128 bits.

Al igual que los certificados, las llaves simétricas deberían tener una copia de seguridad, pero lamentablemente esto no es posible debido a que dentro de las sentencias T-SQL para manejar las llaves simétricas no hay ninguna que sirva para sacar copias de seguridad.

Parámetros para poder recuperar una llave simétrica:

- IDENTITY_VALUE: El cual se usa para generar un valor GUID para la llave.
- KEY_SOURCE: El cual se usa como material para poder generar la llave en sí.

Si estos dos atributos fueron especificados en la creación de la llave simétrica, entonces esta podrá ser recreada siempre y cuando se tengan copias de seguridad de los objetos usados para generarla, en este caso del certificado usado.

- ❖ Cifrar mediante una clave asimétrica:

Una clave asimétrica es una entidad protegible en el nivel de base de datos. De manera predeterminada, esta entidad contiene una clave pública y otra privada. Si se ejecuta sin la cláusula FROM, CREATE ASYMMETRIC KEY genera un nuevo par de claves. Si se ejecuta con la cláusula FROM, CREATE ASYMMETRIC KEY importa un par de claves desde un archivo o importa una clave pública desde un ensamblado.

De manera predeterminada, la clave privada está protegida por la clave maestra de la base de datos. Si no se ha creado una clave maestra de base de datos, se necesita una contraseña para proteger la clave privada. Si hay una clave maestra de base de datos, la contraseña es opcional.

La clave privada puede ser de 512, 1024 o 2048 bits.

- Cifrar mediante un certificado:

Un certificado es un objeto de seguridad firmado digitalmente que contiene una clave pública (y opcionalmente una privada) para SQL Server. Pueden utilizarse certificados generados externamente o generados por SQL Server.

Los certificados son útiles debido a que ofrecen la opción de exportar e importar claves a archivos de certificado X.509. La sintaxis para crear certificados ofrece opciones de creación para los certificados, como establecer una fecha de expiración.

Las claves privadas generadas por SQL Server tienen una longitud de 1024 bits. Las claves privadas importadas de un origen externo presentan una longitud mínima de 384 bits y una máxima de 4.096 bits. La longitud de una clave privada importada debe ser un entero múltiplo de 64 bits.

La clave privada debe corresponder con la clave pública especificada por certificate_name.
No se requiere la opción ENCRYPTION BY PASSWORD si se cifra la clave privada con la clave maestra de base de datos.

No es necesario especificar una contraseña de descifrado si se cifra la clave privada con la clave maestra de base de datos.

Se utiliza para proteger las conexiones, en la creación de reflejo de la base de datos, para firmar paquetes y otros objetos, o para cifrar datos o conexiones.

- Cifrar Mediante ENCRYPTBYPASSPHRASE

Cifra datos mediante una frase de contraseña usando el algoritmo TRIPLE DES con una longitud de clave de 128 bits.

La ventaja de usar una frase de contraseña es que es más fácil recordar una frase con significado que una cadena larga de caracteres.

Esta función no comprueba la complejidad de la contraseña.

El tipo de valor devuelto es un ***varbinary*** con un tamaño máximo de 8.000 bytes.

Capítulo 4 Diagramas

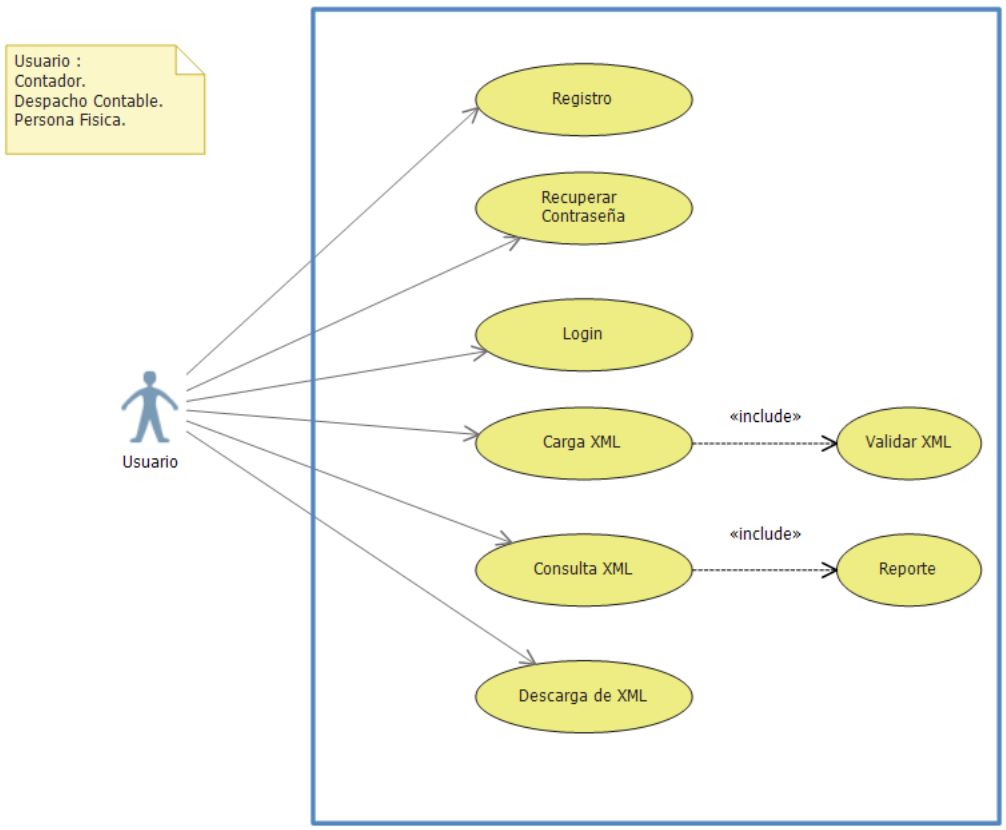
4.1. Casos de Uso

Un Caso de uso es una descripción de las acciones de un sistema desde el punto de vista del usuario. Para los desarrolladores del sistema, esta es una herramienta valiosa, ya que es una técnica de aciertos y errores para obtener los requerimientos del sistema desde el de vista del usuario. Esto es importante si la finalidad es crear un sistema que pueda ser utilizado por la gente en general (no sólo por expertos en computación).

Es una estructura para describir la forma en que un sistema lucirá para los usuarios potenciales. Es una colección de escenarios iniciados por una entidad llamada actor (una persona, componente de hardware, un lapso u otro sistema).Un caso de uso debería dar por resultado algo de valor ya sea para el actor que lo inició o para otro.
Es posible volver a utilizar casos de uso de uso. Una forma (“inclusión”) es utilizar los pasos de un caso de uso como parte de la secuencia de pasos de otro caso de uso. Otra forma (“extensión”) es crear un nuevo caso de uso mediante la adición de pasos a un caso de uso existente.

4.1.1. Diagrama General

A continuación se muestra el diagrama de casos de uso general del sistema.

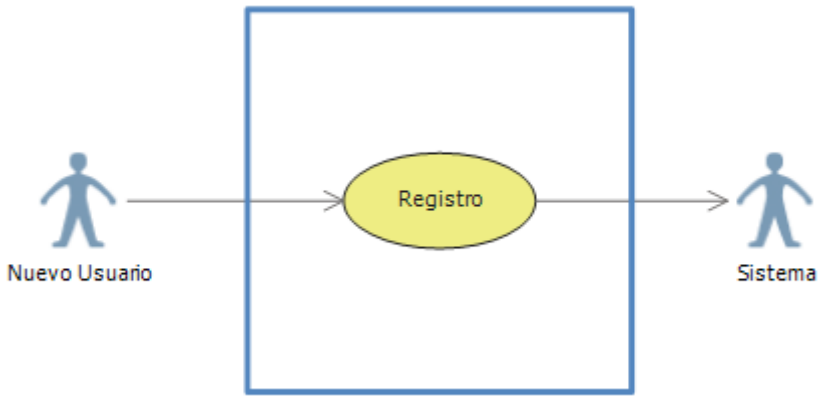


4.1.2. Diagrama Detallado

Existen diferentes formatos para detallar un caso de uso dependiendo del nivel de que se quiera alcanzar. Los grados de formalidad con los que podemos escribir un caso de uso son:

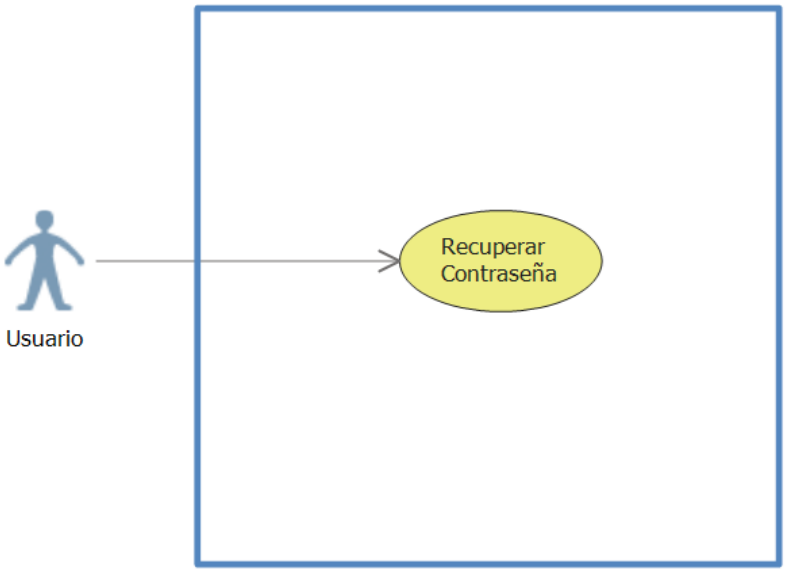
- Breve: Descripción en pocas líneas.
- Informal: Varios párrafos que cubren varios escenarios, entendiendo por escenario a una instancia de caso de uso.
- Completo: El más elaborado, se trata de una descripción detallada con una plantilla.

4.1.2.1. Registro



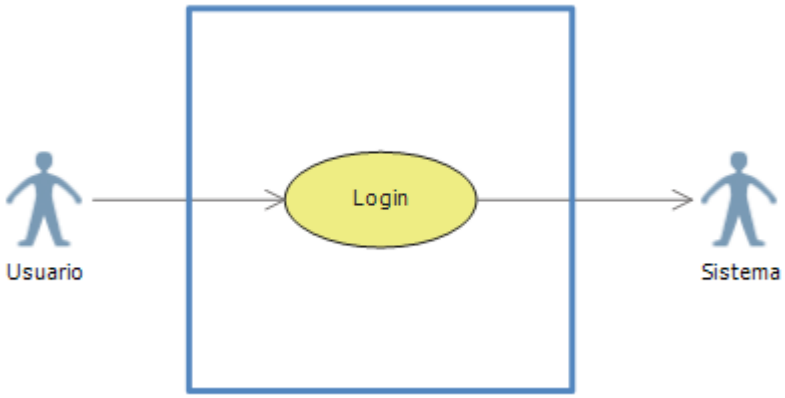
Caso de uso	Registro
Actores	Nuevo Usuario
Resumen	El usuario registrará sus datos para ingresar al sistema.
Pre condiciones	Que el email del usuario no exista en la BD.
Pos condiciones	El usuario habrá quedado registrado en el sistema.

4.1.2.2. Recuperar Contraseña



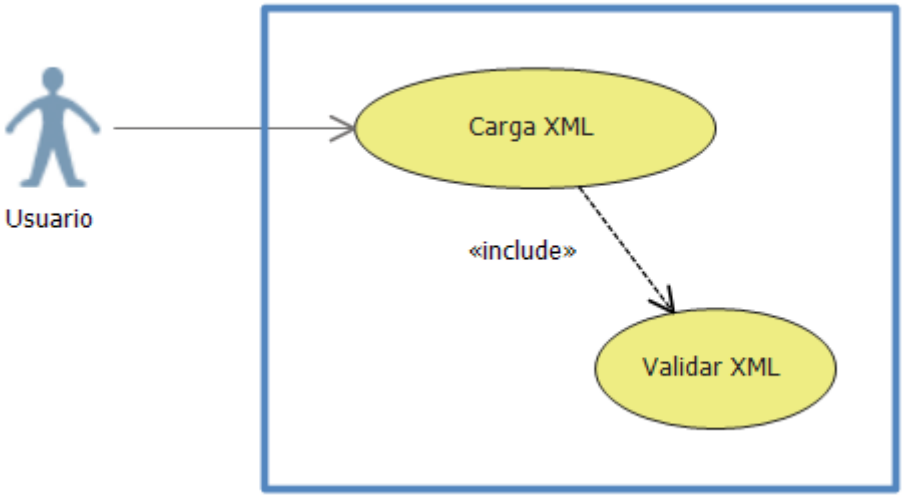
Caso de uso	Recuperar Contraseña
Actores	Usuario
Resumen	El usuario ingresara su correo electrónico con él que se registró en el sistema. El sistema validara que este se encuentre en los registros de la BD y procederá a reenviarle su password.
Pre condiciones	Estar registrado y activo en el sistema.
Pos condiciones	Ingresara nuevamente al sistema

4.1.2.3. Login



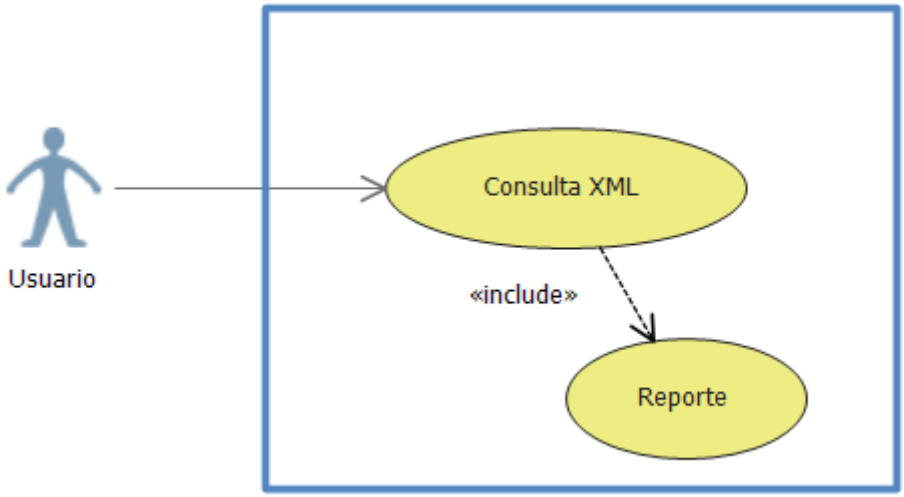
Caso de uso	Login
Actores	Usuario
Resumen	El usuario ingresa sus credenciales, para utilizar el sistema.
Pre condiciones	Haberse registrado antes.
Pos condiciones	Si las credenciales del usuario son válidas podrá ingresar al sistema.

