SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE TLÁHUAC

"LA ESENCIA DE LA GRANDEZA RADICA EN MIS RAÍCES"

INFORME TÉCNICO DE RESIDENCIAS PROFESIONALES

TÍTULO DEL PROYECTO:

SISTEMA INTEGRAL DE INDICADORES PARA EL SGC DEL ITT

PRESENTA:
URBANO BALLESTEROS RODRIGUEZ
VIANET VARELA CHIRINOS

ASESORES:
ING. JUAN CARLOS CAMPOS CABELLO
ING. EDSON JESÚS ROSAS MARTÍNEZ

INGENIERÍA SISTEMAS COMPUTACIONALES

Fecha de entrega: 7 de febrero de 2017**.**

Tabla de contenido

Lista de figuras

Lista de tablas

Lista de algoritmos

Lista de siglas

Capítulo 1

Generalidades

En este capítulo se presenta de manera detallada los objetivos del proyecto, además de la información de la institución donde será desarrollado el mismo.

1.1. Objetivos

En esta sección se detallan los objetivos tanto generales como específicos a cumplir a lo largo del desarrollo del Sistema Integral de Indicadores.

1.1.1. Objetivo general

Desarrollar un sistema que permita medir, evaluar y controlar los procesos realizados por cada departamento en el Instituto Tecnológico de Tláhuac, basado en los lineamientos establecidos por el TecNM en el Sistema de Gestión de Calidad, para lograr realizar una gestión eficaz y de calidad de los mismos. Este sistema permitirá visualizar los resultados de los indicadores de tal manera que los procesos podrán ser monitoreados de una manera práctica y accesible.

1.1.2. Objetivos específicos

- Análisis de requerimientos.
- Toma de datos y entrevistas.
- Desarrollo del sistema.
- Implementación de sistema en ambiente de pruebas.
- Implemetación de sistema en ambiende de producción.

1.2. Justificación

Dicho proyecto está basado en la necesidad de contar con una herramienta que permita la evaluación automática de los procesos pertenecientes en el Sistema de Gestión de Calidad del Instituto Tecnológico de Tláhuac, esto debido a que no se cuenta con ningún sistema para la realización de dicha evaluación, ya que actualmente los procesos se realizan de manera manual, teniendo así procesos extensos y complicados. El Sistema Integral de Indicadores facilitará el trabajo realizado, agilizara los tiempos requeridos y permitirá con esto brindar un mejor servicio a la comunidad del Instituto Tecnológico de Tláhuac, además de que dichos indicadores influyen directamente para la toma de decisiones ya que muestran información acerca del cumplimiento de los objetivos, logrando como beneficio el perfeccionamiento de los procesos de la institución.

1.3. Caracterización de la empresa en la que participó

En esta sección se detalla la información general de la institución, así como el área para la cual se desarrolla el proyecto.

1.3.1. Datos generales de la empresa

- **Empresa:** Instituto Tecnológico de Tláhuac.
- **Dirección:** Av. Estanislao Ramírez No.301 Colonia Ampliación Selene C.P. 13420 Tláhuac D.F.

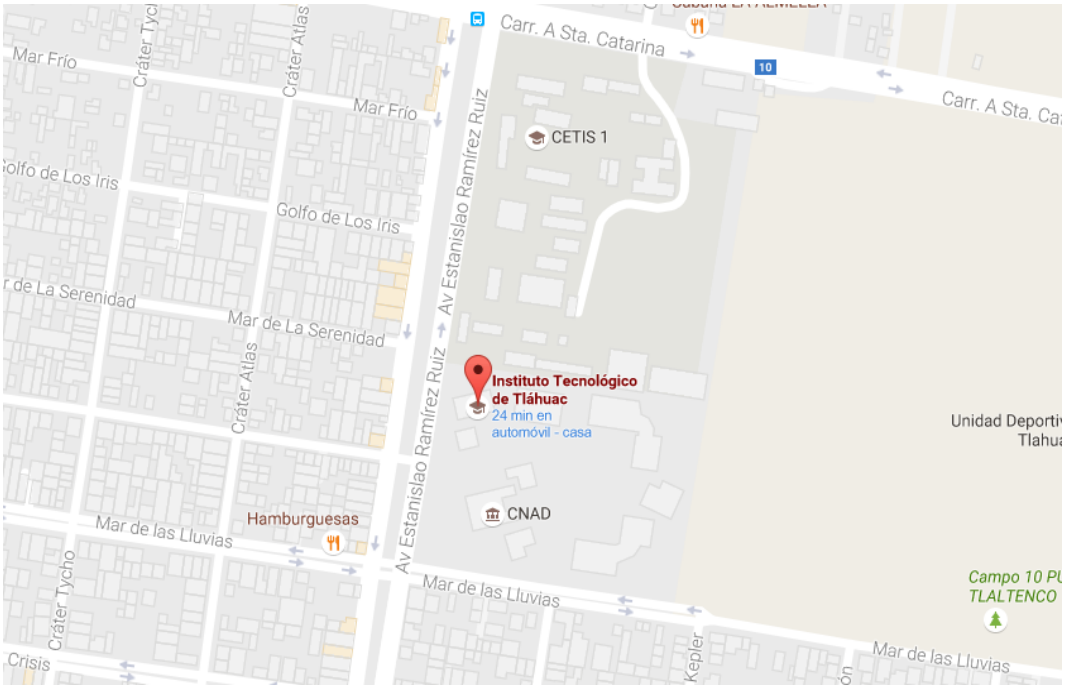

Figura 1.1: Croquis de ubicación.

- **Teléfono:** 5866-0927, 7312-5616, 5841-0560, 2594-4096
- **Dirección de correo electrónico:** sub.academica@ittlahuac.edu.mx
- **Giro:** Educativo.
- **Misión:** Ofrecer un servicio educativo de calidad a través de la mejora continua, personal capacitado, compromiso global e infraestructura de vanguardia.
- **Visión:** Ser una institución de alto desempeño académico, basada en la formación integral de profesionistas competitivos a nivel internacional con responsabilidad global.
- **Valores:** Los valores que manejamos en la institución son los que nos ayudan a identificarnos como seres humanos, como personas que están al servicio de la educación, que les gusta la actividad que desarrollan y que están orgullosos de promover un servicio de calidad a la comunidad.
  - Honestidad.
  - Respeto.

- Trabajo en equipo.
- Vocación de servicio.
- Comunicación.

- **Políticas de calidad:** La organización establece el compromiso de implementar todos sus procesos orientándolos hacia la satisfacción de sus estudiantes, sustentada en la calidad del proceso educativo, para cumplir con sus requisitos, mediante la eficacia de un sistema de gestión de calidad de mejora continua, conforme a la norma ISO 9001:2008/NMX-CC-9001-IMNC-2008.

- **Estructura organizacional:** La estructura organizacional del Instituto Tecnológico de Tláhuac se encentra como se muestra en la figura 1.2

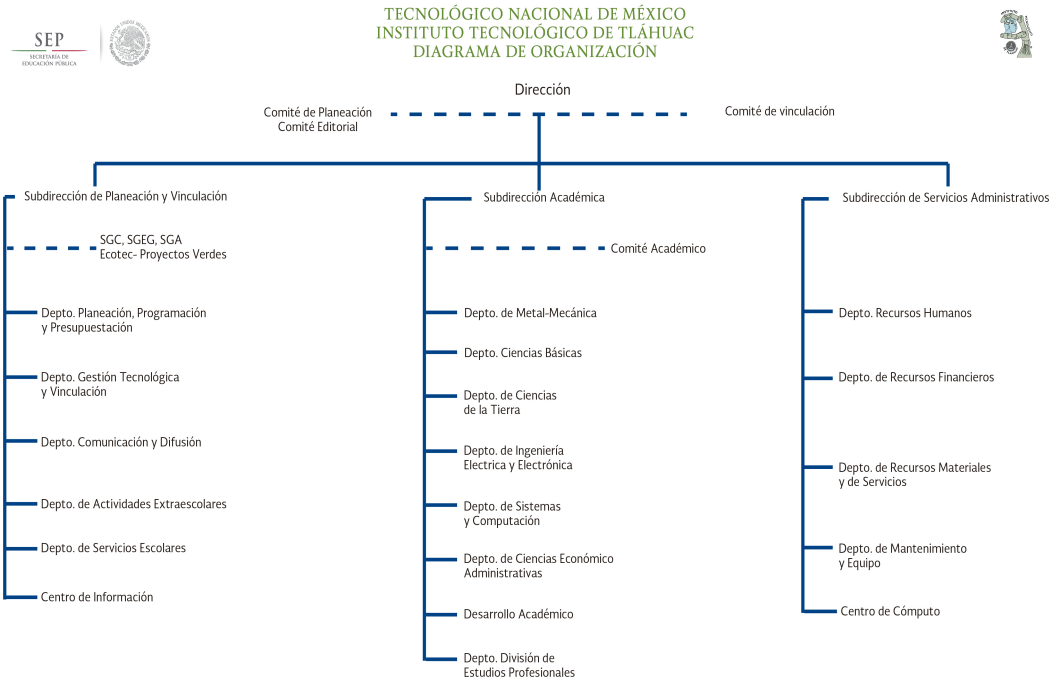

Figura 1.2: Croquis de ubicación.

