UNIVERSIDAD NACIONAL DE SAN CRISTÓBAL DE HUAMANGA

FACULTAD DE INGENIERÍA DE MINAS, GEOLOGÍA Y CIVIL

ESCUELA DE FORMACIÓN PROFESIONAL DE INGENIERÍA INFORMÁTICA

DISEÑO DE LA ARQUITECTURA DE SOFTWARE QUE PERMITA LA SINCRONIZACIÓN DE DATOS ENTRE LOS SISTEMAS COMERCIAL Y CONTABLE DEL ESTUDIO NAVARRO Y ASOCIADOS, AYACUCHO 2013.

TESIS PRESENTADA POR:

BACH. JENY CASTELLARES PÉREZ

PARA OPTAR EL TÍTULO PROFESIONAL DE

INGENIERO INFORMÁTICO

AYACUCHO, 30 DE DICIEMBRE DE 2014

DISEÑO DE LA ARQUITECTURA DE SOFTWARE QUE PERMITA LA SINCRONIZACIÓN DE DATOS ENTRE LOS SISTEMAS COMERCIAL Y CONTABLE DEL ESTUDIO NAVARRO Y ASOCIADOS, AYACUCHO 2013.

ING. JUAN C. CARREÑO GAMARRA
PRESIDENTE

ING. ELINAR CARRILLO RIVEROS
MIEMBRO

ING. EDITH V. GUEVARA MOROTE
MIEMBRO

ING. FLORO A. XANGALI GUERRA
SECRETARIO DOCENTE

DISEÑO DE LA ARQUITECTURA DE SOFTWARE QUE PERMITA LA SINCRONIZACIÓN DE DATOS ENTRE LOS SISTEMAS COMERCIAL Y CONTABLE DEL ESTUDIO NAVARRO Y ASOCIADOS, AYACUCHO 2013.

ING. JUAN C. CARREÑO GAMARRA
PRESIDENTE

ING. ELINAR CARRILLO RIVEROS
MIEMBRO

ING. EDITH P. GUEVARA MOROTE
MIEMBRO

ING. FLORO N. XANGALI GUERRA
SECRETARIO DOCENTE

A mi familia, que es apoyo e impulso continuo.

AGRADECIMIENTO

A Dios por permitir mi formación profesional y a mi alma máter, que me dio conocimiento y habilidades, transmitidos concretamente a través de sus docentes. Estas cualidades me prepararon para ser capaz de proponer soluciones tecnológicas ante problemas del entorno social.

RESUMEN

Este trabajo de investigación plantea que al aplicar el diseño de la arquitectura de software apropiado permite la sincronización de datos entre el Sistema Comercial @NTO y la Base de Datos del Estudio Contable Navarro & Asociados, manteniendo la integridad de los datos, la independencia de las base de datos de ambos sistemas y la ejecución automática de este proceso.

Con la aplicación de conceptos de sistemas distribuidos, patrones de arquitectura, comunicación de datos y sistemas de base de datos se logra alcanzar el objetivo planteado, En el desarrollo del presente trabajo se observa que la conexión a internet en la localidad no es constante, que los sistemas que se buscan comunicar son independientes y no se pueden acceder a su código fuente sino solamente a sus respectivas bases de datos.

Como resultado se obtiene la sincronización de los sistemas, mediante el consumo del servicio web, asegurando la integridad de datos, y modificando las entidades dentro del Sistema Comercial @NTO, para controlar los registros sincronizados.

CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Este trabajo se enfoca en el diseño de la arquitectura de una solución que permita la sincronización automática de datos entre dos sistemas. Es así que se tienen los servicios Windows como iniciadores automáticos de tareas, los servicios web que logran la comunicación entre sistemas (sistemas distribuidos) a través de internet; uso de los sistemas gestores de base de datos transaccionales para asegurar la integridad de datos y herramientas de desarrollo de software que soporten las transacciones. Ítems desarrollados en los capítulos 1 y 2.

Para lograr el objetivo se realiza la investigación de tipo aplicada, con un alcance exploratorio; teniendo como diseño de investigación el no experimental del tipo transeccional exploratorio. Se define la población y muestra; variables e indicadores; medios e instrumentos. Se muestran los resultados obtenidos y las discusiones surgidas dentro del capítulo 3.

Finalmente en el capítulo 4, se encuentran las conclusiones y recomendaciones de la investigación.

CAPÍTULO I
PLANTEAMIENTO DE LA INVESTIGACIÓN

1.1 DIAGNÓSTICO Y ENUNCIADO DEL PROBLEMA

El estudio contable impulsa el uso del Sistema Comercial @NTO, que contiene una Base de Datos Local, éste permite exportar los datos de los movimientos de ventas, que son importados desde el Sistema Contable del estudio contable, para su posterior procesamiento. Los datos se recolectan desde las instalaciones de cada cliente; esta acción demanda asignar determinado tiempo y recursos, que serán costos fijos del estudio contable. Se espera sincronizar esta información automáticamente entre estos dos sistemas.

1.2 DEFINICIÓN DEL PROBLEMA

1.2.1 PROBLEMA PRINCIPAL

¿Cómo lograr la sincronización de datos entre el sistema comercial @NTO y la Base de Datos del Estudio Contable?

1.2.2 PROBLEMAS ESPECÍFICOS

a. ¿Cómo lograr la comunicación el sistema comercial @NTO y la Base de Datos del Estudio Contable?
b. ¿Cómo lograr la verificación de la conexión a internet, para el envío de datos?
c. ¿Cómo lograr la transferencia de datos asegurando su integridad?
d. ¿Qué modificaciones se realizarán dentro del sistema comercial @NTO y el motor de base de datos en el Estudio Contable para tener identificados los datos sincronizados o por sincronizar?

1.3 OBJETIVO GENERAL

Diseñar la Arquitectura de Software de un componente que permita la sincronización de datos a través de internet entre el Sistema Comercial @NTO y la Base de Datos del Estudio Contable.

1.4 OBJETIVOS ESPECÍFICOS

a. Aplicar los Patrones de Diseño de Arquitectura de Software que hagan posible la comunicación entre el sistema comercial @NTO y la Base de Datos del Estudio Contable; específicamente la Arquitectura Orientada a Servicios (SOA por sus siglas en inglés).
b. Programar en el componente la verificación de la conexión a internet para la transmisión automática de datos, utilizando el Administrador de excepciones de la herramienta de desarrollo de software que será ejecutado por un servicio Windows.
c. Transferir los datos dentro de transacciones que aseguren su integridad, en archivos XML y con bases de datos transaccionales.
d. Modificar las entidades de la Base de Datos del sistema comercial @NTO y de la Base de Datos del Estudio Contable, manteniendo la independencia, para que se tengan identificados los datos sincronizados o por sincronizar, usando una bandera que identifique el estado de los datos a transmitir.

1.5 JUSTIFICACIÓN Y DELIMITACIÓN DE LA INVESTIGACIÓN

1.5.1 JUSTIFICACIÓN

Se quiere evitar los gastos y costos de la recolección de datos implementando la sincronización de datos de las Bases de Datos con el objetivo de que programáticamente los datos de los movimientos diarios sean transmitidos a la base de datos del Estudio Contable (Base de Datos Central) a través del internet, cuando la conexión esté disponible. Y

después de su revisión, si es que hubiera cambios, esta información será actualizada en la Base de Datos Local. Asegurando la integridad de datos.

Si mientras se envían los datos la conexión a internet se interrumpe, entonces debe estar plenamente identificado el último dato transmitido satisfactoriamente y cuando la conexión se haya re establecido se debe proseguir con el envío del resto de datos.

El tener la información del cliente en la Base de Datos Central permitirá agilizar el procesamiento, con la seguridad de que los datos disponibles son veraces e íntegros. Si éstos fueran modificados entonces serán retornados a la Base de Datos Local para que el cliente tenga la información actualizada.

1.5.2 DELIMITACIÓN

La investigación se realizará en el Estudio Contable Navarro y Asociados, para sus clientes que usen el sistema comercial @NTO, en el año 2013.

1.6 **DEFINICIÓN DE LA HIPÓTESIS**

1.6.1 HIPÓTESIS GENERAL

Si se aplica el diseño de la Arquitectura de Software del componente se logrará la sincronización de datos entre los Sistemas comercial y contable del Estudio Navarro y Asociados.

1.6.2 HIPÓTESIS ESPECÍFICAS

a. La elección apropiada de los Patrones para el Diseño de la Arquitectura de Software del componente de sincronización de datos permitirán lograr la comunicación entre el Sistema comercial @NTO y la Base de Datos del Estudio Contable.
b. El componente de sincronización de datos verificará que la conexión a internet esté disponible para la transmisión de datos.
c. La transferencia de datos dentro de bloques de transacciones asegurará la integridad de los datos.

d. La independencia de la base de datos permitirá la modificación de las entidades, teniendo identificados los datos sincronizados o por sincronizar.

CAPÍTULO II
MARCO TEÓRICO

2.1 MARCO TEÓRICO

2.1.1 APLICACIONES DISTRIBUIDAS

En el mundo distribuido los términos como "sistema distribuido", "programación distribuida" y "algoritmo distribuido" originalmente referido a las redes de computadoras donde las computadoras individuales fueron distribuidas físicamente dentro de un área geográfica. (Lynch, 1996) Los términos son hoy en día usados en muchos sentidos, referidos a procesos autónomos que se ejecutan en una misma computadora física e interactúan con otras a través del envío de mensajes. (Andrews, 2000) Mientras que no exista una única definición de un sistema distribuido, las siguientes propiedades definidas son comúnmente usadas:

- Existen muchas entidades computacionales autónomas, cada cual tiene su propia memoria local. (Lynch, 1996)
- Las entidades se comunica con otras por medio de mensajes. (Andrews, 2000)

Las entidades computacionales son llamadas computadoras o nodos.

Un sistema distribuido puede tener un objetivo común, tal como resolver un gran problema computacional. Alternativamente, cada computadora puede tener su propio usuario con necesidades individuales, y el propósito del sistema distribuido es coordinar el uso de recursos compartidos o proveer servicios de comunicación a los usuarios. (Ghosh, 2007)

Otras propiedades típicas de los sistemas distribuidos incluyen lo siguiente:

- El sistema tiene que tolerar fallas en computadoras individuales. (Lynch, 1996)
- La estructura del sistema (topología de la red, latencia de la red, número de computadoras) no se conocen de antemano, el sistema podría consistir en diferentes tipos de computadoras y enlaces de redes, y el sistema podría cambiar durante la ejecución de un programa distribuido. (Lynch, 1996)
- Cada computadora tiene una vista limitada o incompleta del sistema. Cada computadora puede conocer solo una parte del dato de ingreso. (Ghosh, 2007)

La computación distribuida es un campo de las ciencias de la computación que estudia los sistemas distribuidos. Un sistema distribuido consiste en múltiples computadoras autónomas que se comunican a través de una red de computadoras. Las computadoras interactúan con cada uno en orden para cumplir un objetivo común. Un programa de computadora que ejecuta un sistema distribuido es llamado programa distribuido, y la programación distribuida es el proceso de escribir tales programas. (Andrews, 2000)

La computación Distribuida también se refiere al uso de los sistemas distribuidos para resolver problemas computacionales. En la computación distribuida, un problema es dividido en muchas tareas, cada una de las cuales es resuelta por una o más computadoras, (Godfrey, 2002) las cuales se comunican mutuamente a través del envío de mensajes. (Andrews, 2000)

2.1.2 Patrones de arquitectura de software

Los patrones arquitectónicos, o patrones de arquitectura, son patrones de diseño de software que ofrecen soluciones a problemas de arquitectura de software en ingeniería de software. Dan una descripción de los elementos y el tipo de relación que tienen junto con un conjunto de restricciones

sobre cómo pueden ser usados. Un patrón arquitectónico expresa un esquema de organización estructural esencial para un sistema de software, que consta de subsistemas, sus responsabilidades e interrelaciones. En comparación con los patrones de diseño, los patrones arquitectónicos tienen un nivel de abstracción mayor. (Avgeriou & Uwe, 2005)

Aunque un patrón arquitectónico comunica una imagen de un sistema, no es una arquitectura como tal. Un patrón arquitectónico es más un concepto que captura elementos esenciales de una arquitectura de software. Muchas arquitecturas diferentes pueden implementar el mismo patrón y por lo tanto compartir las mismas características. Además, los patrones son a menudo definidos como una cosa "estrictamente descrita y comúnmente disponible". Por ejemplo, la arquitectura en capas es un estilo de llamamiento-y-regreso, cuando define uno un estilo general para interaccionar. Cuando esto es descrito estrictamente y comúnmente disponible, es un patrón. (Bass, Clements, & Kazman, 2005)

Uno de los aspectos más importantes de los patrones arquitectónicos es que encarnan diferentes atributos de calidad. Por ejemplo, algunos patrones representan soluciones a problemas de rendimiento y otros pueden ser utilizados con éxito en sistemas de alta disponibilidad. A primeros de la fase de diseño, un arquitecto de software escoge qué patrones arquitectónicos mejor ofrecen las cualidades deseadas para el sistema. (Buschamann, Meunier, Rohnert, Sommerlad, & Stal, 1996)

Ejemplos de patrones arquitectónicos incluyen los siguientes:

- Programación por capas
- Tres niveles
- Pipeline
- Invocación implícita

- Arquitectura en pizarra
- Arquitectura dirigida por eventos, Presentación-abstracción-control
- Peer-to-peer
- Arquitectura orientada a servicios
- Objetas desnudos
- Modelo Vista Controlador

Dominios en el diseño de Patrones

Control de acceso:

Hay muchas situaciones en las cuales el acceso a datos, características y funcionalidad son limitadas a la definición de los usuarios. Desde un punto de vista arquitectónico, acceder a determinadas partes del software debe tener un riguroso control.

