Modelo relacional para el desarrollo de aplicaciones con distribución de datos en múltiples sistemas de gestión de bases de datos

ROBERTO CARLOS ERAZO MUÑOZ

UNIVERSIDAD DEL CAUCA

FACULTAD DE INGENIERIA ELECTRONICA Y TELECOMUNICACIONES

PROGRAMA DE INGENIERIA DE SISTEMAS

POPAYAN

2007

Modelo relacional para el desarrollo de aplicaciones con distribución de datos en múltiples sistemas de gestión de bases de datos

ROBERTO CARLOS ERAZO MUÑOZ

Director
Carlos Alberto Cobos Lozada
Ingeniero de Sistemas, MsC.

UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERIA ELECTRONICA Y TELECOMUNICACIONES
PROGRAMA DE INGENIERIA DE SISTEMAS
POPAYAN
2007

AGRADECIMIENTOS

Por el apoyo que recibí en el desarrollo del presente trabajo de grado, quiero expresar mis agradecimientos en primer lugar a mis padres Roberto Franco Erazo Narváez y Rosa Elia Muñoz Benavides, quienes han sido mi constante e incondicional apoyo en todo mi proceso formativo, de igual forma y no sin menos importancia a mi hermano Daniel Fernando Erazo Muñoz, quien me ha acompañado en todos mis años de estudio y con el cual he aprendido el valor de la dedicación y esfuerzo por lograr lo que queremos, de manera adicional, quiero mencionar también a las siguientes personas:

CARLOS ALBERTO COBOS LOZADA, Magíster en Informática, Ingeniero de Sistemas y director del trabajo de grado, quien siempre estuvo dispuesto a orientarme y animarme durante el desarrollo del mismo.

MIGUEL MENDOZA, Ingeniero en Diseño y Automatización Industrial, quien siempre estuvo dispuesto a orientarme y colaborarme en este trabajo de grado.

SERGIO VICTORIO, Ingeniero de Sistemas, por su apoyo en mi proceso formativo y por brindarme siempre su opinión y apoyo en la culminación de mis estudios en Ingeniería.

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

RESUMEN

TITULO: Modelo relacional para el desarrollo de aplicaciones con distribución de datos en múltiples sistemas de gestión de bases de datos [*]

AUTOR: ERAZO M. Roberto Carlos

PALABRAS CLAVE: Replicación de bases de datos a través de instantáneas, Bloques de Aplicación, Arquitectura Orientada a Servicios, Sistemas de Información Geográfico.

DESCRIPCION:

La realización de sistemas robustos para la administración de información está en auge a nivel global, de igual forma la cantidad de datos que deben ser procesados por dichos sistemas a veces crecen de manera exponencial y en muchas ocasiones esto provoca que se busquen nuevas alternativas para escalar dichos sistemas. Una de las alternativas de solución para esto normalmente es usar replicación de datos, de hecho, es una de las estrategias más utilizadas por ahorro de costos y simpleza de la solución, sin embargo, no siempre se aprovechan todos los servicios que proveen estas herramientas y generalmente se subutilizan dejando así una pérdida en la inversión realizada. Este trabajo de grado propone la realización de una capa intermedia, que se encargue de las tareas de replicación de datos entre diferentes motores de bases de datos, explícitamente la replicación por instantáneas (Snapshot Replication), de manera adicional y como parte integral de este trabajo, se realizo una aplicación de prueba para el turismo que propone alta disponibilidad a través de servicios de replicación. Como resultado de las pruebas realizadas se puede observar que el servicio de replicación realizado y establecido como una capa intermedia entre la aplicación y un motor de base de datos, es el primer paso para realizar sistemas cada vez más interoperables independizando el uso de un solo motor de base de datos y permitiendo escalar un sistema con un costo de inversión acorde a las necesidades de cada

[*] Trabajo de Grado

empresa. De igual forma se obtiene como resultado un sistema para turismo altamente disponible a través de servicios de replicación que permite realizar un crecimiento del sistema acorde a su demanda.

ABSTRACT

TITLE: Relational Model for Development of Applications with data in Multiple Database System Managers *

AUTOR: ERAZO M. Roberto Carlos

KEYWORDS: Database Snapshot Replication, Service Oriented Architecture, Geographic Information Systems, Tourism.

DESCRIPTION:

The development of robust systems to manage information is on the top in the world. In the same way, the quantity of data that have to be processed for these systems sometimes grow exponentiality and many times this situation involve the search of new alternatives to support this tendency.

One of the most popular solution is the data replication, because it is simpler and cheaper than others, but all this services are not taken advantage and they cause lost of the investments.

This grade work purpose the realization of a middle tier that realize the data replications tasks between several database engines, explicitly, snapshot replication. Additionally and complementing the grade work the development of a Tourism application with high availability that include the replication services to test them.

The results of the tests shows that the replication service developed and established like a middle tier in the architecture of the traditional applications is the first step to realize interoperable systems, independent of the database engines and allow it with the minimal cost, according to the capabilities of each company. In the same way, a tourism system with high availability is

* Grade Work

obtained with the use of replication services allowing the grow according the demand.

INTRODUCCIÓN

Este documento contiene la descripción, implementación, prueba y resultados de la aplicación de un Modelo relacional para el desarrollo de aplicaciones con distribución de datos en múltiples sistemas de gestión de bases de datos, el cual divide la realización del mismo en la implementación de un modelo que pretende establecer unos lineamientos base para la delegación de la responsabilidad de replicación a una capa independiente del motor de base de datos que se encargue específicamente de las tareas de replicación basadas en instantáneas y en el desarrollo de un sistema de información turístico que utilice el modulo de replicación de datos, a manera de prueba.

En cuanto a la profundización en los temas de distribución de bases de datos, se pretende dar el primer paso en la realización de un conjunto de servicios que realicen las tareas de distribución que sistemas gestores de base de datos empresariales ya soportan. Los servicios de distribución de datos tienen la ventaja de hacer que los sistemas de información sean mucho más tolerantes a fallos con una disponibilidad más alta. Sin embargo los costos de licenciamiento de productos que soporten dicha distribución son altos y no siempre se aprovechan todas sus características, es decir para hacer una replicación simple de información a veces no es necesario comprar todo un paquete empresarial que además de características de distribución cuenta con otras características adicionales, esto, dado que no siempre se cuenta con la disponibilidad para realizar tales inversiones. De igual forma es común encontrar en ambientes empresariales la implementación de diferentes sistemas en distintos motores de base de datos, así como la realización de alianzas estratégicas con otras compañías, que hacen que el trabajo del grupo de desarrollo por brindar compatibilidad entre los diferentes sistemas se vea comprometido cuando existe variedad de motores de bases de datos. Por esto, se propone con este proyecto el inicio de la construcción de una capa intermedia entre las aplicaciones y las bases de datos que no dependa del

motor. Un servicio general que permita realizar diferentes tipos de distribución y que dicha tarea no dependa del motor de base de datos.

Si bien este proyecto no implementa totalmente dicha capa, si pretende dar unos lineamientos base para el desarrollo de una capa intermedia totalmente independiente, generando un bloque de aplicación que pueda ser reutilizado en el desarrollo de sistemas que deseen implementar tareas de distribución sin necesidad de depender de un motor específico. El enfoque específico del proyecto implementa la distribución de datos a través de un método de replicación, denominado replicación por instantánea asíncrona. A partir de esta experiencia se generaron a lo largo del proyecto una serie de recomendaciones para implementar modelos de replicación que soporten varios motores de base de datos.

Por otro lado y como parte del impacto social en la consecución de trabajos de investigación se incluyo dentro del desarrollo del proyecto una herramienta que permita tener información turística georeferenciada y que esté disponible en dispositivos móviles (Sistema de Información Geográfico). Esta aplicación además de brindar una herramienta de gran interés dentro de la comunidad usa los servicios de replicación del modelo planteado anteriormente para probar de manera real el uso del modelo y su impacto dentro del desarrollo de sistemas que puedan ser implementados a futuro con un costo más bajo que el actual.

Para la realización de este proyecto fue necesario estudiar los conceptos teóricos relacionados con: Replicación de Bases de Datos, Bases de Datos Distribuidas, desarrollo de aplicaciones de cliente inteligente, inclusión de tecnologías móviles para el turismo, administración de motores de base de datos relacionales, sistemas de información geográfico y su inclusión dentro del turismo.

Este documento se divide en 6 capítulos que a continuación se mencionan y describen:

- *Capítulo I – Contexto de la Investigación:* En este capítulo se describe el problema y la pregunta de investigación, junto con la justificación de la investigación. Además se plantean los objetivos que se llevaron a cabo con la investigación, y una descripción general de los resultados obtenidos.
- *Capítulo II – Contexto teórico:* En este capítulo se muestran los conceptos teóricos más importantes sobre, la replicación de bases de datos, los métodos existentes para la realización de la replicación, las principales características de los sistemas de información geográficos y los conceptos relacionados en el desarrollo de aplicaciones móviles.
- *Capítulo III - Manager Distribution, una implementación del modelo relacional de distribución de datos en múltiples motores de base de datos:* En este capítulo se describen el modelo relacional para soportar la replicación de datos, los servicios Web que dan soporte a la replicación por instantáneas, los servicios Windows que monitorean las tareas de replicación y la aplicación de administración de la replicación.
- *Capítulo IV – Easy Tour Guide 2007:* En este capítulo se describe el proceso de construcción de la herramienta SIG para turismo. *Además* se muestra como se implementan los servicios de replicación en el sistema Easy Tour Guide 2007.
- *Capítulo VI – Conclusiones y Trabajo futuro:* En esta parte se presentan las conclusiones relacionados con el desarrollo de la experimentación y el uso de los servicios de replicación, y del trabajo de investigación en sí mismo. Por último se presentan los lineamientos base para la construcción de un modelo de replicación que podrán ser utilizados para soportar la construcción de servicios de replicación en múltiples motores de base de datos así como múltiples técnicas de replicación.

La realización del modelo relacional para la distribución de datos, permitió modelar y entender de mejor manera como los motores de base de datos implementan tareas de replicación. De igual forma se logro entender la importancia que genera una capa intermedia independiente del motor de base de datos y de una aplicación en particular. La generación de modelos que generalicen cada vez las tareas de distribución de datos es un avance importante dentro de los desarrollos de aplicaciones que no desean depender

de un motor en particular. De igual forma este tipo de servicios independientes de las diferentes capas de aplicación son cada vez más comunes en el mercado, lo cual indica que existe una necesidad evidente por hacer sistemas cada vez más interoperables. Uno de los resultados más importantes dentro de este trabajo es dar el primer paso y aportar cada vez más al desarrollo de herramientas y componentes que permitan la interacción con diferentes tipos de plataformas y productos.

Por otro lado la posibilidad de probar el funcionamiento real de este tipo de iniciativas con aplicaciones "tradicionales", permiten constatar la buena concepción del modelo de replicación planteado.

CAPITULO I: CONTEXTO DE LA INVESTIGACIÓN

En esta parte se describe el problema, el cual está relacionado con la búsqueda de nuevas soluciones para el desarrollo de sistemas escalables con replicación de datos basándose en una capa intermedia independiente del motor de base de datos y su uso en el desarrollo de una herramienta tecnológica para el turismo que posea alta disponibilidad y escalabilidad sin incurrir en mayores costos.

1 DESCRIPCION DEL PROBLEMA

1.1 PLANTEAMIENTO DEL PROBLEMA

Cuando se realiza un aplicación (solución informática) soportada con bases de datos distribuidas como Oracle Enterprise Server o SQL Advanced Server, el desarrollador se aísla de las operaciones propias de la distribución, ya que ellas son realizadas en forma transparente por el motor de base de datos que esté usando y estas tareas son gestionadas o configuradas por un Administrador de la base de datos (global). En este escenario, la aplicación depende de los servicios de distribución que ofrece el motor, y normalmente estos servicios "avanzados" cuestan más, es decir si yo tengo mi aplicación en la versión Oracle Workgroup y necesito el soporte de distribución debo comprar la Licencia Empresarial que es la que incorpora estos servicios. En lo anterior vemos un primer problema, el costo adicional (problema económico) por una nueva versión que hace muchas cosas más de las que se necesitan, en este caso la distribución. En otro contexto, cuando las aplicaciones se despliegan bajo el esquema de hosting, el problema es aún mayor, ya que los Proveedores de Servicios de Internet (Internet Services Providers, ISP), normalmente no pueden cubrir necesidades tan específicas en cuanto a tecnología o lo hacen bajo unos costos exorbitantes.

Ahora bien, si la aplicación debe correr de manera simultánea en dos motores de bases de datos distintos, el problema en algunos casos se soluciona comprando versiones superiores de los motores (que cuestan elevadas sumas de dinero) pero que en algunos casos no permiten interoperar con todos los motores (problema tecnológico), por ejemplo, una base de datos Oracle con MySQL, implementar replicación entre estos dos motores de base de datos no es posible porque los servicios de replicación de Oracle no incluyen replicas de motores como MySQL.

La alternativa de solución que se plantea (y con la idea de aclarar el problema) consiste en mover los servicios de distribución que ofrecen los motores de bases de datos, a una capa del software que se desarrolló en este proyecto y que se complementa con un Bloque de Aplicación para la Distribución de Datos (ver Figura 1).

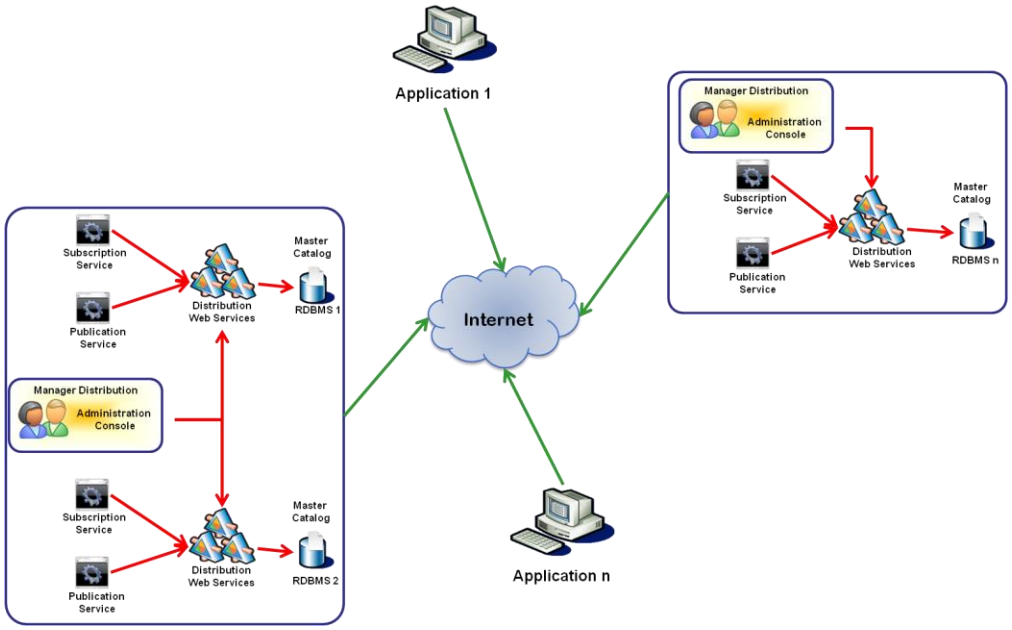

Figura 1 - Propuesta de lógica de distribución y catalogo maestro de distribución

Para que este aplicativo de Distribución funcione apropiadamente se hace necesario contar con un Catálogo Maestro de distribución en todas las bases de datos de los motores que hagan parte del sistema (como un todo), algunos

procesos que permitan realizar la comprobación de consistencia e integridad de los datos (que son parte del aplicativo) y aplicar algunos lineamientos a las tablas a replicar.