1.3.2. Descripción del departamento o área de trabajo.

Definicion y detalle de el departamento.

1.4. Problemas a resolver

A continuación se detallaran los problemas a resolver con el Sistema Integral de Indicadores.

Dentro de los problemas más importantes a resolver se encuentra la digitalización de los procesos de indicadores se realizan de manera manual. Esto por ende lleva demasiado tiempo y requiere de una gran sincronización entre departamentos para que la información sea confiable. Realizar estas actividades requiere de demasiado esfuerzo por parte de los trabajadores del Instituto Tecnológico de Tláhuac lo que provoca que los trámites sean lentos y tediosos. El Sistema Integral de Indicadores pretende solucionar este problema permitiendo tener la información de los procesos en cualquier momento y de una manera sencilla, dando como resultado una sincronización de información eficiente y por consecuencia procesos más rápidos.

Como segundo problema a resolver es la generación de datos indicadores para la toma de decisiones, la cual de igual manera que los procesos realizados en los departamentos, el sistema permitirá configurar reportes de indicadores para los directivos, permitiendo con esto tener información general de los procesos.

Como último punto el sistema contará con la facilidad de adaptarse a nuevos procesos de indicadores, lo cual permite configurarse de tal manera que cuando la información de un indicador cambie, esta modificación no necesite realizarse un cambio en código.

Capítulo 2

Fundamento teórico

En el tercer capítulo se presentan las técnicas y herramientas utilizadas para adoptar una buena orientación que permita sustentar el desarrollo del Sistema Integral de indicadores mostrando una investigación detallada con todo lo necesario para entender cada fase del mismo.

2.1. Indicador

El primer concepto que nos debe quedar claro es saber que es un indicador, un indicador es un dato que nos ayuda a medir de manera objetiva la evolución de un sistema de gestión, por consiguiente un sistema de indicadores son datos que nos brindan información cualitativa o cuantitativa, que permiten seguir el desarrollo de un proceso y su evaluación.

2.2. Sistema de gestíon

Es una herramienta que permite a una organización planear, ejecutar y controlar las actividades realizadas en esta, aquellas que sean necesarias para el buen desarrollo de su misión. El objetivo de los sistemas es aportar a la institución un camino correcto para lograr el cumplimiento de las metas establecidas.

Todo sistema de medición debe satisfacer los siguientes objetivos:

- Comunicar la estrategia.
- Comunicar las metas.
- Identificar problemas y oportunidades.
- Diagnosticar problemas.
- Entender procesos.
- Definir responsabilidades
- Mejorar el control de la institución.
- Identificar iniciativas y acciones necesarias.
- Medir comportamientos.
- Facilitar la delegación en las personas.
- Integrar la compensación con la actuación.

El Sistema Integral de Indicadores es un sistema de gestión que organiza indicadores, permitiendo mediante su implementación obtener múltiples beneficios a la comunidad del Instituto Tecnológico de Tláhuac los cuales son:

- Mejorará la imagen ante los alumnos que deseen ingresar a dicha institución.
- Se lograra brindar un servicio caracterizado por la tolerancia y la responsabilidad.
- Permitirá contar con información útil para la mejora continua de procesos.
- Disminuirá las demoras en la realización de trámites internos.
- Se realizará una gestión enfocada al beneficio del alumno.
- Se Logrará el compromiso de los directivos con los objetivos organizacionales.
- Permitirá conocer la información de los avances en cuanto a la evaluación de procesos para poder tomar decisiones estratégicas y permitir alcanzar los objetivos.

Con dicho Sistema se pretende abarcar cada uno de los departamentos estratégicos que sustentan al Instituto Tecnológico de Tláhuac permitiendo a los directivos tener a su alcance los datos de los indicadores en cada uno de ellos, buscando una vinculación entre estos para obtener estadísticas correctas y precisas. Los departamentos abordados se muestran en la tabla 2.1.

Tabla 2.1: Tabla de departamentos organizadas por subdirecciones

SUBDIRECCION ACADÉMICA	CIENCIAS BÁSICAS
	CIENCIAS DE LA TIERRA
	ELECTRICA Y ELECTRONICA
	SISTEMAS Y COMPUTACION
	CIENCIAS ECONÓMICO-ADMINISTRATIVAS
	METAL-MECÁNICA
	DESARROLLO ACADÉMICO
	DIVISIÓN DE ESTUDIOS PROFESIONALES
SUBDIRECCIÓN ADMINISTRATIVA	RECURSOS HUMANOS
	RECURSOS FINANCIEROS
	RECURSOS MATERIALES Y DE SERVICIOS
	CENTRO DE COMPUTO
	MANTENIMIENTO Y EQUIPO
SUBDIRECCIÓN DE PLANEACIÓN Y VINCULACIÓN	GESTIÓN TECNOLÓGICA Y VINCULACIÓN
	COMUNICACIÓN Y DIFUSIÓN
	ACTIVIDADES EXTRAESCOLARES
	SERVICIOS ESCOLARES
	CENTRO DE INFORMACIÓN
	PLANEACIÓN, PROGRAMACIÓN Y PRESUPUESTACIÓN

Los departamentos antes mencionados se encuentran detallados en el sitio oficial del Instituto Tecnológico de Tláhuac en la seccion de departamentos.

Para el desarrollo del Sistema Integral de Indicadores se realizó una investigación acerca de las tecnologías que se utilizarían. Estas se encuentran clasificadas en dos partes importantes en el desarrollo web, la parte del cliente y la parte del servidor.

Las tecnologías que se encuentran en la parte del cliente son todas aquellas que son ejecutadas por el navegador web como son Javascript, HTML y CSS.

Las tecnologías del lado del servidor son todas aquellas en las cuales su ejecución se encuentra del lado del servidor como son C#, VB, Java, Python y PHP. Estos lenguajes tienen la capacidad de comunicarse directamente con la base de datos, y con esto realizar los procesos necesarios para formar el resultado solicitado por el cliente.

Dentro de los lenguajes de servidor se encuentran los lenguajes de consulta de base de datos, los cuales como su nombre lo dice sirven para manipular la información de una base de datos. El lenguaje de manipulación de datos que se utiliza en los diferentes motores de base de datos es SQL.

A continuación se definirán las tecnologías del cliente utilizadas para el desarrollo del sistema.

2.3. Arquitectura Cliente-Servidor

Se puede definir la computacion cliente servidor como una arquitectura distribuida que permite obtener acceso a informacion de manera transparente para el usuario final aun en entornos multiplataforma.

La arquitectura cliente servidor es el resultado de la integracion de dos culturas, por un lado la del Mainframe que aporta la capacidad de almacenamiento, integridad y acceso a la informacion, y por otro lado la del computador, la cual aporta la facilidad de uso a bajo costo ademas, provee de una presentacion atractiva y una gran variedad en productos y aplicaciones.

El funcionamiento de la arquitectura cliente-servidor es la siguiente:

- Solicitud de información del cliente al servidor.
- El servidor recibe la petición del cliente.
- El servidor procesa la solicitud.
- El servidor regresa el resultado con la información solicitada.
- El cliente recibe la información y la procesa.

La arquitectura cliente servidor está basado en la idea del servicio, en el que el cliente es un proceso consumidor de servicios y el servidor es un proceso proveedor de servicios, esta relación está establecida en función del intercambio de mensajes que es el único elemento de acoplamiento entre ambos.

Las partes fundamentales de la arquitectura cliente servidor son las siguientes:

- **Proceso en el cliente:** Es el que inicia el dialogo mediante una petición.
- **Proceso en el servidor:** Es quien se encuentra en espera de que lleguen peticiones de servicio y da respuesta a las mismas.

- **Middleware:** Es el medio por el cual se transmite información entre cliente y servidor, permitiendo el envío y recepción de mensajes.

En la figura 2.1 se muestra el diagrama que ilustra lo anterior.