Concurrencia:

Muchas aplicaciones deben manejar múltiples tareas de forma que simule el paralelismo. Hay muchas formas de manejar esta concurrencia, y cada una puede ser presentada por un patrón arquitectónico diferente.

Distribución:

El problema de distribución dirige el problema de forma en que los sistemas o componentes se comunican con otros en un entorno distribuido. El patrón más común para afrontar el problema es "the broker". Actuando como un "middleman" entre el componente cliente y el servidor. El cliente envía un mensaje al "broker" y éste se encarga de completar la conexión.

Persistencia:

Los datos persistentes son almacenados en bases de datos o archivos y pueden ser leídos o modificados por otros procesos más adelante. En los

entornos orientados a objetos esto va más allá, y lo que puede ser accedido o modificable son las propiedades de los objetos.

2.1.3 Patrones de diseño de software

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces.

Un patrón de diseño resulta ser una solución a un problema de diseño. Para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser reutilizable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias. (Gamma, Helm, Johnson, & Vlissides, 1995)

2.1.3.1 Objetivos de los patrones

Los patrones de diseño pretenden:

- Proporcionar catálogos de elementos reusables en el diseño de sistemas software.
- Evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente.
- Formalizar un vocabulario común entre diseñadores.
- Estandarizar el modo en que se realiza el diseño.
- Facilitar el aprendizaje de las nuevas generaciones de diseñadores condensando conocimiento ya existente.

Asimismo, no pretenden:

- Imponer ciertas alternativas de diseño frente a otras.
- Eliminar la creatividad inherente al proceso de diseño.

2.1.3.2 Categorías de patrones

Según la escala o nivel de abstracción:

- Patrones de arquitectura: Aquellos que expresan un esquema organizativo estructural fundamental para sistemas de software.
- Patrones de diseño: Aquellos que expresan esquemas para definir estructuras de diseño (o sus relaciones) con las que construir sistemas de software.
- Dialectos: Patrones de bajo nivel específicos para un lenguaje de programación o entorno concreto.

Además, también es importante reseñar el concepto de "anti patrón de diseño", que con forma semejante a la de un patrón, intenta prevenir contra errores comunes de diseño en el software. La idea de los anti patrones es dar a conocer los problemas que acarrean ciertos diseños muy frecuentes, para intentar evitar que diferentes sistemas acaben una y otra vez en el mismo callejón sin salida por haber cometido los mismos errores. (Brown, Malveau, Mc Cormick III, & Mowbray, 1998)

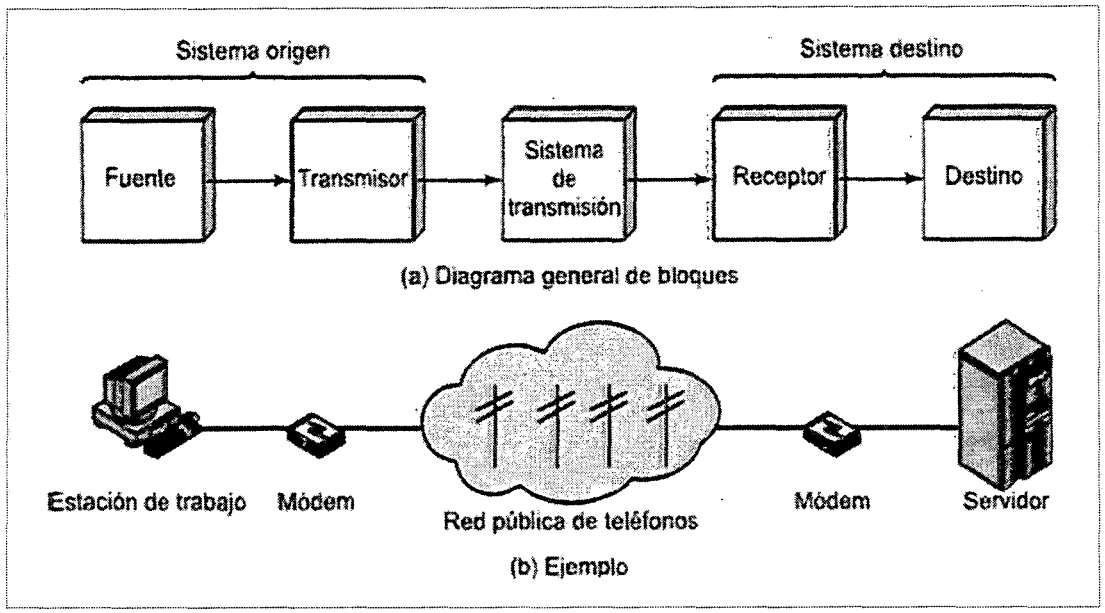
Además de los patrones ya vistos actualmente existen otros patrones como el siguiente:

- Interacción: Son patrones que nos permiten el diseño de interfaces web.

2.1.4 Comunicación de Datos

Comenzaremos nuestro estudio considerando el modelo sencillo de sistema de comunicación mostrado en la Figura II-1, en la que se propone un diagrama de bloques.

FIGURA II-1 MODELO SIMPLIFICADO PARA LAS COMUNICACIONES



Fuente: (Stallings, 2004, pág. 11)

El objetivo principal de todo sistema de comunicaciones es intercambiar información entre dos entidades. Figura II-2 muestra un ejemplo particular de comunicación entre una estación de trabajo y un servidor a través de una red telefónica pública. Otro posible ejemplo consiste en el intercambio de señales de voz entre dos teléfonos a través de la misma red anterior. Los elementos clave en este modelo son los siguientes:

- **La fuente.** Este dispositivo genera los datos a transmitir. Ejemplos de fuentes pueden ser un teléfono o un computador personal.
- **El transmisor.** Normalmente los datos generados por la fuente no se transmiten directamente tal y como son generados. Al contrario, el transmisor transforma y codifica la información, generando señales electromagnéticas susceptibles de ser transmitidas a través de algún sistema de transmisión. Por ejemplo, un módem convierte las cadenas de bits generadas por un computador personal y las transforma en señales analógicas que pueden ser transmitidas a través de la red de telefonía.

- **El sistema de transmisión.** Puede ser desde una sencilla línea de transmisión hasta una compleja red que conecte a la fuente con el destino.
- **El receptor.** El receptor acepta la señal proveniente del sistema de transmisión y la transforma de tal manera que pueda ser manejada por el dispositivo de destino. Por ejemplo, un módem captará la señal analógica de la red o línea de transmisión y la convertirá en una cadena de bits.
- **El destino.** Toma los datos del receptor

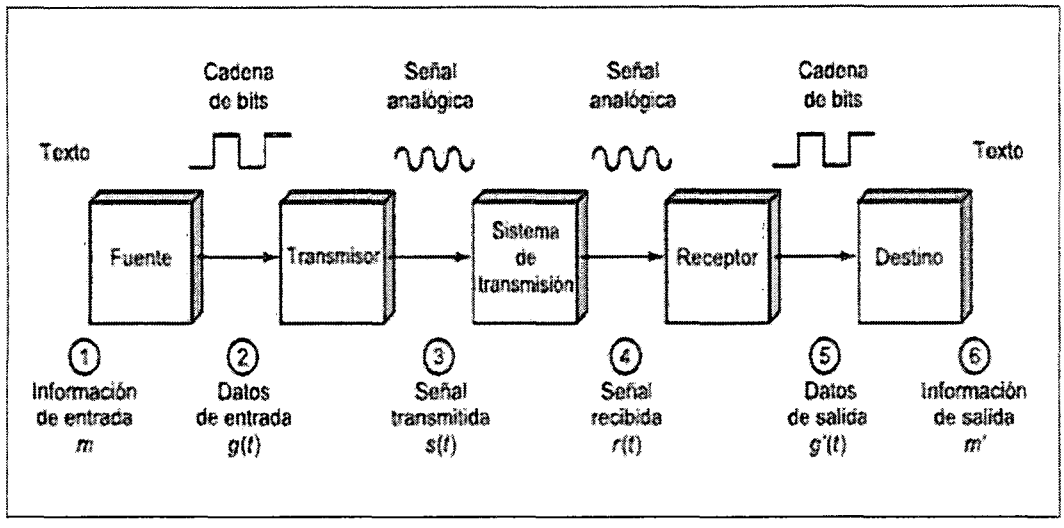
Aunque el modelo presentado pueda parecer aparentemente sencillo, en realidad implica una gran complejidad. (Stallings, 2004, pág. 11)

Para explicar todos los conceptos abordados en Comunicación de datos, la Figura II-2 muestra una perspectiva novedosa del modelo tradicional para las comunicaciones de la Figura II-1. Dicha figura se explica a continuación, paso a paso, con la ayuda de un ejemplo: la aplicación de correo electrónico. Supóngase que tanto el dispositivo de entrada como el transmisor están en un computador personal. Y que, por ejemplo, el usuario de dicho PC desea enviar el mensaje *m* a otro. El usuario activa la aplicación de correo en el PC y compone el mensaje con el teclado (dispositivo de entrada). La cadena de caracteres se almacenará temporalmente en la memoria principal como una secuencia de bits (*g*). El computador se conecta a algún medio de transmisión, por ejemplo una red local o una línea de telefonía, a través de un dispositivo de *E/S* (transmisor), como por ejemplo un transceptor en una red local o un módem. Los datos de entrada se transfieren al transmisor como una secuencia de niveles de tensión *[g(t)]* que representan los bits en algún tipo de bus de comunicaciones o cable. El transmisor se conecta directamente al medio y convierte la cadena *[g(t)]* en la señal a transmitir *[s(t)]*.

Al transmitir *s(t)* a través del medio, antes de llegar al receptor. Por tanto, la señal recibida *r(t)* puede diferir de alguna manera de la transmitida *s(t)*. El receptor intentará estimar la señal original *s(t)*, a partir de la señal *r(t)* y de su conocimiento acerca del medio, obteniendo una secuencia de bits *g'(t)*. Estos bits se envían al computador de salida, donde se almacenan temporalmente en memoria como un bloque de bits *g'*. En muchos casos, el destino intentará determinar si ha ocurrido un error y, en su caso, cooperar con el origen para, eventualmente, conseguir el bloque de datos completo y sin errores. Los datos, finalmente, se presentan al usuario a través del dispositivo de salida, que por ejemplo puede ser la impresora o la pantalla de su terminal. El mensaje recibido por el usuario *(m')* será normalmente una copia exacta del mensaje original *(m)*.

Consideremos ahora una conversación usando el teléfono. En este caso, la entrada al teléfono es un mensaje *(m)* consistente en una onda sonora. Dicha onda se convierte en el teléfono en señales eléctricas con los mismos componentes en frecuencia. Estas señales se transmiten sin modificación a través de la línea telefónica. Por tanto, la señal de entrada *g(t)* y la señal transmitida *s(t)* son idénticas. La señal *s(t)* sufrirá algún tipo de distorsión a través del medio, de tal manera que *r(t)* no será idéntica a *s(t)*. No obstante, la señal *r(t)* se convierte recuperando una onda sonora, sin aplicar ningún tipo de corrección o mejora de la calidad. Por tanto, *m'* no será una réplica exacta de *m*. Sin embargo, el mensaje sonoro recibido es normalmente comprensible por el receptor. En el ejemplo anterior no se han considerado otros aspectos fundamentales en las comunicaciones de datos, como lo son las técnicas de control del enlace, necesarias para regular el flujo de datos, o como la detección y corrección de errores. Tampoco se han considerado las técnicas de multiplexación, necesarias para conseguir una utilización eficaz del medio de transmisión. (Stallings, 2004, págs. 13, 14)

FIGURA II-2 MODELO SIMPLIFICADO PARA LAS COMUNICACIONES DE DATOS



Fuente (Stallings, 2004, pág. 14)

2.1.5 SISTEMAS DE GESTIÓN DE BASE DE DATOS

Definimos un **Sistema Gestor de Bases de Datos** o **SGBD**, también llamado DBMS (Data Base Management System, por sus siglas en inglés) como una colección de datos relacionados entre sí, estructurados y organizados, y un conjunto de programas que acceden y gestionan esos datos. La colección de esos datos se denomina Base de Datos o BD, (DB DataBase).