De manera adicional el aplicativo de distribución se prueba en un SIG para el turismo que sea altamente disponible a través de servicios de replicación, que permitan mantener el sistema estable y escalable.

La inclusión de las TI dentro del turismo en Colombia se ha hecho de manera pasiva y con un bajo impacto, tal es el caso, que hoy en día si bien existen algunos sitios donde se puede encontrar información útil para un visitante, dichos sitios presentan información desactualizada, con un contenido insuficiente y con muy pocas bases para la orientación de un turista. En estos sitios (organizaciones de turismo, sitios Web, como www.bogotaturismo.gov.co, www.idct.gov.co, www.bogota-dc.com, www.turismobogota.com, entre otros) la información se vuelve estática y si bien se hacen actualizaciones, ellas demoran periodos muy largos de tiempo y esto es un gran problema al momento de ofrecerle al visitante contenido que posiblemente ya no es vigente. Desde la perspectiva de los propietarios o administradores de negocios (hoteles, restaurantes, teatros, entre otros) esto es un problema, porque hacen una inversión "alta" en publicidad pero debido a la falta de dinamismo de la información resultan afectados ofreciendo a los visitantes condiciones de negocio que no van acordes con la realidad del mercado.

Por otro lado, para un visitante (o inclusive un residente) es necesario contar con la información adecuada de acuerdo a su posición geográfica y en algunos casos desde su PDA, por ejemplo para visitar los parques cercanos. También, puede ser necesario contar con un ayudante automático que permita organizar un itinerario de visitas de acuerdo a los intereses específicos y por una prioridad dada, luego sería interesante tener la descripción de cada uno de los sitios que se visitaran, entre otras cosas. Para lograr lo anterior, es necesario contar con un SIG que adicionalmente relacione información de horarios de atención, precios, entre otros. Al hacer un uso más efectivo de las TI en la

publicación de información turística, se puede ofrecer la posibilidad de tener mayor acceso a dicha información y ofrecer una información completa y actualizada para que el visitante se pueda guiar de acuerdo a su presupuesto y confort requeridos. Aunque este proyecto no pretende resolver todas estas necesidades de usuario final, con lo anterior se muestra un panorama altamente favorable para el desarrollo de proyectos tecnológicos en el sector turístico en Colombia.

1.2 JUSTIFICACION

Desde la perspectiva tecnológica, en este proyecto se integran varias tecnologías que en Colombia por lo general (con contadas excepciones) se han utilizado por separado, ellas son: los sistemas de información geográfica, utilizados principalmente para el registro catastral, la previsión de desastres ecológicos por derrames de petróleo, entre otros; los portales Web en Internet, cuya aplicación en turismo ha sido principalmente con páginas estáticas o con un bajo nivel de actualización, mientras que en otras áreas son altamente dinámicas, como en los sistemas de planeación de recursos empresariales (Enterprise Resource Planning, ERP), sistemas de manejo de la relación con el cliente (Customer Relationship Management, CMR), entre otros. Se plantea un modelo distribuido de datos relacionales que se acoplan con información relacional y georeferenciada y que se gestionan por el mismo sistema, además se utilizó una arquitectura de clientes inteligentes[1] y arquitecturas orientadas a servicios. Lo anterior sumado a la integración de dispositivos móviles, permitirá construir una herramienta software innovadora en el marco del Turismo de Colombia, y de Latinoamérica.

Desde la perspectiva académica y de investigación el trabajo se basa en el OGC[2] para lo cual fue preciso estudiar, conocer y apropiar los conocimientos relativos a este estándar y posteriormente estos conceptos se interrelacionaron con los servicios que la herramienta ofrece que permita la manipulación e interacción con los mapas y datos geográficos, logrando así no sólo un producto software, sino la apropiación de conceptos fundamentales

para desarrollar futuras investigaciones en el área. En forma transversal a todo el trabajo, se integraron conceptos de distribución de datos que comúnmente son implementados por los motores de bases de datos, y que en la mayoría de los casos son una caja negra para los ingenieros de sistemas que las usan. Si bien estos conceptos no se corre la barrera del conocimiento en la distribución de datos, si se convierte en un ejercicio creativo de recolección bibliográfica, análisis, síntesis, evaluación, formulación, aplicación y evaluación de algunos conceptos fundamentales que se usan en la construcción de dichos sistemas gestores de bases de datos.

El nuevo modelo propuesto desarrolla una lógica de distribución dentro de un aplicativo independiente, que plantea una forma diferente de atacar el problema de la distribución de datos, usando conceptos avanzados del tema en cuestión y permite orientar a los investigadores, en trabajos futuros sobre el mismo. Desde esta perspectiva, el proyecto es una investigación exploratoria que permitió determinar que el tema de distribución de datos como un servicio independiente de la aplicación y del motor de base de datos, es un tema de gran interés para ser profundizado por parte del GTI.

La utilización de los conceptos de ingeniería de software, patrones de software y el proceso unificado en el marco de la arquitectura de cliente inteligente, permitió dar algunos aportes significativos, relacionados con los problemas en la integración de todos los conceptos, lo apropiado que son ellos cuando trabajan juntos, sus fortalezas y algunas recomendaciones.

2 CONTRIBUCIÓN EN LA SOLUCIÓN

Para poder probar la idea de investigación, se formularon un objetivo general y tres objetivos específicos, dirigidos a proponer el método, desarrollar el ambiente computacional que soporte el método y la realización de un proceso de experimentación. Además se plantearon los productos que se perseguían como logro de los objetivos planteados. Los objetivos presentados a continuación, están redactados en futuro, para mantener su integridad con

respecto al plan de trabajo de investigación aprobado por la Facultad de Ingeniería Electrónica y Telecomunicaciones de la Universidad del Cauca.

2.1 OBJETIVOS

A continuación se presentan los objetivos en futuro para que coincidan con la propuesta del anteproyecto de grado aprobado por la Universidad.

2.1.1 OBJETIVO GENERAL

Desarrollar un modelo de datos relacional que permita la construcción de aplicaciones que manejen la lógica de la distribución (replicación) de datos sobre múltiples sistemas de gestión de bases de datos de distintos proveedores.

2.1.2 OBJETIVOS ESPECIFICOS

Los objetivos específicos contemplados en este proyecto son:

1. Definir un modelo conceptual y físico de datos que se puedan distribuir en múltiples sitios, teniendo en cuenta que estos datos pueden ser gestionados por diversos motores de bases de datos relacionales sin hacer uso de los esquemas propios de distribución provistos por esos motores.

2. Desarrollar una aplicación específica, en este caso un sistema de información geográfico (SIG) para el turismo, con los siguientes componentes:
   - Aplicación de Administración: Permite la administración de los sitio de interés turístico, usuarios, entre otros.
   - Aplicación móvil: Permite la visualización de información turística a través de un visor de mapas, y detalla cada sitio mostrando sus aspectos relevantes. Además permitirá realizar consultas tipo Query By Example (QBE).

3. Definir un conjunto de servicios Web XML, un catalogo maestro de distribución y unos procesos que permitan gestionar la distribución consistente de los datos relacionales en los distintos sitios del SIG basado en replicación asíncrona de instantáneas.

4. Definir unos lineamientos para diseñar un modelo relacional, cuando a través de este, se desea realizar la distribución de datos independizándose del motor de base de datos.

5. Desarrollar el SIG distribuido utilizando el paradigma orientado a objetos, modelado con los artefactos definidos en el Lenguaje Unificado de Modelado (Unified Modeling Language, UML), la arquitectura de cliente inteligente y los patrones asociados al mismo en Visual Studio .NET 2003

2.2 PRODUCTOS OBTENIDOS

Al final del proyecto de grado y acorde con los objetivos y la metodología planteada para el desarrollo del mismo, se obtuvo una serie de productos tangibles, de diferente clase: teóricos, tecnológicos y prácticos. Estos productos son:

- El presente documento, como una síntesis general del desarrollo de la investigación realizada para el desarrollo del proyecto, en ella se describe como se cumplieron los objetivos y el proceso seguido para la proposición del modelo, el ambiente computacional y la experimentación; los aportes significativos del trabajo, las conclusiones y las recomendaciones para trabajos futuros.

- Un modelo relacional de base de datos que soporta la replicación de datos en múltiples motores de base de datos, junto a una serie de lineamientos que permiten esbozar un trabajo futuro para la realización de nuevos componentes relacionados con el tema de replicación de base de datos.

- Un software denominado "Manager Distribution" que permite la administración de los servicios de replicación por instantáneas. Esta aplicación permite tener control total sobre las bases de datos que maneja una instancia determinada dentro de un servidor de base de datos y contiene una serie de servicios que son útiles al momento de administrar las actividades de replicación de datos.

- Un conjunto de servicios Web de publicación, distribución y suscripción que pueden ser utilizados para el desarrollo de aplicaciones que busquen personalizar los métodos de replicación a través del uso de referencias Web o bien una federación de servicios Web.

- Dos servicios Windows (agentes) que se encargan de monitorear las tareas de publicación y suscripción a través de los servicios web de replicación.

- Una aplicación para turismo con capacidad de utilizar los servicios de replicación y que prueban el correcto funcionamiento del modelo.

- Un artículo a publicar en la International Conference of Web Engineering (ICWE 2007), Julio 16-20 de 2007, Como, Italia.

CAPITULO II: CONTEXTO TEORICO

En esta parte se brindan los conceptos, principios teóricos y prácticos más importantes sobre la distribución/replicación de bases de datos, su definición, los elementos básicos y fundamentales, la descripción de algunos métodos de replicación existentes.

3 BASES DE DATOS DISTRIBUIDAS

3.1 DEFINICION[3]

Una base de datos distribuida es un tipo de base de datos virtual cuyos componentes están almacenados en diferentes bases de datos "reales" que pueden encontrarse distribuidas en sitios distintos, en otras palabras, es una sola base de datos lógica que físicamente se compone de diferentes bases de datos físicas. Un sistema distribuido de bases de datos debe tener en cuenta los siguientes principios: autonomía local, no dependencia de un sitio central, operación continua, independencia de ubicación, independencia de fragmentación, independencia de replicación, procesamiento de consultas distribuidas, administración de transacciones distribuidas, independencia del hardware, independencia del sistema operativo, independencia de red e independencia de DBMS.

Si bien, un sistema distribuido de datos ofrece muchas ventajas, también se deben tener en cuenta los siguientes inconvenientes:

- Se disminuye el procesamiento óptimo de las consultas.
- Requiere más soporte a la administración del catálogo. El catalogo del sistema debe incluir no solamente los datos usuales, como autorizaciones, vistas, etc, también debe guardar la información de control necesaria para permitir que el sistema proporcione la independencia de la ubicación, fragmentación y replicación necesaria. Luego se debe decidir como guardar el catálogo, para esto existen las siguientes opciones: Centralizado: Todo el

catálogo es almacenado en un sitio central. Completamente replicado: todo el catalogo se almacena por completo en cada uno de los sitios. Dividido: cada sitio mantiene su propio catálogo de los objetos almacenados en ese sitio y el catalogo total es la unión de todos los catálogos disjuntos. Centralizado/Dividido: cada sitio tiene su propio catálogo local, pero además hay un único sitio central que mantiene una copia unificada de todos los catálogos locales.

- **Actualización de datos**: Cuando se realiza una actualización en un objeto, éste debe replicarse a los otros sitios. Si en el momento cuando se propaga la actualización no está disponible el objeto que se va a actualizar, por ejemplo por un fallo de red, la actualización fallará. Para solventar dicho problema se usa un esquema denominado copia primaria.

- **Control de la recuperación**: Se basa en el protocolo de confirmación en dos fases. La confirmación en dos fases es necesaria en cualquier ambiente en donde una transacción única puede interactuar con varios administradores de recursos autónomos, escenario que se presenta más claramente en un sistema distribuido de bases de datos.

- **Control de la concurrencia**: en los sistemas distribuidos el control de la concurrencia se maneja con bloqueos, solamente que dichos bloqueos se tratan como mensajes, lo cual se convierte en un problema cuando existe sobrecarga de los mismos.

Para entender mejor la parte anterior, es importante tener claros los siguientes conceptos:

Confirmación en dos fases (commit en 2 fases): Esta es indispensable donde una transacción única puede interactuar con varios administradores de recursos independientes y más aun al tratarse de sistemas de bases de datos distribuidas. Para esto entonces se realiza lo que se llama una confirmación en dos fases, donde se requiere de mínimo un coordinador, quien es el encargado de garantizar que ambos administradores de recursos (para este caso los

diferentes DBMS's) confirmen o deshagan al unísono, las actualizaciones de las que son responsables y además proporcionar esa garantía incluso si el sistema falla en la mitad del proceso. Esto se logra cuando por ejemplo se emite una operación de confirmación (commit) o de deshacer (rollback) en la base de datos, entonces el coordinador realiza las siguientes dos fases: Primero: Informa a cada participante que debe forzar todos los registros de bitácora de los recursos usados por la transacción hacia su propia bitácora física, si la escritura se realizó satisfactoriamente el/los participantes envían un mensaje de confirmación al coordinador (en caso contrario confirma la operación como no exitosa). Segunda: Cuando el coordinador recibe todas las confirmaciones fuerza la entrada en su propia bitácora física, si algún mensaje de los participantes fue de operación no exitosa, deshace la transacción lógicamente.

Replicación de datos: Se da cuando un objeto dado puede ser representado por muchas copias o réplicas distintas, guardadas en diferentes sitios. La utilización de réplicas se utiliza porque pueden representar un mejor rendimiento así como mayor disponibilidad del objeto replicado. Su principal desventaja es que cuando el objeto replicado es modificado, sus replicas deben actualizarse. Para solventar ese inconveniente se presentan dos tipos de replicación:

- **Síncrona**: Si una réplica se actualiza, todas las demás réplicas del mismo fragmento se actualizan en la misma transacción. Su desventaja es la de imponer una sobrecarga sobre todas las transacciones que actualizan una réplica.
- **Asíncrona**: las actualizaciones de una réplica son actualizadas en un momento posterior, no en la misma transacción, por lo tanto hay momentos en donde las réplicas no son iguales. Si bien su ventaja es un alto rendimiento ya que no hay sobrecarga de replicación, su desventaja es que en algún momento los datos pueden ser inconsistentes y esto puede no ser deseable.

Redundancia Controlada: Disminuye las operaciones de E/S. Se dice la redundancia se controla cuando es administrada por el DBMS y esta oculta para los usuarios. Existen dos tipos de redundancia:

- Mantener copias exactas, o réplicas de los datos básicos.
- Mantener datos derivados además de los básicos, frecuentemente en tablas de resumen o de columnas calculadas o derivadas.