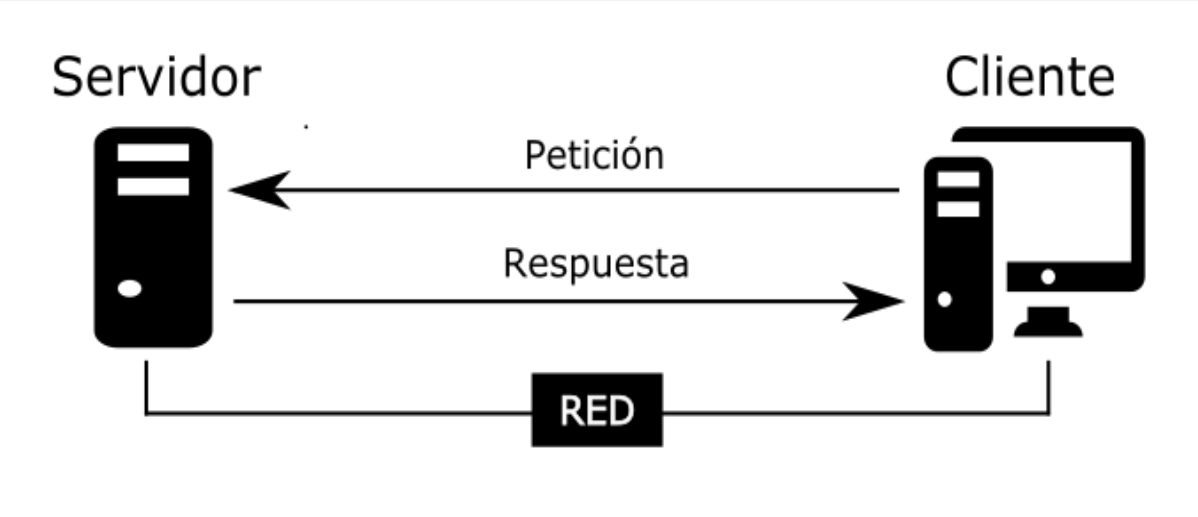

Figura 2.1: Diagrama de modelo cliente-servidor.

2.4. Lenguajes en cliente (Frontend)

2.4.1. HTML Y HTML 5

Definiéndolo de forma sencilla, *"HTML es lo que se utiliza para crear todas las páginas web de Internet"*. Más concretamente, HTML es el lenguaje con el que se *.esçriben"* la mayoría de páginas web.

Los diseñadores utilizan el lenguaje HTML para crear sus páginas web, los programas que utilizan los diseñadores generan páginas escritas en HTML y los navegadores que utilizamos los usuarios muestran las páginas web después de leer su contenido HTML.

Aunque HTML es un lenguaje que utilizan los ordenadores y los programas de diseño, es muy fácil de aprender y escribir por parte de las personas.

El lenguaje HTML es un estándar reconocido en todo el mundo y cuyas normas define un organismo sin ánimo de lucro llamado World Wide Web Consortium, más conocido como W3C. Como se trata de un estándar reconocido por todas las empresas relacionadas con el mundo de Internet, una misma página HTML se visualiza de forma muy similar en cualquier navegador de cualquier sistema operativo.

El propio W3C define el lenguaje HTML como *ün lenguaje reconocido universalmente y que permite publicar información de forma global"*. Desde su creación, el lenguaje HTML ha pasado de ser un lenguaje utilizado exclusivamente para crear documentos electrónicos a ser un lenguaje que se utiliza en muchas aplicaciones electrónicas como buscadores, tiendas online y banca electrónica.

El origen de HTML se remonta al año de 1980, cuando el físico Tim Berners-Lee, trabajador del CERN propuso un nuevo sistema de hipertexto para compartir documentos.

Los sistemas de hipertexto habían sido desarrollados años antes. En el ámbito de la informática el hipertexto permitía que los usuarios accedieran a la información relacionada con los documentos electrónicos que estaba visualizando. De cierta manera, los primitivos sistemas de hipertexto podrían asimilarse a los enlaces de las páginas web actuales.

Con el tiempo este sistema de hipertexto fue evolucionando convirtiéndose en el lenguaje de marcado más utilizado en la actualidad, siendo su última versión la conocida HTML5.

HTML5 establece una serie de nuevos elementos y atributos que reflejan el uso típico de los sitios web modernos, permitiendo concentrar básicamente tres características:

- Estructura.
- Estilo.
- Funcionalidad.

Aunque oficialmente no fue declarado, HTML5 es considerado como la combinación de HTML, CSS y Javascript ya que estas tecnologías son altamente dependientes y actúan como una sola unidad organizada bajo la especificación de HTML5. HTML está a cargo de la estructura, CSS presenta esta estructura y Javascript se encarga de darle funcionalidad.

2.4.2. CSS

Como ya se mencionó anteriormente el conjunto de tecnologías que trabajan en conjunto con HTML para formar HTML5 no solamente incluye los nuevos elementos que permiten que la definición de la estructura de un sistema se mas fácil, si no también es de vital importancia hablar de la parte que permite darle una buena presentación.

Oficialmente CSS no tiene nada que ver con HTML5 ya que es un complemento desarrollado para superar las limitaciones y reducir la complejidad de HTML. En un principio HTML proveía de atributos que permitían definir estilos esenciales para cada elemento, pero a medida que el lenguaje evoluciono, la escritura de códigos se volvió compleja y no pudo satisfacer las necesidades demandadas por los diseñadores, en consecuencia, CSS pronto fue adoptado gracias a que este permite separar la estructura de la presentación. Desde entonces CSS se ha convertido en uno de los lenguajes más importantes de la actualidad enfocado en las necesidades de los diseñadores.

En la versión 3 de CSS es considerado en el desarrollo de HTML5 , debido a esto la integración entre ambos lenguajes es vital para el desarrollo web y es la razón por la que cada que se habla de HTML5 también se hace referencia a CSS3 aunque oficialmente se trate de dos tecnologías completamente separadas. En estos momentos las nuevas características que se incorporan a CSS3 se han ido incorporando a los navegadores web al igual que las características de HTML5.

2.4.3. Javascript

Javascript es un lenguaje interpretado usado para múltiples propósitos pero solo considerado como un complemento hasta ahora. Una de las innovaciones que ayudó a cambiar el modo en que vemos Javascript fue el desarrollo de nuevos motores de interpretación, creados para acelerar el procesamiento de código. La clave de los motores más exitosos fue transformar el código Javascript en código máquina para lograr velocidades de ejecución similares a aquellas encontradas en aplicaciones de escritorio. Esta mejorada capacidad permitió superar viejas limitaciones de rendimiento y confirmar el lenguaje Javascript como la mejor opción para la web.

Para aprovechar esta prometedora plataforma de trabajo ofrecida por los nuevos navegadores, Javascript fue expandido en relación con portabilidad e integración.Ala vez, interfaces de programación de aplicaciones (APIs) fueron incorporadas por defecto en cada navegador para asistir al lenguaje en funciones elementales. Estas nuevas APIs (como Web Storage, Canvas, y otras) son interfaces para librerías incluidas en navegadores. La idea es hacer disponible poderosas funciones a través de técnicas de programación sencillas y estándares, expandiendo el alcance del lenguaje y facilitando la creación de programas útiles para la web.

Despues de hablar de las tecnologías del lado del cliente, a continuacion hablaremos de las que se ejecutan en el servidor.

2.5. Lenguajes en servidor (Backend)

2.5.1. C#

C# es el nuevo lenguaje de propósito general diseñado por Scott Wiltamuth y Anders Hejlsberg, este último también es conocido por diseñar el lenguaje Turbo Pascal y la herramienta RAD Delphi.

Aunque es posible escribir código para la plataforma .NET en distintos lenguajes, C# es el único diseñado específicamente para ser utilizado en ella, por lo que programarla usando C# es mucho más sencillo e intuitivo que hacerlo con cualquiera de los otros lenguajes ya que C# carece de elementos heredados innecesarios en .NET. Por esta razón, se suele decir que C# es el lenguaje nativo de .NET.

En resumen, C# es un lenguaje de programación que toma las mejores características de lenguajes preexistentes como Visual Basic, Java o C++ y las combina en uno solo. El hecho de ser relativamente reciente no implica que sea inmaduro, pues Microsoft ha escrito la mayor parte de la BCL usándolo, por lo que su compilador es el más depurado y optimizado de los incluidos en el .NET Framework SDK.

Las características principales de C# son las siguientes:

- **Sencillez:** C# elimina muchos elementos que otros lenguajes incluyen y que son innecesarios en .NET. Por ejemplo:
  - El código escrito en C# es auto contenido, lo que significa que no necesita de ficheros adicionales al propio fuente tales como ficheros de cabecera o ficheros IDL.
  - El tamaño de los tipos de datos básicos es fijo e independiente del compilador, sistema operativo o máquina para quienes se compile (no como en C++), lo que facilita la portabilidad del código.
  - No se incluyen elementos poco útiles de lenguajes como C++ tales como macros, herencia múltiple o la necesidad de un operador diferente del punto (.) acceder a miembros de espacios de nombres (::)
- **Modernidad:** C# incorpora en el propio lenguaje elementos que a lo largo de los años ha ido demostrándose son muy útiles para el desarrollo de aplicaciones y que en otros lenguajes como Java o C++ hay que simular, como un tipo básico decimal que permita realizar operaciones de alta precisión con reales de 128 bits, la inclusión de una instrucción foreach que permita recorrer colecciones con facilidad y es ampliable a tipos definidos por el usuario, la inclusión de un tipo básico string para representar cadenas o la distinción de un tipo bool específico para representar valores lógicos.
- **Orientación a objetos:** Como todo lenguaje de programación de propósito general actual, C# es un lenguaje orientado a objetos, aunque eso es más bien una característica del CTS que de C#. Una diferencia de este enfoque orientado a objetos respecto al de otros lenguajes como C++ es que el de C# es más puro en tanto que no admiten ni funciones ni variables globales sino que todo el código y datos han de definirse dentro de definiciones de tipos de datos, lo que reduce problemas por conflictos

de nombres y facilita la legibilidad del código.