4.1.2.4. Carga XML



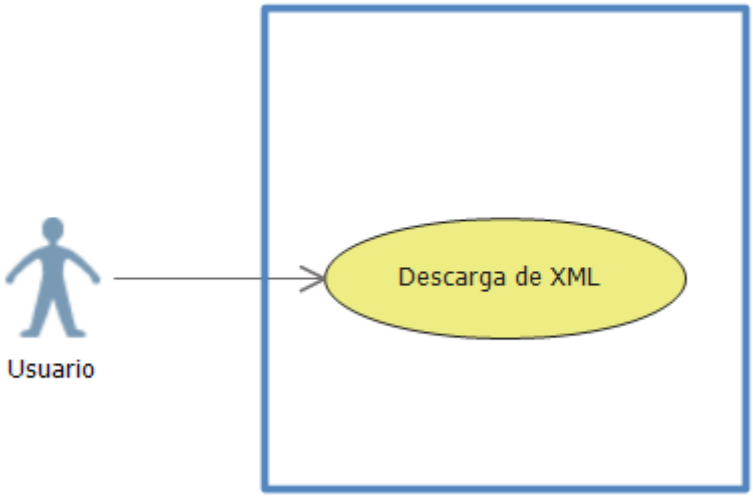
Caso de uso	Carga XML
Actores	Usuario
Resumen	El usuario cargará sus XML en el sistema.
Pre condiciones	• Haberse logueado. • Tener créditos disponibles. • Que los archivos sean válidos.
Pos condiciones	Los archivos se almacenarán en la base de datos para futuras consultas.

4.1.2.5. Consulta XML



Caso de uso	Consulta XML
Actores	Usuario
Resumen	El usuario podrá realizar diferentes tipos de búsquedas para localizar sus archivos y si lo desea generar su reporte.
Pre condiciones	Tener archivos guardados en la BD.
Pos condiciones	Poder generar su reporte de gastos.

4.1.2.6. Descarga XML



Caso de uso	Descarga de XML
Actores	Usuario
Resumen	El usuario podrá recuperar sus archivos xml individualmente las veces que quiera.
Pre condiciones	Tener archivos xml guardados en el sistema.
Pos condiciones	Descargar los archivos individualmente.

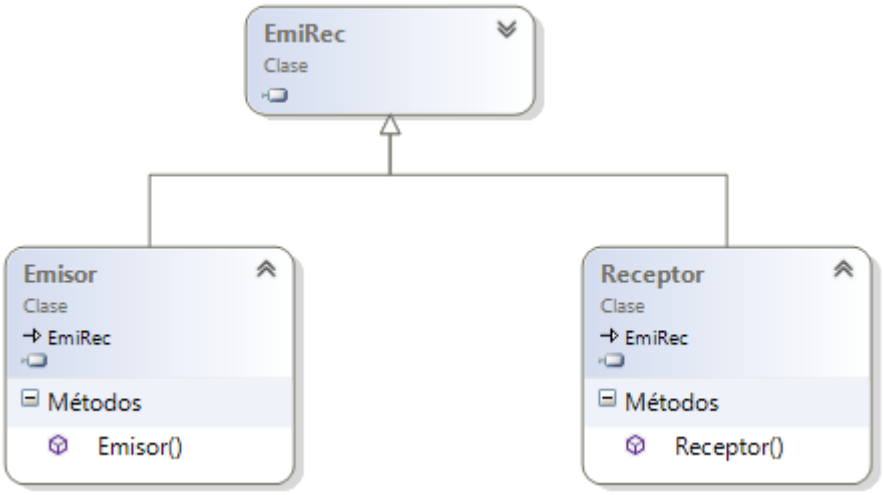
4.2. Diagrama de Clases

Una clase es una categoría o grupo de cosas que tienen atributos y acciones similares.
Un rectángulo es el símbolo que representa a la clase, y se divide en 3 áreas. El área superior contiene el nombre, el área central contiene los atributos, y el área inferior las acciones.

Un diagrama de clases está formado por varios rectángulos de este tipo conectados por líneas que muestran la manera en que las clases se relacionan entre sí.

Los diagramas de clases permiten al analista hablarle a los clientes en su propia terminología, lo cual hace posible que los clientes indiquen importantes detalles de los problemas que requieren ser resueltos.

En la figura siguiente se muestra un ejemplo de diagrama de clases.



4.3. Diagrama de Actividades

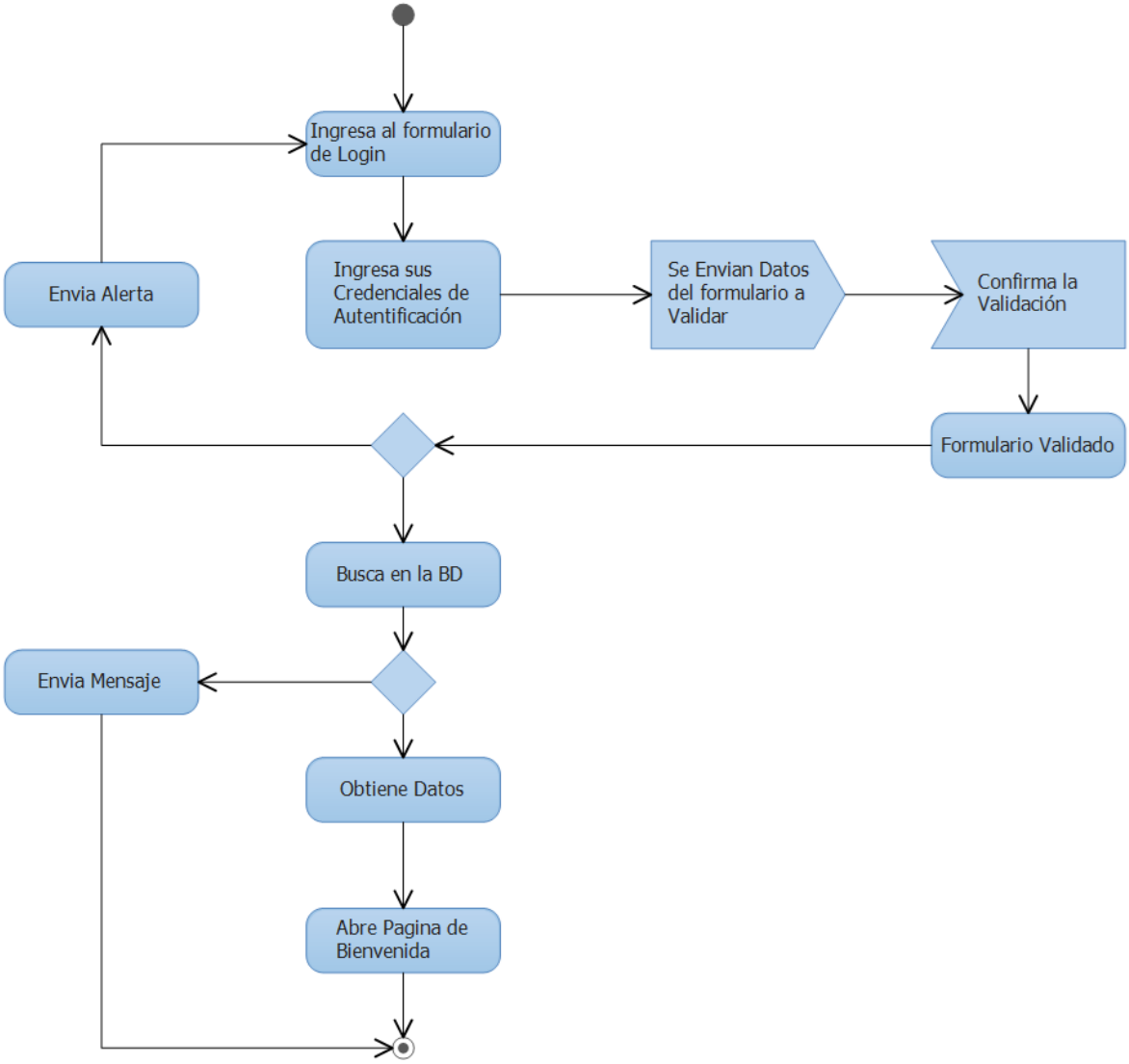
Un diagrama de actividades ha sido diseñado para mostrar una visión simplificada de lo que ocurre durante una operación o proceso. A cada actividad se le representa con un rectángulo con las esquinas redondeadas. El procesamiento de una actividad se lleva a cabo y, al realizarse, se continúa con la siguiente actividad. Una flecha representa la transición de una a otra actividad. Al igual que el diagrama de estados, el de actividad cuenta con un punto inicial (representado por un círculo relleno) y uno final (representado por una diana).

El diagrama de Actividades del UML es muy parecido al ***diagrama de flujo.*** Muestra los pasos, puntos de decisión y bifurcaciones.

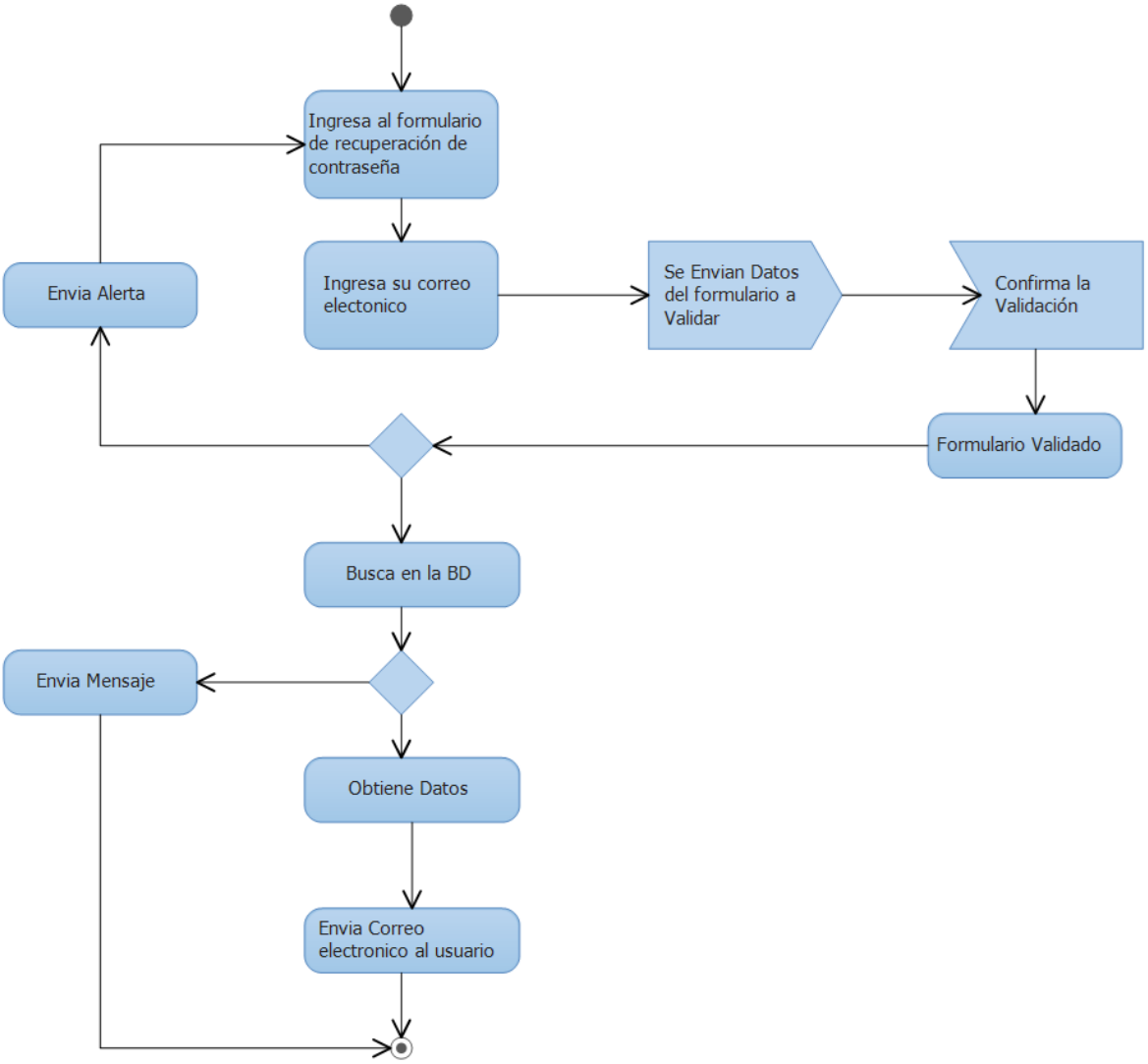
Además de ser una extensión de los diagramas de estados.