3.2 CONCEPTOS BASICOS DE LA REPLICACION DE DATOS[3]

Un sistema de base de datos distribuida es un sistema que está compuesto físicamente por 2 o más servidores con sistemas operativos y sistemas gestores de base de datos (DBMS) diferentes, pero lógicamente representan la misma unidad de información.

La interconexión (cualquiera que sea) entre dichos servidores hace posible su comunicación sin importar su ubicación geográfica.

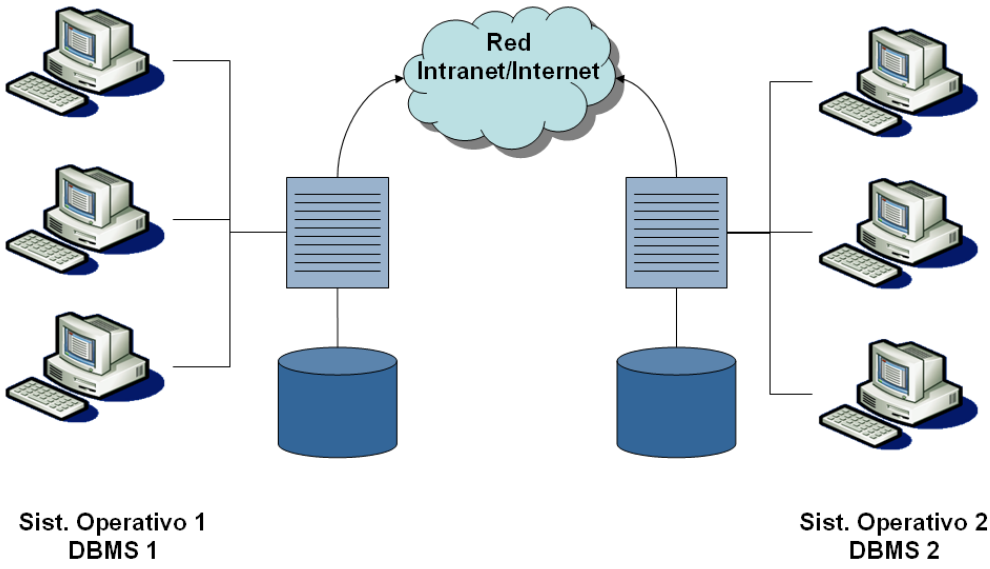

Figura 2 – Sistema de base de datos distribuida

Existe autonomía e independencia en cada DBMS para el tratamiento de los datos. El sistema de base de datos distribuido, permite establecer una sociedad entre los diferentes DBMS, agregándole componentes que hacen que dicho DBMS se comporte como parte de un solo DBMS que gestiona y sincroniza información de forma que lógicamente se trate como un solo sistema de administración de base de datos distribuida.

3.2.1 VENTAJAS DE LOS SISTEMAS DE BASE DE DATOS DISTRIBUIDOS[3]

Un sistema de base de datos distribuida permite enlazar información que físicamente se encuentra aislada. Generalmente las organizaciones se encuentran distribuidas y cada distribución implica que los datos e información tratada en cada parte se aíslen del conjunto lógico que representa la empresa. Estas divisiones son denominadas islas de información.
Las bases de datos distribuidas permiten acceder de manera remota a la información relevante que de igual forma puede ser tratada localmente y accedida de manera remota sin perder el concepto de unidad que debe representar la empresa.

Es importante distinguir entre un sistema de base de datos distribuida a un sistema que accede a datos remotos (cliente/servidor). Los sistemas que hacen uso de datos remotos, son sistemas que de una u otra manera hacen de forma evidente dicho uso. Los sistemas de bases de datos distribuidas no hacen uso evidente y tratan a la información como una unidad, al menos para el usuario final es así.
Existen 12 objetivos que deben cumplir los sistemas de bases de datos distribuidos para considerarse como tal, estos objetivos son importantes para caracterizar el comportamiento de los sistemas de base de datos distribuidos.

1. **Autonomía Local:** Los sitios que componen el sistema distribuido deben ser tan autónomos como puedan. Esto es, las operaciones en un sitio X son controladas localmente y su operación y funcionamiento no dependen de otros. Puede surgir la posibilidad que un sitio X le dé cierto control a

cierto sitio Y, por tanto los sitios deben ser autónomos en el mayor grado posible.

2. **No dependencia de un sitio local**: Si existe autonomía local todos los sitios deben tratarse como iguales. No debe haber un sitio "maestro" que sea el centro de operación. Si existe autonomía local este objetivo se da por añadidura, de no existir autonomía local completa es importante que se cumpla este objetivo para evitar cuellos de botella y que el sistema sea vulnerable.

3. **Operación continua**: Los sistemas de base de datos distribuido aseguran mayor confiabilidad y disponibilidad. Un fallo en algún punto del sistema no deshabilita a todo el sistema, éste puede seguir operando de manera completa o parcial aumentando el grado de disponibilidad y confiabilidad del sistema.

4. **Independencia de ubicación**: Los usuarios no deben saber donde se encuentran los datos ubicados físicamente. Para el usuario los datos deben ser tratados como una unidad de información, como si estos fueran almacenados en un único sitio central a pesar de que físicamente se encuentren esparcidos. Esto permite que las aplicaciones accedan a datos que no necesariamente se encuentren almacenados localmente.

5. **Independencia de fragmentación**: Esto implica que se dé un soporte a la fragmentación. Los usuarios deben poder hacer uso de los datos sin percatarse de que estos están fragmentados. Esto permite que las aplicaciones no se invaliden porque los datos se encuentren fragmentados en respuesta a requerimientos de rendimiento. Acceder a información fragmentada puede optimizar el rendimiento dado que el optimizador de consultas puede analizar una petición y saber a que sitio debe acceder para obtener la información solicitada. La actualización de una tupla que haga que sus condiciones de fragmentación cambien, hace que dicha tupla se traslade al fragmento que lo soporta.

6. **Independencia de replicación**: La replicación permite tener conjuntos de datos replicados en diferentes sitios, para dar mayor disponibilidad de los datos. Estas réplicas deben ser controladas y usadas para dar soporte a la disponibilidad de las aplicaciones. El usuario debe tratar la información de manera transparente sin percatarse de si los datos están o no replicados. La replicación implica directamente mayor procesamiento en actualización de datos, el optimizador del sistema debe saber que réplicas pueden o deben ser accedidas, habilitadas o deshabilitadas.

7. **Procesamiento de consultas distribuidas**: Implica acceder a diferentes partes físicas para retornar resultados (en el peor de los casos). La optimización es importante para mejorar los tiempos de respuesta. El soporte a estas operaciones se debe dar en el menor tiempo posible.

8. **Administración de transacciones distribuidas**: Por tratarse de un sistema distribuido se involucran dos a más sitios en los cuales una operación puede implicar acceder y manipular los datos en cada uno de esos sitios. Para esto un agente se encarga de realizar la o las operaciones para cumplir con la transacción. Se debe entonces controlar la recuperación y la concurrencia. La recuperación implica que una operación sea atómica, esto implica que el conjunto de agentes que hacen parte de una transacción (uno por cada sitio) confirme o rechace la operación (commit en dos fases). En cuanto a la concurrencia se debe controlar el bloqueo, de modo que no se presente un bloqueo mortal.

9. **Independencia de Hardware**: El sistema distribuido no debe depender del hardware. Cada uno de sus componentes deben ser socios igualitarios en la sociedad que representa el sistema distribuido de base de datos.

10. **Independencia del sistema operativo**: Para el sistema de base de datos distribuido el sistema operativo que este en cada máquina no debe ser evidente al usuario. Los sistemas operativos no deben ser restricción ni ventaja para el sistema de base de datos distribuido.

11. **Independencia de red**: No hay dependencia alguna con el tipo de red que soporte el sistema. Debe poder interactuar con diferentes tipos de redes.

12. **Independencia del DBMS**: Teóricamente un sistema de base de datos distribuido ideal debería soportar diferentes DBMSs, sin embargo, esto no siempre es tan fácil. Para que el sistema sea escalable es necesario el concepto de heterogeneidad. Pero no siempre es posible.

3.2.2 PROBLEMAS EN LOS SISTEMAS DISTRIBUIDOS[3]

El principal problema de los sistemas distribuidos de base de datos, es el uso de la red, bien sea WAN o LAN, es por esto que estos sistemas deben minimizar la cantidad de mensajes en la red y el volumen de los mismos. Este objetivo hace que se presenten los siguientes inconvenientes:

- Procesamiento de consultas.
- Administración del catálogo.
- Propagación de la actualización.
- Control de recuperación.
- Control de concurrencia.

- **Procesamiento de consultas**: El minimizar el uso de la red hace que el optimizador de consultas distribuya su tarea, haciendo una optimización global, seguido de optimizaciones locales en cada sitio afectado.
  El optimizador debe seleccionar la(s) estrategia(s) de acuerdo a la velocidad de datos y el retardo en el acceso. Se debe tener en cuenta que un sistema distribuido puede también realizar operaciones en paralelo en diferentes sitios lo que mejora el tiempo de respuesta.

- **Administración de catálogo**: El catálogo del sistema no solo incluye los datos base en relación a tablas, vistas, etc., sino también información de control que permite la fragmentación y replicación, por esto la

administración del catálogo puede almacenarse de diferentes formas entre las que están:

1. **Centralizado**: Se almacena en un solo sitio central.
2. **Completamente replicado**: El catálogo total se almacena completamente en cada uno de los sitios.
3. **Dividido**: Cada sitio (de distribución) tiene un catálogo con los objetos locales de cada sitio. El catálogo total es la unión de todos los catálogos.
4. **Centralizado y dividido**: Cada sitio tiene su propio catálogo local. El catálogo total es la unión de todos los catálogos pero se almacena en un único sitio central.

Inconvenientes:

1. **Centralizado**: Viola el objetivo de la no dependencia de un sitio central.
2. **Completamente replicado**: Pérdida de la autonomía, la actualización del catálogo debe ser replicada en todos los sitios.
3. **Dividido**: Aumenta el costo de operaciones, para acceder al menos a un objeto remoto hay que acceder al menos a la mitad de sitios.
4. **Centralizado y dividido**: Es mejor que tener el catalogo totalmente dividido pero se cae nuevamente en la dependencia de un sitio central.

En la vida real no se usa ninguno de estos enfoques. El enfoque que se usa es: Por cada objeto en la base de datos distribuida se tiene un sinónimo para que pueda accederse de manera independiente a su ubicación. Es decir, hay una tabla de sinónimos con la ubicación del objeto real. Esa tabla de sinónimos se encuentra en todos los sitios. Asi en el caso de acceder a un objeto que no esté localmente almacenado, se realizará una búsqueda en el sitio que esta referenciado en la tabla de sinónimos (primer acceso remoto), en caso de que este objeto haya migrado a otro sitio, se tendrá una nueva referencia al sitio que contenga el objeto requerido (segundo acceso remoto), en caso de una nueva migración, esta se

actualiza en la tabla de sinónimos en todas las tablas. Como se ve máximo son dos accesos remotos.

Hay catálogos locales y tablas de sinónimos en todos los sitios de todos los objetos globales. Dicha tabla tiene el sinónimo, nombre el objeto, quien lo creo y donde está almacenado. No hay un catálogo central.

- **Propagación de la actualización**: El inconveniente con la propagación es que en el momento de la propagación es posible que no todos los sitios estén disponibles. Luego la transacción fallaría. Por esta razón existen estrategias, una de ellas es:

  Esquema de copia primaria: Cada objeto replicado tiene designada una copia primaria y el resto son copias secundarias. Las copias primarias están en sitios diferentes de los objetos copiados. Una replicación se da si sus copias primarias son actualizadas, las copias primarias se encargan de replicar a las copias secundarias.
  En una replicación sincrónica puede existir el inconveniente de que no todos los sitios a los cuales se deben replicar estén disponibles, por esto se piensa en replicación asíncronas, las cuales traen como consecuencia que no siempre la base de datos sea consistente en todo momento.

  Control de la recuperación: Basado en la confirmación en dos fases (commit en dos fases). Esta es necesaria cuando una única transacción interactúa con varios DBMS en diferentes sitios. Por la autonomía que esto conlleva, surgen los siguientes puntos:

  1. *No dependencia de un sitio central*. Quien coordina la transacción generalmente debe ser alguien que inicia la transacción y no solo un sitio central. Luego todos los sitios deben ser capaces de coordinar y ser coordinados.
  2. *Confirmación en 2 fases*. Implica un alto tráfico en la red porque el coordinador debe controlar la confirmación en cada sitio.

3. *Pérdida de la autonomía local*. En el sentido de que los sitios coordinados deben responder a un solo sitio coordinador para confirmar la transacción.

Existen variaciones a la confirmación en dos fases que pueden volver más eficiente este proceso que son confirmar y deshacer de manera presupuesta. Confirmar de manera presupuesta, implica que los participantes notifican solamente los mensajes de cancelar y no los de confirmar y el coordinador puede olvidar la transacción tan pronto como haya trasmitido su decisión, siempre y cuando su decisión haya sido confirmar. Si falla algún participante entonces se interrogará al coordinador durante el reinicio. Si el coordinador aun recuerda la transacción, entonces la decisión debió haber sido deshacer, de lo contrario debió haber sido confirmar. Deshacer de manera presupuesta es un procedimiento similar, solo que los participantes no notifican los mensajes de deshacer.

Confirmar de manera presupuesta, debería ser la estrategia a seguir porque se presupone que todas las operaciones serán exitosas, pero puede darse el caso de confirmaciones falsas. Actualmente en los sistemas implementados, deshacer de manera presupuesta es la estrategia que siguen, porque evita sobrecarga de mensajes y asumen que las operaciones de deshacer se ejecutaron.

- **Control de Concurrencia**: En todos los sistemas de bases de datos el control de la concurrencia se hace basado en bloqueos. Los bloqueos son responsabilidad de cada DBMS, pero en un sistema de base de datos distribuido el reporte de bloqueos y desbloqueos son mensajes.

  Si se considera una operación con n equipos, para una transacción se habla que se requieren 5n mensajes, de la siguiente forma:
  - n solicitudes de bloqueo.
  - n confirmaciones de bloqueo.
  - n actualizaciones.

- n confirmaciones
- n solicitudes de desbloqueo.

Se podrían reducir el número de mensajes haciendo que en el mismo mensaje vaya una solicitud de bloqueo y actualización al igual que mensajes de confirmación y desbloqueo. Esto reduce el número de mensajes pero de igual forma el sistema se torna más lento que un sistema centralizado. Para agilizar el proceso se puede usar una estrategia de copia primaria y así solamente se haga el bloqueo en la copia primaria y que esta finalmente distribuya la operación a las copias secundarias en otro momento, pero por tratarse de un sistema distribuido, se puede presentar un bloqueo mortal. Sin embargo existen estrategias para detectar de manera eficiente bloqueos mortales aunque esto implique mayor mensajería a nivel de red.