C# soporta todas las características propias del paradigma de programación orientada a objetos: encapsulación, herencia y polimorfismo.

En lo referente a la encapsulación es importante señalar que aparte de los típicos modificadores public, private y protected, C# añade un cuarto modificador llamado internal, que puede combinarse con protected e indica que al elemento a cuya definición precede sólo puede accederse desde su mismo ensamblado.

Respecto a la herencia -a diferencia de C++ y al igual que Java- C# sólo admite herencia simple de clases ya que la múltiple provoca más quebraderos de cabeza que facilidades y en la mayoría de los casos su utilidad puede ser simulada con facilidad mediante herencia múltiple de interfaces. De todos modos, esto vuelve a ser más bien una característica propia del CTS que de C#.

Por otro lado y a diferencia de Java, en C# se ha optado por hacer que todos los métodos sean por defecto sellados y que los redefinibles hayan de marcarse con el modificador virtual (como en C++), lo que permite evitar errores derivados de redefiniciones accidentales. Además, un efecto secundario de esto es que las llamadas a los métodos serán más eficientes por defecto al no tenerse que buscar en la tabla de funciones virtuales la implementación de los mismos a la que se ha de llamar. Otro efecto secundario es que permite que las llamadas a los métodos El lenguaje de programación C# Tema 2: Introducción a C# José Antonio González Seco Página 23 virtuales se puedan hacer más eficientemente al contribuir a que el tamaño de dicha tabla se reduzca.

- **Orientación a componentes:** La propia sintaxis de C# incluye elementos propios del diseño de componentes que otros lenguajes tienen que simular mediante construcciones más o menos complejas. Es decir, la sintaxis de C# permite definir cómodamente propiedades (similares a campos de acceso controlado), eventos (asociación controlada de funciones de respuesta a notificaciones) o atributos (información sobre un tipo o sus miembros).

- **Gestión automática de memoria:** Como ya se comentó, todo lenguaje de .NET tiene a su disposición el recolector de basura del CLR. Esto tiene el efecto en el lenguaje de que no es necesario incluir instrucciones de destrucción de objetos. Sin embargo, dado que la destrucción de los objetos a través del recolector de basura es indeterminista y sólo se realiza cuando éste se active ya sea por falta de memoria, finalización de la aplicación o solicitud explícita en el fuente-, C# también proporciona un mecanismo de liberación de recursos determinista a través de la instrucción using.

- **Seguridad de tipos:** C# incluye mecanismos que permiten asegurar que los accesos a tipos de datos siempre se realicen correctamente, lo que permite evita que se produzcan errores difíciles de detectar por acceso a memoria no perteneciente a ningún objeto y es especialmente necesario en un entorno gestionado por un recolector de basura. Para ello se toman medidas del tipo:

  - Sólo se admiten conversiones entre tipos compatibles. Esto es, entre un tipo y antecesores suyos, entre tipos para los que explícitamente se haya definido un operador de conversión, y entre un tipo y un tipo hijo suyo del que un objeto del primero almacenase una referencia del segundo (downcasting) Obviamente, lo último sólo puede comprobarlo en tiempo de ejecución el CLR y no el compilador, por lo que en realidad el CLR y el compilador colaboran para asegurar la corrección de las conversiones.
  - No se pueden usar variables no inicializadas. El compilador da a los campos un valor por defecto consistente en ponerlos a cero y controla mediante análisis del flujo de control del fuente que no se lea ninguna variable local sin que se le haya asignado previamente algún valor.

  - Se comprueba que todo acceso a los elementos de una tabla se realice con índices que se encuentren dentro del rango de la misma.
  - Se puede controlar la producción de desbordamientos en operaciones aritméticas, informándose de ello con una excepción cuando ocurra. Sin embargo, para conseguirse un mayor rendimiento en la aritmética estas comprobaciones no se hacen por defecto al operar con variables sino sólo con constantes (se pueden detectar en tiempo de compilación).
  - A diferencia de Java, C# incluye delegados, que son similares a los punteros a funciones de C++ pero siguen un enfoque orientado a objetos, pueden almacenar referencias a varios métodos simultáneamente, y se comprueba que los métodos a los que apunten tengan parámetros y valor de retorno del tipo indicado al definirlos.
  - Pueden definirse métodos que admitan un número indefinido de parámetros de un cierto tipo, y a diferencia lenguajes como C/C++, en C# siempre se comprueba que los valores que se les pasen en cada llamada sean de los tipos apropiados.

- **Instrucciones seguras:** Para evitar errores muy comunes, en C# se han impuesto una serie de restricciones en el uso de las instrucciones de control más comunes. Por ejemplo, la guarda de toda condición ha de ser una expresión condicional y no aritmética, con lo que se evitan errores por confusión del operador de igualdad (==) con el de asignación (=); y todo caso de un switch ha de terminar en un break o goto que indique cuál es la siguiente acción a realizar, lo que evita la ejecución accidental de casos y facilita su reordenación.

- **Sistema de tipos unificado:** A diferencia de C++, en C# todos los tipos de datos que se definan siempre derivarán, aunque sea de manera implícita, de una clase base común llamada System.Object, por lo que dispondrán de todos los miembros definidos en ésta clase (es decir, serán .ºbjetos").

  A diferencia de Java, en C# esto también es aplicable a los tipos de datos básicos Además, para conseguir que ello no tenga una repercusión negativa en su nivel de rendimiento, se ha incluido un mecanismo transparente de boxing y unboxing con el que se consigue que sólo sean tratados como objetos cuando la situación lo requiera, y mientras tanto puede aplicárseles optimizaciones específicas.

  El hecho de que todos los tipos del lenguaje deriven de una clase común facilita enormemente el diseño de colecciones genéricas que puedan almacenar objetos de cualquier tipo.

- **Extensibilidad de tipos básicos:** C# permite definir, a través de estructuras, tipos de datos para los que se apliquen las mismas optimizaciones que para los tipos de datos básicos. Es decir, que se puedan almacenar directamente en pila (luego su creación, destrucción y acceso serán más rápidos) y se asignen por valor y no por referencia. Para conseguir que lo último no tenga efectos negativos al pasar estructuras como parámetros de métodos, se da la posibilidad de pasar referencias a pila a través del modificador de parámetro ref.

- **Extensibilidad de operadores:** Para facilitar la legibilidad del código y conseguir que los nuevos tipos de datos básicos que se definan a través de las estructuras estén al mismo nivel que los básicos predefinidos en el lenguaje, al igual que C++ y a diferencia de Java, C# permite redefinir el significado de la mayoría de los operadores -incluidos los de conversión, tanto para conversiones implícitas como explícitas- cuando se apliquen a diferentes tipos de objetos.

  Las redefiniciones de operadores se hacen de manera inteligente, de modo que a partir de una única definición de los operadores ++ y – el compilador puede deducir automáticamente como ejecutarlos de manera prefijas y postifja; y definiendo operadores simples (como +), el compilador deduce cómo aplicar su versión de asignación compuesta (+=). Además, para asegurar la consistencia, el compilador vigila que los operadores con opuesto siempre se redefinan por parejas (por ejemplo, si se redefine ==,

también hay que redefinir !=).

También se da la posibilidad, a través del concepto de indizador, de redefinir el significado del operador [] para los tipos de dato definidos por el usuario, con lo que se consigue que se pueda acceder al mismo como si fuese una tabla. Esto es muy útil para trabajar con tipos que actúen como colecciones de objetos.

- **Extensibilidad de modificadores:** C# ofrece, a través del concepto de atributos, la posibilidad de añadir a los metadatos del módulo resultante de la compilación de cualquier fuente información adicional a la generada por el compilador que luego podrá ser consultada en tiempo ejecución a través de la librería de reflexión de .NET . Esto, que más bien es una característica propia de la plataforma .NET y no de C#, puede usarse como un mecanismo para definir nuevos modificadores.

- **Versionable:** C# incluye una política de versionado que permite crear nuevas versiones de tipos sin temor a que la introducción de nuevos miembros provoquen errores difíciles de detectar en tipos hijos previamente desarrollados y ya extendidos con miembros de igual nombre a los recién introducidos.

  Si una clase introduce un nuevo método cuyas redefiniciones deban seguir la regla de llamar a la versión de su padre en algún punto de su código, difícilmente seguirían esta regla miembros de su misma signatura definidos en clases hijas previamente a la definición del mismo en la clase padre; o si introduce un nuevo campo con el mismo nombre que algún método de una clase hija, la clase hija dejará de funcionar. Para evitar que esto ocurra, en C# se toman dos medidas:

  - Se obliga a que toda redefinición deba incluir el modificador override, con lo que la versión de la clase hija nunca sería considerada como una redefinición de la versión de miembro en la clase padre ya que no incluiría override. Para evitar que por accidente un programador incluya este modificador, sólo se permite incluirlo en miembros que tengan la misma signatura que miembros marcados como redefinibles mediante el modificador virtual. Así además se evita el error tan frecuente en Java de creerse haber redefinido un miembro, pues si el miembro con override no existe en la clase padre se producirá un error de compilación.
  - Si no se considera redefinición, entonces se considera que lo que se desea es ocultar el método de la clase padre, de modo que para la clase hija sea como si nunca hubiese existido. El compilador avisará de esta decisión a través de un mensaje de aviso que puede suprimirse incluyendo el modificador new en la definición del miembro en la clase hija para así indicarle explícitamente la intención de ocultación.