Antes de aparecer los SGBD (década de los setenta). La información se trataba y se gestionaba utilizando los típicos sistemas de gestión de archivos que iban soportados sobre un sistema operativo. Éstos consistían en un conjunto de programas que definían y trabajaban sus propios datos. Los datos se almacenan en archivos y los programas manejan esos archivos para obtener la información. Si la estructura de los datos de los archivos cambia, todos los programas que los manejan se deben modificar; por ejemplo, un programa trabaja con un archivo de datos de alumnos, con una estructura o registro ya definido; si se incorporan elementos o campos a la estructura del archivo, los programas que utilizan

ese archivo se tienen que modificar para tratar esos nuevos elementos. En estos sistemas de gestión de archivos, la definición de los datos se encuentra codificada dentro de los programas de aplicación en lugar de almacenarse de forma independiente, y además el control de acceso y la manipulación de los datos vienen impuestos por los programas de aplicación.

Esto supone un gran inconveniente a la hora de tratar grandes volúmenes de información. Surge así la idea de separar los datos contenidos en los archivos de los programas que los manipulan, es decir, que se pueda modificar la estructura de los datos de los archivos sin que por ello se tengan que modificar los programas con los que trabajan. Se trata de estructurar y organizar los datos de forma que se pueda acceder a ellos con independencia de los programas que los gestionan.

Inconvenientes de un sistema de gestión de archivos:

- Redundancia e inconsistencia de los datos.
- Dependencia de los datos física-lógica.
- Dificultad para tener acceso a los datos.
- Separación y aislamiento de los datos.
- Dificultad para el acceso concurrente.
- Dependencia de la estructura del archivo con el lenguaje de programación.
- Problemas en la seguridad de los datos.
- Problemas en la integridad de datos.

Todos estos inconvenientes hacen posible el fomento y desarrollo de SGBD. El objetivo primordial de un gestor es proporcionar eficiencia y seguridad a la hora de extraer o almacenar información en las BD. Los sistemas gestores de BD están diseñados para gestionar grandes bloques

de información, que implica tanto la definición de estructuras para el almacenamiento como de mecanismos para la gestión de la información.

Un BD es un gran almacén de datos que se define una sola vez; los datos pueden ser accedidos de forma simultánea por varios usuarios; están relacionados y existe un número mínimo de duplicidad; además en las BD se almacenarán las descripciones de esos datos, lo que se llama metadatos en el diccionario de datos.

El SGBD es una aplicación que permite a los usuarios definir, crear y mantener la BD y proporciona un acceso controlado a la misma. Debe prestar los siguientes servicios:

- Creación y definición de la BD: especificación de la estructura, el tipo de los datos, las restricciones y relaciones entre ellos mediante lenguajes de definición de datos. Toda esta información se almacena en el diccionario de datos, el SGBD proporcionará mecanismos para la gestión del diccionario de datos.
- Manipulación de los datos realizando consultas, inserciones y actualizaciones de los mismos utilizando lenguajes de manipulación de datos.
- Acceso controlado a los datos de la BD mediante mecanismos de seguridad de acceso a los usuarios.
- Mantener la integridad y consistencia de los datos utilizando mecanismos para evitar que los datos serán perjudicados por cambios no autorizados.
- Acceso compartido a la BD, controlando la interacción entre usuarios concurrentes.
- Mecanismos de respaldo y recuperación para restablecer la información en caso de fallos en el sistema.

  (Facultad Ingeniería de Sistemas, 2014, págs. 7 - 9)

2.2 DEFINICIÓN DE TERMINOS

2.2.1 WINDOWS COMMUNICATION FOUNDATION

Windows Communication Foundation (WCF) es un marco de trabajo para la creación de aplicaciones orientadas a servicios. Con WCF, es posible enviar datos como mensajes asincrónicos de un extremo de servicio a otro. Un extremo de servicio puede formar parte de un servicio disponible continuamente hospedado por IIS, o puede ser un servicio hospedado en una aplicación. Un extremo puede ser un cliente de un servicio que solicita datos de un extremo de servicio. Los mensajes pueden ser tan simples como un carácter o una palabra que se envía como XML, o tan complejos como una secuencia de datos binarios.

Características de WCF

WCF incluye el siguiente conjunto de características:

2.2.1.1 ORIENTACIÓN A SERVICIOS

Como consecuencia del uso de los estándares de WS, WCF le permite crear aplicaciones orientadas a servicios. SOA, la arquitectura orientada a servicios es el uso de servicios web para enviar y recibir datos. Los servicios tienen la ventaja general de estar débilmente acoplados entre una aplicación y otra en lugar de incluidos en el código. Una relación de acoplamiento débil implica que cualquier cliente creado en cualquier plataforma puede conectar con cualquier servicio siempre y cuando se cumplan los contratos esenciales.

2.2.1.2 INTEROPERABILIDAD

WCF implementa los estándares del sector modernos para la interoperabilidad de servicios web.

2.2.1.3 VARIOS MODELOS DE MENSAJES

Los mensajes se intercambian mediante uno de los distintos modelos. El más común es el de solicitud/respuesta, en que un extremo solicita datos de otro extremo y el otro extremo responde. Existen otros modelos, como un mensaje unidireccional, en que un único extremo envía un mensaje sin esperar ninguna respuesta. Un modelo más complejo es el modelo de intercambio dúplex donde dos extremos establecen una conexión y envían datos hacia delante y hacia atrás, similar a un programa de mensajería instantánea.

2.2.1.4 Metadatos de servicios

WCF admite la publicación de metadatos de servicios utilizando los formatos especificados en los estándares de la industria, como WSDL, Esquemas XML y WS-Policy. Estos metadatos pueden utilizarse para generar y configurar automáticamente clientes para el acceso a los servicios de WCF. Los metadatos se pueden publicar sobre HTTP y HTTPS, o utilizando el estándar Intercambio de metadatos de servicios web.

2.2.1.5 Contratos de datos

Dado que WCF se basa en .NET Framework, también incluye métodos con código sencillo para proporcionar los contratos que desea aplicar. Uno de los tipos de contrato universales es el contrato de datos. Básicamente, mientras se escribe el código del servicio usando Visual C# o Visual Basic, la forma más sencilla de controlar los datos consiste en crear clases que representan una entidad de datos con propiedades que pertenecen a la misma. WCF incluye un completo sistema para trabajar con datos de esta manera fácil. Cuando se han creado las clases que representan los datos, el servicio genera automáticamente los metadatos que permiten a los clientes ajustarse a los tipos de datos que se han diseñado.

2.2.1.6 Seguridad

Es posible cifrar los mensajes para proteger la privacidad, así como obligar a los usuarios a que se autentiquen antes de permitirles recibir mensajes.

La seguridad puede implementarse utilizando estándares conocidos como SSL o WS-Secure Conversation.

2.2.1.7 Varios transportes y codificaciones

Los mensajes pueden enviarse con cualquiera de los protocolos y codificaciones integrados. La combinación más frecuente de protocolo y codificación consiste en enviar mensajes SOAP codificados de texto utilizando el Protocolo de transferencia de hipertexto (HTTP) usado en World Wide Web. WCF también le permite enviar mensajes sobre TCP, canalizaciones con nombre o MSMQ. Estos mensajes pueden codificarse como texto o utilizando un formato binario optimizado. Los datos binarios pueden enviarse de manera eficaz utilizando el estándar MTOM. Si ninguno de los transportes o codificaciones proporcionados satisface sus necesidades, puede crear uno personalizado.

2.2.1.8 Mensajes confiables y en cola

WCF admite intercambio de mensajes confiable usando sesiones confiables implementadas sobre mensajería WS-Reliable y mediante MSMQ.

2.2.1.9 Mensajes duraderos

Un mensaje duradero es aquel que nunca se pierde debido a una interrupción de la comunicación. Los mensajes que forman parte de un modelo de mensajes duraderos siempre se guardan en una base de datos. Si se produce una interrupción, la base de datos le permite reanudar el intercambio de mensajes cuando se restablezca la conexión. También puede crear un mensaje duradero utilizando Windows Workflow Foundation (WWF).

2.2.1.10 Transacciones

WCF también admite las transacciones que usan uno de los tres modelos de transacción: las transacciones WS-Atomic, las API del espacio de

nombres System.Transactions y Coordinador de transacciones distribuidas de Microsoft.

2.2.1.11 Compatibilidad con AJAX y REST

REST es un ejemplo de una tecnología de la Web 2.0 en evolución. WCF se puede configurar para procesar datos XML "sin formato" que no se ajustan en un sobre SOAP. WCF también se puede extender para admitir formatos XML concretos, como ATOM (un estándar popular de RSS), e incluso formatos no XML, como notación de objetos JavaScript (JSON).

2.2.1.12 Extensibilidad

La arquitectura de WCF tiene varios puntos de extensibilidad. Si se necesita una función adicional, existen una serie de puntos de entrada que le permiten personalizar el comportamiento de un servicio.

2.2.2 Enterprise Application Integration

Los componentes de integración, también llamado Enterprise Application Integration (EAI) son una aproximación a la integración de los sistemas, diseñados para integrar software empresarial de diferentes proveedores, dado que la industria requiere el mejor software para áreas específicas de su negocio.

Los software de integración de talla mundial disponibles en el mercado son los siguientes:

- **Microsoft BizTalk Server**, es una plataforma de integración e interconexión de procesos de negocio. Por medio del uso de adaptadores diseñados para comunicarse con diferentes tipos de software usados en una empresa de **gran tamaño.** Permitiendo a las compañías automatizar e integrar los procesos de negocio (sistemas corporativos) heterogéneos.
- **Oracle Enterprise Application Integration Services**, provee un conjunto de soluciones basadas en los principios de una

arquitectura orientada a servicios para implementar las soluciones de integración. Estos componentes permiten únicamente la integración de los productos de la Suite de Oracle, lo que significa una limitante, dado que las necesidades de la industria son la integración de diferentes aplicaciones incluso entre diferentes tecnologías.

2.2.3 Arquitectura Orientada a Servicios

La arquitectura orientada a servicios de cliente (en inglés Service-Oriented Architecture), es un concepto de arquitectura de software que define la utilización de servicios para dar soporte a los requisitos del negocio.

Permite la creación de sistemas de información altamente escalables que reflejan el negocio de la organización, a su vez brinda una forma bien definida de exposición e invocación de servicios (comúnmente pero no exclusivamente servicios web), lo cual facilita la interacción entre diferentes sistemas propios o de terceros.

SOA define las siguientes capas de software:

- Aplicaciones básicas - Sistemas desarrollados bajo cualquier arquitectura o tecnología, geográficamente dispersos y bajo cualquier figura de propiedad;
- De exposición de funcionalidades - Donde las funcionalidades de la capa aplicativa son expuestas en forma de servicios (generalmente como servicios web);
- De integración de servicios - Facilitan el intercambio de datos entre elementos de la capa aplicativa orientada a procesos empresariales internos o en colaboración;
- De composición de procesos - Que define el proceso en términos del negocio y sus necesidades, y que varía en función del negocio;

- De entrega - donde los servicios son desplegados a los usuarios finales.

SOA proporciona una metodología y un marco de trabajo para documentar las capacidades de negocio y puede dar soporte a las actividades de integración y consolidación.

2.2.3.1 Diseño y desarrollo de SOA

La metodología de modelado y diseño para aplicaciones SOA se conoce como análisis y diseño orientado a servicios. La arquitectura orientada a servicios es tanto un marco de trabajo para el desarrollo de software como un marco de trabajo de implementación. Para que un proyecto SOA tenga éxito los desarrolladores de software deben orientarse ellos mismos a esta mentalidad de crear servicios comunes que son orquestados por clientes o middleware para implementar los procesos de negocio. El desarrollo de sistemas usando SOA requiere un compromiso con este modelo en términos de planificación, herramientas e infraestructura. (Bieberstein, Bose, Fianmante, Jones, & Shah, 2005)

Cuando la mayoría de la gente habla de una arquitectura orientada a servicios están hablando de un juego de servicios residentes en Internet o en una intranet, usando servicios web. Existen diversos estándares relacionados a los servicios web. Incluyen los siguientes:

- XML
- HTTP
- SOAP
- REST
- WSDL
- UDDI

En un ambiente SOA, los nodos de la red hacen disponibles sus recursos a otros participantes en la red como servicios independientes a los que

tienen acceso de un modo estandarizado. La mayoría de las definiciones de SOA identifican la utilización de Servicios Web (empleando SOAP y WSDL) en su implementación, no obstante se puede implementar SOA utilizando cualquier tecnología basada en servicios. (Bieberstein, Bose, Fianmante, Jones, & Shah, 2005)

2.2.3.2 Beneficios

Los beneficios que puede obtener una organización que adopte SOA son:

- Mejora en los tiempos de realización de cambios en procesos.
- Facilidad para evolucionar a modelos de negocios basados en tercerización.
- Facilidad para abordar modelos de negocios basados en colaboración con otros entes (socios, proveedores).
- Poder para reemplazar elementos de la capa aplicativa SOA sin disrupción en el proceso de negocio.
- Facilidad para la integración de tecnologías disímiles.