4.3.1. Registro

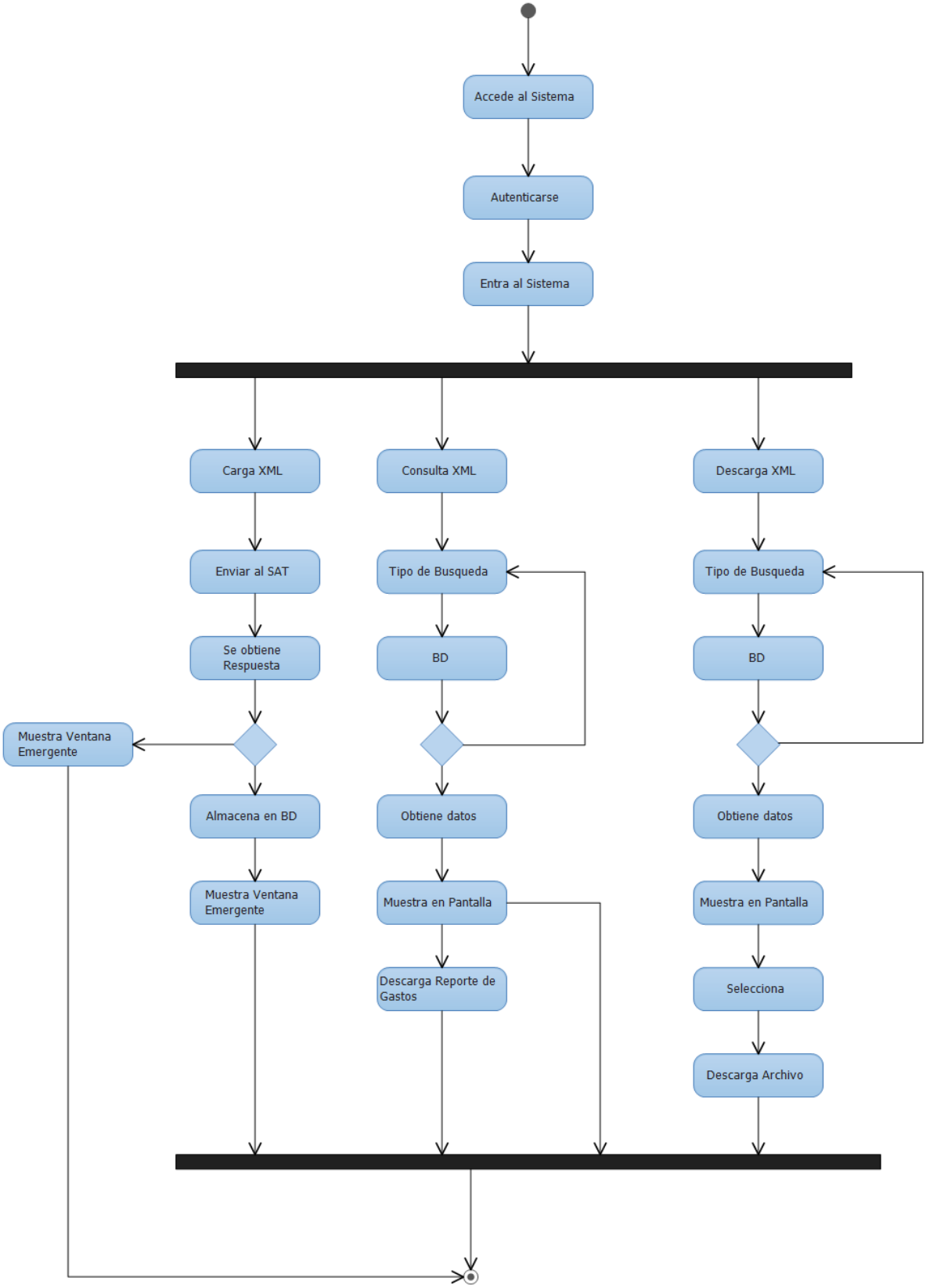
4.3.2. Login


Ingresa al formulario de Login
Ingresa sus Credenciales de Autentificación
Se Envian Datos del formulario a Validar
Confirma la Validación
Formulario Validado
Envia Alerta
Busca en la BD
Envia Mensaje
Obtiene Datos
Abre Pagina de Bienvenida

4.3.3. Recuperar Contraseña



4.3.4. Módulos del Sistema



Capítulo 5 Desarrollo

5.1. Patrones de Diseño

Los ***patrones de diseño*** son el esqueleto de las soluciones a problemas comunes en el desarrollo de software. Brindan una solución ya probada, que están sujetos a contextos similares.

Debemos tener presente los siguientes elementos de un patrón:

- Nombre: Que representa el problema y solución en términos de uno o dos palabras
- Problema: Describirá cuando aplicar el patrón. Explica el problema y su contexto.
- Solución: Describe los problemas que participan en el diseño.

❖ Tipos de Patrones

Patrones Creacionales: Inicialización y configuración de objetos.

Patrones Estructurales: Separan la interfaz de la implementación. Se ocupan de cómo las clases y objetos se agrupan, para formar estructuras más grandes.

Patrones de Comportamiento: Más que describir objetos o clases, describen la comunicación entre ellos.

5.1.1. Singleton

El patrón Singleton garantiza que una clase sólo tenga una instancia y proporciona un punto de acceso global a ésta instancia.

El funcionamiento de este patrón es muy sencillo y podría reducirse a los siguientes conceptos:

1. Ocultar el constructor de la clase Singleton, para que los clientes no puedan crear instancias.
2. Declarar en la clase Singleton una variable miembro privada que contenga la referencia a la instancia única que queremos gestionar.
3. Proveer en la clase Singleton una función o propiedad que brinde acceso a la única instancia gestionada por el Singleton. Los clientes acceden a la instancia a través de esta función o propiedad.

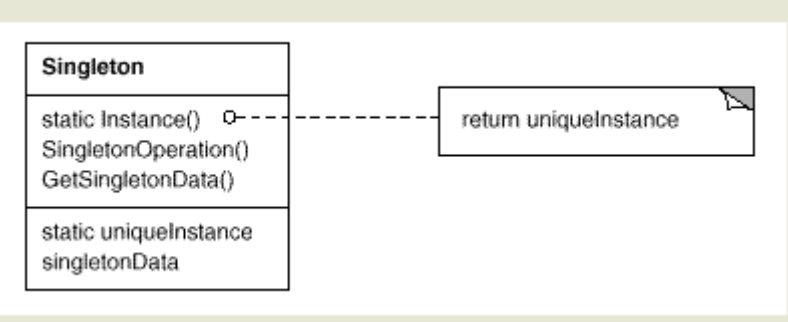
- **Conexión a Base de Datos con Singleton**

El uso de singleton es el de obtener siempre la misma y única instancia de una clase. Concretamente estoy hablando de la conexión con la base de datos.

Una vez establecida la conexión, debemos mantenerla instanciada para poderla utilizar durante toda la ejecución de la aplicación. Una única conexión es suficiente para ejecutar todos los accesos a la base de datos, por lo tanto tenemos que evitar instanciarla más de una vez ya que el hecho de establecerla es costoso en términos de procesamiento, tráfico de red y tiempo.

Por lo que este patrón nos ayudara a establecer la conexión de manera más simple y ágil.

Diagrama OMT del singleton tomada del libro de GoF.



Ejemplo Básico de un Singleton en C Sharp

```
public class Singleton
{
  private static Singleton instance = null;

  protected Singleton() {}

  public static Singleton Instance
  {
        get {
        if (instance == null)
        instance = new Singleton();
        return instance;
        }
  }
}
```

5.1.2. Inversión de Control (IoC)

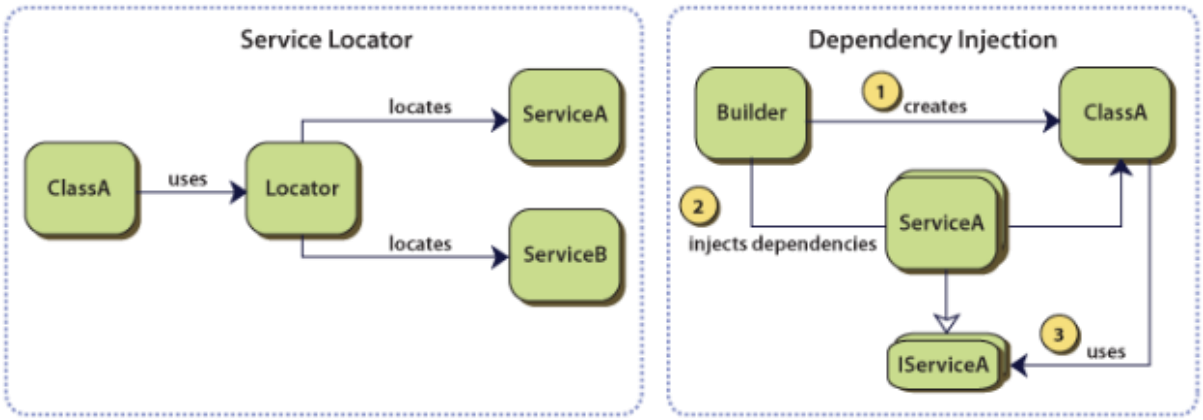
Un IoC es una herramienta muy útil para implementar un sistema enfocado a un diseño orientado a componentes ya que nos proporciona la flexibilidad necesaria para implementar un diseño libre de dependencias entre componentes.

Para tener una verdadera independencia de componentes debemos crear un repositorio llamado "Contenedor de IoC".

Un contenedor IoC es un componente de software que nos ayuda a manejar y soportar un IoC sin problema, gracias a que nos permite almacenar en los componentes que deseemos y luego en base a las directivas que nosotros definamos.

El IoC se puede implementar de varias maneras. El patrón de Inyección de Dependencias y el patrón Service Locator son versiones especializadas de este patrón que delian diferentes implementaciones

La siguiente Figura Ilustra la vista conceptual de ambos patrones



- ❖ Dependencia de Inyección

También llamada DI (Dependency Injection), la inyección de dependencias es un patrón de diseño que nos permite construir software con poco acoplamiento. Básicamente, el patrón funciona con un objeto que se encarga de construir las dependencias que una clase necesita y se las suministra, de ahí el término "inyección". Esto implica que la clase ya no crea directamente los objetos que necesita, sino que los recibe de otra clase.

Existen 2 formas principales

- *Inyección de Constructor*
- *Inyección de Setter*

- **Uso del IoC**

Uno de los principales usos es el de evitar que nuestros objetos de negocios creen objetos que necesiten utilizar, proveyéndoselos ya listos para usar. Desacoplándolo con interfaces e inyectándolos por constructor, lo cual es bueno porque no nos va a permitir pasarle ni devolver cualquier cosa que no se encuentre definido en la interfaz.

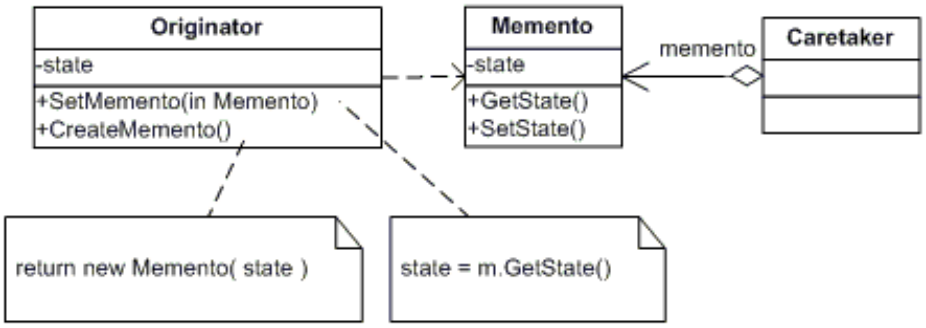
5.1.3. Memento

El objetivo principal de memento es guardar el estado de un objeto sin romper el encapsulamiento del mismo.

- **Uso de memento en formularios**

Memento es una buena alternativa dentro de los formularios, para facilitar el hacer y deshacer determinadas operaciones.

La siguiente figura muestra el diagrama de clases del patrón



Originator:

- Crea el memento y recupera su estado.
- Esta es la clase del objeto en cuestión, es decir, el objeto a guardar o restaurar.

Memento:

- Clase interna estática de originator que contiene su estado. El originator determina que datos almacenar en el Memento y solo el Originator debe ser capaz de leer el Memento.

Caretaker:

- Es la clase que maneja los Mementos.
- Nunca necesita saber que hay en Memento, solo necesita saber que el objeto que recibe permite restaurar el estado del Originator.

Capítulo 6 Funcionalidad del Sistema

6.1. Registro

El registro es muy sencillo pues únicamente se solicita información mínima necesaria para hacer uso del sistema. La figura siguiente se muestra la pantalla de registro.

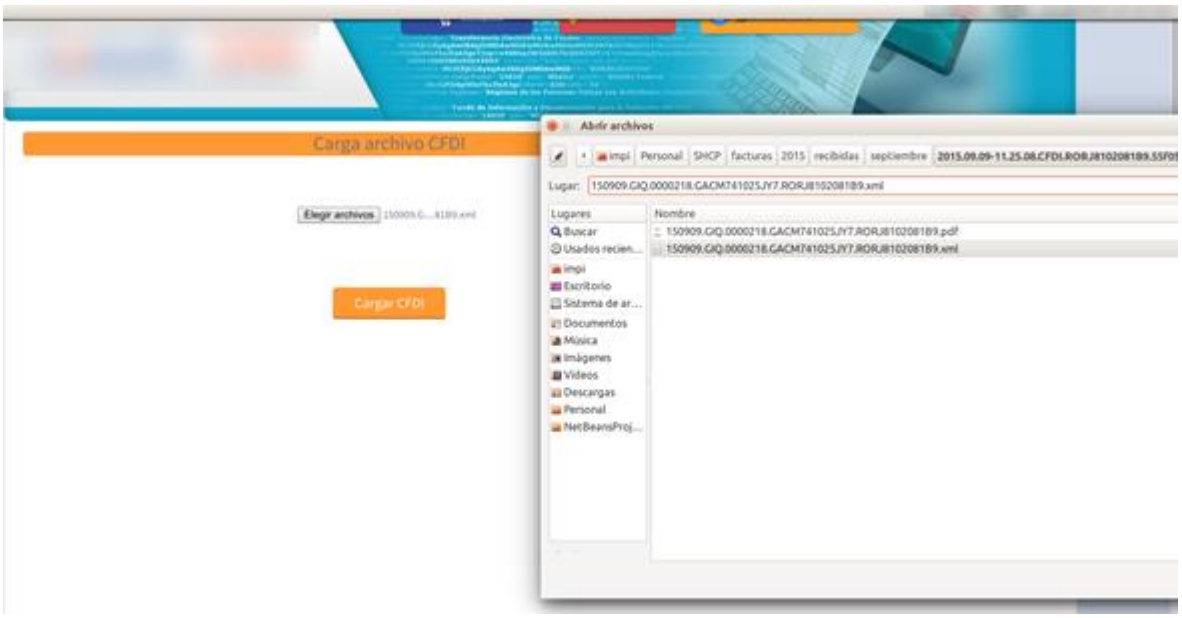
6.2. Login

Después de haberse registrado y haber obtenido el correo de confirmación con su código de verificación podrá ingresar al sistema. El código de seguridad es muy importante ya que nos ayuda a prevenir ataques al sitio web. La figura siguiente muestra la pantalla de logeo.