- **Eficiente:** En principio, en C# todo el código incluye numerosas restricciones para asegurar su seguridad y no permite el uso de punteros. Sin embargo, y a diferencia de Java, en C# es posible saltarse dichas restricciones manipulando El lenguaje de programación C# Tema 2: Introducción a C# José Antonio González Seco Página 26 objetos a través de punteros. Para ello basta marcar regiones de código como inseguras (modificador unsafe) y podrán usarse en ellas punteros de forma similar a cómo se hace en C++, lo que puede resultar vital para situaciones donde se necesite una eficiencia y velocidad procesamiento muy grandes.

- **Compatible:** Para facilitar la migración de programadores, C# no sólo mantiene una sintaxis muy similar a C, C++ o Java que permite incluir directamente en código escrito en C# fragmentos de código escrito en estos lenguajes, sino que el CLR también ofrece, a través de los llamados Platform Invocation Services (PInvoke), la posibilidad de acceder a código nativo escrito como funciones sueltas no orientadas a objetos tales como las DLLs de la API Win32. Nótese que la capacidad de usar punteros en código inseguro permite que se pueda acceder con facilidad a este tipo de funciones, ya que éstas muchas veces esperan recibir o devuelven punteros.

También es posible acceder desde código escrito en C# a objetos COM. Para facilitar esto, el .NET Framework SDK incluye una herramientas llamadas tlbimp y regasm mediante las que es posible generar automáticamente clases proxy que permitan, respectivamente, usar objetos COM desde .NET como si de objetos .NET se tratase y registrar objetos .NET para su uso desde COM.

Finalmente, también se da la posibilidad de usar controles ActiveX desde código .NET y viceversa. Para lo primero se utiliza la utilidad aximp, mientras que para lo segundo se usa la ya mencionada regasm.

2.6. Frameworks

El concepto framework se emplea en muchos ámbitos del desarrollo de sistemas software, no solo en el ámbito de aplicaciones Web. Podemos encontrar frameworks para el desarrollo de aplicaciones médicas, de visión por computador, para el desarrollo de juegos, y para cualquier ámbito que pueda ocurrírsenos.

En general, con el término framework, nos estamos refiriendo a una estructura software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. En otras palabras, un framework se puede considerar como una aplicación genérica incompleta y configurable a la que podemos añadirle las últimas piezas para construir una aplicación concreta.

Éxisten frameworks tanto para lenguajes del cliente como del servidor, permitiendo realizar desarrollos mas rÃ¡pidos y eficientes. Dentro de los frameworks utilizados se encuentran los siguientes:

2.6.1. Frameworks en cliente

Bootstrap

Bootstrap es un framework de Twitter el cual está enfocado para el desarrollo Web, específicamente diseñado para la maquetación de sitios Web mediante el lenguaje de estilos CSS y Javascript. Este está conformado por un conjunto de archivos CSS y Javascript los cuales al incluirlos en un sitio Web se facilita el maquetado y la edición de estilos de la misma sin tocar en absoluto el código CSS, esto agiliza el desarrollo de aplicaciones Web dándonos un diseño elegante y bueno gracias a sus clases predefinidas.

El Framework trae varios elementos con estilos predefinidos fáciles de configurar: Botones, Menús desplegables, Formularios incluyendo todos sus elementos e integración jQuery para ofrecer ventanas y mensajes dinámicos.

JQuery

JQuery es una biblioteca de JavaScript creada inicialmente por John Resig, la cual nos permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar integración con la técnica AJAX a páginas Web.

El DOM (Document Object Model), proporciona un conjunto estándar de objetos para representar documentos HTML y XML, un modelo estándar sobre cómo pueden combinarse dichos objetos, y una interfaz estándar para acceder a ellos y manipularlos. A través del DOM, los programas pueden acceder y modificar el contenido, estructura y estilo de los documentos HTML y XML, que es para lo que se diseñó principalmente.

2.6.2. Frameworks en servidor

.NET Framework

El framework utilizado en el servidor es .NET Framework de Microsoft el cual es un componente de software que puede ser añadido al sistema operativo Windows, este provee un extenso conjunto de soluciones predefinidas para necesidades generales de la programacion de aplicaciones y administra la ejecución de los programas escritos específicamente con la plataforma.

.NET Framework podría ser considerada una respuesta de Microsoft al creciente mercado de los negocios en entornos WEB, como competencia a la plataforma Java de Oracle Corporation y a los diversos Frameworks de desarrollo web usados en PHP. Dicho Framework propone ofrecer una manera rápida y económica, a la vez que segura y robusta para desarrollar aplicaciones permitiendo una integración rápida y ágil, permitiendo con esto acceso a la información desde cualquier tipo de dispositivo.

A continuación se muestra una breve descripción de las versiones de .NET Framework:

- **.NET Framework 1.0:** Lanzado en 2002 (Visual Studio .NET), Esta es la primera versión de .NET Framework, publicado el 13 de febrero de 2002 y disponible para Windows 98,ME,NT 4.0,2000 y XP. El soporte estándar de Microsoft para esta versión finaliza 10 de julio de 2007 y el soporte extendido terminó el 14 de julio de 2009, con la excepción de XP Media Center y Tablet PC ediciones.

- **.NET Framework 1.1:** Lanzado en 2003 (Visual Studio 2003), El soporte integrado para teléfonos ASP.NET controles. Previamente disponible como un add-on para .NET Framework, que ahora forma parte del marco. Los cambios en la seguridad permiten utilizar ensamblados para ejecutar aplicaciones en internet con restricciones de confianza permitiendo acceesos restringidos a las aplicaciones de ASP.NET. El soporte para ODBC y bases de datos le permite una mayor funcionalidad ademas de implementar el soporte para Internet versíon 6.

- **.NET Framework 2.0:** Lanzado en 2005 (Visual Studio .NET 2005), con un nuevo CLR (para manejar los genéricos y tipos anulables) y los compiladores de C# y VB 2 8. El paquete redistribuible 2.0 se puede descargar de forma gratuita desde Microsoft, y fue publicado el 22 de enero de 2006. 2.0 El SDK(Software Development Kit) se puede descargar de forma gratuita desde Microsoft. Se incluye como parte de Visual Studio 2005 y Microsoft SQL Server 2005. La versión 2.0 sin ningún Service Pack es la última versión con soporte para Windows 98 y Windows Me. Versión 2.0 con Service Pack 2 es la última versión con soporte oficial para Windows 2000, aunque ha habido algunas soluciones no oficiales publicados en línea para utilizar un subconjunto de la funcionalidad de la Versión 3.5 en Windows 2000.

- **.NET Framework 3.0:** Lanzado en 2006 , básicamente es sólo .NET 2.0 con nuevas bibliotecas como Windows Presentation Foundation, Windows Communication Foundation, Workflow Foundation y Cardspace, .NET Framework 3.0 anteriormente llamado WinFX, fue lanzado el 21 de noviembre de 2006 e incluye un nuevo sistema de código administrado API que son una parte integral de Windows Vista y Windows Server 2008 sistemas operativos. TambiÃ©n estÃ¡ disponible para Windows XP SP2 y Windows Server 2003 como descarga. No hay grandes cambios de arquitectura que se incluyen con esta versión;. NET Framework 3.0 utiliza el mismoCLR (Common Language Runtime) como NET Framework 2.0 .

- **.NET Framework 3.5:** Lanzado en 2007, basado en .NET 3.0 incluyendo nuevas bibliotecas como LINQ y TimeZoneInfo, además de nuevos compiladores de C# y VB 3.9.

- **.NET Framework 4.0:** Lanzado en 2010 con un nuevo CLR, nuevas bibliotecas y el DLR(Dynamic Languaje Runtime). La parte principal de esta versión son las extensiones que permiten mejorar el apoyo para el computo paralelo, dirigida a multiples núcleos o sistemas distribuidos. Se implementa la tecnología de computo paralelo para LINQ llamada PLINQ (Paralel LINQ).

- **.NET Framework 4.5:** Lanzado en 2012 permite el desarrollo en Windows 8, así como bibliotecas adicionales que permiten la resolución más eficiente de expresiones regulares. También permite definir una cultura para un dominio de aplicación y soporte para la Unicode(UTF-16).

Estas tecnologías en conjunto son de vital importancia debido a que las funcionalidades que se agregan al sistema dependen del trabajo en conjunto de las mismas.

Para mejorar el funcionamiento del SII fue necesario aplicar un modelo de arquitectura de software ya que en la actualidad de no hacerlo de esta manera, puede que el sistema no tenga el rendimiento o funcionamiento esperado.