2.2.4 Smart Clients

Smart client (traducido del inglés, cliente inteligente) es un tecnicismo usado en el desarrollo de software. Generalmente se refiere a aplicaciones que:

- Son entregadas sobre la red (internet).
- No requieren instalación (o proveen una instalación y actualizaciones automáticas).
- Actualizadas automáticamente sin intervención del usuario.
- Tienen "Look and Feel" (aspecto) de aplicación de escritorio.
- Aprovechan los recursos de hardware y software del computador donde son ejecutados

El término Cliente Inteligente tiene la intención de referirse simultáneamente a la captura de los beneficios de un Cliente liviano (cero

instalaciones, auto-actualizaciones) y un Cliente pesado (alta presentación, alta productividad).

Los términos "Rich Internet Application" (RIA) y "Rich Web Application" (Aplicaciones Ricas por Internet o Web) son usados para referirse al aprovechamiento de otras tecnologías, incluyendo Flash y Applets de Java.

Las aplicaciones Cliente Inteligente estrecharon las diferencias entre las aplicaciones web y las aplicaciones de escritorio. Proveyeron los beneficios de una aplicación web (tales como apoyarse en Internet y ofrecer acceso remoto a los datos) mientras siguen proveyendo ese aspecto vivaz, productivo y elegante inherente a las aplicaciones de escritorio.

2.2.5 XML (EXTENDED MARKUP LANGUAGE).

Es un lenguaje (o Metalenguaje) de marcas creado para resolver las limitaciones de HTML. Los lenguajes de marcas se crearon ante la necesidad de utilizar un formato estándar para representar la información. HTML se definió para soportar un sistema de hipertexto distribuido, que fuera portable y fácil de aprender. El problema es que HTML es un lenguaje de marcas fijo y con un conjunto limitado de marcas. Resulta útil para formatear un documento pero no permite informar sobre la semántica de la información codificada en él, no indica lo que está representando, se preocupa principalmente de que algo se visualice con unas características determinadas, pero no te dice que lo que está mostrando es el título de un libro o el precio de un artículo.

El lenguaje XML hace precisamente esto, describe el contenido de lo que etiqueta (Golfarb & Prescod, 1999). La potencia de esta forma de trabajar radica en que se está etiquetando e identificando el contenido, olvidando en un principio la forma de presentarlo. Esto entre otras cosas agilizará el intercambio de información y la cooperación entre las empresas facilitando el comercio electrónico.

Tanto XML como HTML tienen su base en el SGML. El SGML (Standard Generalized Markup Language) es el estándar internacional para la definición de la estructura y el contenido de diferentes tipos de documentos electrónicos, es un metalenguaje de marcas (un lenguaje utilizado para crear nuevos lenguajes de marcas). Por ejemplo HTML, es un lenguaje de marcas que ha sido creado utilizando SGML. El problema es que SGML es complicado, difícil de aprender y bastante complejo.

XML es una versión abreviada de SGML optimizada para su utilización en Internet. Su diferencia fundamental con SGML es que es mucho más simple y más portable. Supone un compromiso entre la potencia de SGML y la simplicidad y portabilidad de HTML. Esto significa que con él cada usuario va a poder definir sus tipos de documentos propios, así como sus etiquetas adaptadas a cada problema particular. (Cuesta Morales, 1999, págs. 17, 18)

2.2.6 Integridad de Datos

- Calidad de la información (perspectiva de la integridad):

> *"los datos deben estar estructurados reflejando adecuadamente los objetos, relaciones y las restricciones existentes en la parcela del mundo real que modela la base de datos"*

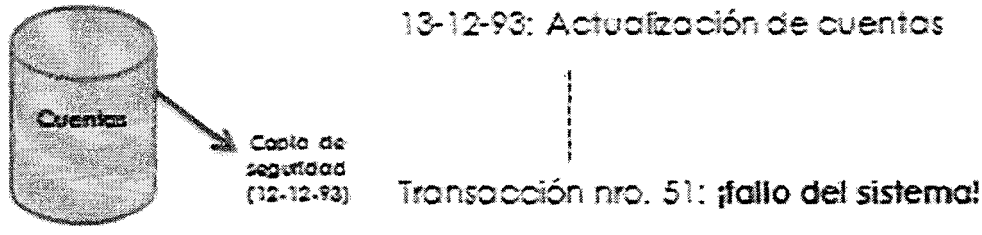
  - SGBD debe asegurar que los datos se almacenan correctamente.
  - SGBD debe asegurar que las actualizaciones de los usuarios sobre la base de datos se ejecutan correctamente y que se hacen permanentes.

Herramientas del SGBD orientadas a la integridad:

- Comprobar (frente a actualizaciones) las restricciones de integridad del esquema.

- Controlar la ejecución correcta de las actualizaciones (entorno concurrente).
- Recuperar (reconstruir) la base de datos en caso de pérdidas o accidentes.

FIGURA II-3 Integridad: Recuperación



Fuente: (Hernández-Orallo, 2006, pág. 8)

- Procedimiento de recuperación:
  - Sustituir el fichero de Cuentas por su copia de seguridad
- Efecto negativo:
  - Se han perdido las actualizaciones de 50 transacciones

2.2.6.1 INTEGRIDAD: TRANSACCIONES

- La integridad de la base de datos se ve en peligro generalmente por las operaciones de acceso de las aplicaciones.
- Las operaciones de acceso a una base de datos se organizan en transacciones.

TRANSACCIÓN ⇒ Secuencia de operaciones de acceso a la base de datos que constituyen una unidad lógica de ejecución.

Emp(dni, nombre, dir, dept)

CP: {dni}

CAj: {dept} → Dep

Dep(cod, nombre, ubicación)

CP: {cod}

R1: ∀Dx:Dep (∃Ex:Emp (Dx.cod = Ex.dept))

Inserción de un nuevo departamento:

<d2, "Personal", "Planta 3ª">

cuyo primer empleado es el de *dni* 20

1ra. idea

1) Inserción en Dep: <d2, "Personal", "Planta 3ª>

ERROR: la restricción *R1* no se cumple

2) Modificación de Emp en la tupla con *dni* 20

2da. idea

1) Modificación de Emp en la tupla con *dni* 20

ERROR: la clave ajena sobre *dept* en *Emp* no se cumple

2) Inserción en Dep: <d2, "Personal", "Planta 3ª>

Operaciones de las transacciones relevantes para el SGBD:

- **leer(X)**: lectura o consulta del dato X de la base de datos sobre la variable del programa del mismo nombre.

  1. buscar la dirección del bloque que contiene el dato X
  2. copiar el bloque a un *buffer* de memoria principal
  3. copiar el dato X del *buffer* a la variable X del programa

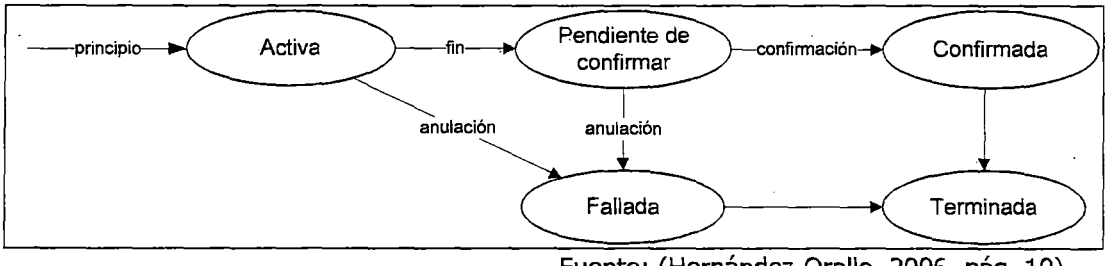
- **escribir(X)**: actualización (inserción, borrado o modificación) del dato X de la base de datos usando la variable del mismo nombre del programa.

Si no se ha leído antes
1. buscar la dirección del bloque que contiene el dato X
2. copiar el bloque a un *buffer* de memoria principal
3. copiar el dato X de la variable del programa a la posición adecuada en el *buffer*
4. copiar el bloque actualizado del *buffer* al disco

Operaciones de definición de las transacciones:

- **principio**: indica el comienzo de la transacción.
- **fin**: indica que se han terminado todas las operaciones de la transacción.
- **confirmación**: indica el éxito de la transacción, permitiendo que el SGBD guarde los cambios efectuados en la base de datos.
- **anulación**: indica el fracaso de la transacción debido a algún motivo. El SGBD deshace todos los posibles cambios efectuados por la transacción.

FIGURA II-4 Operaciones de Definición de las Transacciones



Fuente: (Hernández-Orallo, 2006, pág. 10)

Propiedades que deben cumplir las transacciones:

A. **atomicidad**: una transacción es una unidad atómica de ejecución (o se ejecutan todas sus operaciones o ninguna).

B. **consistencia**: la transacción debe dar lugar a un estado de la base de datos consistente (se cumplen todas las restricciones de integridad).
C. **aislamiento**: las modificaciones introducidas por una transacción no confirmada no son visibles al resto de transacciones.
D. **persistencia**: la confirmación implica la grabación de los cambios introducidos en la base de datos, de forma que no se puedan perder por fallo del sistema o de otras transacciones.

Dos tipos de funcionamiento de las transacciones (según SGBD):

A. **Actualización Inmediata**: las actualizaciones tienen efecto inmediato en memoria secundaria y en caso de anulación se tienen que deshacer.
B. **Actualización Diferida**: las actualizaciones sólo tiene efecto inmediato en memoria principal y se transfieren a memoria secundaria cuando se confirman. (Hernández-Orallo, 2006, págs. 7 -11)

2.2.7 ADO.NET DATA PROVIDERS

El modelo ADO.NET Data Provider provee una interface de gestión común en el.NET Framework para conectar e interactuar con una fuente de datos.

CAPÍTULO III
DESARROLLO DE LA INVESTIGACIÓN

3.1 MATERIALES Y MÉTODOS.

En esta parte se define la fuente de datos y los procedimientos para su obtención. Es necesario hacer una descripción detallada de la muestra y la metodología seguida durante la investigación, de tal manera que otro u otros investigadores, siguiendo los mismos procedimientos y con muestras similares puedan encontrar resultados parciales.

3.1.1 TIPO DE INVESTIGACIÓN

El tipo de investigación es aplicada, y el alcance de investigación es exploratorio, donde el objetivo es examinar un tema poco estudiado y se desea indagar sobre temas y áreas desde nuevas perspectivas. (Hernández, Fernández, & Baptista, 2010, págs. 77, 79), para definir una Arquitectura de software que permita la sincronización de datos entre los sistemas comercial y contable del Estudio Navarro y Asociados.

3.1.2 DISEÑO DE LA INVESTIGACIÓN

El propósito de responder a las preguntas de investigación planteadas y cumplir con los objetivos de estudio, el investigador debe seleccionar o desarrollar un diseño de investigación específico. Cuando se establecen y formulan hipótesis, los diseños sirven también para someterlas a prueba. Los diseños cuantitativos pueden ser experimentales o no experimentales.

El investigador debe visualizar de manera práctica y concreta de responder a las preguntas de investigación, además de cubrir los objetivos fijados. Esto implica seleccionar o desarrollar uno o más diseños de investigación y aplicarlos al contexto particular de su estudio. El término

diseño se refiere al plan o estrategia concebida para obtener la información que se desea.

El Diseño es el Plan o estrategia que se desarrolla para obtener la información que se requiere en una investigación. (Hernández, Fernández, & Baptista, 2010, pág. 158)

El diseño de investigación, es un tipo no experimental; puesto que no se manipula deliberadamente una o más variables independientes para analizar sus posibles resultados. Se plantea intencionalmente un diseño de arquitectura de software, a través de la observación, que permita la sincronización de datos como efecto final. Además que este diseño de investigación es transeccional exploratorio, ya que trata de una exploración inicial en un momento específico, sin ideas prefijadas y con apertura.

3.1.3 POBLACIÓN Y MUESTRA

La unidad de análisis para esta investigación es un cliente del Estudio Contable Navarro y Asociados que use el sistema Comercial @NTO y que tenga conexión al servicio de internet.

3.1.3.1 POBLACIÓN

Una vez que se ha definido cuál será la unidad de análisis, se procede a delimitar la población que va ser estudiada y sobre la cual se pretende generalizar los resultados, así una población es el conjunto de todos los casos que concuerdan con una serie de especificaciones (Hernández, Fernández, & Baptista, 2010, pág. 238).

La población, también conocida como universo, se denomina al conjunto de elementos que tienen características comunes. (Tafur, 1995, pág. 170)

La población considerada dentro de esta investigación está compuesta por los clientes del Estudios Contable Navarro y Asociados que harán uso del

sistema comercial @NTO, qué es el origen de datos para el procesamiento en el sistema contable, Base de Datos, de dicho Estudio contable. Y que tengan el servicio de un proveedor de internet en sus instalaciones.