6.3. Cargar Archivos

Para cargar un comprobante fiscal digital será necesario dar click sobre el botón ( ) ubicado en el menú lateral Izquierdo. Dar click sobre el botón que dice "***seleccionar archivo***" y ubicaremos el cfdi (archivo con extensión .XML) a cargar.

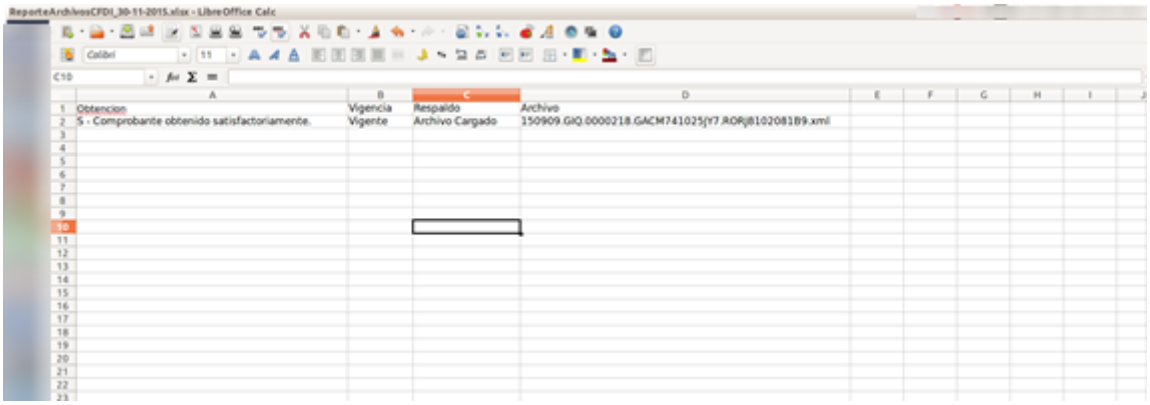
La figura siguiente muestra la carga de un comprobante fiscal digital.



6.3.1. Comprobación

Dar click sobre el botón "***Cargar CFDI***" y nos saldrá un mensaje que se guardó correctamente el comprobante fiscal (valida el CFDI por medio de un Web Service del SAT e indica si fue almacenado correctamente) y nos muestra la opción de descargar un reporte para ver el resultado de la validación del comprobante, y de si fue almacenado correctamente.

En la figura siguiente se muestra el resultado de la carga de un CFDI

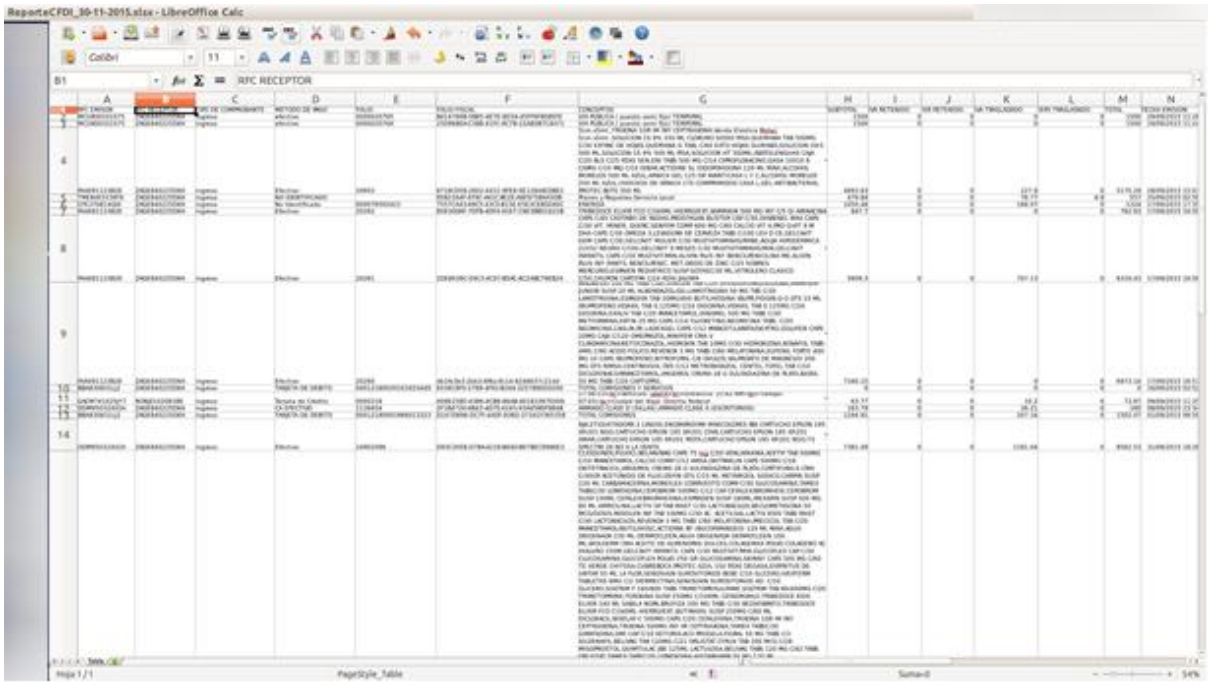


6.4. Búsqueda

Para consultar uno o varios cfdi's será necesario dar click sobre el botón () y llenar cualquiera de los parámetros presentados, se puede hacer búsquedas por Nombre, RFC del Emisor/Receptor y/o por fecha de emisión del comprobante fiscal. Al mostrar la tabla de resultados se mostrará al final de esta la sumatoria total de impuestos. Que se muestra a continuación

6.4.1. Reporte de Gastos

De igual manera se puede descargar el reporte de gastos que muestra el desglose de los conceptos más importantes del comprobante fiscal digital. Que se muestra a continuación.



6.5. Recuperación del XML

Para descargar uno o varios cfdi's será necesario dar click sobre el botón () y llenar cualquiera de los parámetros presentados, se puede hacer búsquedas por Nombre, RFC del Emisor/Receptor y/o por fecha de emisión del comprobante fiscal. Al mostrar la tabla de resultados dar click sobre el icono () de la columna archivos XML con lo que se generará la descarga.

Capítulo 7 Optimización de Consultas

7.1. Índices

Un índice es una estructura de disco asociada con una tabla o una vista que acelera la recuperación de filas de la tabla o de la vista. Un índice contiene claves generadas a partir de una o varias columnas de la tabla o la vista. Dichas claves están almacenadas en una estructura (árbol b) que permite que SQL Server busque de forma rápida y eficiente la fila o filas asociadas a los valores de cada clave.

La selección de los índices apropiados para una base de datos y su carga de trabajo es una compleja operación que busca el equilibrio entre la velocidad de la consulta y el costo de actualización. Los índices estrechos, o con pocas columnas en la clave de índice, necesitan menos espacio en el disco y son menos susceptibles de provocar sobrecargas debido a su mantenimiento. Por otra parte, la ventaja de los índices anchos es que cubren más consultas.

Los índices mal diseñados y la falta de índices constituyen las principales fuentes de atascos en aplicaciones de base de datos. El diseño eficaz de los índices tiene gran importancia para conseguir un buen rendimiento de una base de datos y una aplicación.

Estrategia recomendada para el diseño de índices.

- Comprender las características de la propia base de datos.
- Comprender las características de las consultas utilizadas con frecuencia.
- Comprender las características de las columnas utilizadas en las consultas.
- Determinar qué opciones de índice podrían mejorar el rendimiento al crear o mantener el índice.
- Determinar la ubicación de almacenamiento óptima para el índice.

Características de los Índices

- Índices agrupados y no agrupados
- Índices exclusivos y no exclusivos
- Índices de una sola columna y de varias columnas
- Orden ascendente o descendente en las columnas del índice
- Índices de tabla completa y filtrados en índices no clúster

7.1.1. Tipos de Índices

7.1.1.1. No cluster

Un índice no clúster contiene los valores de clave del índice y localizadores de fila que apuntan a la ubicación de almacenamiento de los datos de tabla. Se pueden crear varios índices no clúster en una tabla o una vista indizada. Por lo general, se deben diseñar índices no clúster para mejorar el rendimiento de consultas utilizadas con frecuencia no cubiertas por el índice clúster.

Arquitectura de los Índices No Clúster

Los índices no clúster tienen la misma estructura de árbol b que los índices clúster, excepto por las siguientes diferencias importantes:

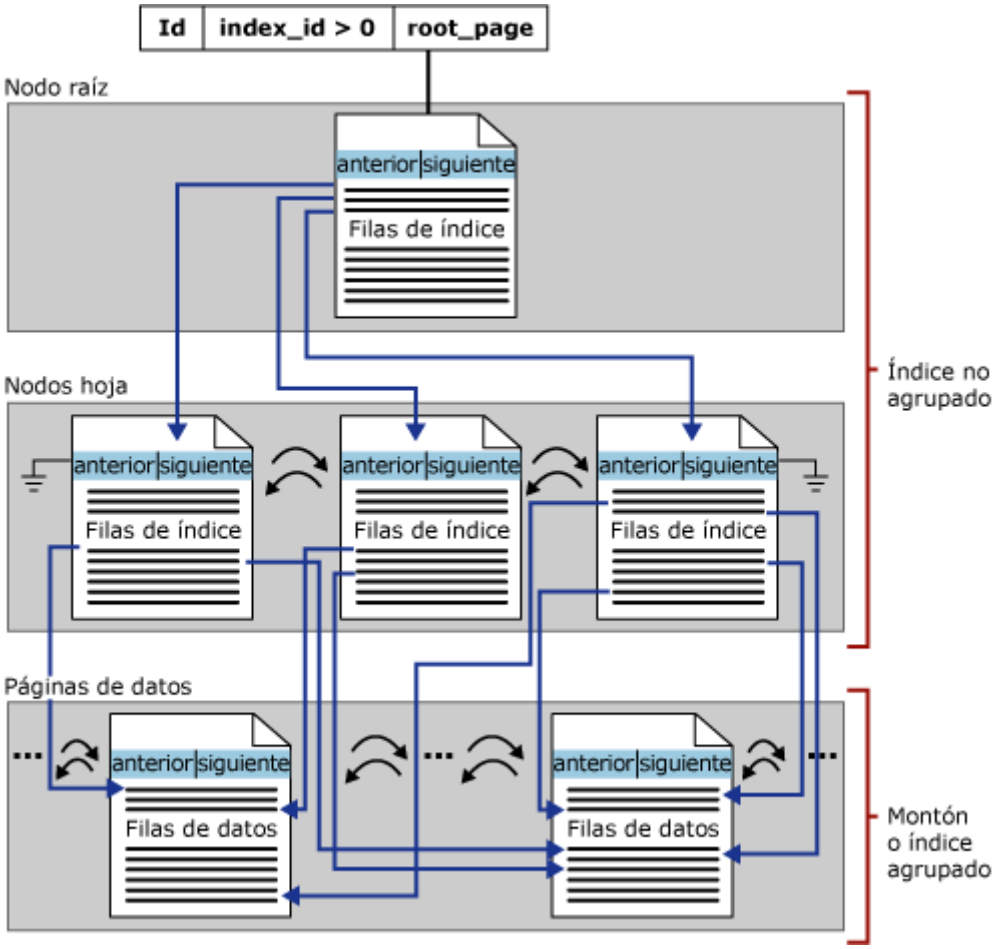
- Las filas de datos de la tabla subyacente no están ordenadas ni almacenadas basándose en sus claves no agrupadas.
- La capa de hoja de un índice no clúster está compuesta por páginas de índices, en lugar de páginas de datos.

Los localizadores de filas de las filas de índices no clúster pueden ser un puntero a la fila o una clave de índice clúster para una fila, tal como se describe a continuación:

- Si la tabla es un montón, lo que significa que no tiene ningún índice clúster, el localizador de fila es un puntero a la fila. El puntero se genera a partir del identificador (Id.) de archivo, el número de página y el número de la fila dentro de la página. El puntero completo se conoce como Id. de fila (RID).
- Si la tabla tiene un índice clúster o si el índice está en una vista indizada, el localizador de fila es la clave del índice clúster para la fila.

Los índices no agrupados tienen una fila en ***sys.partitions***, con **index_id** >1 para cada partición que el índice usa. De forma predeterminada, un índice no clúster tiene una sola partición. Cuando un índice no clúster tiene varias particiones, cada una tiene una estructura de árbol b que contiene las filas de índice de esa partición específica. Por ejemplo, si un índice no clúster tiene cuatro particiones, habrá cuatro estructuras de árbol b, una en cada partición.

En la siguiente ilustración se muestra la estructura de un índice no clúster en una sola partición



Puede incluir columnas sin clave en un índice no clúster para evitar que supere las limitaciones actuales de tamaño del índice de un máximo de 16 columnas de clave y un tamaño máximo de las claves de índice de 900 bytes. El Motor de base de datos no tome en cuenta las columnas sin clave al calcular el número de columnas de clave de índice o el tamaño de clave de índice.

7.1.1.2. Cluster

Los índices agrupados ordenan y almacenan los registros de una tabla o vista basándose en los valores de su campo clave (un campo de ID por ejemplo). Son campos (o columnas) incluidos en la definición del índice. Solo puede haber un índice clúster por cada tabla, porque las filas de datos solo pueden estar ordenadas de una forma.

La única manera en que los registros de una tabla o vista estén almacenados de forma ordenada es cuando contiene un índice agrupado. Cuando una tabla tiene un índice agrupado, la tabla se denomina *tabla agrupada*. Si una tabla no tiene un índice clúster, sus filas de datos están almacenadas en una estructura sin ordenar denominada heap.

Los índices cluster se implementan de la siguiente manera.