3.2.3 INDEPENDENCIA DEL DBMS[3]

Para lograr este objetivo en los sistemas de base de datos distribuidos, se requiere que dos o más motores de base de datos compartan una misma interfaz, por ejemplo SQL. A continuación se presentan unos conceptos a tener en cuenta cuando se habla de replicación:

Pasarela: La pasarela es un programa especial propio de un motor de base de datos para que dicho motor parezca estar en los distintos otros sitios donde no está, independiente de en donde resida dicho programa y cumpliendo con las siguientes funcionalidades:

- Protocolos para intercambiar sentencias SQL y resultados de un motor a otro y viceversa.
- Permitir la ejecución de cualquier instrucción SQL no planeada. Generalmente esto se logra usando el soporte existente para SQL dinámico, o bien, usar una interfaz al nivel de llamada (como SQL/CLI u

ODBC). La pasarela puede hacer uso del procesador interactivo de SQL que viene ya con algunos motores como Oracle.

- Hacer transformaciones entre los tipos de datos soportados por los motores de base de datos involucrados.
- Transformar el dialecto SQL de un motor a otro. La semántica entre ellos puede ser diferente. La pasarela puede usar un medio de paso por medio del cual se puede escribir la sentencia SQL según como la entenderá el motor de base de datos que ejecutará la sentencia SQL.
- Transformar la información de retroalimentación.
- Suponiendo dos motores de base de datos. Transformar el catalogo del motor dos al formato del catalogo del motor uno. De esta forma los usuarios del motor uno saben que objetos hay en el motor dos y como comunicarse con dichos objetos (Ver Figura 1).
- Manejar el desacoplo semántico (diferencia de nombramiento, tipos de datos, diferencia de unidades, entre otros).
- Servir como participante en el protocolo de confirmación de dos fases. Para que la pasarela haga esta tarea, se necesita que el administrador de transacciones en el motor dos le de las propiedades para hacerlo.
- Asegurarse que el bloqueo en el motor dos se haga cuando el motor uno así lo requiera. La capacidad de la pasarela para hacer esto dependerá de las arquitecturas de bloqueo.

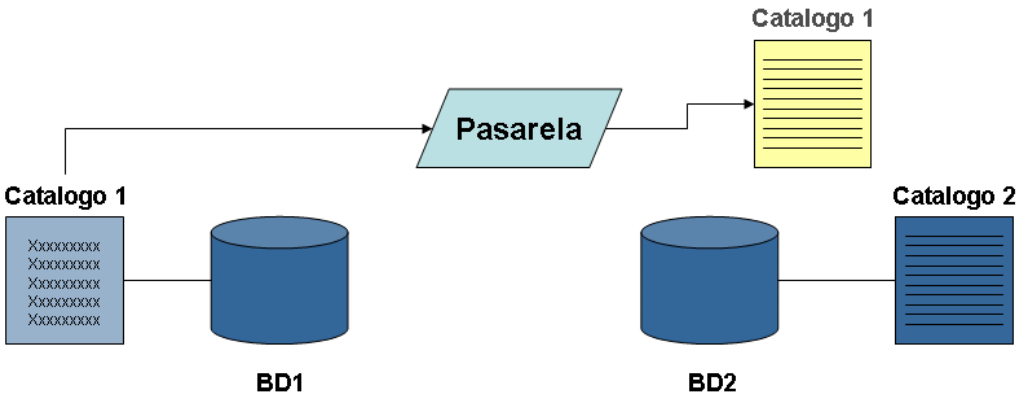

Figura 3 - Pasarela como transformación del catálogo

3.2.4 MIDDLEWARE PARA ACCESO A DATOS[3]

Los middleware es una clase de tecnologías software diseñadas para administrar la complejidad y heterogeneidad inherente en los sistemas distribuidos. Este es definido como una capa de software entre el sistema operativo y la aplicación que provee una abstracción común de programación a lo largo del sistema distribuido. Las middleware son informalmente llamadas "plomería", porque conecta partes de aplicaciones distribuidas con tuberías de datos y los pasa a través de ellas.[4]

Existen varios productos que son middleware de datos, en particular este software suele funcionar así (Ver Figura 4):

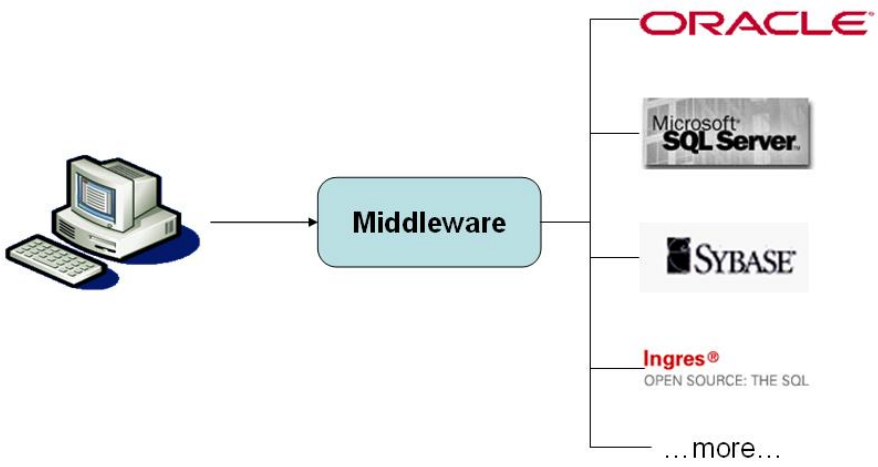

Figura 4 - Funcionamiento del middleware

El middleware actúa como un servidor de base de datos capaz de tener un catalogo local, procesar consultas SQL, etc., pero además puede incluso guardar datos, realmente esos datos están almacenados en una serie de sitios tras bambalinas que son DBMSs independientes.

Hasta este punto el sistema no es un sistema distribuido del todo, ya que las bases de datos no son concientes de la existencia de otras. Si se agrega un nuevo cliente del lado de las bases de datos y este solicita consultas al middleware se tendrá lo que se conoce como un sistema federado o bien un sistema de bases de datos múltiples.

Un sistema federado es un sistema distribuido usualmente heterogéneo, con una autonomía local cercana a la total. Las transacciones en estos sistemas locales son administradas por el DBMS local pero las globales son responsabilidad del middleware, quien mantiene un catálogo global.

4 ALGUNOS METODOS DE DISTRIBUCION

Los métodos de distribución varían de acuerdo a la latencia y autonomía, dos factores a tener en cuenta de acuerdo al ambiente donde se realizará la distribución. A continuación se listan algunos de los métodos más utilizados en la replicación de datos.

4.1 TRANSACCIONES DISTRIBUIDAS[5]

Una transacción de una sola instancia de motor de base de datos que abarque dos o más bases de datos es, de hecho, una transacción distribuida. La instancia administra la transacción distribuida internamente; para el usuario funciona como una transacción local.
En este esquema, todos los sitios tienen los mismos datos al mismo tiempo. Un coordinador de transacciones, facilita la distribución de transacciones usando el protocolo de confirmación en dos fases (commit en 2 fases) que garantiza que la transacción se complete en todos los sitios participantes al mismo tiempo.
En la aplicación, una transacción distribuida se administra de forma muy parecida a una transacción local. Al final de la transacción, la aplicación pide que se confirme (commit) o se revierta (rollback) la transacción. El administrador de transacciones debe administrar una confirmación distribuida

de forma diferente para reducir al mínimo el riesgo de que, si se produce un error en la red, algunos administradores de recursos realicen confirmaciones mientras los demás revierten la transacción. Esto se consigue mediante la administración del proceso de confirmación en dos fases (la fase de preparación y la fase de confirmación), que se conoce como confirmación en dos fases (2 Phase Commit).

- **Fase de preparación:** Cuando el administrador de transacciones recibe una solicitud de confirmación, envía un comando de preparación a todos los administradores de recursos implicados en la transacción. Cada administrador de recursos hace lo necesario para que la transacción sea duradera y todos los búferes que contienen imágenes del registro de la transacción se pasan a disco. A medida que cada administrador de recursos completa la fase de preparación, notifica si la preparación ha tenido éxito o no al administrador de transacciones.

- **Fase de confirmación:** Si el administrador de transacciones recibe la notificación de que todas las preparaciones son correctas por parte de todos los administradores de recursos, envía comandos de confirmación a cada administrador de recursos. A continuación, los administradores de recursos pueden completar la confirmación. Si todos los administradores de recursos indican que la confirmación ha sido correcta, el administrador de transacciones envía una notificación de éxito a la aplicación. Si algún administrador de recursos informó de un error al realizar la preparación, el administrador de transacciones envía un comando para revertir la transacción a cada administrador de recursos e indica a la aplicación que se ha producido un error de confirmación.

4.2 REPLICACION TRANSACCIONAL[6]

La replicación transaccional, hace una instantánea de los datos y esta es replicada a los suscriptores, luego cuando hay modificación de datos en el

publicador, las transacciones son individualmente capturadas y propagadas a los suscriptores. Este tipo de replicación es útil cuando:

- Se desea que los cambios sean propagados de manera incremental a los suscriptores apenas ocurren.
- Necesita que las transacciones se adhieran a las propiedades ACID[7].
- Los suscriptores están frecuentemente conectados al publicador.

Este tipo de replicación usa el log de transacciones para capturar cambios incrementales hechos sobre la tabla publicada. Los cambios detectados son propagados a los suscriptores y aplicados en el mismo orden en el que ocurrieron. De igual forma los cambios incrementales hechos en el publicador fluyen de acuerdo a la agenda de distribución del agente distribuidor. Esta agenda puede ser configurada para tener una mínima latencia o bien para asignar intervalos a los suscriptores. Cuando este tipo de replicación es utilizado sin actualización inmediata u opciones de colas de actualización, los cambios hechos en el publicador se hacen evitando conflictos de actualización. Dado el caso de que las opciones de actualización inmediata o que las opciones de colas de actualización estén activas, es posible realizar cambios dentro de los suscriptores, lo cual genera colas de actualización que pueden tener conflictos. Cuando los suscriptores necesitan tener la actualización de los datos cercanos al tiempo real, estos necesitan tener una conexión de red con el publicador. Este tipo de replicación puede proveer una baja latencia a los suscriptores.

4.3 REPLICACION POR INSTANTANEAS[6]

La replicación por instantáneas distribuye los datos tal cual como aparecen en el momento exacto que se solicita la replicación, este método no monitorea la actualización de datos. Este método es usado cuando los datos no cambien frecuentemente en el tiempo, o bien cuando se tolera altos niveles de latencia. Cuando la replicación ocurre, toda la instantánea es entregada a los suscriptores. El uso más frecuente de este tipo de replicación, ocurre cuando los datos tratados son generalmente de solo lectura y con poco cambio. En resumen los casos en los que este tipo de replicación es útil es:

- Los datos son en su mayoría estáticos y no cambian de manera frecuente.

- Cuando los datos publicados no se desactualizan en corto tiempo.
- Replicando pequeños volúmenes de datos en donde una entera actualización de datos es razonable.

Si bien este método de replicación como ya se menciono antes es apropiado cuando se necesita distribuir copias de datos de solo lectura (read-only), este también provee la opción de actualizar los datos en el suscriptor. Cuando los suscriptores leen datos, la consistencia transaccional es mantenida entre el publicador y el suscriptor. Cuando un suscriptor a una replicación por instantáneas, debe actualizar datos, la consistencia transaccional se mantiene entre el publicador y el suscriptor, gracias a que los datos son propagados usando el protocolo de confirmación en dos fases (2 Phase Commit), una característica de la opción de actualización inmediata. Este tipo de replicación requiere de una menor sobrecarga del procesador que una replicación transaccional, dado que no requiere de un continuo monitoreo de los cambios de los datos en los servidores fuente. Dado el caso de que los datos a trasmitir, es decir la instantánea se demasiado grande, se requiere un alto consumo de recursos de red para trasmitir.

4.4 REPLICACION MERGE[6]

Este método permite que tanto los publicadores como los suscriptores hagan actualizaciones sobre los datos mientras están conectados o desconectados y luego fusionar (merging) los cambios entre los sitios cuando estos están conectados. La replicación merge permite trabajar a cada uno de los sitios de manera independiente y un tiempo después fusionar todos los cambios obteniendo un resultado uniforme. Inicialmente una instantánea es distribuida a todos los suscriptores, después se marca los cambios de los datos publicados tanto en el publicador así como en los suscriptores. Los datos son sincronizados de manera continua a través de una agenda o bien por demanda. Dado que las actualizaciones se realizan en más de un servidor, los mismos datos deben ser actualizados en todos los servidores conectados, esto provoca que ocurran conflictos cuando las actualizaciones son fusionadas. Este tipo de replicación tiene una serie de configuraciones para la resolución de conflictos que pueden ser configuradas cuando se crea una publicación de este

tipo. De esta forma cuando un conflicto ocurre un solucionador de conflictos es invocado por el agente Merge y determina que datos son aceptados y propagados. Este tipo de replicación es útil cuando:

Múltiples suscriptores necesitan actualizar datos en varios momentos y propagar esos cambios al publicador o a los suscriptores.

Los suscriptores necesitan recibir datos, hacer cambios fuera de línea (offline) y después sincronizar esos cambios con el publicador y otros suscriptores.

No se esperan múltiples conflictos cuando los datos son actualizados en varios sitios, esto porque los datos fueron filtrados en particiones y luego publicados en diferentes suscriptores o bien por el uso de su aplicación, sin embargo, si existen conflictos, las violaciones de las propiedades ACID son aceptadas.

5 APLICACIONES DE CLIENTE INTELIGENTE

Las aplicaciones de cliente inteligente [1], son un paso en la evolución del desarrollo de aplicaciones software, ellas aprovechan las fortalezas de las aplicaciones de escritorio y las aplicaciones Web (ver Figura 5).

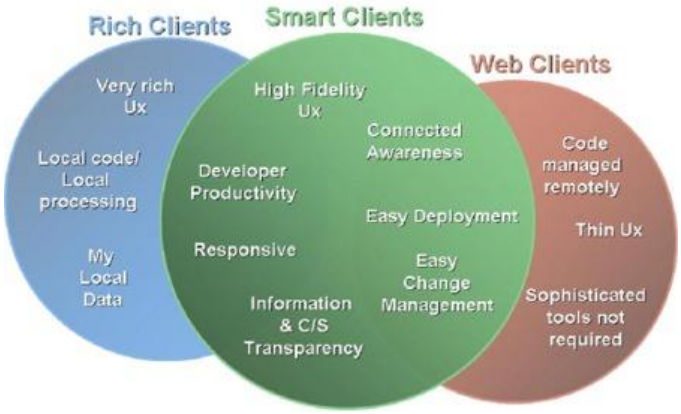

Figura 5 - Aplicaciones de Cliente Inteligente

Estas aplicaciones definen los siguientes atributos:

- **Experiencia de usuario de alta fidelidad**: Explotar las interfaces de usuario y la tecnología gráfica, así como personalizar cada usuario basado en el contexto.

- **Conexión Inteligente**: Inteligencia en línea y fuera de línea, tomando ventaja de los datos locales guardados en cache. Almacenamiento de datos centralizado y distribuido y un pre y post procesamiento en el servidor conectado a través de servicios Web.

- **Centralización de Información**: Los datos son fáciles de mantener en cache, pero se cuenta con un repositorio centralizado que es fácil de acceder y modificar.