A continuación se describen los conceptos de arquitectura de software utilizados en el SII.

2.7. Arquitectura de Software

En un principio la programación fue considerada un arte y se desarrollaba como tal, esto debido a la dificultad que representaba realizar dicha actividad. Con el paso del tiempo se han ido desarrollando nuevas formas para el desarrollo de software con las cuales la resolución de problemas ha reducido su complejidad. A estas formas de desarrollo se les denomina Arquitectura de Sofware.

La arquitectura de software es definida como un nivel de diseño que pone especial interés en aspectos más allá de los algoritmos y estructuras de datos de la computación; el diseño y especificación de la estructura global del sistema es un nuevo tipo de problema.

La arquitectura de software es el diseño de mas alto nivel de la estructura de un sistema permitiendo con esto contar con un panorama amplio acerca del funcionamiento del sistema.

Para elegir una arquitectura de software se necesita realizar un análisis de objetivos y requerimientos, además de las restricciones, por ejemplo, no es recomendable utilizar una arquitectura de 3 capas para un sistema que manejará información en tiempo real.

Generalmente, no es necesario inventar una nueva arquitectura para cada sistema a desarrollar, mas bien es recomendable adoptar alguna de las existentes.

Dentro de las arquitecturas mas conocidas se encuentran las siguientes:

- **Descomposición modular:** Donde el software se estructura en grupos funcionales muy acoplados.
- **Cliente-Servidor:** Donde el software reparte su carga de cómputo en dos partes independientes pero sin reparto claro de funciones.
- **Arquitectura de tres niveles:** Especialización de la arquitectura cliente-servidor donde la carga se divide en tres partes (o capas) con un reparto claro de funciones: una capa para la presentación (interfaz de usuario), otra para el cálculo (donde se encuentra modelado el negocio) y otra para el almacenamiento (persistencia). Una capa solamente tiene relación con la siguiente.
- **Modelo Vista Controlador:** Es un patrón de arquitectura de software, que separa los datos y la lógica de negocios de un aplicación de la interfaz de usuario y el modulo encargado de gestionar los eventos y las comunicaciones.

La arquitectura seleccionada para el SII es el Modelo Vista Controlador (MVC) el cual debido a la forma de organizar el funcionamiento del sistema cubre muy bien las necesidades del sistema.

2.7.1. MVC (Modelo Vista Controlador)

Modelo Vista Controlador (MVC) es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. Se trata de un modelo muy maduro y que ha demostrado su validez a lo largo de los años en todo tipo de aplicaciones, y sobre multitud de lenguajes y plataformas de desarrollo.

Los tres componentes en los que se separa el Modelo Vista Controlador son:

- **Modelo:** Contiene una representación de los datos que maneja el sistema, su lógica de negocio y sus mecanismos de persistencia.
- **Vista:** Esta es en otras palabras la interfaz, la cual contiene la información que se envía al cliente y los mecanismos de interacción con el mismo.
- **Controlador:** Este actual como intermediario entre el modelo y la vista, gestionando el flujo de información entre ellos y las transformaciones para adoptar los datos a las necesidades.

El modelo se encarga de las siguientes acciones:

- Acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.
- Define las reglas de negocio (la funcionalidad del sistema). Un ejemplo de regla puede ser: "Si la mercancía pedida no está en el almacén, consultar el tiempo de entrega estándar del proveedor".
- Lleva un registro de las vistas y controladores del sistema.
- Si estamos ante un modelo activo, notificará a las vistas los cambios que en los datos pueda producir un agente externo (por ejemplo, un fichero por lotes que actualiza los datos, un temporizador que desencadena una inserción, etc).

El controlador se encarga de las siguientes acciones:

- Recibe los eventos de entrada (un clic, un cambio en un campo de texto, etc).
- Contiene reglas de gestión de eventos, del tipo "SI Evento Z, entonces Acción W". Estas acciones pueden suponer peticiones al modelo o a las vistas. Una de estas peticiones a las vistas puede ser una llamada al método .ªctualizar ()". Una petición al modelo puede ser ObtenerTiempoDeEntrega (nuevaOrdenDeVenta).

La vista se encarga de las siguientes acciones:

- Recibir datos del modelo y la muestra al usuario.
- Tienen un registro de su controlador asociado (normalmente porque además lo instancia).
- Pueden dar el servicio de Actualización(), para que sea invocado por el controlador o por el modelo (cuando es un modelo activo que informa de los cambios en los datos producidos por otros agentes).

El flujo que sigue una petición dentro del MVC se describe en la fígura 2.2

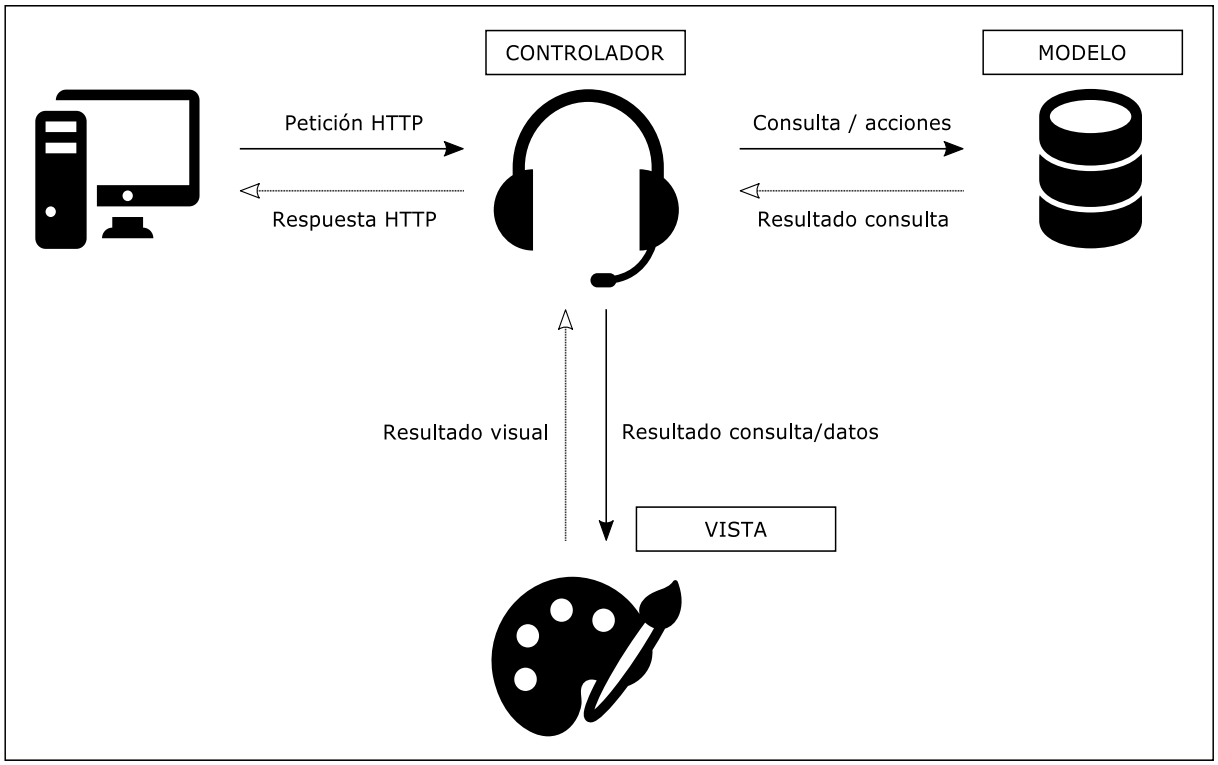

Figura 2.2: Diagrama de flujo de una petición en MVC.

Los pasos que sigue la petición son los siguientes:

a) El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, enlace, etc).

b) El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos (handler) o callback.

c) El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario (por ejemplo, el controlador actualiza el carro de la compra del usuario). Los controladores complejos están a menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.

d) El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se refleja los cambios en el modelo (por ejemplo, produce un listado del contenido del carro de la compra). El modelo no debe tener conocimiento directo sobre la vista. Sin embargo, se podría utilizar el patrón Observador para proveer cierta indirección entre el modelo y la vista, permitiendo al modelo notificar a los interesados de cualquier cambio. Un objeto vista puede registrarse con el modelo y esperar a los cambios, pero aun así el modelo en sí mismo sigue sin saber nada de la vista. El controlador no pasa objetos de dominio (el modelo) a la vista aunque puede dar la orden a la vista para que se actualice. Nota: En algunas implementaciones la vista no tiene acceso directo al modelo, dejando que el controlador envÃe los datos del modelo a la vista.

e) La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

2.8. Base de Datos

En los últimos tiempos, se ha empezado a considerar la información como un recurso estratégico de una organización; pues facilita su supervivencia y le permite, además, ser competitiva en el mercado. Dicha información le permite tener una visión de los posibles cambios, y con esto mejorar la toma de decisiones.

Una base de datos se define como una colección de datos almacenados de una manera permanente, que pueden ser compartidos y usados con distintos propósitos por múltiples usuarios.