Restricciones PRIMARY KEY y UNIQUE

- Cuando se crea una restricción PRIMARY KEY, se crea automáticamente un índice clúster único en las columnas si aún no existe un índice clúster en la tabla o no se ha especificado un índice no clúster. La columna de clave principal no puede permitir valores NULL.
- Cuando cree una restricción UNIQUE, se creará un índice no clúster único para exigir una restricción UNIQUE de forma predeterminada. Puede especificarse un índice clúster único si todavía no existe un índice clúster en la tabla.
- Un índice creado como parte de la restricción recibe automáticamente el mismo nombre que la restricción.

Índice independiente de una restricción

- Puede crear un índice clúster en una columna que no sea la de clave principal si se especificó una restricción de clave principal no agrupada.

7.1.1.3. Full Text Index

Se refiere a la funcionalidad de SQL Server que admite consultas de texto con los datos basados en caracteres. Este tipo de consultas pueden incluir palabras y frases, así como múltiples formas de una palabra o frase. Para apoyar las consultas de texto completo, índices de texto deben ser implementadas en las columnas de referencia en la consulta. Las columnas se pueden configurar con tipos de datos de caracteres (tales como char, varchar, nchar, nvarchar, text, ntext, xml) o con tipos de datos binarios (tales como varbinary e imagen).

Un índice de texto completo se compone de fichas de palabras que se derivan del texto que se está indexando. Por ejemplo, si el texto indexado contiene la frase "tablas pueden incluir índices," el índice de texto contendría cuatro fichas: "tablas", "pueden", "incluir" y "índices" El proceso de SQL Server utiliza los siguientes componentes para la búsqueda de texto completo:

- Tablas de usuario. Estas tablas contienen los datos para ser indexada.
- Recolector de texto completo. El recolector de texto completo trabaja con los hilos de rastreo de texto completo. Es responsable de la planificación y conducción de la población de los índices de texto completo, y también para el seguimiento de los catálogos de texto.

- Archivos de sinónimos. Estos archivos contienen sinónimos de los términos de búsqueda.
- Objetos stoplist. Contienen una lista de palabras comunes que no son útiles para la búsqueda.
- Procesador de consultas de SQL Server. El procesador de consultas compila y ejecuta consultas SQL. Si una consulta SQL incluye una consulta de búsqueda de texto completo, la consulta se envía al motor de texto completo, tanto durante la compilación y durante la ejecución. El resultado de la consulta se compara con el índice de texto completo.
- Motor de texto completo. El motor de texto completo en SQL Server está totalmente integrado con el procesador de consultas. El motor de Full-Text compila y ejecuta consultas de texto. Como parte de la ejecución de la consulta, el motor de texto completo puede recibir la entrada del diccionario de sinónimos y stoplist.
- Escritor Index (indizador). El escritor índice construye la estructura que se utiliza para almacenar los tokens indexados.
- Filtra gerente demonio. El gerente de demonio de filtro es responsable de supervisar el estado del texto completo de host de demonio de filtro del motor.

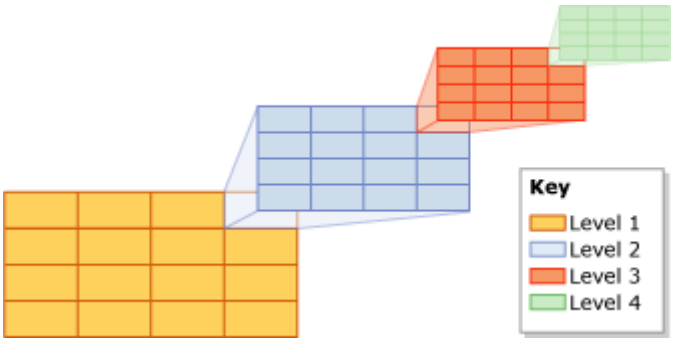
7.1.1.4. Spatial Index

Un *índice espacial* es un tipo de índice extendido que permite indexar una columna espacial. Una columna espacial es una columna de tabla que contiene los datos de un tipo de datos espaciales, como la **geometría o geografía.**

En SQL Server, los índices espaciales se construyen con los árboles B, lo que significa que los índices deben representar los datos espaciales de 2 dimensiones en el orden lineal de los árboles B. Por lo tanto, antes de leer datos en un índice espacial, SQL Server implementa una descomposición jerárquica uniforme del espacio. El proceso de creación de índice *se descompone* el espacio *en* una jerarquía *de cuadrícula* de cuatro niveles. Estos niveles se refieren como *el nivel 1 (nivel* superior), *nivel 2, nivel 3 y* nivel *4.*

Cada nivel sucesivo se descompone aún más en el nivel por encima de ella, por lo que cada celda de nivel superior contiene una cuadrícula completa en el siguiente nivel. En un nivel dado, todas las rejillas tienen el mismo número de células a lo largo de los dos ejes (por ejemplo, 4x4 o 8x8), y las celdas son todos de un tamaño.

La siguiente ilustración muestra la descomposición de la celda superior derecha en cada nivel de la jerarquía de la rejilla en una cuadrícula de 4x4.



Los Spatial Index admite una cláusula de rejillas que permite especificar diferentes densidades de cuadrícula a distintos niveles. La densidad de rejilla para un nivel dado se especifica mediante una de las siguientes palabras clave.

Palabra clave	Configuración de la red	Número de células
BAJO	4X4	16
MEDIO	8X8	64
ALTO	16X16	256

Los índices espaciales admiten los siguientes métodos de geometría set-oriented bajo ciertas condiciones: **STContains (), STDistance (), STEquals (), STIntersects (), STOverlaps (), STTouches (), y STWithin ().**

Para el apoyo de un índice espacial, estos métodos deben ser utilizados dentro del WHERE o JOIN ON de una consulta, y deben ocurrir dentro de un predicado de la siguiente forma general:

geometría1. method_name (geometría2) valid_number comparison_operator

❖ **Tipos de Indices Utilizados**

La utilización de índices varía de acuerdo a la optimización de los querys:

- Agrupados: Generalmente nos referimos a las llaves primarias y aquellos campos que pueden ser solicitados con mayor frecuencia.
- No agrupados: Son generados de acuerdo a los campos que regresa una consulta, así como las condiciones que aplican a esta.
- Únicos: Son utilizados para no repetir información de los campos, en este el correo electrónico.

7.2. Plan de Ejecución

Muestran gráficamente los métodos de recuperación de datos elegidos por el optimizador de consultas de SQL Server.

El plan de ejecución gráfico utiliza iconos para representar la ejecución de instrucciones y consultas específicas en SQL Server en lugar de la representación tabular que crean las opciones de la instrucción SET (SET SHOWPLAN_ALL o SET SHOWPLAN_TEXT) de Transact-SQL, o el formato XML para la representación que crea SET SHOWPLAN_XML.

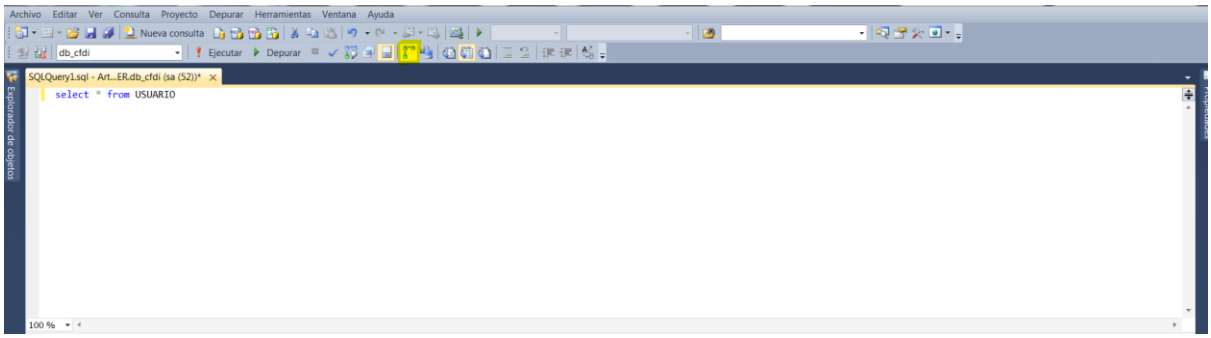
La visualización gráfica es muy útil para comprender las características de rendimiento de una consulta. SQL Server Management Studio muestra las estadísticas que faltan, por lo que exige al optimizador de consultas que estime la selección de los predicados y, después, permite crear fácilmente las estadísticas que faltaban.

7.2.1. Métodos para obtener un plan de ejecución

7.2.1.1. Modo Gráfico

Abrir el SQL Server Management Studio y antes de ejecutar una consulta o procedimiento almacenado (Sin cifrar) seleccionar el icono de “Incluir plan de ejecución actual”, que está marcado en amarillo.

Como se muestra en la siguiente figura.



Obteniendo el siguiente resultado.

A continuación se proporcionan las directrices para interpretar la salida del plan de ejecución:

- Cada nodo de la estructura en árbol se representa como un icono que especifica el operador lógico y físico utilizado para ejecutar esa parte de la consulta o instrucción.
- Cada nodo está relacionado con un nodo principal. Los nodos secundarios que tienen el mismo nodo principal se dibujan en la misma columna. Sin embargo, no todos los nodos de la misma columna tienen necesariamente el mismo nodo principal. Cada nodo se conecta a su nodo principal mediante reglas con puntas de flecha.
- Los operadores se muestran como símbolos relacionados con un nodo principal específico.
- El ancho de la flecha es proporcional al número de filas. Se utiliza el número real de filas cuando está disponible. Si no, se utiliza el número estimado de filas.
- Cuando la consulta contiene varias instrucciones, se dibujan varios planes de ejecución de la consulta.

- Las partes de las estructuras en árbol se determinan por el tipo de instrucción ejecutada.
- Para las consultas en paralelo, en las que intervienen varias CPU, las Propiedades de cada nodo en el plan de ejecución gráfico muestran información acerca de los subprocesos del sistema operativo utilizados.

7.2.1.1.1. Iconos

A continuación se detallaran los iconos básicos que debes conocer en la ejecución de planes.

Table Scan: Recupera todas las filas de la tabla especificada en la columna **Argument** del plan de ejecución de consultas. Si hay un predicado WHERE en la columna **Argument**, sólo se devuelven las filas que cumplan el predicado. Es un operador lógico y físico.

Y se representa de la siguiente manera: 

En la mayoría de los casos su aparición quiere decir que estamos haciendo mal las cosas, y necesitamos urgente crear un índice o reestructurarlo si ya existe alguno. Aunque no siempre es así, el motor siempre intenta predecir los costos de ejecución basados en las estadísticas que va almacenando, si él estima que va ser más rápido leer toda la tabla en vez de leer un índice, usará ese método. Esto suele suceder con tablas poco pobladas y la query en cuestión no conlleva ningún filtro, ya que reduce considerablemente el **overhead.**

Clustered Index Scan: Examina el índice clúster especificado en la columna Argument del plan de ejecución de la consulta. Si hay un predicado WHERE, sólo se devuelven las filas que lo cumplen. Si la columna Argument contiene la cláusula ORDER, el procesador de consultas ha solicitado que la salida de las filas se devuelva en el orden en que las haya ordenado el índice clúster. Si no hay una cláusula ORDER, el motor de almacenamiento recorre el índice de la forma óptima (sin tener que ordenar el resultado).

Su representación es la siguiente: 

Clustered Index Seek: Utiliza la capacidad de búsqueda de los índices para recuperar filas de un índice no agrupado. La columna **Argument** contiene el nombre del índice no agrupado utilizado. También contiene el predicado SEEK. El motor de almacenamiento utiliza el índice para procesar solamente las filas que cumplen el predicado SEEK. También se puede incluir un predicado WHERE:(), predicado que evaluará el motor de almacenamiento respecto a todas las filas que cumplan el predicado SEEK (no se utilizan los índices para esta comprobación).

Si la columna **Argument** contiene la cláusula ORDER, significa que el procesador de consultas ha determinado que se deben devolver las filas en el orden en el que el índice no agrupado las haya clasificado. Si no hay una cláusula ORDER, el motor de almacenamiento busca el índice de la forma óptima (que no garantiza que el resultado se ordene). Permitir que el resultado mantenga su ordenación puede ser menos eficiente que generar un resultado no ordenado.

Se representa de la siguiente manera: 

Ver esto es síntoma de un correcto uso de los índices **clustered** en la base de datos.

BookMark Lookup: Utiliza un marcador (identificador de fila o clave de agrupación en clústeres) para buscar la fila correspondiente en la tabla o índice agrupado. La columna Argument contiene la etiqueta del marcador utilizado para buscar la fila en la tabla o el índice agrupado. La columna Argument también contiene el nombre de la tabla o el índice agrupado donde se busca la fila. Si en la columna Argument aparece la cláusula WITH PREFETCH, el procesador de consultas habrá determinado que resulta óptimo utilizar una captura previa asincrónica (lectura anticipada) al buscar marcadores en la tabla o el índice agrupado.

No se utiliza en SQL Server 2008. En su lugar, **Clustered Index Seek** y **RID Lookup** proporcionan la función de búsqueda de marcadores. El operador **Key Lookup** también proporciona esta funcionalidad.

Se representa de la siguiente manera: 

Una de las maneras para evitar su aparición excesiva es limitar los campos requeridos en el query, sólo solicitar los que están incluidos en el índice o agregar los campos al índice.

RID Lookup: Es una búsqueda de marcadores en un heap que utiliza un identificador de fila suministrado (RID). La columna Argument contiene la etiqueta de marcador utilizada para buscar la fila en la tabla y el nombre de la tabla en la que se busca la fila. RID Lookup siempre está acompañado por una instrucción NESTED LOOP JOIN.

Se representa de la siguiente manera: 

Este operador no es muy frecuente de ver, y normalmente aparece cuando el motor intenta optimizar por su cuenta la query y no tenemos un índice agrupado (**Clustered**) entonces buscará a través del índice único **ROW ID** (de ahí el nombre **RID**).

Sort: Ordena todas las filas entrantes. La columna **Argument** contiene un predicado DISTINCT ORDER BY, si esta operación elimina las réplicas, o un predicado ORDER BY con una lista de las columnas, separadas por comas, que se van a ordenar. Las columnas llevan como prefijo el valor ASC, si el orden de las columnas es ascendente, o el valor DESC, si es descendente.