- **Diseñado para operaciones**: Operaciones de seguridad, despliegue centralizado, un promedio de procesamiento de CPU local y adaptable al control de versiones.

Dentro de las ventajas que proporciona desarrollar aplicaciones de cliente inteligente se destacan las siguientes:

- **Combinar lo mejor de los dos tipos de aplicaciones (Desktop y Web):** De esta forma se mezcla el alcance de las aplicaciones para Internet con el poder de cómputo local.

- **Confiabilidad mejorada en ambientes de red heterogéneos**: Esto implica que pueden guardar datos localmente y usar la red de manera inteligente para proveer funcionalidad y mejorar las operaciones incluso cuando se está fuera de línea. Esta capacidad es de vital importancia sobre todo cuando el usuario móvil necesita ser productivo.

- **Incrementar el desempeño y la estabilidad**: Usa la capacidad de un sistema operativo local y el poder de cálculo de una CPU local, de esta forma se logra alto desempeño. La capacidad de correr código local

cargando muchas tareas que son hechas por los servidores en una aplicación Web típica.

- **Desarrollo de aplicaciones más rápidamente**: Los desarrolladores encuentran las aplicaciones inteligentes más fáciles de escribir, dado que ya no tienen que preocuparse por el manejo de sesiones y los viajes redondos al servidor. El modelo de programación es más intuitivo y la riqueza de código provista por las librerías de clases, ayuda a los desarrolladores a volverse más productivos.

- **Acceso a la funcionalidad de la máquina local**: En muchos casos las aplicaciones necesitan usar funcionalidad de hardware que existe en la máquina que puede ser accedida solo a través de una aplicación inteligente. Esta funcionalidad puede ser un dispositivo de entrada/salida, un acelerador gráfico, un software instalado en el cliente (DirectX), entre otros.

- **Integración con aplicaciones y sistemas Desktop existentes**: Se cuenta con un mecanismo de interoperabilidad entre aplicaciones que permite a los programadores desarrollar en una forma nativa. Además las aplicaciones construidas usando el Framework de .NET pueden conectarse con sistemas existentes y aplicaciones empaquetadas.

- **Facilidad de despliegue y seguridad**: Con un despliegue No-Touch[8], las aplicaciones inteligentes basadas en Windows, pueden desplegarse y actualizarse simplemente copiando los componentes necesarios desde un servidor Web que pueden ser accedidos por usuarios finales. Además usando un código de acceso de seguridad puede realizarse un mejor control del código ejecutado en el cliente inteligente, resultando una adecuada experiencia para el usuario.

- **Capacidades de sincronización de datos y soporte a la movilidad**: las aplicaciones inteligentes pueden ser entregadas en varias plataformas móviles con diferentes dispositivos (Smartphones, PDAs, Tablet PCs,

Laptops), compartiendo un modelo de programación unificado. Además si los recursos de la red no están habilitados para las aplicaciones móviles, los desarrolladores pueden adicionar sincronización de datos en la aplicación asegurando una operación transparente.

- **Soporte a Web Services y XML Nativo**: El Framework .NET, fue diseñado para soportar XML y servicios Web, esto facilita la manipulación de los datos y una fácil integración de la aplicación con sistemas heterogéneos.

- **Mejor interfaz y experiencia de usuario**: Las aplicaciones inteligentes, pueden usar funcionalidad del sistema operativo para proveer una rica e intuitiva experiencia de usuario, esto no depende de los viajes al servidor o a las capacidades de despliegue del navegador.

- **Acceso flexible a datos y Caching local de datos**: Las aplicaciones inteligentes pueden ser inteligentes en el sentido de que los datos son manejados con la aplicación. De acuerdo a las diferentes consideraciones, los desarrolladores pueden escoger entre guardar y usar datos locales o conectarse a una base de datos remota, esto trabaja cuando es necesario y no solo cuando la conexión con la red se habilita.

5.1 XML Y WEB SERVICES

XML[13] es el acrónimo del eXtensible Markup Language (lenguaje de marcado ampliable o extensible) desarrollado por el World Wide Web Consortium (W3C). Es una versión simple de SGML. Su objetivo principal es conseguir una página Web más semántica. Aunque una de las principales funciones con las que nace sería suceder al HTML, separando la estructura del contenido y permitiendo el desarrollo de vocabularios modulares, compatibles con cierta unidad y simplicidad del lenguaje (objetivo que se viene desarrollando a través de la especificación XHTML), tiene otras aplicaciones entre las que destaca su uso como estándar para el intercambio de datos entre diversas aplicaciones o software con lenguajes privados como en el caso

del SOAP. Al igual que el HTML, se basa en documentos de texto plano en los que se utilizan etiquetas para delimitar los elementos de un documento. Sin embargo, XML define estas etiquetas en función del tipo de datos que está describiendo y no de la apariencia final que tendrán en pantalla o en la copia impresa, además de permitir definir nuevas etiquetas y ampliar las existentes. Son varios los vocabularios desarrollados en XML con el fin de ampliar sus aplicaciones. Podemos considerar fundamentales: XHTML, XSL-FO y XSLT, XLink, XPointer y Schema. Además, existen también versiones para usos específicos, como MathML (fórmulas matemáticas), SVG (gráficos vectoriales), RSS (sindicación de noticias), o XBRL (partes financieros).

Un Web Service[14] es una colección de protocolos y estándares usados para intercambiar datos entre aplicaciones. Los programas escritos en varios lenguajes de programación y corriendo en diferentes plataformas pueden usar Web Services para intercambiar datos a través de Internet, de manera similar lo pueden hacer a través de comunicaciones entre procesos en un solo computador. Esta interoperabilidad es posible dado que está basada en estándares abiertos. OASIS y la W3C son los comités responsables de la arquitectura y estandarización de los servicios Web. Para mejorar la interoperabilidad entre las implementaciones de los Web Services la organización WS-I ha desarrollado una serie de perfiles para definir estándares involucrados.

6 BLOQUES DE APLICACIÓN (APPLICATION BLOCK)[15]

Los bloques de aplicación ayudan a ubicar los problemas comunes entre los desarrolladores y los proyectos software. Estos bloques encapsulan las mejores prácticas que recomienda Microsoft para las aplicaciones en .NET y pueden ser incluidos en los proyectos, fácil y rápidamente. Por ejemplo el bloque de aplicación de acceso a datos provee acceso a las características que son más usadas por ADO.NET, exponiéndolas a través de clases de fácil uso. Dentro de los bloques de aplicación que están implementados para el Framework de .NET son:

- **Bloque de aplicación de Caching**: Permite a los desarrolladores incorporar un cache local en su aplicación.
- **Bloque de aplicación de configuración**: Permite que las aplicaciones puedan leer y escribir información de configuración.
- **Bloque de aplicación de acceso a datos**: Permite a los desarrolladores incorporar funcionalidad estándar de bases de datos a sus aplicaciones (servicios mínimos de consulta y manejo de datos).
- **Bloques de aplicación de criptografía**: Permite incluir encriptación y funciones de hashing en las aplicaciones.
- **Bloques de aplicación de manejo de excepciones**: Permite a los desarrolladores y a los creadores de políticas, crear una estrategia consistente para procesar excepciones que ocurran en las capas de la arquitectura de las aplicaciones empresariales.
- **Bloques de aplicación de instrumentación y Logging**: Permite incorporar un logging estándar e instrumentar la funcionalidad de sus aplicaciones.
- **Bloque de aplicación de seguridad**: Permite a los desarrolladores incorporar funcionalidad de seguridad en las aplicaciones, como por ejemplo servicios de autenticación y autorización.

7 TURISMO Y LAS TECNOLOGIAS DE LA INFORMACION

Dentro de los estudios que se han realizado en el mundo acerca del sector del turismo y su relación con las tecnologías de la información, se encuentran ya varios trabajos adelantados en otros países en donde se demuestra que la inclusión de tecnologías dentro del sector turismo es un paso indispensable en la evolución de este mercado. A continuación se exponen ciertas posiciones que analizan el impacto de las tecnologías de la información dentro del sector del turismo.

7.1 SITUACION ACTUAL

La organización IMACMEXICO[1] dentro de un estudio acerca de la tecnología y el turismo, asegura que la tecnología facilita a los proveedores de turismo la flexibilidad para reaccionar ante la demanda del mercado y la capacidad para integrarse diagonalmente con otros proveedores al objeto de proporcionar nuevas combinaciones de servicios y mejorar la efectividad de los costes. El mismo impacto de un computador y la integración de sistemas y telecomunicaciones ofrecidos como servicios para el cliente hacen que hoy en día sea muy fácil encontrar un plan turístico con todo incluido desde un solo punto, que las agencias de viajes puedan acceder a información de disponibilidad de hoteles, vuelos, bares, restaurantes, etc, con el fin de armar un solo paquete turístico atendido en un solo punto. Adicional a esto, se encuentra el alto impacto que Internet genera en los clientes, puesto que ahora es posible consultar información de vuelos, hoteles e incluso información acerca de una localidad en particular a través de servicios basados en localización, mostrando mapas, videos, imágenes y todo esto al alcance de la mano del cliente. Por esto hoy en día es muy fácil encontrar mucha información disponible en Internet que ayude a armar un propio plan de turismo que se ajuste a las necesidades y gustos del cliente. Por otro lado, según Carolina Disegni en la publicación del 9 de agosto de 2004 del diario La Tercera de Chile, asegura que definitivamente gracias a Internet el mercado del turismo no solo se ha expandido sino que ahora es mucho más fácil de acceder, esto es una ventaja no solo para los turistas sino también para las empresas que ofrecen estos servicios, puesto que se han logrado alianzas entre las grandes y pequeñas y medianas empresas para ofrecer servicios de manera conjunta unificando todo el servicio en un solo portal. Cito: "Según explica Helen Kouyoumdjian, gerente general de la Corporación de Promoción Turística de Chile (CPT), quien presentó este tema en la última Cumbre Nacional de Turismo, "*a través de estos sistemas se tranza la mitad de las ventas internacionales de turismo". Una cifra que no es menor si se considera que las estadísticas de la Organización Mundial de Turismo registran que en*

[1] Iniciativa Mexicana de aprendizaje para la conservación. Disponible en Web: http://www.imacmexico.org

2003 hubo movimientos por US$ 500 mil millones en la industria, logrando un crecimiento anual del 4%"[9].

Concluyendo, el turismo y su relación con la tecnología han cambiado la forma de hacer turismo y es necesario que existan suficientes herramientas tecnológicas para soportar este cambio de no ser así, no se estaría soportando las nuevas tendencias y el nuevo turismo y se perdería el protagonismo en este sector. A continuación se explica una de las tecnologías más utilizadas para la creación de aplicaciones para el turismo.

7.2 SISTEMAS DE INFORMACION GEOGRAFICO

Existen muchas definiciones y estudios sobre los SIG, dado que a dichos sistemas convergen actualmente muchas disciplinas, entre ellas, la geografía, la ingeniería civil, ingeniería de sistemas, entre otras. Dentro de las definiciones más destacadas de lo que es un SIG se encuentran:

- Un sistema de hardware, software y procedimientos diseñados para facilitar la obtención, gestión, manipulación, análisis, modelación y salida de datos espacialmente referenciados, para resolver problemas complejos de planificación y gestión[10][11].
- Un sistema de computador capaz de mantener y usar datos con localizaciones exactas en una superficie terrestre[11].
- Un sistema de información geográfica, es una herramienta de análisis de información. La información debe tener una referencia espacial y debe conservar una inteligencia propia sobre la topología y representación[10][11].
- Un SIG se define como un conjunto de métodos, herramientas y datos que están diseñados para actuar coordinada y lógicamente para capturar, almacenar, analizar, transformar y presentar toda la información geográfica y de sus atributos con el fin de satisfacer múltiples propósitos[12].

Existen otras muchas definiciones de SIG, algunas de ellas acentúan su componente de base de datos, otras sus funcionalidades y otras enfatizan el hecho de ser una herramienta de apoyo en la toma de decisiones, pero todas

coinciden en referirse a un SIG como un sistema integrado para trabajar con información espacial, herramienta esencial para el análisis y toma de decisiones en muchas áreas vitales para el desarrollo, incluyendo la relacionada con la infraestructura de un municipio, estado o incluso a nivel nacional y/o mundial. Los componentes principales de un SIG son: Hardware, Software, Datos, Métodos y Personal (ver Figura 6). Todos estos componentes se relacionan sinérgicamente para lograr el objetivo específico para el cual fue diseñado el SIG[11].

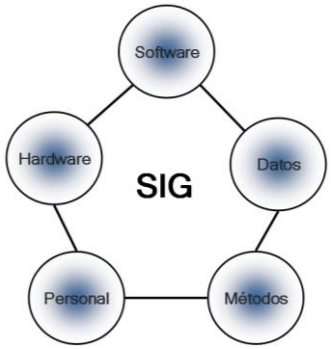

Figura 6 - Representación de un SIG

A continuación se presenta una descripción del papel que juegan las partes que componen el SIG y su comportamiento dentro del mismo.

Hardware: Es donde opera el SIG. Hoy por hoy, programas de SIG se pueden ejecutar en un amplio rango de equipos, desde servidores hasta computadores personales usados en red o trabajando fuera de línea (offline/desconectado) y en Asistentes Personales Digitales (Personal Digital Assistant, PDA).

Software: Los programas de SIG proveen las funciones y las herramientas necesarias para almacenar, analizar y desplegar la información geográfica. Los principales componentes de los programas para SIG son: Herramientas para la entrada y manipulación de la información geográfica, un sistema manejador de base de datos (DBMS – DataBase Management System), una herramientas que permitan búsquedas geográficas, análisis y visualización, y una Interfaz gráfica para el usuario (GUI) para acceder fácilmente a las herramientas.

Datos: Se requiere de adecuados datos de soporte para que el SIG pueda resolver los problemas y contestar a consultas de la forma más acertada posible. La consecución de datos correctos generalmente absorbe entre un 60% y 80% del presupuesto de implementación del SIG, y la recolección de los datos es un proceso largo que frecuentemente demora el desarrollo de productos que son de utilidad. Mantener, organizar y manejar los datos debe ser política de la organización.

Personal: Recurso humano que posee los conocimientos y habilidades para cumplir con cada una de las etapas de desarrollo y operación del SIG. La gestión de un SIG depende principalmente del potencial humano involucrado en el mismo.

Métodos: Un SIG operará acorde con un plan bien diseñado y con unas reglas claras del negocio, que son los modelos y las prácticas operativas características de cada organización.

La manera como se agrupan los diversos elementos constitutivos de un SIG quedan determinados por una serie de características comunes a varios tipos de objetos en el modelo, estas agrupaciones son dinámicas y generalmente obedecen a condiciones y necesidades especificas de los usuarios. La definición formal del concepto categoría o cobertura, queda determinado como una unidad básica de agrupación de varios mapas que comparten algunas características comunes en forma de temas relacionados con los objetos contenidos en los mapas. Sobre un mapa se definen objetos (tienen una dimensión y localización respecto a la superficie de la tierra), estos poseen atributos, y éstos últimos pueden ser de tipo gráfico o de tipo alfanumérico. A un conjunto de mapas relacionados se le denomina entonces categoría, a un conjunto de categorías se les denomina un tema y al conjunto de temas dispuesto sobre un área específica de estudio se agrupa en forma de índices temáticos o geoindice del proyecto SIG. De tal forma que la arquitectura jerárquica de un proyecto queda expuesta por el concepto de índice, tema, categoría, objetos y atributos.