3.1.3.2 MUESTRA

La muestra es, en esencia, un subgrupo de la población. Digamos que es un subconjunto de elementos que pertenecen a ese conjunto definido en sus características al que llamamos población (Hernández, Fernández, & Baptista, 2010, pág. 240).

La muestra es representativa en una investigación cuando las características de los elementos constitutivos de la muestra tienen exactamente el mismo carácter que el universo, es decir, que los aspectos característicos de las muestras son comunes al universo. (Tafur, 1995, pág. 171)

Muestra no probabilística, subgrupo de la población en la que la elección de los elementos no depende de la probabilidad sino de las características de la investigación. (Hernández, Fernández, & Baptista, 2010, pág. 176)

Al ser ésta una investigación exploratoria, donde el objetivo consiste en examinar un tema poco estudiado. La muestra que se toma es no probabilística, entonces se dirige la muestra a un cliente que hace uso del sistema comercial @NTO, y tiene el servicio de internet en sus oficinas, por ser un caso – tipo.

3.1.4 VARIABLES E INDICADORES

3.1.4.1 DEFINICIÓN CONCEPTUAL DE LAS VARIABLES

INDEPENDIENTE

Diseño de la Arquitectura de Software de un componente:

Las opciones de arquitectura deben promover ciertas cualidades al tiempo que implementan la funcionalidad deseada, alineándose con los requisitos de calidad. El sistema se descompone en elementos para lograr el propósito deseado.

INDICADORES INDEPENDIENTES

Patrones de Arquitectura de Software.

Un patrón arquitectónico expresa un esquema de organización estructural esencial para un sistema de software, que consta de subsistemas, sus responsabilidades e interrelaciones.

Verificación de Conexión a Internet.

El sistema verifica técnicamente que exista una conexión confiable al servicio de internet.

Transferencia de datos en transacciones.

El sistema agrupa las acciones de transferencia de datos en transacciones que garanticen la integridad y consistencia de datos.

Modificación de entidades.

Las entidades de los sistemas comercial y contable sufren una modificación para tener rastro de los datos transferidos.

DEPENDIENTES

Sincronización de Datos

En términos informáticos se habla de sincronización cuando varios procesos se ejecutan a la vez con el propósito de completar una tarea y evitar así condiciones de carrera, que pudieran desembocar en un estado inesperado. También se habla de sincronización de datos cuando dos dispositivos se actualizan de forma que contengan los mismos datos.

INDICADORES DEPENDIENTES

Comunicación entre sistemas.

Un sistema distribuido consiste en múltiples computadoras o sistemas autónomos que se comunican a través de una red de computadoras, para lograr un fin común.

Transmisión de datos

Es la transferencia física de datos (un flujo digital de bits) por un canal de comunicación punto a punto o punto a multipunto.

Integridad de los datos

La exigencia de integridad de los datos garantiza la calidad de los datos de la base de datos.

Identificación de datos

Refiere a tener identificados los datos que han sido sincronizados entre dos sistemas.

3.1.4.2 Definición Operacional de las Variables

INDEPENDIENTE

X: Diseño de la Arquitectura de Software de un componente

INDICADORES INDEPENDIENTES

X1: Patrones de Diseño de Arquitectura de Software.

X2: Verificación de Conexión a Internet

X3: Transferencia de datos en transacciones

X4: Modificación de entidades

DEPENDIENTES

Y: Sincronización de Datos

INDICADORES DEPENDIENTES

Y1: Comunicación entre sistemas

Y2: Transmisión de datos

Y3: Integridad de los datos

Y4: Identificación de datos

3.1.5 MEDIOS E INSTRUMENTOS

Se usan las técnicas de observación, medición directa, análisis documental, para recolectar los datos e información necesarios que permitan el diseño arquitectónico.

Entre los instrumentos que permiten comprender cuál es el comportamiento de la solución para lograr la sincronización automática de datos, están descritos en la Tabla III-1.

TABLA III-1 INSTRUMENTOS DE LA INVESTIGACIÓN

Herramienta	Descripción	Objetivo
Microsoft® Office Visio® 2007 Versión 12.0.4518.1014	Software de dibujo vectorial para Microsoft Windows. Las herramientas que lo componen permiten realizar diagramas de oficinas, diagramas de bases de datos, diagramas de flujo de programas, UML, y más, que permiten iniciar al usuario en los lenguajes de programación.	Modelar la arquitectura de software de la solución. Modelar del patrón de desarrollo de software de componente.
Notepad **Microsoft® Windows®** Versión 6.1	Sencillo editor de texto que viene incluido en todas las versiones de Microsoft® Windows®.	Observar la información de los logs, que contienen datos de la conexión y de los tiempos de transferencia de los mismos.
Microsoft® Visual Studio ® Tools for Applications 2.0 **Microsoft® Visual Studio ® 2008** Versión 9.0.35191	Visual Studio Tools for Applications (VSTA) es un conjunto de herramientas que los proveedores de software independientes (ISV) pueden utilizar para construir capacidades de personalización en sus aplicaciones, tanto para la automatización y extensibilidad.	Observar la información que la solución ha transmitido, es decir los datos en el formato XML.
Resource Monitor **Microsoft® Windows® Operating System** Versión 6.1.7600	Aplicación del sistema operativo.	Obtener información detallada en tiempo real acerca de los recursos de un computador.

Microsoft® Excel 2010 **Microsoft® Office® Professional Plus** Versión 14.0.4760.1000(64 bits)	Software que permite crear tablas, y calcular y analizar datos. Es de tipo hoja de cálculo. Excel permite crear tablas que calculan de forma automática los totales de los valores numéricos que especifica, imprimir tablas con diseños cuidados, y crear gráficos simples.	Graficar las distribuciones de frecuencia de los volúmenes de datos, del tiempo de transferencia de datos.
AllFusion® Erwin® Data Modeler Versión 7.1.01075	CA ERwin® Data Modeling ofrece un entorno de modelado de datos de colaboración para administrar datos empresariales con una interfaz intuitiva y gráfica.	Modelar las entidades a modificar, para obtener la sentencia de modificación de entidades de base de datos.
Microsoft® SQL Server® Management Studio **Microsoft® SQL Server® 2008 R2** Versión 10.50.1600.1	SQL Server Management Studio es un ambiente integrado para obtener acceso, configurar, gestionar, administrar y desarrollar todos los componentes de SQL Server. SQL Server Management Studio combina un amplio grupo de herramientas gráficas con una serie de editores de sentencias para facilitar el acceso a SQL Server a desarrolladores y administradores de todos los niveles.	Ejecutar sentencias para modificar las entidades de base de datos.

Fuente: Elaboración Propia.

A partir del estudio y análisis de la literatura disponible para diseñar la Arquitectura de software del "componente", que permite la sincronización de datos entre los sistemas Comercial y Contable del Estudio Navarro y Asociados.

CONSIDERACIONES

El volumen de datos es un factor a considerar para el diseño de la arquitectura, es por ello que se selecciona el cliente que produce mayor cantidad de datos mensual que es 4 587 KB al mes. De las que partieron las siguientes consideraciones:

A. Volumen de datos

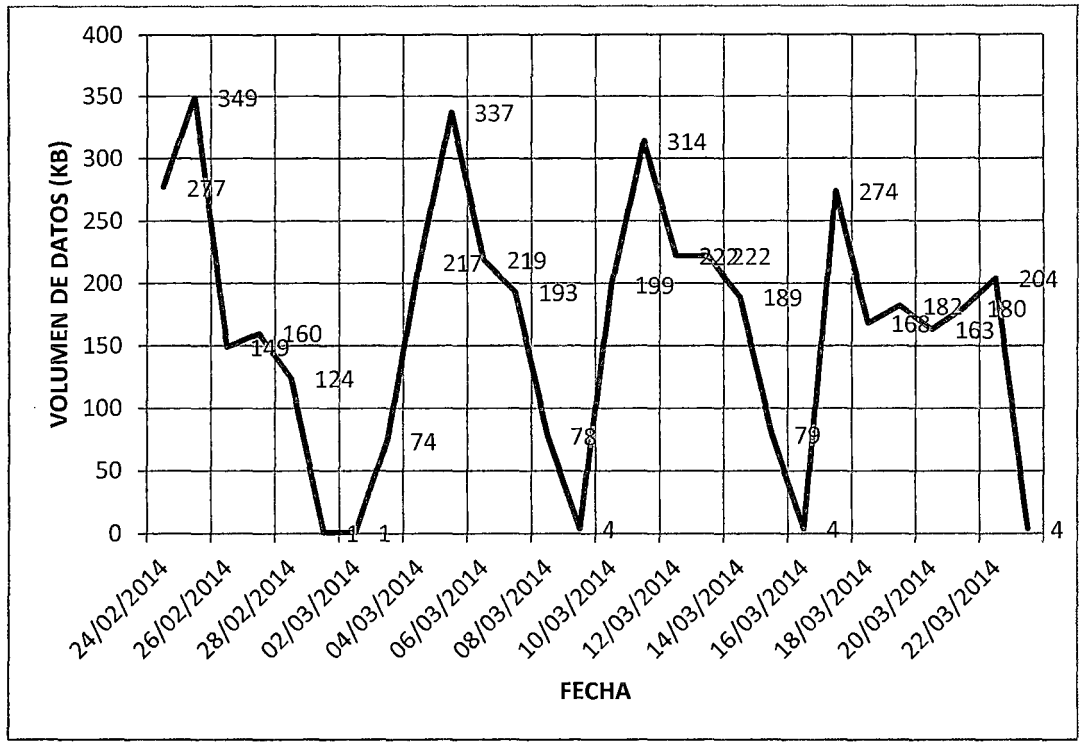
   Las observaciones fueron realizadas, durante el periodo 24 de febrero de 2014 y 23 de marzo de 2014, del volumen de datos de cada archivo XML generado para la transmisión y con ello se determina cuál es el volumen de datos más alto generado en un día. Detalle mostrado en la Figura III-1, a partir de la Tabla 0-1 del ANEXO. En la Figura 0-2 del ANEXO A se muestran los archivos XML generados.

B. Tiempo de espera (Time out)

   Considerando la moda del volumen de datos a transmitir analizado en el ítem anterior y el tiempo máximo empleado para dicha transmisión el cual es de 3 889 milisegundos (información obtenida desde los log de datos Figura 0-1, resumido en la

Tabla 0-2 Tiempo de Transferencia de datos Diario y Figura III-2), y teniendo en cuenta que el time out por defecto de un servicio web es de 15 000 milisegundos (Microsoft Developer Network, 2014) se concluye que la posibilidades de que ocurra un time out son mínimas, conociendo además que la propiedad time out es configurable programáticamente (Microsoft Developer Network, 2014).

FIGURA III-1 VOLUMEN DE DATOS DIARIO



CRITERIOS

A partir de las consideraciones, para lograr la sincronización, se definieron los siguientes criterios para desarrollar el componente:

A. Integridad

Para lograr la integridad de los datos se consideran dos puntos importantes:

- Transacciones: la transferencia de datos se debe realizar mediante transacciones usando los ADO .NET Data Providers que permiten mantener una transacción viva mientras se conecta a la Base de Datos del Sistema Contable.
- Banderas de Confirmación: Dependiendo de la ejecución de la transacción se debe obtener una respuesta mediante banderas:

  "True": ejecución exitosa.

  "False": excepción, no se realizó ninguna acción.

B. Estructura de Datos

Revisar cuál es la estructura de datos que recibe el servicio web. Figura 0-3.