Se representa de la siguiente manera: 

Stream Aggregate: Agrupa las filas por una o varias columnas y, a continuación, calcula una o varias expresiones agregadas devueltas por la consulta. Los operadores posteriores de la consulta pueden hacer referencia al resultado de este operador. El operador requiere que la información esté ordenada por las columnas dentro de sus grupos.

Se representa de la siguiente manera: 

Aparece principalmente cuando agrupamos los datos, y mezclamos con funciones agregadas como **MIN**, **SUM**, **AVG**, también con la cláusula **HAVING**. También es frecuente ver que este operador lleva acompañado a **SORT**, a quien utiliza para ordenar primeramente los datos antes de agrupar.

7.2.1.2. Por Instrucciones

Para la utilización de este método es necesario primero ejecutar las siguientes instrucciones en el manejador de SQL server establecidas en ***ON*** una vez terminado el análisis se deben desactivar cambiándolas a ***OFF.***

SHOWPLAN_TEXT: Devuelve la información como un conjunto de filas en forma de árbol jerárquico que representa los pasos que sigue el procesador de consultas de SQL Server al ejecutar cada instrucción.

Cuando esta opción está establecida en ON, se devuelve la información acerca del plan de ejecución de todas las instrucciones SQL Server siguientes hasta que se vuelve a establecer en OFF.

SHOWPLAN_ALL: Se establece en tiempo de ejecución, no en tiempo de análisis. Cuando esta opción está establecida en ON, se devuelve información acerca de todas las instrucciones Transact-SQL siguientes hasta que se vuelve a establecer en OFF.

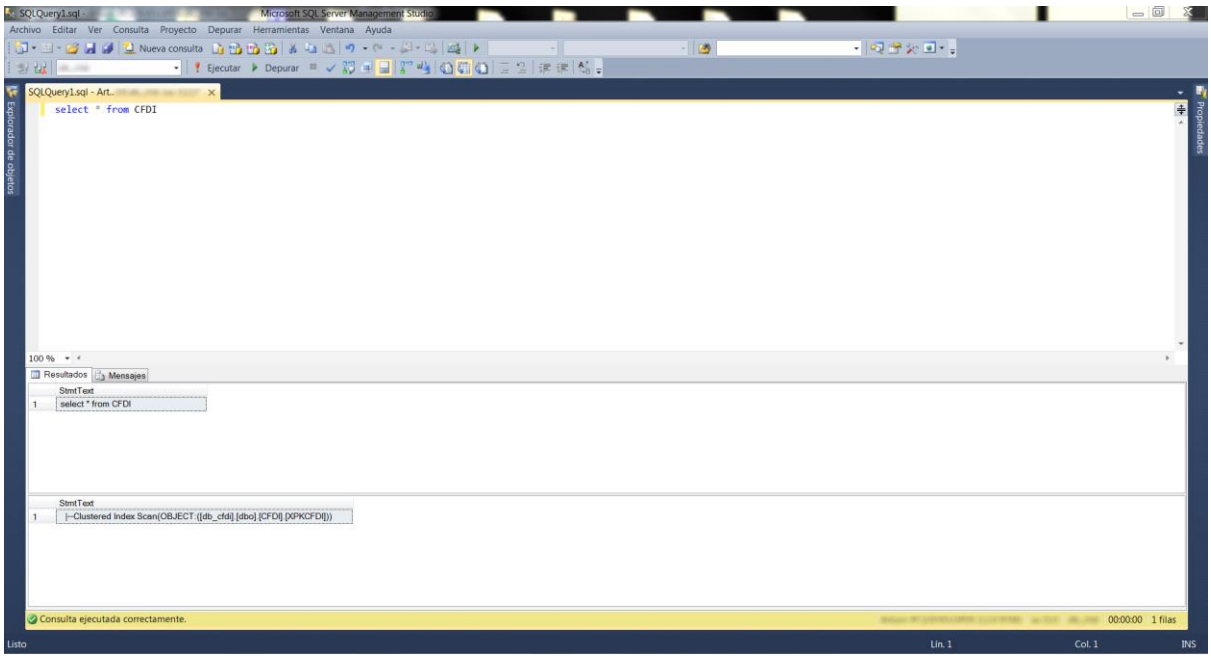
Devuelve la información como un conjunto de filas en forma de árbol jerárquico que representa los pasos que sigue el procesador de consultas de SQL Server al ejecutar cada instrucción. Cada instrucción reflejada en la salida contiene una fila con el texto de la instrucción, seguida de varias filas con los detalles de los pasos de su ejecución.

STATISTICS PROFILE: Cuando está activada, cada consulta ejecutada devuelve su conjunto de resultados regular, seguido de un conjunto de resultados adicional que muestra un perfil de la ejecución de la consulta.Produce una salida de texto.

Trabaja para ad hoc queries, views y stored procedures.

STATISTICS XML: Se establece en tiempo de ejecución, no en tiempo de análisis. Devuelve la información como un conjunto de documentos XML. Cada declaración después de la instrucción se refleja en la salida de un documento único. Cada documento contiene el texto de la declaración, seguida de los detalles de los pasos de ejecución

La siguiente figura muestra un ejemplo de este método



.

7.2.1.3. Analizar Caché de Consultas

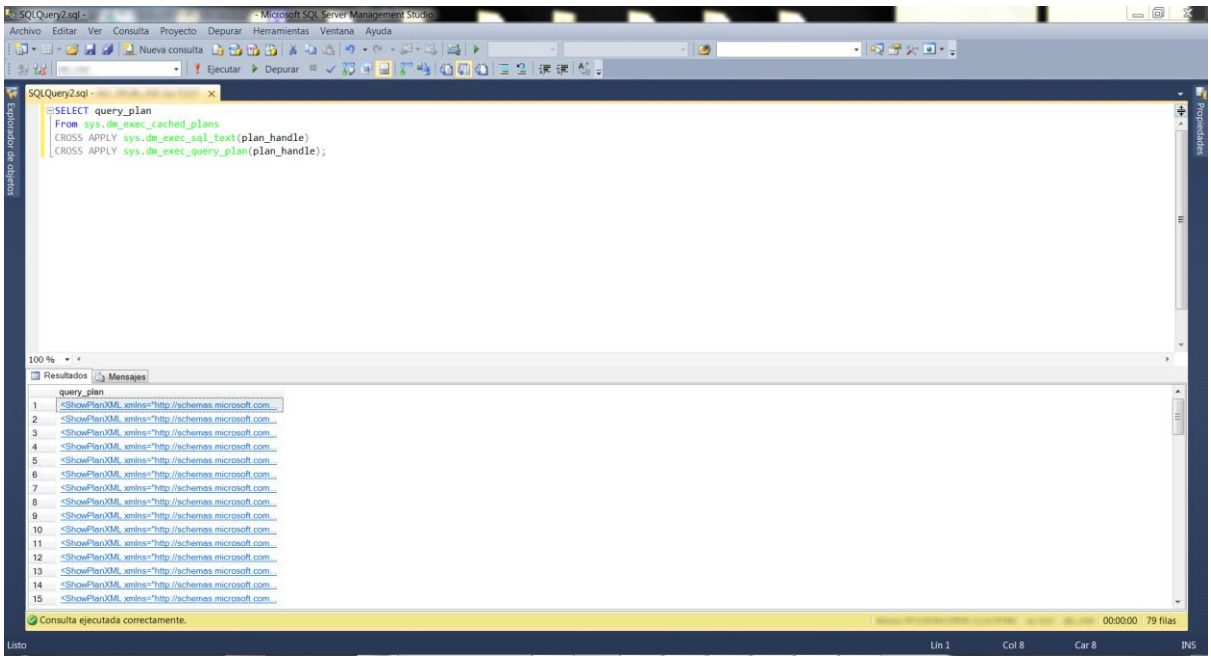
Cuando no se puede obtener el plan de ejecución a través del manejador de SQL server o por instrucciones, tenemos la opción de analizar el plan de caché de consultas. La siguiente es una consulta básica que obtiene los planes de consulta en caché en XML con su texto SQL.

Se tendrá que agregar filtros adicionales en la consulta para filtrar solo los planes que nos interesan

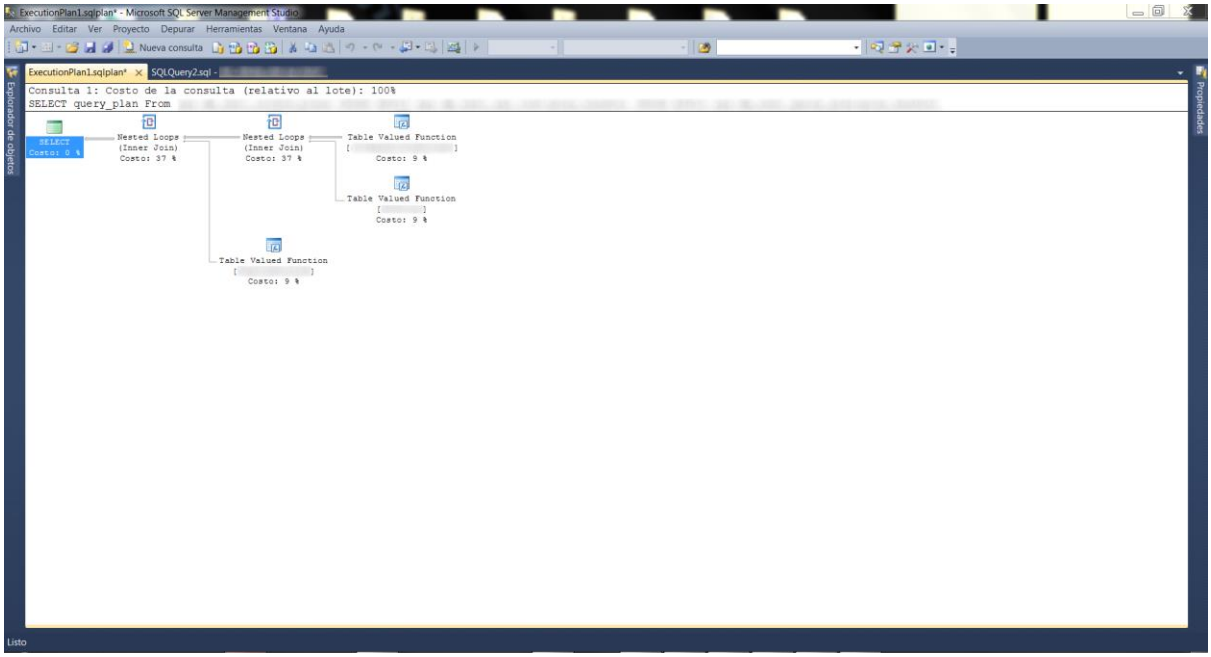
A continuación se muestra el código de la consulta.

```
SELECT usecounts, cacheobjtype, objtype, text, query_plan
From sys.dm_exec_cached_plans
CROSS APPLY sys.dm_exec_sql_text(plan_handle)
CROSS APPLY sys.dm_exec_query_plan(plan_handle);
```

En la figura siguiente se muestra el resultado de la consulta.



Con lo que podemos obtener el plan de ejecución de cualquier query que hayamos ejecutado antes, dando click en la columna ***query_plan*** de cada fila. Como se muestra en la figura siguiente.



7.3. Costo de consultas.

SQL Server usa un optimizador de consultas basado en el costo, es decir, intenta generar un plan de ejecución con el costo estimado más bajo. La estimación se basa en las estadísticas de distribución de datos que están disponibles para el optimizador cuando evalúa cada tabla implicada en la consulta. Si esas estadísticas se pierden o quedan obsoletas, el optimizador de consultas carecerá de la información esencial necesaria para el proceso de optimización de consultas y, por lo tanto, es probable que las estimaciones queden fuera de la marca. En dichos casos, el optimizador seleccionará un plan menos óptimo tanto sobrestimando como subestimando los costos de ejecución de los distintos planes.

7.3.1. Estadísticas

Las estadísticas para la optimización de consulta son objetos que contienen información acerca de la distribución de valores en una o más columnas de una tabla o vista indizada. El optimizador de consultas utiliza estas estadísticas para estimar la *cardinalidad*, o número de filas, en el resultado de la consulta. Estas *estimaciones de cardinalidad* habilitan al optimizador de consultas para crear un plan de consulta de alta calidad. Por ejemplo, el optimizador de consultas podría utilizar las estimaciones de cardinalidad para elegir el operador Index Seek en lugar del operador Index Scan, con un uso más intensivo de los recursos, mejorando con ello el rendimiento de la consulta.

Cada objeto de estadísticas, se crea en una lista de una o más columnas de la tabla e incluye un histograma que muestra la distribución de valores en la primera columna. Los objetos de estadísticas en varias columnas también almacenan la información estadística relativa a la correlación de valores entre las columnas. Estas estadísticas de la correlación, o *densidades*, derivan del número de filas distintas de valores de columna.

DBCC SHOW_STATISTICS muestra las estadísticas de optimización de consulta actuales de una tabla o vista indizada.