Los SIG funcionan con dos tipos diferentes de información geográfica: el modelo vector y el modelo raster. El modelo raster funciona a través de una retícula que permite asociar datos a una imagen; es decir, se pueden relacionar paquetes de información a los píxeles de una imagen digitalizada. Los modelos raster además, tratan con fenómenos del mundo real que varían continuamente en el espacio. En el modelo vector, la información sobre puntos, líneas y polígonos se almacena como una colección de coordenadas x,y. La ubicación de una característica puntual, puede describirse con un sólo punto x,y, por tanto este modelo trata con fenómenos discretos, que no tienen un cambio constante con el tiempo. Las características lineales, pueden almacenarse como un conjunto de puntos de coordenadas x,y. Las características poligonales, pueden almacenarse como un circuito cerrado de coordenadas. Todo el conjunto de puntos o líneas conforman una capa.

Para el desarrollo del sistema de información geográfico, es necesario mencionar que el modelo que se aplicará para el diseño del mismo, se basa en el OpenGIS Reference Model (ORM)[2], esto para que el desarrollo del sistema cumpla con un estándar internacional que asegura la interoperabilidad entre los SIGs existentes y la calidad del producto.

CAPITULO III: MANAGER DISTRIBUTION, UNA IMPLEMENTACION DEL MODELO RELACIONAL DE DISTRIBUCION DE DATOS EN MULTIPLES MOTORES DE BASE DE DATOS

Teniendo en cuenta los conceptos asociados con la distribución y replicación de datos se presenta una descripción del desarrollo del componente más importante dentro del trabajo de grado que hemos denominado Manager Distribution, el cual representa un sistema de administración de replicación por instantáneas. Este producto software implementa el modelo relacional para el desarrollo de aplicaciones con distribución de datos en múltiples sistemas de gestión de bases de datos que a continuación se explica de manera detallada.

8 MANAGER DISTRIBUTION

Para entender bien el modelo y su implementación, a continuación se presenta una descripción general del mismo, luego se describen los componentes que hacen parte de la herramienta de distribución y finalmente se muestra su interacción con la herramienta para turismo.

8.1 DESCRIPCION

El modelo relacional para el desarrollo de aplicaciones con distribución de datos en múltiples sistemas de gestión de bases de datos, consiste en un conjunto de entidades relacionadas que dan soporte a las tareas de distribución de datos. Explícitamente y para este caso, la replicación por instantáneas. En este modelo, la responsabilidad de dichas tareas se centran en una serie de servicios Web que están disponibles para ser consumidos. De manera adicional el sistema cuenta con una aplicación que es utilizada para la administración de los servicios de replicación por parte de un administrador de base de datos, permitiéndole programar tareas de replicación de manera fácil

e intuitiva. Esta aplicación también va acompañada de una serie de servicios Windows (que trabajan como agentes/actores autónomos) que son los responsables de ejecutar las acciones programadas por el administrador de la base de datos.

De manera independiente al tipo de replicación se pueden separar claramente tres roles que están involucrados en las tareas de replicación que son: Publicador, Suscriptor y Distribuidor. Cada uno de ellos ejecuta tareas independientes bien definidas pero todas ellas se integran para completar la replicación de datos. Estos roles realizan sus tareas gracias a los servicios y agentes relacionados con cada una de sus tareas. A continuación se describe la misión de cada uno de los roles.

Publicador: Es el/los servidor(es) que se encargan de generar la publicación es decir el conjunto de datos que serán replicados a otros servidores (suscriptores). El publicador es el encargado de habilitar la publicación en el momento que se le solicite.

Suscriptor: Es el/los servidor(es) que reciben las replicas de los datos publicados por el/los publicador(es). "Recibir" esas replicas de datos consiste en actualizar la información que actualmente maneja cada uno de los suscriptores por la información recibida de los publicadores.

Distribuidor: Es el/los servidor(es) que se encargan de realizar la distribución de los datos publicados por el/los publicador(es) a el/los suscriptores(es).

Para ilustrar mejor el funcionamiento e interacción de estos roles se pueden referenciar a la Figura 7

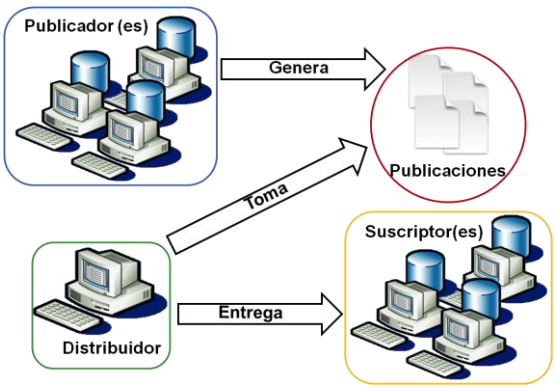

Figura 7 - Interacción de roles en la replicación

"Manager Distribution" es una aplicación que consta de tres servicios responsables de las tareas de distribución, publicación y de la carga de datos dentro de los suscriptores. Desarrolla la tarea de replicación por instantáneas y está compuesto de la siguiente manera. Dos servicios Windows encargados de las tareas de publicación y suscripción de publicaciones, una aplicación Windows como consola de administración para permitirle a un administrador de base de datos que realice la gestión de las publicaciones, las suscripciones y las formas de distribuir la replicación de los datos. Tanto los servicios Windows como la consola de administración consumen la lógica de distribución que está expuesta en un conjunto de servicios Web, estos servicios son indispensables para la realización de las tareas de distribución. Toda la información pertinente a esta distribución se encuentra en un motor de base de datos específico. A continuación se explica la interacción de todos los componentes descritos en la parte anterior para la replicación por Instantáneas. Además se hace claridad entre los modelos de suscripción existentes y como su implementación se ve reflejada en los servicios de replicación.

8.1.1 SUSCRIPCION POR INSERCION (PUSH)

El modelo de replicación por instantáneas con distribuidor local implica que el agente distribuidor reside en la máquina que genera la publicación. La suscripción por inserción también conocida como suscripción Push se comporta haciendo que los servicios de replicación residan en el distribuidor. Esto implica que las distribuciones se realizan con base en la agenda de publicaciones. El modelo de replicación por instantáneas con suscripción por inserción se muestra en la Figura 8.

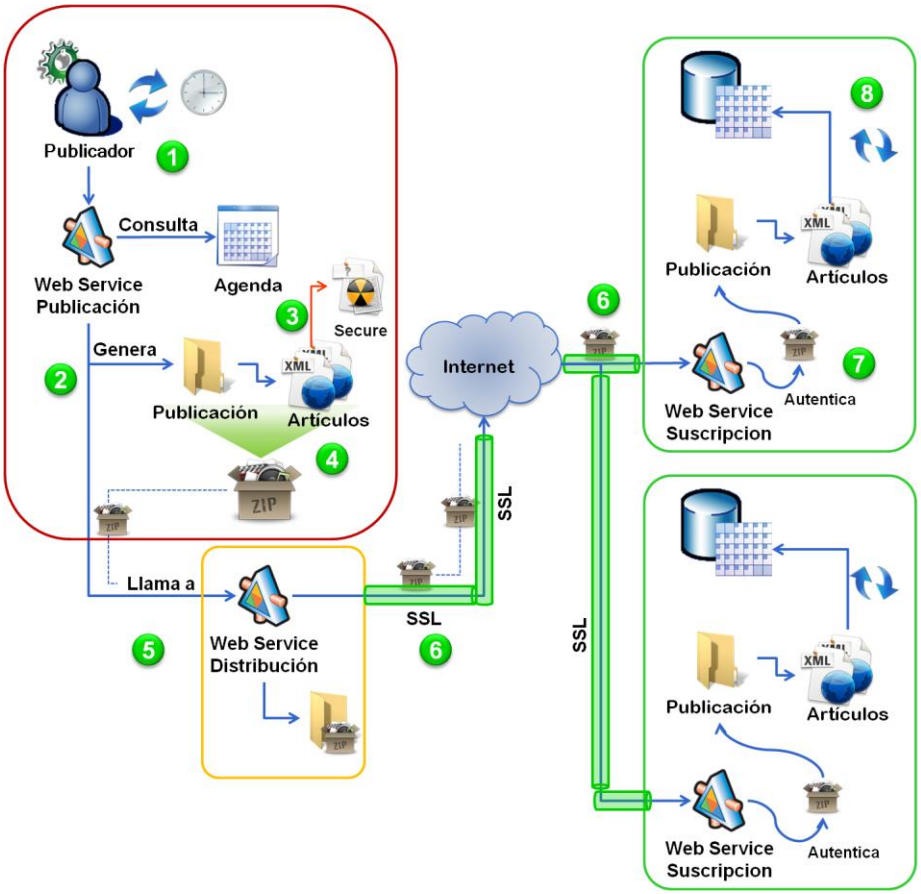

Figura 8 - Modelo de replicación por instantáneas con suscripción por inserción

La descripción del proceso de la replicación por instantáneas con suscripción por inserción se describe en los siguientes pasos:

1. El agente publicador cada cierto intervalo de tiempo (5 segundos) está verificando a través del servicio Web de publicación si dentro de la agenda hay una nueva publicación que generar.
2. El servicio web de publicación genera en la carpeta de publicaciones, la publicación en su respectiva carpeta y los respectivos artículos, repartidos en diferentes archivos XML.
3. Se genera un MD5 de cada uno de los artículos en un archivo que también se guarda en la carpeta de la publicación.

4. Se comprime toda la publicación, esto implica todos los artículos y el archivo de seguridad.
5. Ese archivo se pasa al servicio Web de distribución y se almacena en una carpeta de publicaciones a distribuir.
6. El servicio Web de distribución hace un llamado a los servicios Web de suscripción de los suscriptores que tienen suscripción por inserción (push) a esa publicación, pasándole el archivo comprimido que recibió en el llamado anterior a través de SSL.
7. El servicio Web de suscripción descomprime el archivo que le llega verificando el CRC del mismo y saca un MD5 a cada artículo comparándolo con el archivo de seguridad que viene dentro del archivo comprimido.
8. La publicación se sincroniza con la base de datos de suscripción.

Los pasos descritos anteriormente, así como la Figura 8 representan un esquema de replicación en donde el distribuidor se encuentra en un servidor independiente al de publicación y el de suscripción. Este esquema permite denominar al distribuidor, como un distribuidor remoto. En caso de que el distribuidor se encuentre configurado en el mismo servidor que está configurado como publicador, el distribuidor es denominado distribuidor local. Sin importar como esté configurado el distribuidor (local o remoto) las tareas descritas en la parte anterior se llevan a cabo de igual forma.

8.1.2 SUSCRIPCION POR EXTRACCION (PULL)

Este modelo al igual que el anterior puede tener un distribuidor local o bien, un distribuidor remoto. Lo que cambia en este caso, es el tipo de suscripción. Con la suscripción por extracción se define una agenda en donde se especifica el momento en el tiempo en el cual la sincronización de los datos en el suscriptor se debe llevar a cabo. Luego el llamado a los servicios Web de distribución se hace con base en la agenda de suscripción por extracción (Pull) y no con base en la agenda de publicación como se hace cuando se utiliza una suscripción por inserción (Push). (Ver Figura 9).

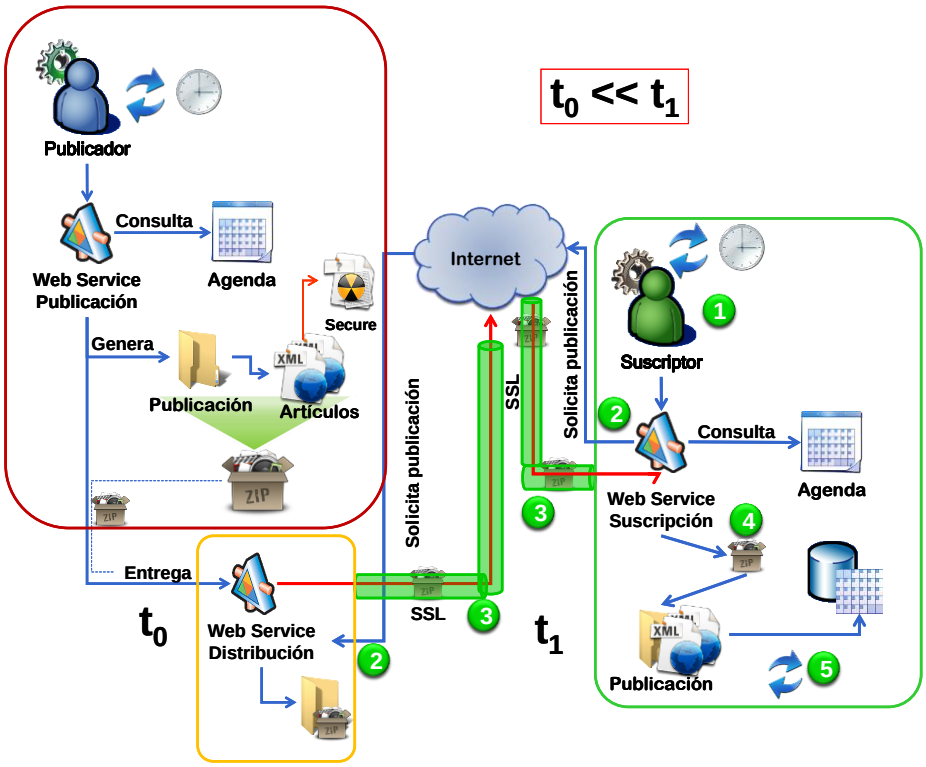

Figura 9 - Modelo de replicación con suscripción por extracción.

En una suscripción por extracción el comportamiento es diferente, básicamente en cuanto a los momentos (t) en los que ocurren el evento de creación de la publicación y el de solicitud de la misma por parte del suscriptor. Además, porque el origen del segundo evento lo genera el propio suscriptor con base en una agenda propia. Los pasos 1, 2, 3, 4, y 5 del lado del publicador son iguales a la suscripción por inserción. Luego en el lado del suscriptor, los pasos se describen a continuación:

1. El agente suscriptor se activa cada determinado intervalo de tiempo (5 segundos) y a través del servicio Web consulta la agenda de suscripciones.
2. El servicio Web de suscripción solicita al servicio Web de distribución la publicación correspondiente a la suscripción activa.
3. El servicio Web de distribución entrega la última publicación generada correspondiente al Id de publicación que solicitó el suscriptor y la entrega al servicio Web de suscripción a través de SSL.

4. El servicio Web de suscripción descomprime el archivo que le llega verificando el CRC del mismo y saca un MD5 a cada artículo comparándolo con el archivo de seguridad que viene dentro del archivo comprimido.
5. El servicio Web de suscripción sincroniza la publicación con la base de datos de suscripción.