La información se debe considerar como otro recurso corporativo de la misma manera que se considera el recurso humano o los recursos físicos. Por lo tanto, la administración de la información debe implicar:

- Planear su adquisición por anticipado.
- Protegerla contra la destrucción o mal uso.
- Asegurar su calidad.
- Retirarla de la organización cuando ya no se le requiera.

Debido que la cantidad de información que una organización necesita para subsistir es cada vez mayor, deben existir métodos eficientes tanto para el almacenamiento rápido como para la consulta ágil. La tecnología más utilizada para manejar estas grandes cantidades de información son las Bases de Datos.

La tecnología de bases de datos proporciona los medios para que se cumplan los objetivos de lograr máximos beneficios en la administración de información debido a las siguientes razones:

- Se logra que el desarrollo de aplicaciones sea más rápido debido a que los programadores reutilizan datos u procedimientos almacenados en la base de datos.
- Hay una mayor participación del usuario final en la creación de las aplicaciones, haciendo software más tangible y de mayor valor inmediato.
- El acceso a los datos es flexible y rápido.

Gracias a esto las organizaciones tienen más y mejor información para la toma de decisiones, además de incrementar su productividad.

A continuación se presentan las ventajas y desventajas de un sistema de base de datos:

a) Ventajas

1) **Economía de escala:** esencialmente, la concentración de aplicaciones en una sola localidad puede reducir costos.
2) **Mayor información:** existe una mayor facilidad para el análisis y la toma de desiciones.
3) **Datos y programas compartidos:** la reutilización de los mismos datos y programas, permiten minimizar o controlar la redundancia.
4) **Declaración de estándares:** Debido a que la información se vuelve muy grande, es necesario declarar estándares para el acceso y manejo de los datos.
5) **Consistencia de datos:** Esta dada por el control o eliminación de la redundancia.
6) **Integridad:** El sistema de base de datos debe velar por el grado de validez y de corrección de los datos. Debe permitir definir reglas que deben cumplir los datos en la base de datos.
7) **Seguridad:** Se pueden especificar niveles de acceso mediante perfiles de usuario.

b) Desventajas

- **Tamaño:** Un sistema de base de datos es un gran conjunto de programas que a lo largo de su uso su tamaño incrementa considerablemente.
- **Mayor susceptibilidad a fallas:** Debido a la cantidad de información es más probable que se generen fallas.
- **Recuperación a las fallas:** La recuperación de un DBMS interactivo y multiusuario puede ser muy compleja.

Los sitemas de base de datos utilizan un lenguaje el cual le permite realizar modificaciones sobre ella. Este lenguajes es el llamado SQL.

2.8.1. Lenguaje SQL

El lenguaje SQL es un conjunto de onstrucciones que permite crear, consultar y modificar una base de datos. Para realizar estas acciones, SQL se divide en tres grupos de lenguajes de base de datos:

- **DML (Data Manipulation Language):** Permite consultar y modificar datos.
- **DDL (Data Definition Language):** Permite definir la base de datos.
- **DAL (Data Acces Language):** Permite administrar el acceso a la base de datos.

Todos los sistemas gestores de base de datos cuentan con estos tres grupos para poder realizar las acciones sobre las bases de datos. Dentro de los sitemas gestores de base de datos mas comunes se encuentran los siguientes:

- SQL Server.
- MySQL.
- Postgress SQL.
- Oracle.

El lenguaje elegido para realizar el SII es SQL Server, esto debido a la integración que tiene con el ambiente de desarrollo y los lenguajes seleccionados.

2.8.2. Microsoft SQL Server

Microsoft SQL server es un sistema de gestión de bases de datos relacional producido por Microsoft. Su principal lenguaje de consulta es Transact-SQL, una aplicación de las normas ANSI/ISO estÃ¡ndar Structured Query Language (SQL) utilizando por ambas Microsoft y Sybase.
Las características de Microsoft SQL Server son las siguientes:

- Soporte de transacciones.
- Escalabilidad, estabilidad y seguridad.
- Soporta procedimientos almacenados.
- Incluye también un potente entorno gráfico de administración, que permite el uso de comandos DDL y DML gráficamente.
- Permite trabajar en modo cliente-servidor, donde la información y datos se alojan en el servidor y las terminales o clientes de la red sólo acceden a la información.

- Además permite administrar información de otros servidores de datos.

Este sistema incluye una versión reducida, llamada MSDE con el mismo motor de base de datos pero orientado a proyectos más pequeños, que en su versión 2005 pasa a ser el SQL Express Edition, que se distribuye en forma gratuita.

Microsoft SQL Server constituye la alternativa de Microsoft a otros potentes sistemas gestores de bases de datos como son Oracle, Sybase ASE, PostgreSQL o MySQL.

Requisitos de hardware

Tabla 2.2: Requisitos de Hardware de Microsoft SQL Server

COMPONENTE	REQUISITO
Memoría	Minimo: Ediciones Express: 512 MB Todas las demás ediciones: 1 GB Recomendaciones: Ediciones Express: 1 GB Todas las demás ediciones: Al menos 4 GB y debe aumentar a medida que el tamaño de la base d
Velocidad del procesador	Minimo - Procesador x86: 1 GHz. - Procesador x64: 1.4 Ghz. Recomendado: 2 GHz o más.
Tipo de procesador	- Procesador x64: AMD Opteron, AMD Athlon 64, Intel Xeon compatible con Intel EM64T Intel Pentium IV. - Procesador x86: compatible con Pentium III o superior.

Rendimiento

Respecto al rendimiento de Microsoft SQL Server, este se desempeña dentro de cualquier arquitectura .NET o J2EE.

Del mismo modo este se beneficia de RAID (Redundant Array of Independent Disks) el cual es un sistema de almacenamiento de datos entre los que se distribuye o replican los datos, es decir, este se desempeña mejor si se ejecuta en un servidor dedicado específicamente a él.

Microsoft SQL Server como cualquier otro sistema manejador de bases de datos depende de la estructura de las instrucciones que recibe para determinar su rendimiento, esto debido a que existen distintas maneras de consultar o modificar la información, es por ello que es importante seguir la documentación del producto para realizar las instrucciones más rápidas posibles.

Administración y mantenimiento

Microsoft SQL Server ofrece un grupo de acciones que permiten administrar los recursos dentro del mismo, las cuales se enlistan a continuación:

- Se provee de un sistema de copias de seguridad en la cual se vuelcan los datos y estructura en un archivo con extensión ".bak".
- Soporta tres tipos de replicaciń:
  - **Instantanea:** En la replicación de instantáneas los datos se copian tal y como aparecen exactamente en un momento determinado.
  - **Transaccional:** En este caso se propaga una instantánea inicial de datos a los suscriptores, y después, cuando se efectúan las modificaciones en el publicador, las transacciones individuales se propagan a los suscriptores. Microsoft SQL Server almacena las transacciones que afectan a los objetos replicados y propaga esos cambios a los suscriptores de forma continua o a intervalos programados. Al finalizar la propagación de los cambios, todos los suscriptores tendrán los mismos valores que el publicador.
  - **Mixta:** Permite que varios sitios funcionen en línea o desconectados de manera autónoma, y mezclar más adelante las modificaciones de datos realizadas en un resultado único y uniforme. La instantánea inicial se aplica a los suscriptores; a continuación SQL Server 2000 hace un seguimiento de los cambios realizados en los datos publicados en el publicador y en los suscriptores.

Estabilidad

Microsoft SQL Server tiene una gran resistencia a la corrupción de datos ya que los datos van a través de distintos puestos de control y previene contratiempos en caso de que el sistema colapse sin previo aviso. Se puede considerar como un contratiempo a un fallo inesperado del equipo huésped de Microsoft SQL Server o un apagado inesperado del mismo.

Desarrollo de aplicaciones

- Se cuenta con múltiples APIs para acceder a él.
- Se apoya en ODBC y JDBC para conectividad en red, así como los métodos de acceso de base de datos nativos.
- Soporta conexiones desde distintos lenguajes como C/C++, Java, Perl, Python, PHP, C# y Visual Basic.
- Soporta métodos de cifrado SSL.

Licencias

El licenciamiento de Microsoft SQL Server esta disponible de las siguientes dos maneras:

- **Por Procesador:** Requiere una licencia única para cada CPU en el equipo que ejecuta e incluye el acceso limitado de clientes.
- **Servidor/por asiento:** Se requiere una licencia para el servidor y las licencias para cada cliente.

Ya terminados los putos de la parte tÃ©cnica del proyecto, hablaremos de la parte de la metodologÃa utilizada para realizar el mismo.

2.9. Metodología de desarrollo

Se le llama metodología a el conjunto de técnicas y herramientas que dan una documentación formal al proyecto referente a los procesos, las políticas y procedimientos que ayudan a los desarrolladores para la realizaron de un producto (software) .

El objetivo principal de una metodología es tener mayor eficacia al cumplir con los requisitos que se plantearon desde un principio en el proyecto, además de prevenir las pérdidas de tiempo en la realización del mismo, siendo un guÃa que llevará de manera ordenada el equipo de trabajo en el desarrollo de software con respecto a las actividades y a que el software tiene un ciclo de vida, es decir, tiene un tiempo límite para su cumplimiento.