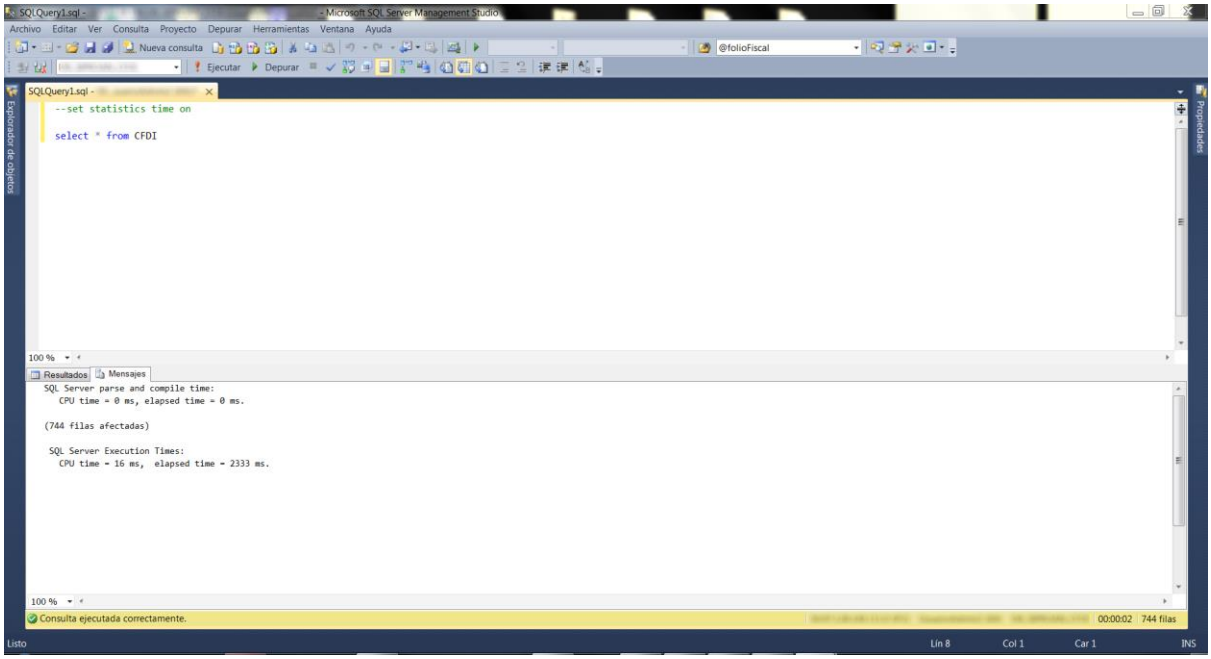
Mientras que ***UPDATE STATISTICS*** actualiza las estadísticas de optimización de consulta para una tabla o vista indizada.

Y para actualizar todas las estadísticas de la BD se deberá ejecutar la siguiente instrucción **EXEC sp_updatestats.**

Para determinar cuándo se actualizaron las estadísticas por última vez, utilice la función ***STATS_DATE.***

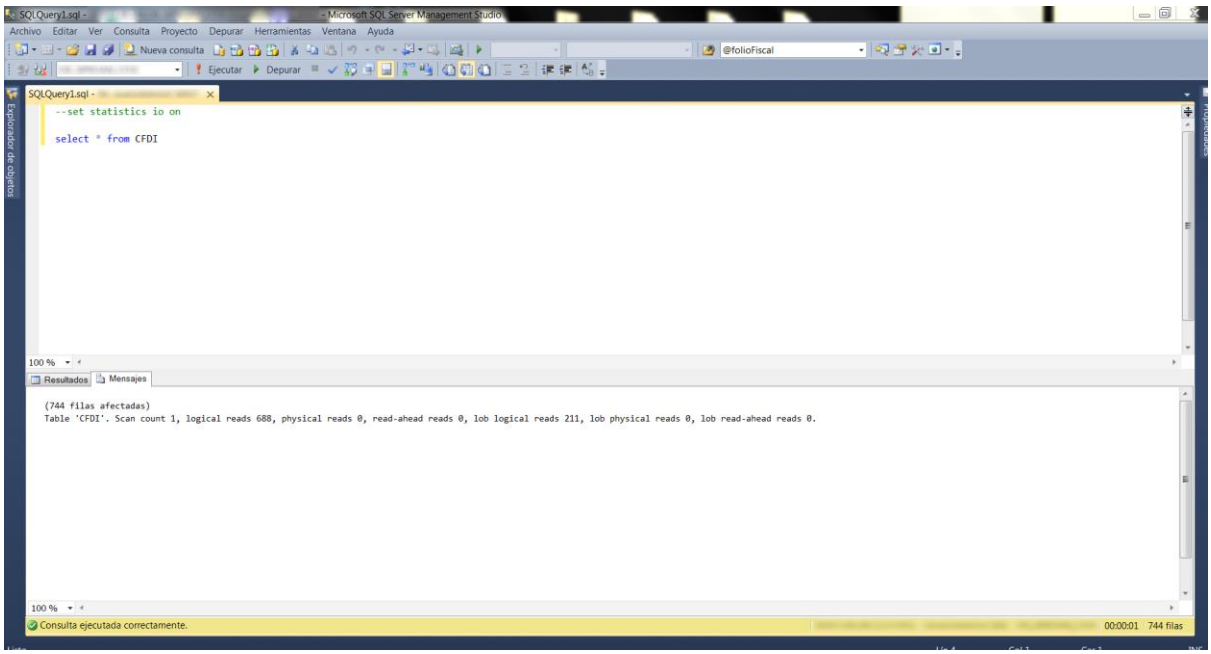
7.3.2. Statistics Time

Muestra el número de milisegundos necesarios para analizar, compilar y ejecutar cada declaración. Como se muestra en la figura siguiente



7.3.3. Statistics Io

Muestra información sobre la cantidad de actividad del disco generada por instrucciones de Transact-SQL. Como se muestra en la figura siguiente.

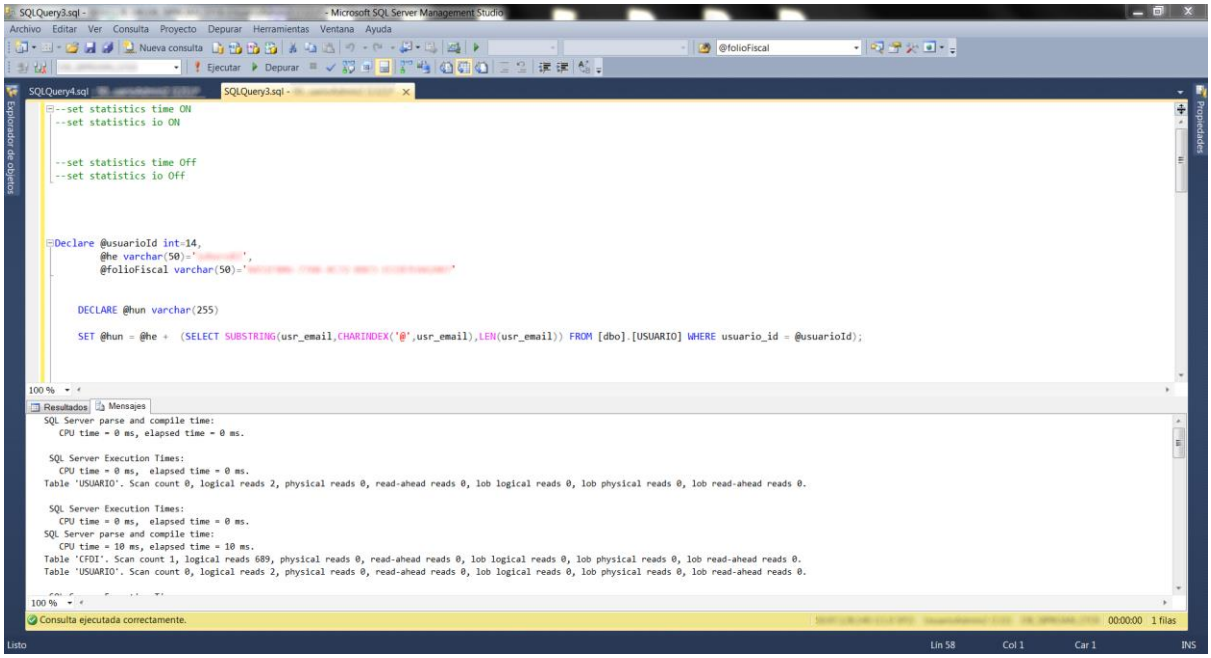


7.4. Ejemplo de Optimización

Pasos para optimizar un Query.

- Paso 1

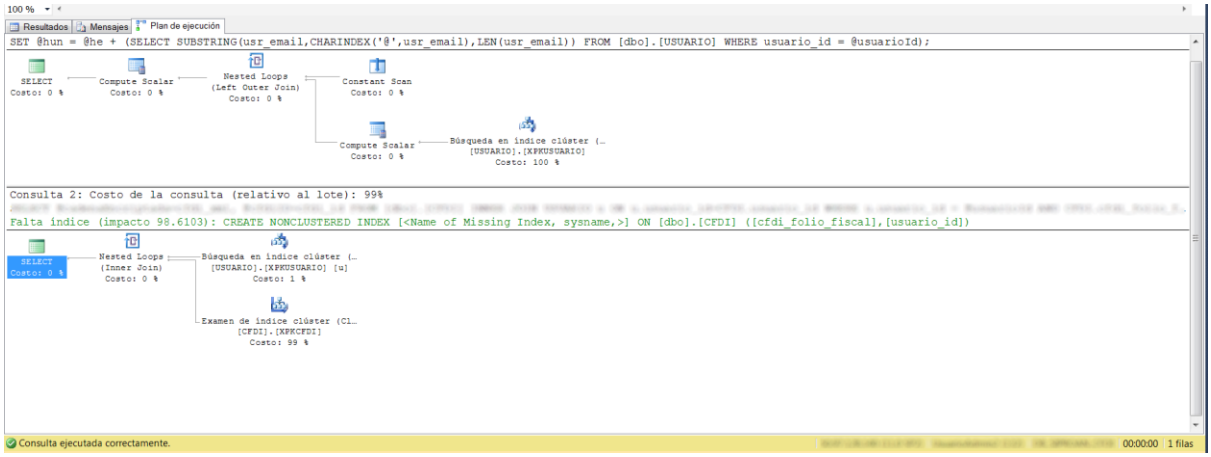
Ejecutar las sentencias "***statistics time***" y "***statistics io***", después ejecutar la consulta o store procedure a analizar. A continuación se muestra el resultado del Query.



Como podemos observar en la imagen la tabla física CFDI contiene la mayor cantidad de lecturas.

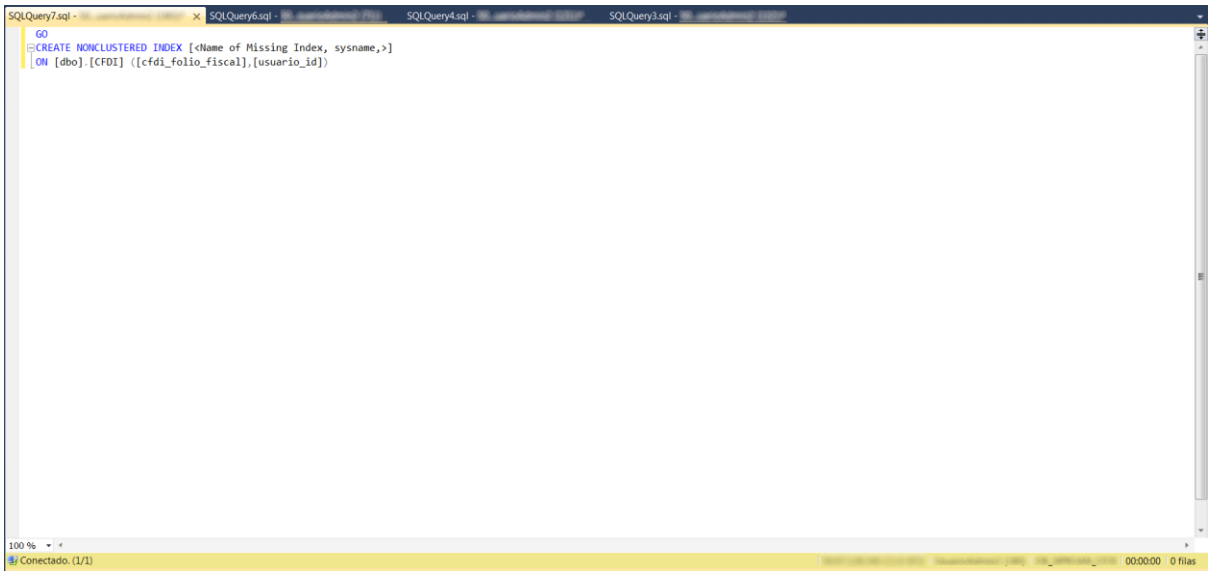
- Paso 2

Desactivar las sentencias "***statistics time***" y "***statistics io***" enseguida activar el plan de ejecución (se utilizará el modo gráfico). En la siguiente figura se ve reflejado que efectivamente la Tabla ***CFDI*** está llevando un porcentaje alto del query y además en este caso el mismo manejador nos está recomendado la creación de un índice non clustered.



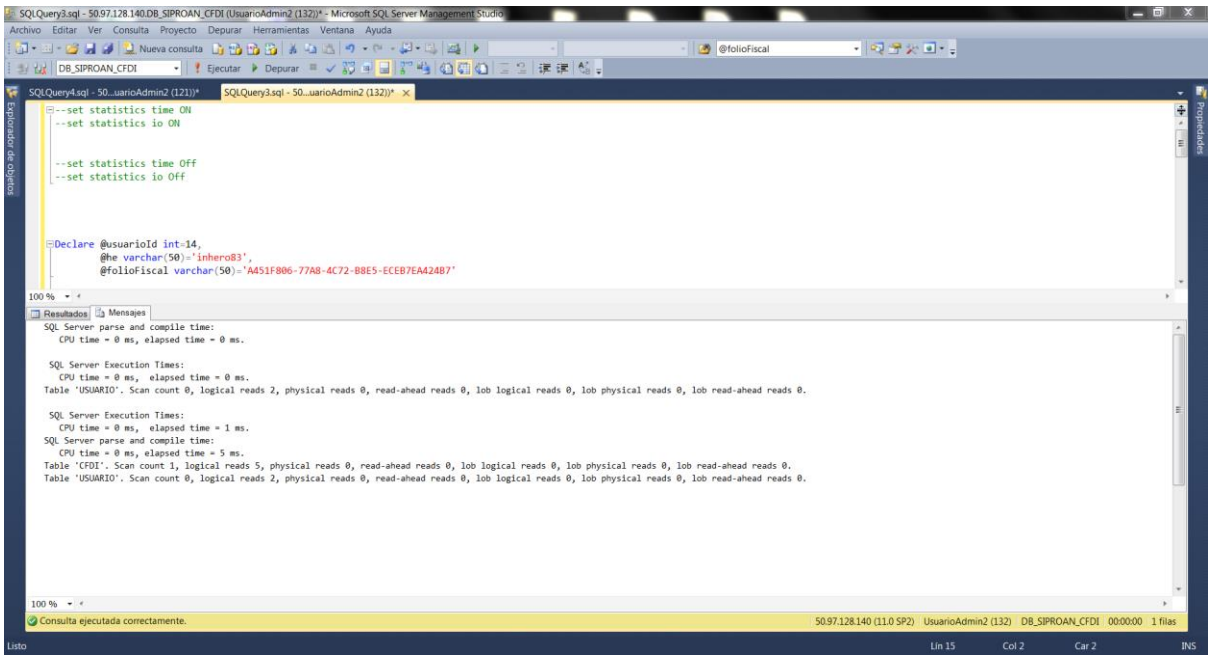
- Paso 3

Se analiza el plan de ejecución, se detectan la falta de índices o restructuración de estos en nuestro caso se tendrá que generar un índice. La siguiente imagen muestra la creación del índice



- Paso 4

Para comprobar que el índice creado incrementó la eficiencia de muestra consulta volvemos a activar las sentencias "***statistics time***" y "***statistics io***", pero antes actualizamos las estadísticas de la tabla utilizando "***update statistics CFDI***". El resultado se observa en la figura siguiente.