8.1.3 CONFIGURACION DE ROLES Y SUS IMPLICACIONES

La configuración de los equipos involucrados en la red de replicación, pueden asumir cualquier de los tres roles o los tres al mismo tiempo. Esto implica que un publicador puede ser distribuidor y suscriptor a su vez. La configuración de los roles que cada uno de los equipos debe desempeñar es una labor del administrador del sistema de replicación. Cada uno de los equipos contendrá entonces un modelo completo del catalogo maestro de replicación (CMR), sin importar el rol que desempeñen dentro del sistema de replicación. Cada uno de los roles utiliza una porción del CMR y usa uno o varios de los agentes y servicios Web. (Ver Figura 10)

A continuación se especifican cada uno de los objetos utilizados por cada uno de los roles. Si un equipo desea configurarse como publicador, distribuidor y suscriptor, este utilizara todos los agentes y servicios involucrados para cada labor.

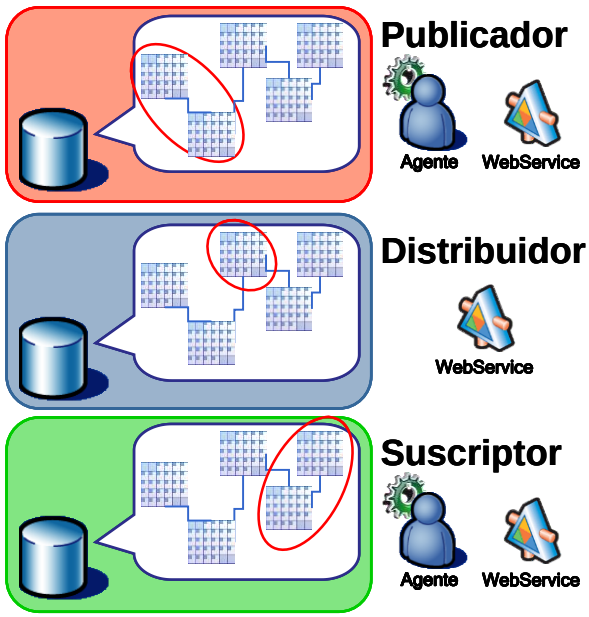

Figura 10 - Recursos usados en cada rol dentro del modelo de replicación.

El rol de Publicador como se muestra en la Figura 10 activa el agente de publicación y este a su vez usa los servicios Web de publicación. Dentro del modelo relacional propuesto cuando un servidor se configura como publicador, los objetos del CMR utilizados por este rol son: **Publication**, almacena la información de la publicación, tal como el nombre de la publicación, la base de datos de publicación y los identificadores de los artículos involucrados en la publicación. **Articles**, almacena la información de los artículos que componen una publicación en particular, los datos más importante a tener en cuenta aquí son la tabla o vista que define el artículo, las columnas que lo conforman y de ser necesario una condición que permita filtrar datos que requiera el administrador del sistema de replicación. **ExecutionPlan**, almacena información de la siguiente ejecución de una publicación (esta tabla se actualiza después de que una publicación se realiza con base a su programación y periodicidad). Esta tabla es consultada por el agente de publicación, para determinar que publicaciones se deben llevar a cabo en una determinada hora y fecha. **Programation**, almacena la información de la hora y fecha en las cuales la publicación debe llevarse a cabo. Dado que esta tarea generalmente puede tener intervalos de repetición y diferentes tipos de

periodicidad las tablas a continuación descritas ayudan a soportar este tipo de comportamientos. **Repetition**, almacena información acerca de qué tipo de periodicidad tiene una publicación en particular. Los tipos de periodicidad son: Diaria, Semanal, y Mensual. **Daily**, almacena información acerca de la periodicidad en días de una programación en particular. Es decir, en cuantos días una publicación debe llevarse a cabo. **Weekly**, almacena información acerca de la periodicidad en semanas de una programación en particular. Es decir en cuantas semanas debe llevarse a cabo una publicación en particular y explícitamente, que días de la semana. **Monthly**, almacena información acerca de la periodicidad en meses de una programación en particular.

El rol de Suscriptor como se muestra en la Figura 10 activa el agente de suscripción y este a su vez el servicio Web de suscripción. Dentro del CMR propuesto cuando un servidor se configura como suscriptor, los objetos de la base de datos utilizados por este rol son: **Suscription**, almacena la información relacionada con una suscripción en particular. Si esta suscripción es por inserción (push) esta no contará con programación independiente lo que implica que la sincronización de los datos se llevara a cabo una vez se emita la publicación. Si la suscripción es por extracción (pull), la suscripción contará con una programación independiente a la publicación. Es labor del administrador del sistema de replicación que exista coherencia entre las dos tareas. **ExecutionPlan**, almacena información de la siguiente ejecución de una suscripción (esta tabla se actualiza después de que una suscripción por extracción se realiza con base a su programación y periodicidad).Esta tabla es consultada por el agente de suscripción, para determinar que suscripciones se deben llevar a cabo en una determinada hora y fecha. **Programation**, Guarda la información acerca de la hora y fecha en las cuales la suscripción por extracción (pull) debe llevarse a cabo. Esta programación también se soporta en las tablas: Repetition, Daily, Weekly and Monthly.

El rol de Distribuidor como se muestra en la Figura 10 activa el servicio Web de distribución. Dentro del modelo relacional propuesto cuando un servidor se configura como distribuidor, los objetos de la base de datos utilizados por este rol son: **ServerRole**, almacena la información relacionada con los servidores

que hacen parte de la red de replicación y que rol desempeñan dentro de la misma. **PubToDistribute**, almacena información relacionada con las publicaciones que se deben distribuir. La información más importante acerca de las publicaciones a distribuir son: Id del servidor donde se encuentra la publicación y el Id de esa publicación para ese servidor en particular, así como una contraseña para poder verificar que un suscriptor determinado puede suscribirse a dicha publicación. **SusToDistribute**, almacena información acerca de las suscripciones hechas a las publicaciones que el distribuidor debe realizar. La información más importante a tener en cuenta en esta tabla, es el Id del equipo suscriptor, el id de la publicación a distribuir (Id de PubToDistribute), el Id del suscriptor y el tipo de suscripción.

8.2 MODELO RELACIONAL

La metodología utilizada para obtener el modelo fue la siguiente:

1. Estudio y análisis de modelos de replicación de varios productos de base de datos. A través de la revisión de esquemas de replicación y de los objetos adicionados a los modelos cuando los servicios mencionados se implementan, se logro tener una visión de los objetos necesarios para plantear el modelo relacional que soporta la persistencia de la información concerniente a la replicación. Se selecciono la metáfora de publicador/suscriptor para el desarrollo del modelo.
2. Con el objeto de confirmar la decisión tomada respecto a la metáfora y el diseño inicial del modelo relacional, se realizó un prototipo software que permitió validar los conceptos y sentar las bases para el diseño final del modelo y de su implementación. Este prototipo fue desechado en su mayoría pero fue clave para el éxito del proyecto.
3. Complemento del modelo relacional con un conjunto de aplicaciones que permitieran la administración de dichos servicios y a través de los cuales se facilite la creación de estrategias de replicación (adicional a lo planteado en el anteproyecto). La definición de interfaces usables para la administración de este tipo de servicios fue tomada con base en la revisión de las interfaces de los motores de base de datos analizados previamente,

de modo que la nueva consola de administración se acerque más al concepto definido por la metáfora de publicador/suscriptor.

4. A los servicios de administración se le incorporaron agentes (a modo de servicios Windows), los cuales se encargan de revisar constantemente, agendas de ejecución que realizan la tarea de replicar la información de acuerdo a una programación pre-establecida por el administrador de base de datos.
5. Se observa que el modelo planteado en primera instancia, ofrece una capa de servicios que se encarguen de la replicación ligado a un catálogo maestro (modelo de replicación) que soporte la información concerniente a la replicación, no son suficientes para soportar servicios independientes del motor de base de datos e independientes de una aplicación en particular. Es así como se plantea con los elementos mencionados con anterioridad un modelo no solo relacional sino un modelo general que abarca: un modelo relacional, un conjunto de servicios web, dos agentes encargados de monitorear las tareas de replicación y una aplicación de administración de replicación. Finalmente cabe notar que el tipo de replicación elegida por la pregunta de investigación planteada responde solo a un tipo de replicación que es la replicación por instantáneas, quedando así los tipos de replicación Merge y Transaccional como temas a desarrollar en trabajos futuros.

De esta forma el trabajo se reorienta debido a que cuando se obtuvo el modelo relacional que soporta la replicación de datos por instantáneas, se vio la necesidad de implementar servicios que hagan de este modelo algo más útil para la comunidad en general como lo son los servicios de administración y monitoreo. De esta manera se supera el objetivo general del trabajo de grado y se abren una serie de nuevas preguntas de investigación lo cual se considera como uno de los resultados más valiosos de este trabajo de grado.
En la Figura 11 se muestra el modelo propuesto.

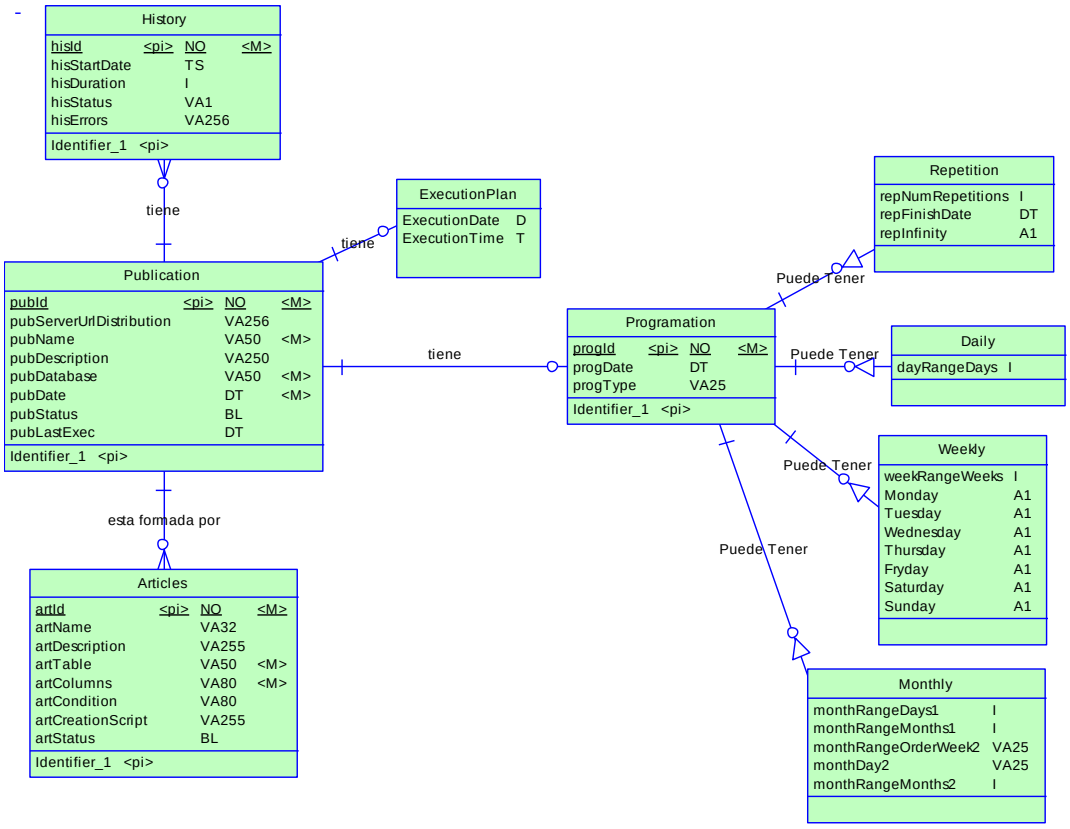

Figura 11 - Modulo de publicación

En la Figura 11 se pueden observar las tablas del CMR relacionadas con el rol de publicación, es decir, cuando un servidor está configurado como Publicador. El uso que se le da a cada una de las tablas se explico en la sección anterior. Cabe notar la aparición de dos nuevas tablas que se utilizan de manera general, sin importar el rol que se configure. Las tablas son: **History**, almacena la información acerca de las publicaciones (o suscripciones) realizadas, la hora de ejecución, la duración en la operación y su estado (Terminado (true)/ Incompleto (false)) En caso de que algún error se presente, el error se registra en el campo hisErrors dentro de la tabla. A continuación la Figura 12 muestra la sección que corresponde a las tablas comprometidas en la tarea de suscripción.

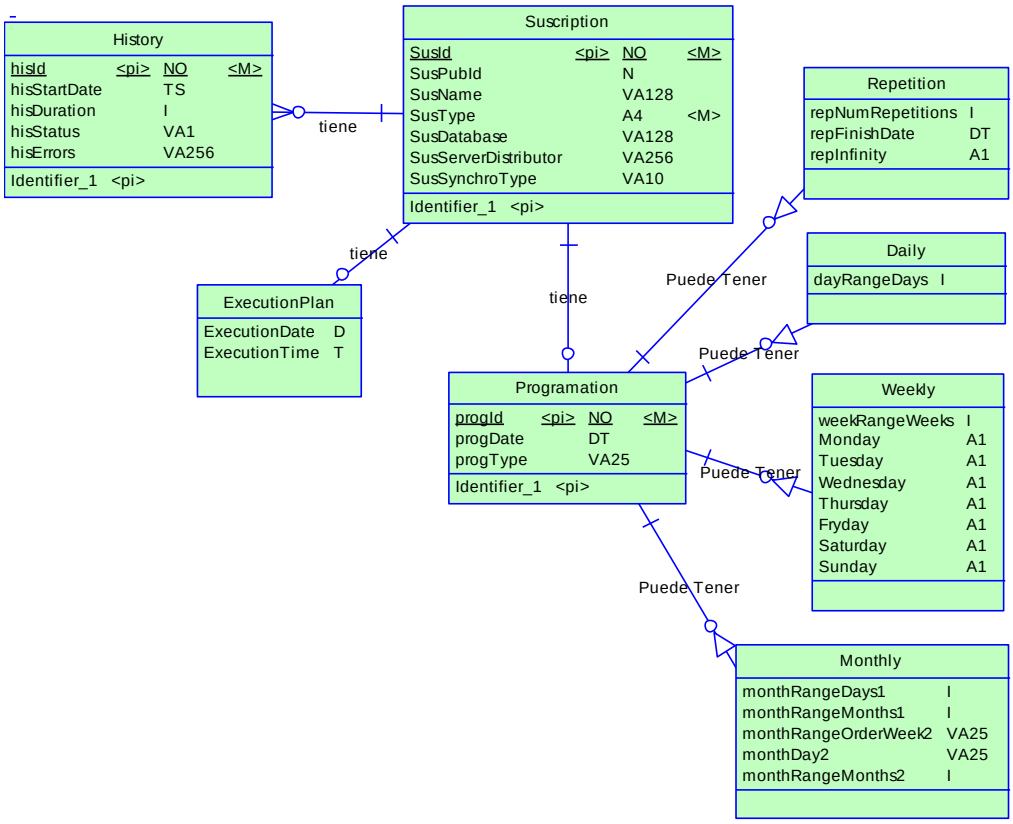

Figura 12 - Modulo de suscripción

Cabe notar que dentro del modelo que se muestra en la Figura 12 hay una existe un campo dentro de la tabla Suscription llamado SusPubId que hace relación al identificador de la tabla Publication, haciendo referencia a que publicación hace parte de una determinada suscripción. Esta relación lógica debe existir pero no puede ser una relación física real que implique integridad referencial. Esto porque el suscriptor a una publicación esta generalmente en un equipo independiente a donde se genera la publicación. Luego, lógicamente debe existir una conexión, pero esta no se representa físicamente en el modelo. De igual forma cabe notar que solo en el caso en que la suscripción sea una suscripción por extracción (Pull) se especifica una programación para esta suscripción, esto porque en el caso en que sea una suscripción por inserción (Push) esta se sincroniza con la agenda de publicación. Luego especificar una agenda diferente para este tipo de suscripción no se valida.