C. Formato XML

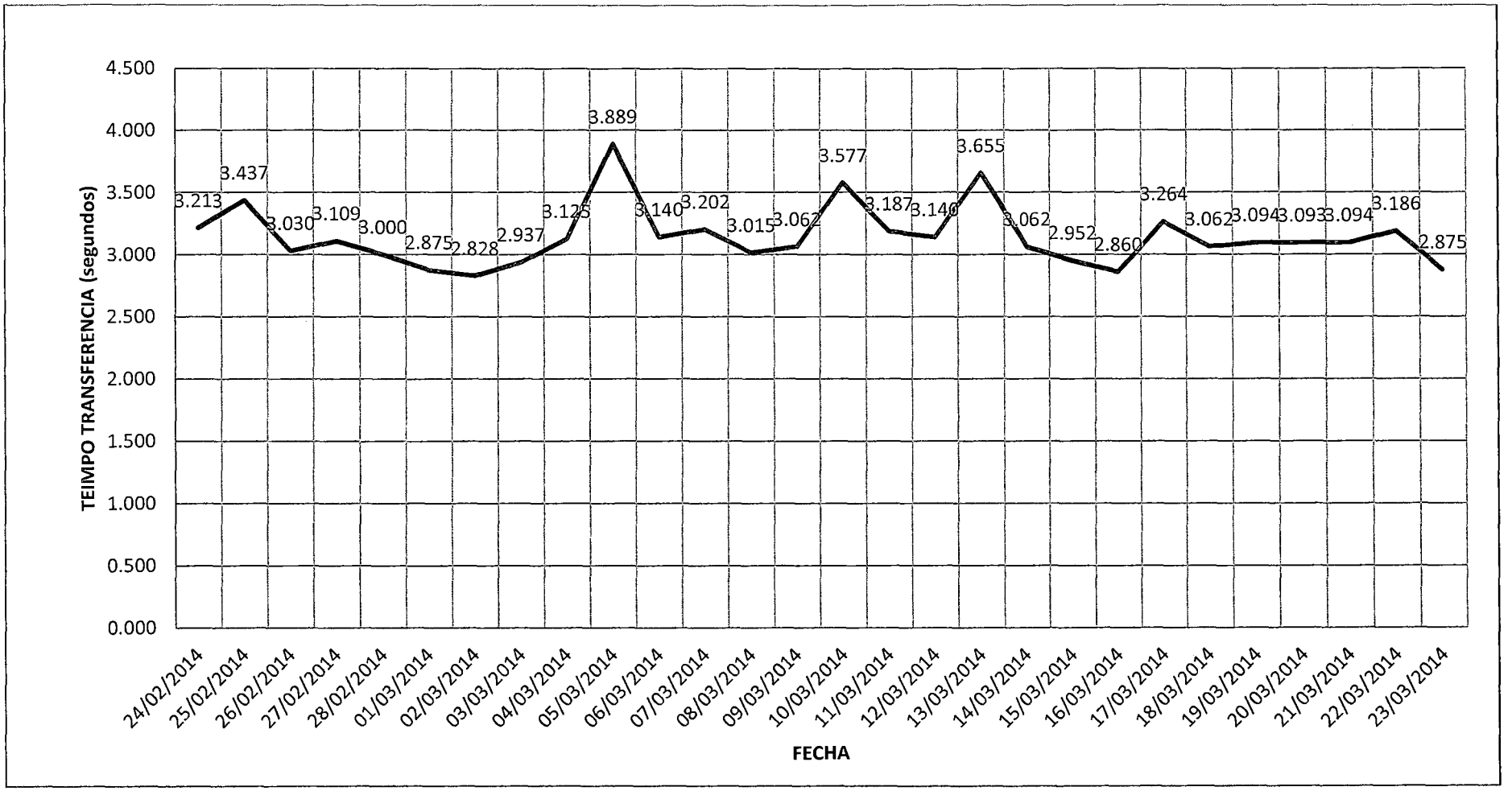
De acuerdo a la estructura de datos, se definirá el formato XML dentro del cual se transmitirán los datos, Figura 0-4.

D. Frecuencia de envíos

En el Estudio Contable se realiza el procesamiento de datos mensualmente; sin embargo el volumen de datos acumulado extendería el tiempo de procesamiento pudiendo generar un time out en el servicio web; hecho que no permita la transmisión de datos.

Para disminuir el volumen de datos, la transferencia se hará progresivamente, es decir en forma diaria, puesto que día a día se hace un cierre contable lo genera independencia de los datos.

FIGURA III-2 TIEMPO DE TRANSFERENCIA DE DATOS DIARIO



3.2 RESULTADOS

El diseño de la arquitectura define la estructura del sistema o software que dará el marco de trabajo. Esta arquitectura de software se selecciona y diseña con base en objetivos y restricciones. El objetivo es aquel que ha sido prefijado en el Capítulo I. Planteamiento de la Investigación dentro de este trabajo de investigación. Las restricciones son aquellas limitaciones derivadas de las tecnologías disponibles para implementar sistemas de información.

Este escenario tiene la necesidad de sincronizar datos de dos sistemas autónomos, que contienen lógicas independientes, que están alojados en diferentes servidores y además los servidores se encuentran en diferentes ubicaciones físicas, existiendo un único proceso de "comunicación", que es la de exportación e importación de datos, y es ejecutada a demanda. La mayor restricción que se tiene es la propiedad de encapsulamiento de ambos sistemas.

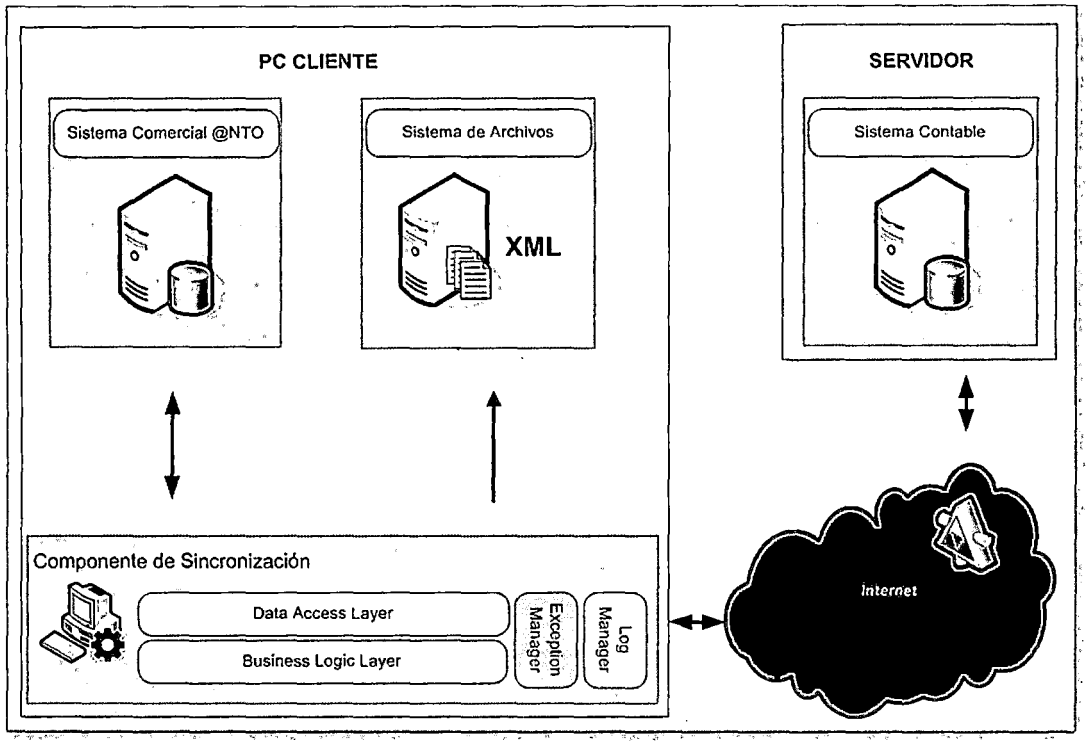
La mayor restricción que se enfrentó es el aislamiento de ambos sistemas que no pueden modificarse, únicamente hacer uso de los métodos expuestos, que no alteren su parte lógica.

El diseño de la arquitectura se basó en el de software distribuido, que permite la ejecución automática del proceso de sincronización sobre una red de comunicación – denominado en adelante "Componente de Sincronización". La base del patrón arquitectónico es el de la Arquitectura Orientada a Servicios (SOA, por sus siglas en inglés) que otorga un marco de trabajo para la construcción del software y da soporte a la integración y consolidación de sistemas. En la Figura II-1, se representa la Arquitectura de Alto Nivel del componente, que otorga una visión general del diseño.

Parte de la solución es la configuración un Sistema de Archivos, que será el repositorio de archivos con datos producto del proceso de

sincronización; además del Componente de Sincronización, ejecutor del proceso. Estos últimos, de acuerdo al diseño, son alojados en la PC Cliente, por ser el generador de datos.

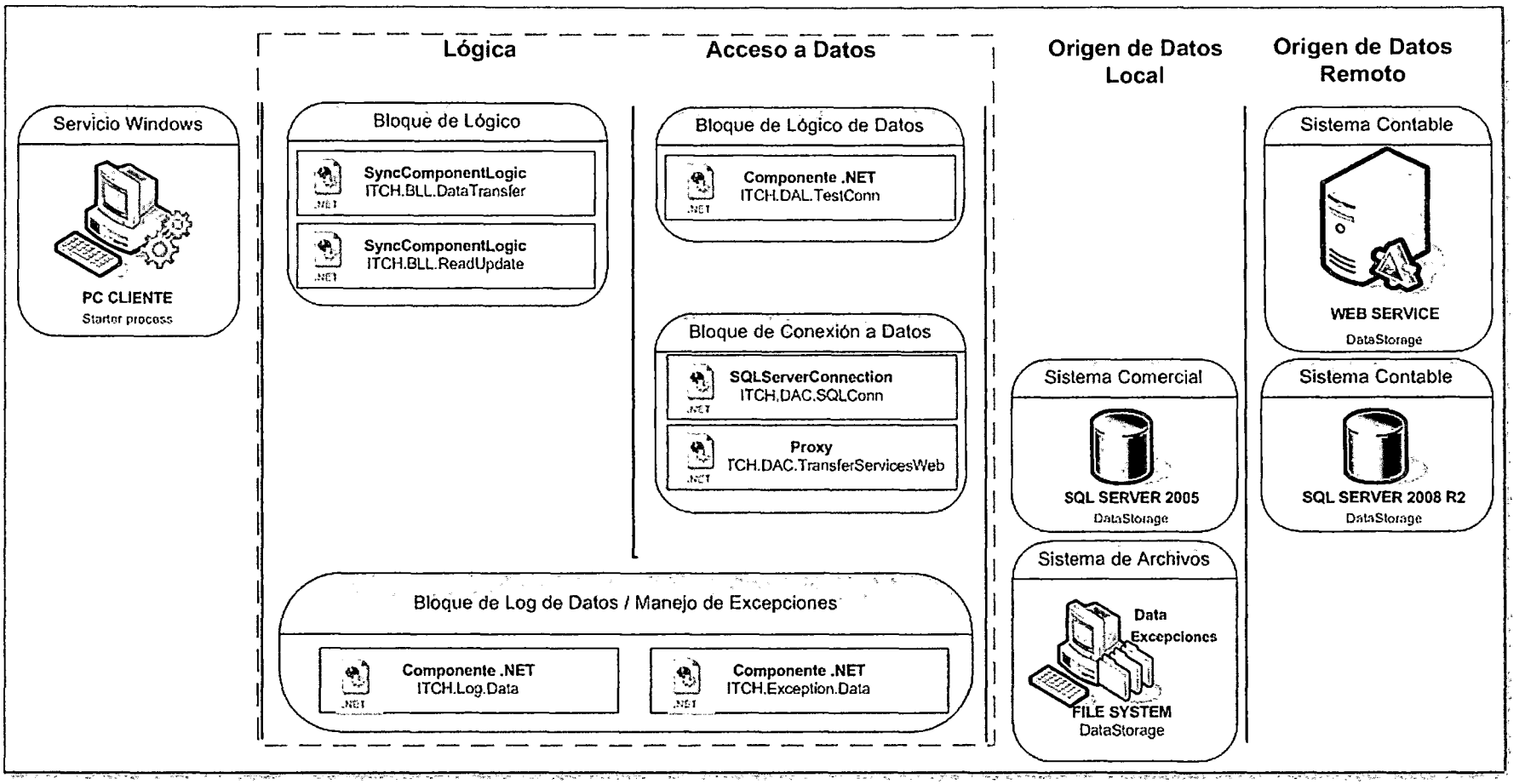
FIGURA III-3 ARQUITECTURA DE SOFTWARE DE ALTO NIVEL



Fuente: Elaboración Propia.

El Componente de Sincronización dentro de sí tiene componentes especializados que logran el cumplimiento del objetivo principal, como se muestra en la Figura III-4. Además consta de otros elementos necesarios dentro de la solución como son el Servicio Windows como arrancador del proceso, el Servicio Web como expositor del método de sincronización, la Base de datos del Sistema Comercial y el Sistema de Archivos.

FIGURA III-4 ARQUITECTURA DETALLADA DEL COMPONENTE



Fuente: Elaboración Propia.

Componente de sincronización

La tarea del componente es en primera instancia obtener los datos de las transacciones comerciales que aún no han sido sincronizados, desde la Base de Datos del Sistema Comercial; estructurar los datos de acuerdo a la plantilla definida, para transferirla a través del Servicio Web publicado en el SERVIDOR (Instalaciones del Estudio Contable) – que realiza la inserción de datos en la Base de Datos del Sistema Contable – finalmente, al obtener el mensaje de *ejecución exitosa* actualiza las banderas de *sincronizado* en la Base de Datos origen.

El componente administra un registro de actividades y de excepciones del proceso, para facilitar el seguimiento de los pasos ejecutados por el proceso de sincronización.

Para la transferencia de datos se decide trabajar con el formato XML por ser un estándar internacional que ofrece la posibilidad de definir los elementos: en él se pueden definir etiquetas y estructurar el documento en función de dichas etiquetas. Este metalenguaje está optimizado para su uso en Internet en el que se define la sintaxis, y dentro de sus características más importantes son las de simplicidad y portabilidad.