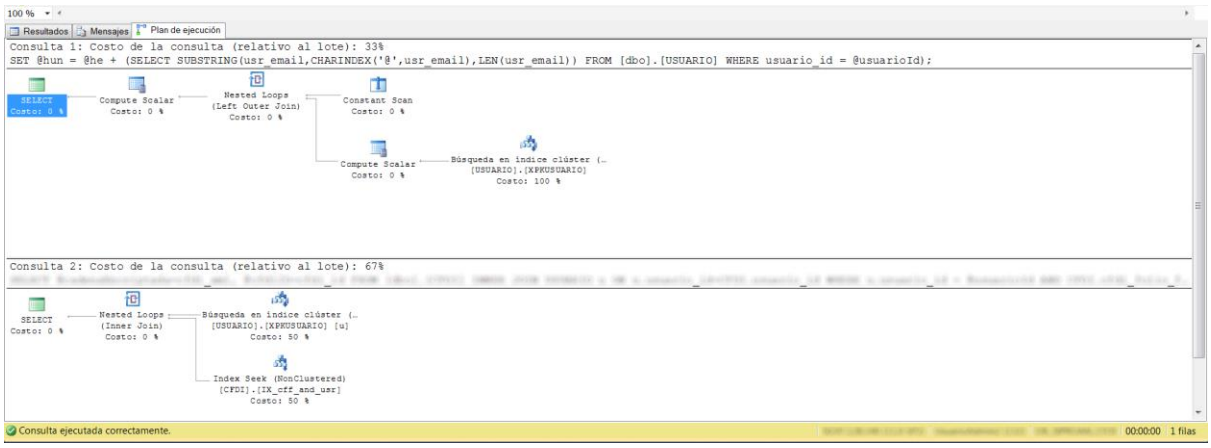
- Paso 5

Como observamos en la pantalla anterior, la tabla ***CFDI*** redujo considerablemente las lecturas, para comprobar el resultado ejecutamos nuevamente el plan de ejecución. Que se muestra en la siguiente figura.

En efecto los porcentajes han cambiado, pero a pesar de tener un ***index seek*** en la tabla de ***CFDI*** nos está generando un ***nested Loops*** y un ***BookMark Lookup*** por lo que será necesario revisar el índice creado.

- Paso 6

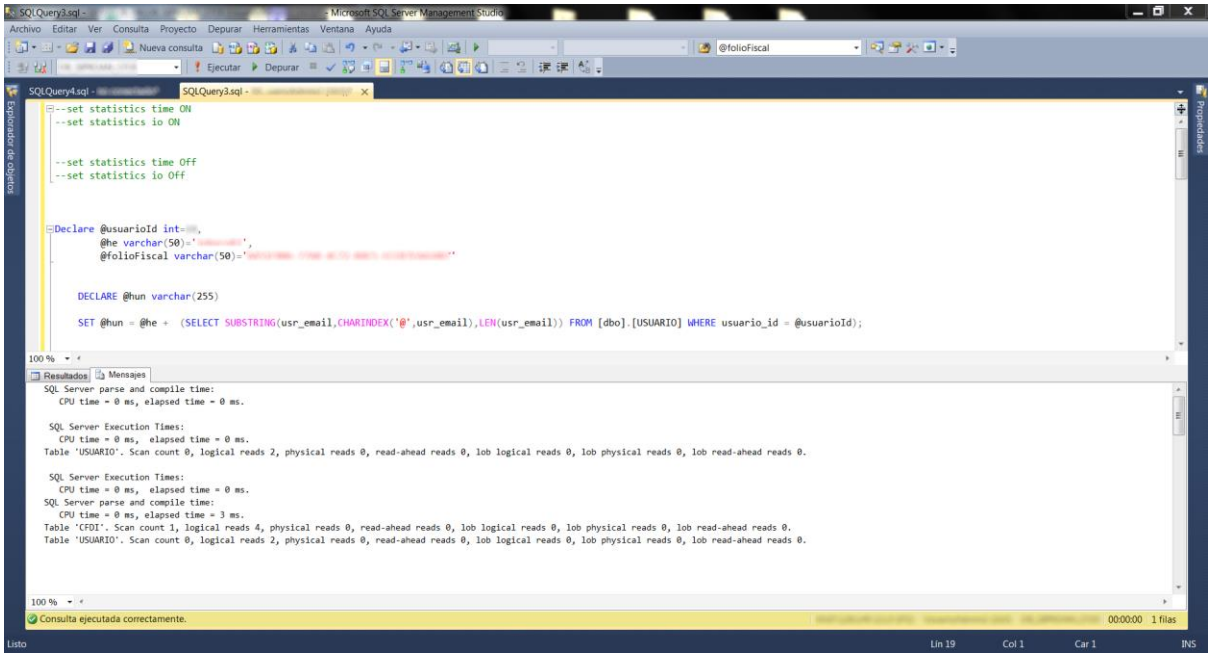
Al analizar el ***BookMark Lookup*** y el ***Index Seek*** nos damos cuenta que nos falta incluir campos en el index por lo que procedemos a modificar este y ejecutamos nuevamente el plan de ejecución.



Al analizar el nuevo plan de ejecución nos daremos cuenta que la consulta 2 ya se encuentra más equilibrada por lo que procederemos a ejecutar las instrucciones "***statistics time***" y "***statistics io***" para ver el resultado final.

- Paso 7

En la figura siguiente podemos observar que las lecturas lógicas de la tabla ***CFDI*** bajaron considerablemente así como el tiempo en el CPU por lo que la optimización resultó satisfactoria.



Finalmente la optimización de consultas no solo se basa en el buen uso de los índices, en ocasiones la reestructuración del query o store procedure es necesaria debido al mal uso de operadores lógicos, aritméticos o funciones propias del lenguaje.

Un mal diseño de la bd también es importante, debido a que se pueden generar tablas con tipos de datos que no sean los adecuados provocando el uso excesivo de operadores o creando tablas que no están normalizadas provocando la creación de queries que ocupan una mayor cantidad de lecturas.

Proyectos a Futuro

- Sistema de Apoyo a la Contabilidad Electrónica

Extender el sistema para timbrado de nómina.

El sistema va ir evolucionando constantemente debido a los cambios que se realizan en la Resolución Miscelánea Fiscal, publicada en el Diario Oficial de la Federación.

El sistema saldrá a internet para que más empresas o contribuyentes la conozcan y puedan realizar su contabilidad más fácilmente.

- Sistema de Aviso de privacidad

Elaborar un modelo de gestión de protección de datos personales y proponer su automatización para la segunda fase del proyecto.

Redactar el informe y notificar los cheros y activos con datos de carácter personal, así como las medidas de seguridad implementadas por el cliente, para la elaboración de los Avisos de Privacidad Básicos.

Realizar las actividades de diagnóstico a nivel procesos de negocio, tecnología y seguridad de la información acorde a normativas internacionales que son prioritarias para el cumplimiento de la Ley Federal de Protección de Datos Personales en Posesión de los Particulares (LFPDPPP).

- Sistema de Directorio Inteligente

Es un sistema que propicia el intercambio de información de los negocios a través de medios electrónicos cuenta con una tarjeta digital que mediante el escaneo de un código QR agrega la información del contacto a los teléfonos inteligentes o tabletas.

Se puede obtener la ubicación física del negocio a través de mapas

El Sistema es ideal para aquellos clientes que no cuentan con presencia en internet o bien si ya cuenta con una página se puede acrecentar la cantidad de visitas y la usabilidad del mismo ya que muestra información básica del negocio como sus historia, horario laboral, fotografías de eventos, productos o promociones ubicación del negocio y tener la posibilidad de entablar una comunicación más cercana con el cliente por medio del uso de las redes sociales.

Conclusiones

Dada la naturaleza de los sistemas de información contable acerca de informar a las empresas o contribuyentes su entrada y salida de efectivo, estos sistemas deben contribuir a satisfacer las exigencias de los usuarios del área y al posicionamiento de las empresas en el ámbito competitivo, por lo que el sistema de información debe ser un mecanismo que facilite la transmisión de la información económica, de tal manera que los responsables de la administración la dispongan oportunamente para tomar decisiones sobre el manejo de los recursos.

Por lo que se plantea el manejo de diferentes patrones de diseño que nos ayude a mejorar tanto el rendimiento como la seguridad del usuario. Además de optimizar las consultas que se realizan en Base datos para que el usuario pueda consultar su información lo más rápido posible y finalmente contar con una interfaz gráfica lo más sencilla que facilite su uso a cualquier persona que ocupe la aplicación.

Bibliografía

[1]. Márquez Avendaño,Bertha M., Zulaica Rugarcía, José M. “Implementación de un reconocedor de voz gratuito a el sistema de ayuda a invidentes Dos-Vox en español”. Univ. De las Américas Puebla. Cholula, Puebla, México, 2004. Consultada. Octubre 2013.

[2]. Tipos de Servidores
http://fccea.unicauca.edu.co/old/siconceptosbasicos.htm

[3]. C sharp
http://www.ecured.cu/Lenguaje_de_Programaci%C3%B3n_C_Sharp

[4]. ASP.NET
https://es.wikipedia.org/wiki/ASP.NET

[5]. Lenguaje C sharp
https://msdn.microsoft.com/es-MX/library/aa287558%28v=vs.71%29.aspx

[6]. Informacion general ASP.NET
https://msdn.microsoft.com/es-es/library/4w3ex9c2%28v=vs.100%29.aspx

[7]. Berson, Alex. “Client/Server Architecture”. Edit. McGraw-Hill. 1992

[8]. Ceballos, Fco. Javier. Java 2 “Curso de programación”. 4ª Edición. Edit. Alfaomega - Ra-Ma. Febrero 2011.

[9]. Tipos de Datos Sql server 2012
https://msdn.microsoft.com/es-mx/library/ms187752%28v=sql.120%29.aspx

[10]. Tipos de Datos Sql Server 2012
https://franklindavidsql.wordpress.com/2013/01/05/tipos-de-datos-en-microsoft-sql-server-2012/

[11]. Cifrado de Sql Server
https://msdn.microsoft.com/es-es/library/bb510663%28v=sql.120%29.aspx

[12]. Algoritmo de Cifrado
https://msdn.microsoft.com/es-ES/library/ms345262%28v=sql.120%29.aspx

[13].Llave Asimétrica
https://technet.microsoft.com/es-es/library/ms174430%28v=sql.110%29.aspx

[14]. Certificados
https://technet.microsoft.com/es-es/library/ms187798%28v=sql.110%29.aspx

[15]. Llave Simétrica
https://msdn.microsoft.com/es-ES/library/ms188357%28v=sql.120%29.aspx

16. [Encriptación de Datos]
https://dbamemories.wordpress.com/2011/10/14/encriptacion-de-datos-en-sql-server-%E2%80%93-parte-3/

17. [ENCRYPTBYPASSPHRASE]
https://msdn.microsoft.com/es-es/library/ms190357%28v=sql.120%29.aspx

18. [Modelo Entidad - Relación]
http://www.belgrano.esc.edu.ar/matestudio/carpeta_de_access_introduccion.pdf

19. [Patrones de Diseño]
https://msdn.microsoft.com/es-es/library/bb972240.aspx

20. [Patrones de Diseño]
http://www.genbetadev.com/metodologias-de-programacion/patrones-de-diseno-que-son-y-por-que-debes-usarlos

21. [Singleton]

https://msdn.microsoft.com/es-es/library/bb972272.aspx#EEAA

22. [IoC]
http://cup-coffe.blogspot.mx/2010/04/inversion-de-control-ioc-y-contenedor.html

23. [IoC]
https://translate.google.com.mx/translate?hl=es-419&sl=en&u=https://msdn.microsoft.com/en-us/library/ff921087.aspx&prev=search

24. [Localizador de Servicio]
https://msdn.microsoft.com/en-us/library/ff921142.aspx

25. [Memento]
http://dofactory.com/net/memento-design-pattern

26. [Índices]
https://msdn.microsoft.com/es-es/library/jj835095%28v=sql.120%29.aspx

27. [Índices Clúster]
https://msdn.microsoft.com/es-es/library/ms190457%28v=sql.120%29.aspx

28. [Índices Clúster]
https://msdn.microsoft.com/es-es/library/ms186342%28v=sql.120%29.aspx

29. [Full Text Index]
https://translate.google.com.mx/translate?hl=es-419&sl=en&u=https://www.simple-talk.com/sql/learn-sql-server/understanding-full-text-indexing-in-sql-server/&prev=search

30. [Estructura de Full Text Index]
https://translate.google.com.mx/translate?hl=es-419&sl=en&u=https://msdn.microsoft.com/en-us/library/ms142571.aspx&prev=search
31. [Index Spatial]
https://translate.google.com.mx/translate?hl=es-419&sl=en&u=https://msdn.microsoft.com/en-us/library/bb895265.aspx&prev=search

32. [Plan de Ejecución]
https://technet.microsoft.com/es-es/library/ms178071%28v=sql.105%29.aspx

33. [showplan_text]
https://msdn.microsoft.com/es-es/library/ms176058%28v=sql.120%29.aspx

34. [showplan_all]
https://msdn.microsoft.com/es-es/library/ms187735%28v=sql.120%29.aspx

35. [Rendimiento en Consultas]
https://technet.microsoft.com/es-es/magazine/2007.11.sqlquery.aspx