De forma general el ciclo de vida del software está definido de los siguientes puntos:

- **Definición de objetivos:** se definen todas las ideas concretas del proyecto permitiendo tener una visión general del mismo, además de establecer un panorama amplio para la definición de alcances.
- **Análisis de requisitos y su viabilidad:** Se recopilan los requisitos del cliente y examina cualquier restricción que se pueda aplicar, es aquí donde se comienzan a visualizar los alcances del proyecto.
- **Diseño general:** Son los requisitos generales de la arquitectura del proyecto.
- **Diseño en detalle:** Es la definición precisa de cada bloque de la aplicación, esto hace referencia al detalle del funcionamiento y acciones a realizar por el sistema.
- **Programación:** Es la implementación de un lenguaje de programación para crear los modulos o funciones definidas en el proyecto.
- **Prueba de unidad:** Es la prueba individual de cada bloque para verificar que cumplen con las especificaciones de diseño y funcionalidad del proyecto.
- **Prueba beta:** Esta prueba se encarga de garantizar que el software cumple con las especificaciones originales y cumpla con todos los objetivos establecidos.
- **Documentación:** Es la guía que servirla a los usuarios que utilizaran el software a aclarar sus dudas respecto al funcionamiento del mismo en caso de usuarios finales, además de la creación de un manual técnico que servirá como guía para los encargados del soporte.
- **Implementación:** Es la puesta en marcha del proyecto para su uso, para llegar a esta parte es necesario que los puntos anteriores se evaluaran de manera satisfactoria y con esto validar que el software esta listo para ponerse en un ambiente de producción.
- **Mantenimiento:** Es la corrección de los posibles errores resultantes del uso cotidiano de los usuarios finales, estos se pueden presentar de dos maneras, errores de datos y errores de sistema.
  - Los errores de datos hacen referencia a una mala captura de información por parte del sistema o el usuario final, truncando con esto alguno de los procesos posteriores debido a información incorrecta o falta de la misma.
  - Los errores de sistema son aquellos en los que la falla es ocasionada por un fallo directamente en la programación del sistema.

Los modelos para el desarrollo de software están clasificados en tres grupos, de los cuales cada uno de ellos representa el proceso de desarrollo de software de una manera particular, y a pesar de estar definidos claramente, no representan necesariamente la realidad de cómo se debe desarrollar el software, si no que establece un enfoque común. Un modelo puede ser modificado y adaptado de acuerdo a las necesidades de software en desarrollo.

Los tres grupos de modelos de desarrollo son los siguientes:

- Modelo en cascada.
- Modelo evolutivo.

- Modelo basado en componentes.

El modelo que se utilizará para el desarrollo de este proyecto es el modelo en cascada, del cual hablaremos a continuación.

2.9.1. Modelo en cascada

Modelo conocido también como modelo clásico, modelo tradicional o modelo lineal secuencial.

Es el más básico de todos los modelos y sirve como bloque de construcción para los demás paradigmas de ciclo de vida del desarrollo de sistemas, se puede decir que es un método puro que implica un desarrollo rígido y lineal. Está basado en el ciclo convencional de una ingeniería y su visión es muy simple: el desarrollo de software se debe realizar siguiendo una secuencia de fases. Cada etapa tiene un conjunto de metas bien definidas y las actividades dentro de cada una contribuyen a la satisfacción de metas de esa.
Las fases normalmente aplicadas son:

- **Análisis del Sistema:** En esta fase se establecen los requisitos de todos los elementos del sistema y luego se asigna algún subconjunto de estos requisitos al software.
- **Análisis de requisitos del software:** El proceso de recopilación de los requisitos se centra e intensifica especialmente en el software. Es necesario comprender el ámbito de la información del software así como la función, el rendimiento y las interfaces requeridas.
- **Diseño del sistema:** Se enfoca en cuatro atributos distintos del programa:
  - La estructura de los datos.
  - La arquitectura del software.
  - El detalle procedimental.
  - La caracterización de la interfaz.
- **Diseño del programa:** El proceso de diseño traduce los requisitos en una representación del software con la calidad requerida antes de que comience la codificación
- **Codificación:** El diseño debe traducirse en una forma legible para la máquina. Si el diseño se realiza de una manera detallada, la codificación puede realizarse mecánicamente.
- **Pruebas:** Generado el código se inicia la prueba del programa. Que se centra en la lógica interna del software y en las funciones externas, realizando pruebas que aseguren que la entrada definida produce los resultados que realmente se requieren.
- **Implantación:** Significa programación. Producto de esta etapa es el código en cualquier nivel, incluido el producido por sistemas de generación automática.
- **Mantenimiento:** el software puede sufrirá algunos cambios después de que se entrega al cliente. Los cambios ocurrirán debidos a que se haya encontrado errores, a que el software deba adaptarse a cambios del entorno externo (sistema operativo o dispositivos periféricos) o a que el cliente requiera ampliaciones funcionales o del rendimiento.

Sin embargo debemos saber que los proyectos reales raramente siguen el flujo secuencial que propone el modelo. Siempre hay iteraciones y se crean problemas en la aplicación del paradigma.

Es difícil para el cliente establecer todos los requisitos explícitamente. El ciclo de vida clásico lo requiere y tiene dificultades en acomodar posibles incertidumbres que pueden existir al comienzo de muchos productos.

Por lo tanto se inicia con la especificación de requerimientos del cliente, continua con la planificación, el modelado, la construcción y el despliegue para finalizar en el enfoque del software. El modelo estÃ¡ dirigido por documentos y no proporciona resultados tangibles de software hasta el final del ciclo de vida de algunas herramientas. El diseño en cascada es una secuencia definida de los acontecimientos y los resultados finales para proporcionar una estructura para cualquier proyecto que siga el contenido específico y detallado. Puede ser apropiado para proyectos de software que son estables especialmente cuando sus requisitos no cambian. Es caracterizado por ordenar de manera rigurosa las etapas del ciclo de vida de software, dado que el comienzo de cada etapa debe esperar a la finalización de la inmediata anterior.

La metodología en cascada es esencialmente:

- El inicio y el alcance del proyecto.
- La planificación del proyecto (calendario, recursos necesarios, costo).
- Definición de las necesidades del negocio y el análisis en detalle de la solución.
- La creación de la solución.
- Prueba que la solución funciona.
- La entrega de la solución a su público objetivo.
- Cierre del proyecto.

Ventajas

- Permite el control de gestión.
- Plazos normalmente adecuados para cada etapa de desarrollo.
- Este proceso conduce a entregar el proyecto a tiempo.
- Es sencilla y facilita la gestión de proyectos.
- Permite tener bajo control el proyecto.
- Limita la cantidad de interacción entre equipos que se produce durante el desarrollo.

Desventajas

- No conocer si la solución es correcta hasta estar cerca de su lanzamiento.
- Poco tiempo para corregir fallas.
- Depuración complicada.
- Los cambios introducidos durante el desarrollo pueden confundir al equipo.
- No es frecuente que el cliente o usuario final explicite clara y completamente los requisitos.
- Es necesaria la paciencia del cliente.
- El cliente podrÃa detectar un error.
- El proceso es lento y pesado.

Probable fracaso del modelo de cascada

- Los clientes quieren ver el producto a medida que se desarrolla.
- El cliente no puede validar el producto a medida que lo desarrollamos.
- No poder tomas decisiones por la falta de avances o resultados.
- Las actividades están estrictamente ligadas causando así impedir que una actividad empiece antes determinar totalmente la anterior, provocando invertir algún tiempo de desarrollo innecesariamente.

Capítulo 3

Desarollo

Capítulo 4

Pruebas y resultados

Capítulo 5

Conclusiones

Bibliografía

[1] S. Abad Mota, "Lineamientos sobre cómo escribir informes técnicos," http://cpc.ldc.usb.ve/documentos/comoEscribirInf07.pdf [Consulta: Marzo 2013].

[2] C. Bash, J. J.R., and R. Benjamins, "What are ontologies, and why do we need them?" pp. 20–26, 1999.

[3] M. Crussière, *Étude et Optimisation de Communications à Haut-débit sur lignes d'énergie: exploitation de la combinaison OFDM/CDMA*. Institut National des Sciences Appliquées de Rennes, Rennes, Francia, 2005.

[4] Open PLC European Research Alliance, "PLC Technology license," http://www.ipcf.org [Consulta: Febrero 2011].

[5] G. Qiang and Z. Jun, "LMS Algorithms for noise Offset in Medium-Voltage Power Line Communication," *Intelligent Systems and Applications, ISA'09. International Workshop*, pp. 1–4, Mayo 2009.

[6] E. R. and N. S. B., "Fundamentals of database systems," 1999.

[7] S. Wermter, E. Riloff, and S. Gabriele, *Connectionist, Statistical and Symbolic Approaches to Learning for Natural Language Processing*. Springer-Verlag, 1996.