Es así que el archivo XML se envía a través del servicio Web, que expone el método de inserción en la Base de Datos del Sistema Contable que recibe un objeto del tipo *OperationClient* (véase Figura 0-3), con una estructura a la cual el componente debe adaptarse; a partir de ello se define la estructura del archivo XML, como se muestra en la Figura 0-4.

A nivel de Desarrollo de Software, el Componente se construye bajo el patrón *N-Capas*. Con dos bloques importantes como el Bloque Lógico y el Bloque de Acceso a Datos, además de un Bloque de Log de Datos/Manejo de Excepciones que es un bloque transversal a los dos anteriores.

- **Capa Lógica:** Bloque lógico que recibe la orden – del servicio Windows – para iniciar el proceso de sincronización, solicita a la Capa de acceso a datos los registros a transferir, los mismos que son transformados al formato XML para luego ser enviados al servicio web mediante la Capa de acceso a datos.
- **Capa de Acceso a Datos:** Consta de:
    - Bloque de Verificación que verifica que el servicio web esté en línea.
    - Bloque de Conexión a Datos, que gestiona dos orígenes de datos:
        - Origen de datos remoto: Conexión al Servicio web a través de un objeto proxy para enviar los datos en el formato XML, provenientes de la capa lógica, y finalmente recibir la bandera de confirmación de: "ejecución exitosa" para ordenar la actualización de los registros enviados (hacia la conexión al origen de datos local), o "excepción" para tratarla en el Bloque de Log de Datos/ Manejo de Excepciones.
        - Origen de datos local: Conexión a la BD del cliente para obtener los registros a enviar y actualizar localmente los registros que han sido sincronizados.
- **Bloque de Log de Datos/ Manejo de Excepciones:** Componente transversal, También construye y escribe archivos de texto (*.txt), que contienen la información de la transferencia de datos y las excepciones que puedan surgir durante el proceso de sincronización en el Sistema de Archivos, que realiza dos tareas:
    - Registrar la información de la transferencia de datos en un Log de datos
    - Registrar las excepciones en un Log de excepciones para su posterior análisis y seguimiento.

Servicio Windows.

Es el componente que inicia el proceso de sincronización de datos, que invoca al *Componente de Sincronización.* Este componente puede ser de ejecución automática diaria y/o de ejecución a demanda, para casos excepcionales. La ejecución principal es automática porque se desea garantizar la calidad en la ejecución y la disponibilidad de la información, disminuyendo errores y optimizando cada recurso material y humano.

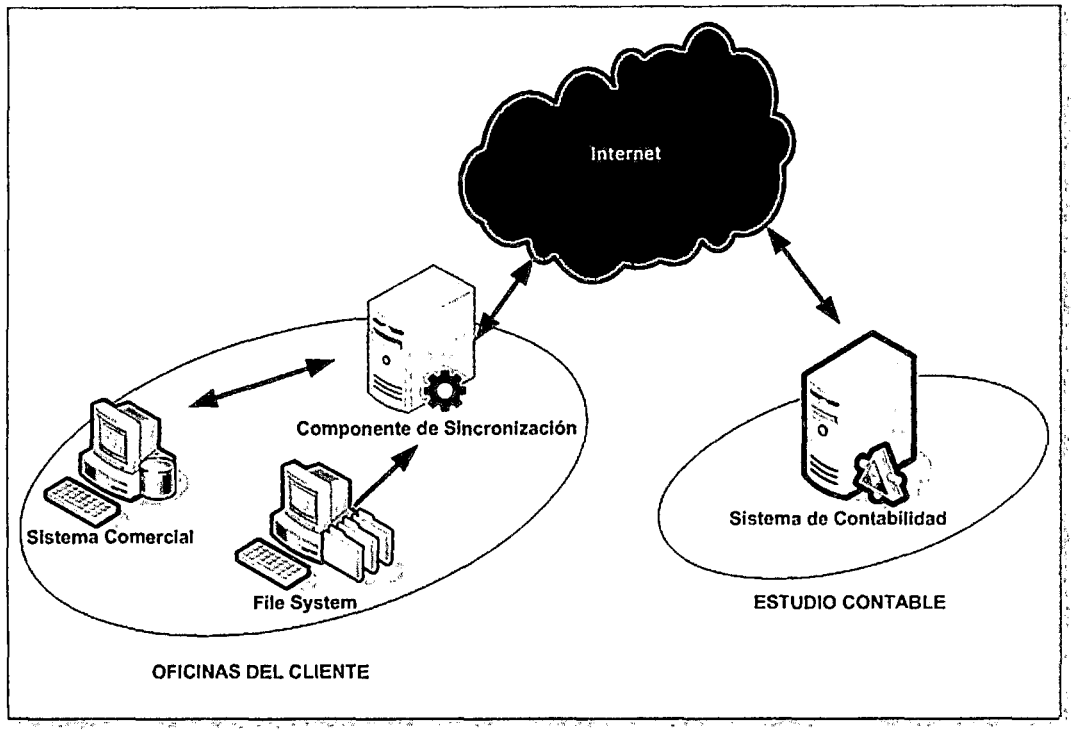
Servicio Web.

Es el método que realiza el proceso de inserción de los datos (transacciones comerciales diarias) en el Sistema Contable, va a estar publicado sobre internet y será invocado y consumido por el "Componente de Sincronización".

La Distribución de Componentes, del software distribuido se organizan del siguiente modo: el "Componente de Sincronización" se despliega en la PC Cliente, de este modo se obtiene la ventaja de que las consultas de datos se ejecuten localmente haciéndola más eficiente; también en este equipo se almacenan el Log de Datos y el Log de Excepciones, obteniendo de igual modo eficiencia en la ejecución de este proceso; además aloja el servicio Windows que es el iniciador de la sincronización.

La comunicación con el Sistema Contable, se da a través del servicio web publicado en el Servidor del Estudio Contable, desde donde se consume el método que permite la inserción de datos y se obtiene la respuesta de ejecución exitosa a dicho proceso. Este planteamiento se aprecia gráficamente en la Figura III-5.

FIGURA III-5 DISTRIBUCIÓN DE COMPONENTES



Fuente: Elaboración Propia.

Además como parte del diseño de la arquitectura se define en detalle el flujo grama del "Componente de Sincronización" que grafica el funcionamiento y las invocaciones que se deben realizar para lograr la sincronización de datos (Figura 0-6).

3.3 DISCUSIÓN.

El resultado obtenido en esta investigación se puede comparar con los Enterprise Application Integration, puesto que permite la integración de sistemas y lo hace según la arquitectura orientada a servicios (SOA).

Por medio de esta investigación se logra la sincronización de datos específicamente para el Sistema Comercial y el Sistema Contable del Estudio Navarro y Asociados, en semejanza al producto Oracle EAI Services, que integra solamente la Suite Oracle.

Los altos costos para la adquisición de licencias de uso e implementación sólo permiten que grandes compañías puedan acceder a estos productos; haciendo inviable su uso por las Mypes de la región Ayacucho, siendo aún estos productos técnicamente recomendables y que responden ampliamente a sus requerimientos de integración.

CAPÍTULO IV
CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUSIONES.

- El patrón de Arquitectura Orientado a Servicios (SOA) fue la base para realizar el diseño del componente que finalmente permite la sincronización de datos a través del internet entre los Sistemas Comercial (Cliente) y Contable (Servidor).
- El componente de sincronización, permite verificar la conexión a internet, además se asegura la disponibilidad del Servicio Web y así se garantiza la transmisión de datos.
- El uso de un formato ligero y estructurado, como el XML. Y, adicionalmente el uso del ADO .NET Data Provider, hacen que se garantice la transmisión de la data a la Base de Datos del Servidor mediante transacciones.
- Mediante la modificación de una entidad (tabla Transacción) dentro de la Base de datos local, agregándose una bandera de tipo booleana (campo bSincronizacion), se logra identificar el estado del registro, es decir, si éste fue o no sincronizado.

4.2 RECOMENDACIONES.

- Gran parte de los sistemas comerciales no contemplan el envío de datos automáticos a través de internet, realizándolos únicamente por medios físicos (USB), impresiones y/o archivos adjuntos en correos electrónicos, situación que puede ser mejorada si se usa como medio de transmisión de datos al internet.
- El proceso de la contabilidad requiere de una re digitación de la data de compras y ventas realizadas por el cliente, luego del cuál

recién inicia el procesamiento contable requerido. Dificultad que puede ser superada si se buscan integrar los sistemas comerciales y contables de diferentes proveedores.

- No se hace uso de transacciones para garantizar la integridad de datos en el almacenamiento dentro de base de datos. Falta que debe ser evitada con el uso de Data Providers, que permiten conexiones a diferentes motores de base de datos, de los lenguajes de programación.

REFERENCIAS BIBLIOGRÁFICAS

Andrews, G. R. (2000). *Foundations of Multithreaded, Parallel, and Distributed Programming.* Addison-Wesley.

Avgeriou, P., & Uwe, Z. (2005). Architectural patterns revisited: a pattern language. *10th European Conference on Pattern Languages of Programs.*

Bass, L., Clements, P., & Kazman, R. (2005). *Software Architecture in Practice* (2da. ed.). Boston: Addison-Wesley.

Bieberstein, N., Bose, S., Fianmante, M., Jones, K., & Shah, R. (2005). *Service-Oriented Architecture (SOA) Compass.* Massachusetts: IBM Press.

Brown, W., Malveau, R., Mc Cormick III, H., & Mowbray, T. (1998). *AntiPatterns. Refactoring Software, Architectures and Projects in Crisis.* New York: Jhon Wiley & Sons, Inc.

Buschamann, F., Meunier, R., Rohnert, H., Sommerlad, P., & Stal, M. (1996). *Pattern-Oriented Software Architecture: A System of Patterns.* New York: John Wiley & Sons.

Cuesta Morales, P. (1999). Desarrollo de Aplicaciones Distribuidas basadas en Tecnologías Web. *Desarrollo de Aplicaciones Distribuidas basadas en Tecnologías Web*, 19.

Facultad Ingeniería de Sistemas. (22 de Enero de 2014). *Sistemas Gestores de Base de Datos.* Recuperado el 12 de Febrero de 2014, de Pontificia Universidad Católica de Ecuador: ftp://ftp.puce.edu.ec/Facultades/Ingenieria/Sistemas/Base%20de%20Datos%20II/Sistemas%20Gestores%20de%20Bases%20de%20Datos%20Capitulo%201.pdf

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design Patterns. Elements of Reusable Object-Oriented Software.* Baarn: Addison Wesley.

Ghosh, S. (2007). *Distributed Systems: An Algorithmic Approach (Chapman & Hall/CRC Computer & Information Science Series).* Boca Raton: Chapman & Hall/CRC.

Godfrey, B. (2002). *A primer on distributed computing.* Recuperado el 20 de 11 de 2012, de Temporary links page: http://www.bacchae.co.uk/docs/dist.html

Golfarb, C., & Prescod, P. (1999). *Manual de XML.* Madrid: Prentice Hall.

Hernández, R., Fernández, C., & Baptista, P. (2010). *Metodología de la investigación* (5ta. ed.). México D. F.: McGraw-Hill.

Hernández-Orallo, J. (2006). *José Hernández-Orallo Web site.* Recuperado el 12 de Junio de 2013, de Universidad Politécnica de España: http://users.dsic.upv.es/~jorallo/docent/BDA/castella/tema3_4x1.pdf

Lynch, N. A. (1996). *Distributed Algorithms* (1ra ed.). California: Morgan Kaufmann Publishers Inc.

Microsoft Developer Network. (25 de Enero de 2014). *HttpWebRequest.Timeout Property.* Recuperado el 03 de Febrero de 2014, de MSDN: http://msdn.microsoft.com/es-es/library/system.net.httpwebrequest.timeout.aspx

Montogomery, D., & Runger, G. (2002). *Probabilidad y Estadística aplicadas a la Ingeniería* (2da ed.). Mexico D.F.: McGraw-Hill.

Stallings, W. (2004). *Comunicaciones y Redes de Computadores* (7ma. ed.). Madrid: PEARSON PRENTICE HALL.

Tafur, R. (1995). *La tesis universitaria* (1ra. ed.). Lima: MANTARO.

ANEXO A

TABLAS Y GRÁFICOS

TABLAS

TABLA 0-1 VOLUMEN DE DATOS DIARIO

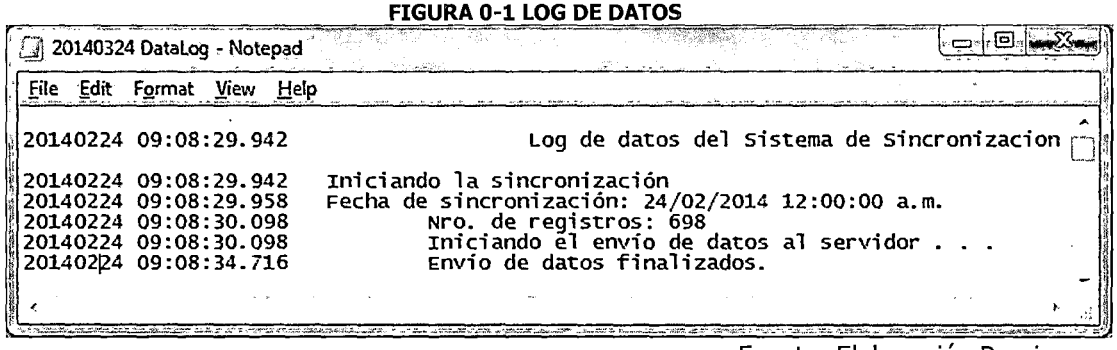
Fecha	Volumen Datos (KB)
24/02/2014	10
25/02/2014	12
26/02/2014	15
27/02/2014	13
28/02/2014	14
01/03/2014	9
02/03/2014	11
03/03/2014	5
04/03/2014	14
05/03/2014	12
06/03/2014	13
07/03/2014	12
08/03/2014	8
09/03/2014	12
10/03/2014	7
11/03/2014	10
12/03/2014	12
13/03/2014	11
14/03/2014	14
15/03/2014	10
16/03/2014	11
17/03/2014	8
18/03/2014	13
19/03/2014	15
20/03/2014	12
21/03/2014	11
22/03/2014	10
23/03/2014	6

TABLA 0-2 TIEMPO DE TRANSFERENCIA DE DATOS DIARIO

Fecha	Tiempo Transferencia (segundos)
24/02/2014	3.213
25/02/2014	3.437
26/02/2014	3.030
27/02/2014	3.109
28/02/2014	3.000
01/03/2014	2.875
02/03/2014	2.828
03/03/2014	2.937
04/03/2014	3.125
05/03/2014	3.889
06/03/2014	3.140
07/03/2014	3.202
08/03/2014	3.015
09/03/2014	3.062
10/03/2014	3.577
11/03/2014	3.187
12/03/2014	3.140
13/03/2014	3.655
14/03/2014	3.062
15/03/2014	2.952
16/03/2014	2.860
17/03/2014	3.264
18/03/2014	3.062
19/03/2014	3.094
20/03/2014	3.093
21/03/2014	3.094
22/03/2014	3.186
23/03/2014	2.875

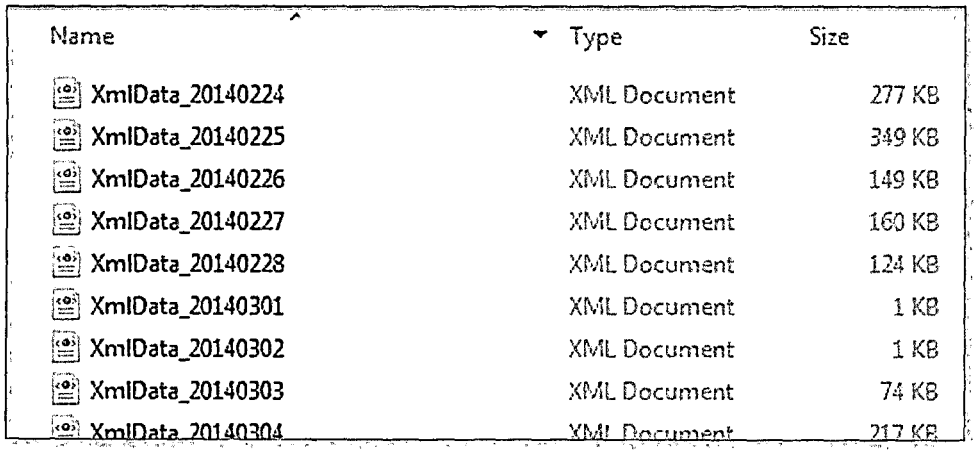
GRÁFICOS

FIGURA 0-1 LOG DE DATOS



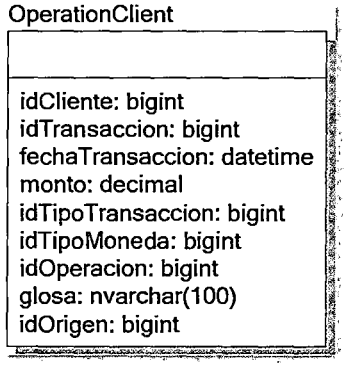
Fuente: Elaboración Propia.

FIGURA 0-2 ARCHIVOS XML GENERADOS



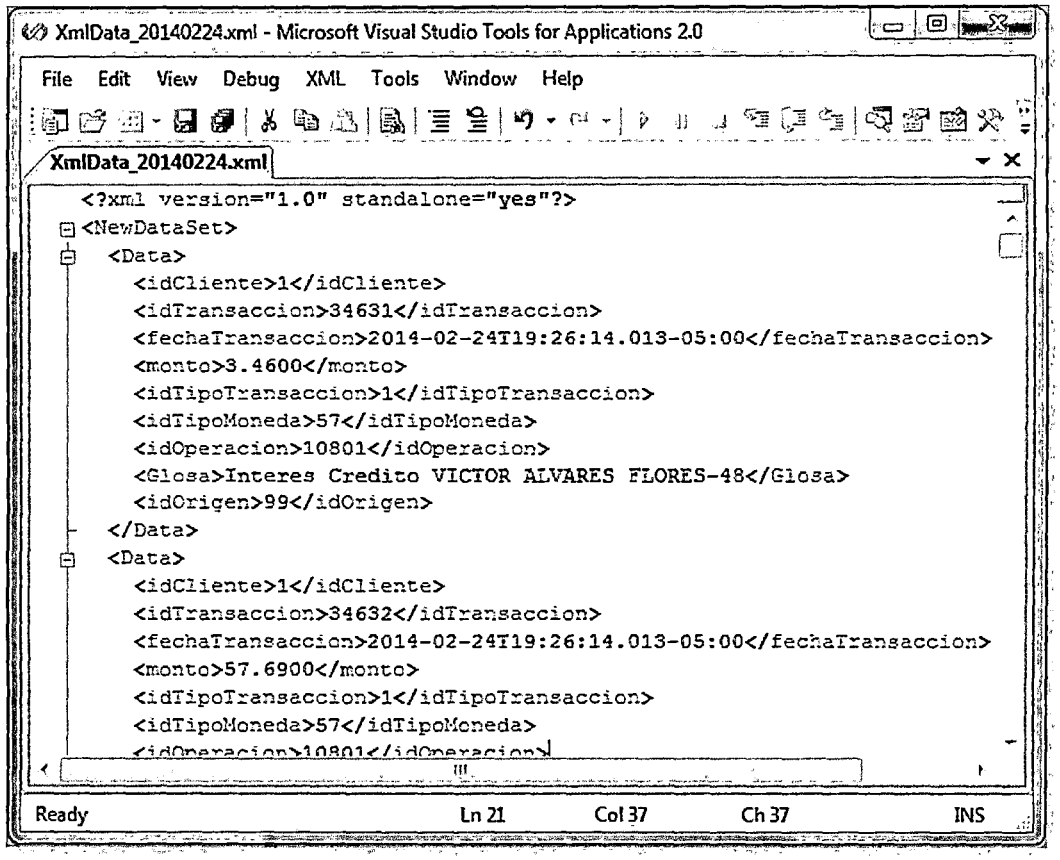
Fuente: Elaboración Propia.

FIGURA 0-3 ESTRUCTURA DE DATOS



Fuente: Elaboración Propia.

FIGURA 0-4 FORMATO XML



Fuente: Elaboración Propia.

FIGURA 0-5 MODIFICACIÓN DE DATOS

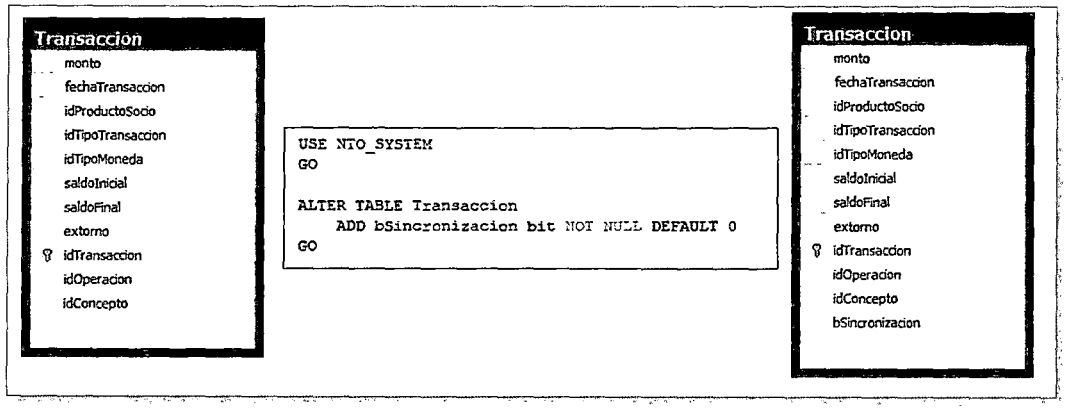
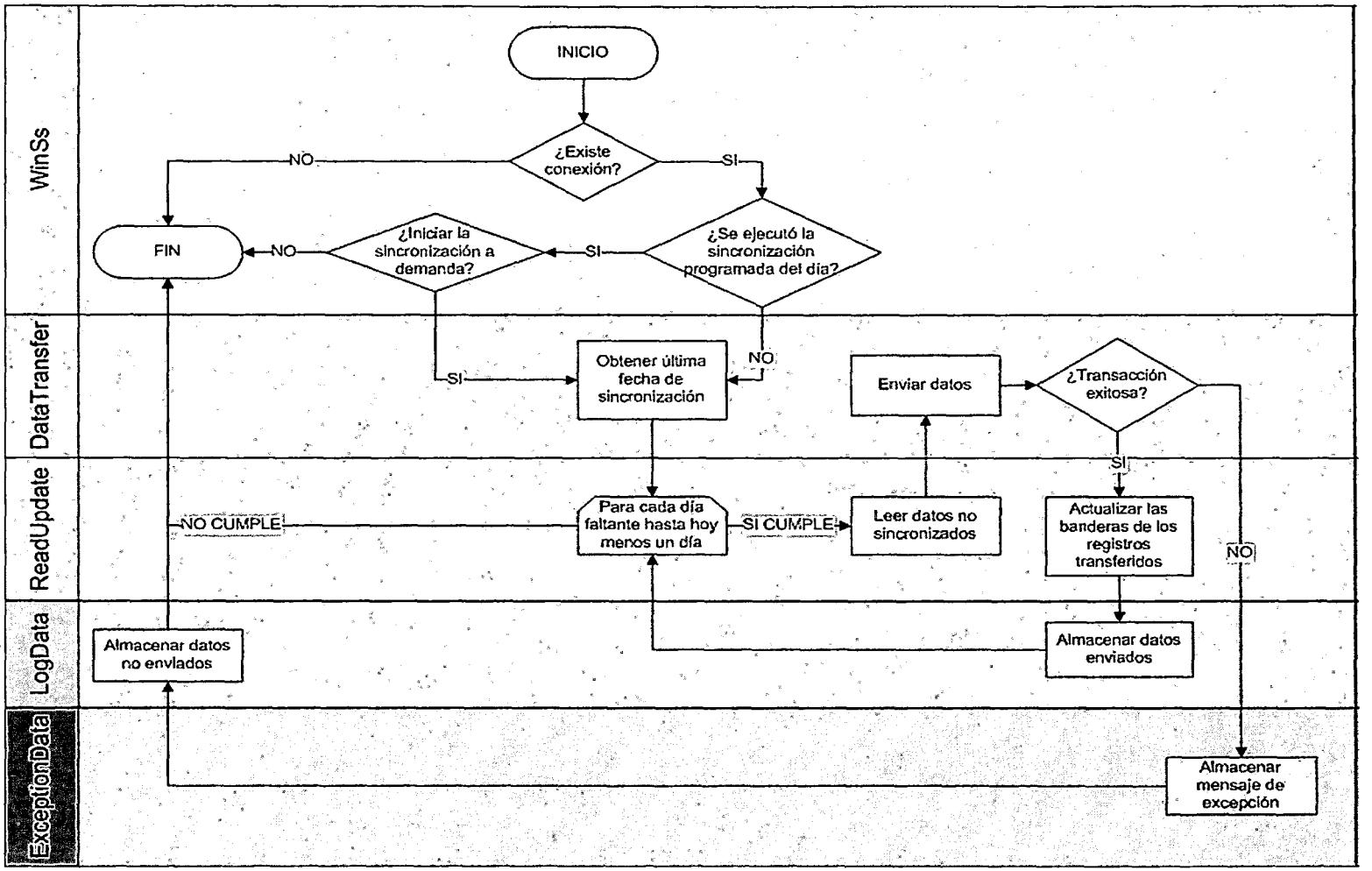


FIGURA 0-6 DIAGRAMA DE FLUJO



Fuente: Elaboración Propia.

FIGURA 0-7 DIAGRAMA ENTIDAD RELACIÓN @NTO