A continuación la Figura 13 muestra la sección del CMR que se usa para dar persistencia a la información de distribución de las publicaciones.

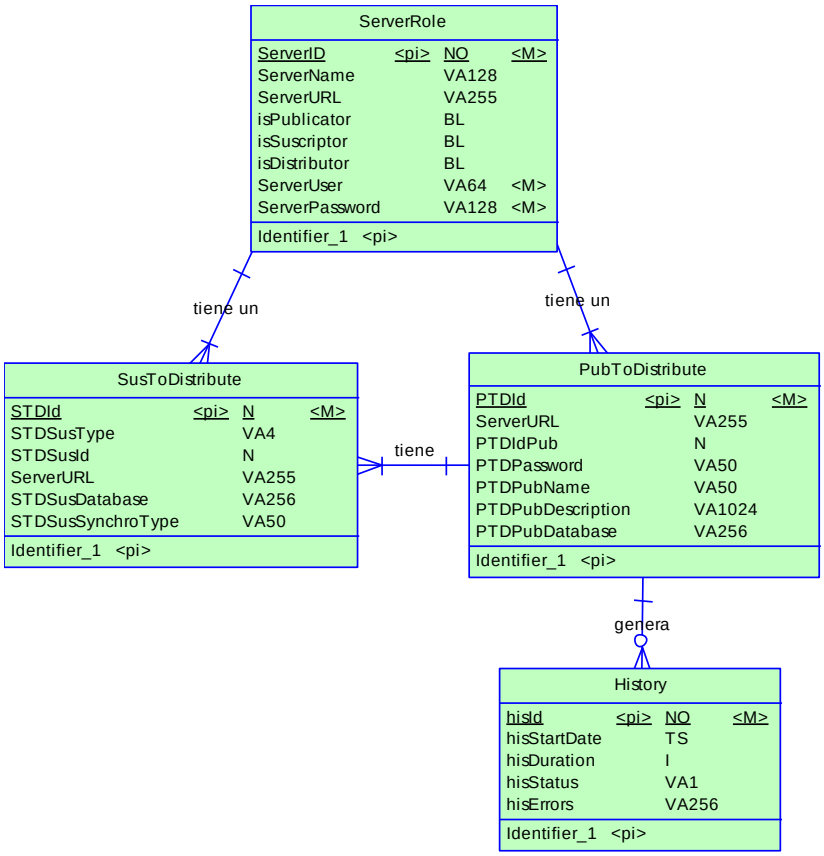

Figura 13 - Modulo de distribución

Al igual que el modulo de suscripción se observa la inclusión de la tabla History, ésta tiene la misma labor que en el anterior modelo, la de persistir información acerca de la realización de las actividades y en caso de la existencia de algún error en cualquiera de las actividades de replicación, en este caso la de distribución, poder persistir esos datos.

8.3 PROGRAMACION DE SERVICIOS WEB DE REPLICACION

La programación/codificación/implementación de los servicios de replicación se realizo con un conjunto de servicios Web que requieren de autenticación para ser consumidos/usados. Dado que la correcta implementación de los servicios

es la base para el correcto funcionamiento de los servicios de replicación, fue necesario realizar un ciclo de desarrollo completo que asegurará el buen funcionamiento de los mismos. A continuación se describe cada una de las fases.

8.3.1 FASE DE ANÁLISIS

En esta fase se construye un modelo conceptual de todos los servicios relacionados. Esta parte es de vital importancia para el entendimiento y delegación de responsabilidades a las clases que representaran los conceptos necesarios para realizar las tareas de distribución.
La Figura 14 muestra el diagrama de casos de uso general del sistema.

Figura 14 - Diagrama de casos de uso Servicio Web de Distribución

El diagrama representado en la Figura 14 muestra las actividades más relevantes a tener en cuenta para la correcta implementación de los servicios, así como los actores que consumirán de manera directa o indirecta los servicios. Los casos de uso presentados en el diagrama fueron llevados a cabo por el sistema a través de funciones que sirvieron de base para llevar a cabo cada una de las tareas presentadas por los casos de uso. Las siguientes

secciones describen las funciones y describen de una forma más detallada cada caso de uso.

8.3.1.1 FUNCIONES DEL SISTEMA

Fabricar Publicación

Ref #	Función
R1.1	Realizo la conexión a la tabla origen
R1.2	Ejecutar los comandos SELECT de las tablas especificadas para cada artículo
R1.3	Guardar los resultados de la consulta en un archivo XML, preservando el esquema
R1.4	Saca un MD5 de cada archivo.
R1.5	Realiza una compresión de los artículos y el archivo de MD5

Tabla 1 - Funciones relacionadas a Fabricar una Publicación

Sincronizar Instantánea

Ref #	Función
R2.1	Abre el archivo XML, leyendo su esquema.
R2.2	Borro el contenido de la tabla destino.
R2.2.1	Si existe condición de borrado, solamente borro el contenido que coincida con la condición
R2.3	Inserta los campos del XML obtenido en la tabla destino.

Tabla 2 - Funciones relacionadas a Sincronizar Instantánea

Definir Programación

Ref #	Función
R3.1	Fijar programación (Fecha, Hora, Periodicidad)
R3.2	Guardar programación en lista de programaciones.

Tabla 3 - Funciones relacionadas a Definir Programación

Revisar Programación

Ref #	Función
R4.1	Abre archivo de programaciones (Instantánea, Sincronización y

	distribución)
R4.2	Recorre el archivo y compara con la hora actual
R4.2.1	Si existe una programación hecha en el tiempo revisado, se llama a la actividad "Fabricar Instantánea", "Sincronizar Instantánea" o "Distribuir Publicación" en cada caso respectivamente.

Tabla 4 - Funciones relacionadas a Revisar Programación

Registrar Servidor

Ref #	Función
R5.1	Determinar la existencia del equipo que se desea agregar (ping)
R5.2	Escribe la URL del equipo en el registro de los equipos (con el campo clave vacío)
R5.2.1	Delegación de permisos al equipo que registró.
R5.3	Envío de clave a través de SSL
R5.4	Almacenamiento de clave con llave privada

Tabla 5 - Funciones relacionadas a Registrar Nodo

Distribuir Publicación

Ref #	Función
R6.1	En cada origen leer los destinos respectivos.
R6.2	Extrae la publicación del origen.
R6.3	Escribir la publicación en los destinos.

Tabla 6 - Funciones relacionadas a Distribuir Publicación

Gestionar Publicación

Ref #	Función
R7.1	Selecciona la tabla a replicar, asignando o no una condición de filtro
R7.1.1	Fijar programación (Fecha, Hora, Periodicidad)
R7.1.2	Guardar programación en lista de programaciones de publicaciones
R7.2	Modificar publicación
R7.3	Borrar publicación
R7.4	Listar publicaciones

Tabla 7 - Funciones relacionadas a Gestionar Publicación

Gestionar Suscripción

Ref #	Función
R8.1	Define la publicación a la que se suscribe
R8.1.1	Fijar programación (Fecha, Hora, Periodicidad)
R8.1.2	Guardar programación en lista de programaciones de sincronización
R8.2	Modificar suscripciones
R8.3	Borrar suscripción
R8.4	Listar Suscripciones

Tabla 8 - Funciones relacionadas a Gestionar Suscripción

Asignar Roles

Ref #	Función
R9.1	Se autentica el servidor al cual se le desea asignar un rol.
R9.2	Delega el rol. (Publicador, Suscriptor o Distribuidor)

Tabla 9 - Funciones relacionadas a Asignar Permisos

8.3.1.2 DESCRIPCIÓN EXTENDIDA DE CASOS DE USO

A continuación se describen cada uno de los casos de uso presentados en la Figura 14.

Caso de Uso:	**Generar Publicación**
Actores:	Agente Publicador, Manager Distribution
Propósito:	Crear una copia de los datos de una o varias tablas especificadas en un momento del tiempo, en base a una programación definida.
Resumen:	El publicador o la aplicación Manager Distribution, inicia la actividad creando una selección de los datos a replicar de cada uno de los artículos. El sistema guarda una copia en memoria local del resultado de la selección en archivos XML en una ruta especificada por el sistema. A cada uno de los archivos se les saca un MD5 y se almacena esa

	información y se comprime toda esa información en una ruta especificada por el sistema.
Tipo:	**Primario, Esencial**
Referencias Cruzadas:	*R1.1, R1.2, R1.3, R1.4, R1.5*

Tabla 10 – Caso de Uso Generar Publicacion - Extendido

Caso de Uso:	**Sincronizar Instantánea**
Actores:	Agente Suscriptor, Manager Distribution
Propósito:	Actualizar los datos de acuerdo a la última publicación
Resumen:	El suscriptor o el Manager Distribution, selecciona una publicación que ya este distribuida borra los datos actuales (todos o los que cumplan la condición) e inserta los datos de cada uno de los artículos que hacen parte de la publicación.
Tipo:	**Primaria, Esencial**
Referencias Cruzadas:	*R2.1, R2.2, R2.2.1, R2.3*

Tabla 11 - Caso de Uso Sincronizar Instantánea - Extendido

Caso de Uso:	**Definir Programación**
Actores:	Manager Distribution
Propósito:	Definir la programación de una actividad (publicar, sincronizar), especificando una periodicidad, para una fácil administración
Resumen:	El administrador del sistema a través de Manager Distribution define una programación para una actividad específica, señalando datos como Hora, Fecha, periodicidad y finalización de la misma.
Tipo:	**Primario, Esencial**
Referencias Cruzadas:	*R3.1, R3.2*

Tabla 12 - Caso de Uso Definir Programación - Extendido

Caso de Uso:	**Registrar Servidor**
Actores:	Manager Distribution
Propósito:	Agregar un servidor a la red de distribución.
Resumen:	El administrador registra un nuevo nodo a través de la URL de dicho nodo. Se espera una confirmación del nodo agregado y se envía la clave de distribución al nodo agregado.
Tipo:	**Primaria, Esencial**
Referencias Cruzadas:	*R5.1, R5.2, R5.2.1, R5.3, R5.4*

Tabla 13 - Caso de Uso Registrar Nodo - Extendido

Caso de Uso:	**Distribuir Publicación**
Actores:	Servicio Web de Publicación y Servicio Web de Suscripción.
Propósito:	Distribuir las publicaciones de los publicadores a los suscriptores
Resumen:	Los servicios Web entregan/solicitan la publicación seleccionada para distribuirla a una lista de suscriptores, disponible en cada publicador o bien solicitar una publicación a un publicador (suscripción pull).
Tipo:	**Primaria, Esencial**
Referencias Cruzadas:	*R6.1, R6.2, R6.3*

Tabla 14 - Caso de Uso Distribuir Publicación - Extendido

Caso de Uso:	**Definir Roles**
Actores:	Manager Distribution
Propósito:	Define el rol que un servidor específico desempeñara dentro de la red de distribución.
Resumen:	El administrador a través de Manager Distribution configura el rol de un servidor registrado en la red para definir su comportamiento dentro de la misma.

Tipo:	**Primaria, esencial**
Referencias Cruzadas:	*R9.1, R9.2*

Tabla 15 - Caso de Uso Definir Roles - Extendido

Caso de Uso:	**Gestionar Publicación**
Actores:	Manager Distribution
Propósito:	Gestionar la publicación y su programación
Resumen:	El administrador selecciona la(s) tabla(s) de las cuales se desea hacer replicación, así como una condición de filtrado. De igual forma define una programación para realizar la actividad con alguna periodicidad. Además puede editar, borrar y listar todas las publicaciones programadas y sus programaciones asociadas.
Tipo:	**Primaria, Esencial**
Referencias Cruzadas:	*R4.1, R4.2, R4.2.1, R7.1, R7.1.1, R7.1.2, R7.1.3, R7.2, R7.3, R7.4*

Tabla 16 - Caso de Uso Gestionar Publicación - Extendido

Caso de Uso:	**Gestionar Suscripción**
Actores:	Manager Distribution
Propósito:	Gestionar la suscripción y la programación de la misma en caso de ser suscripción Pull
Resumen:	El administrador a través de Manager Distribution, selecciona una publicación y asocia un nuevo suscriptor, en caso de ser una suscripción push, la agenda de suscripción se sincroniza con la agenda de publicación. Cuando la suscripción es Pull, la publicación será sincronizada dentro del suscriptor con una programación que puede ser definida, determinando una fecha de inicio, periodicidad y fecha de finalización. Además puede editar, borrar y listar todas las suscripciones programadas.

Tipo:	**Primaria, Esencial**
Referencias Cruzadas:	*R8.1, R8.1.1, R8.1.2, R8.2, R8.3, R8.4*

Tabla 17 - Caso de Uso Gestionar Suscripción – Extendido

8.3.1.3 DIAGRAMAS DE SECUENCIA

En este punto es necesario ver qué tipo de interacción existe entre cada actor y el sistema en cada uno de los casos de uso mencionados. Esta interacción se puede ver fácilmente a través de las siguientes figuras que muestran diagramas de secuencia por cada caso de uso.

Caso de Uso: *Asignarrol*

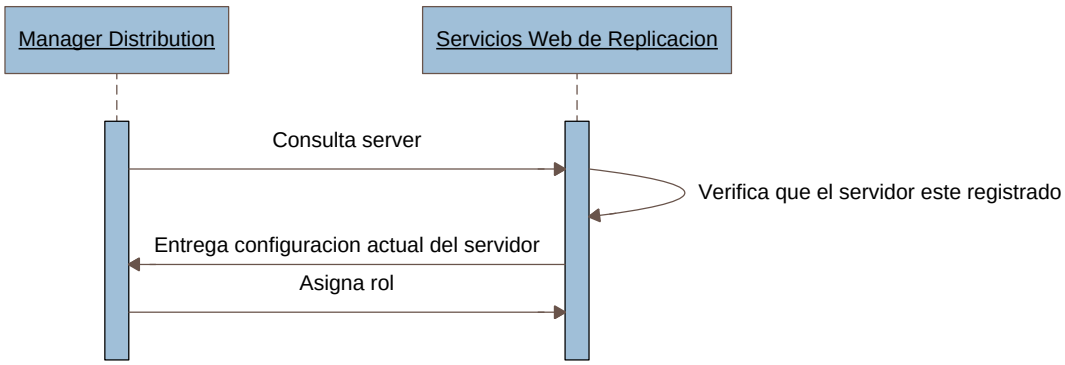

Figura 15 - Diagrama de secuencia – Definir rol

Caso de Uso: *Generar Publicación*

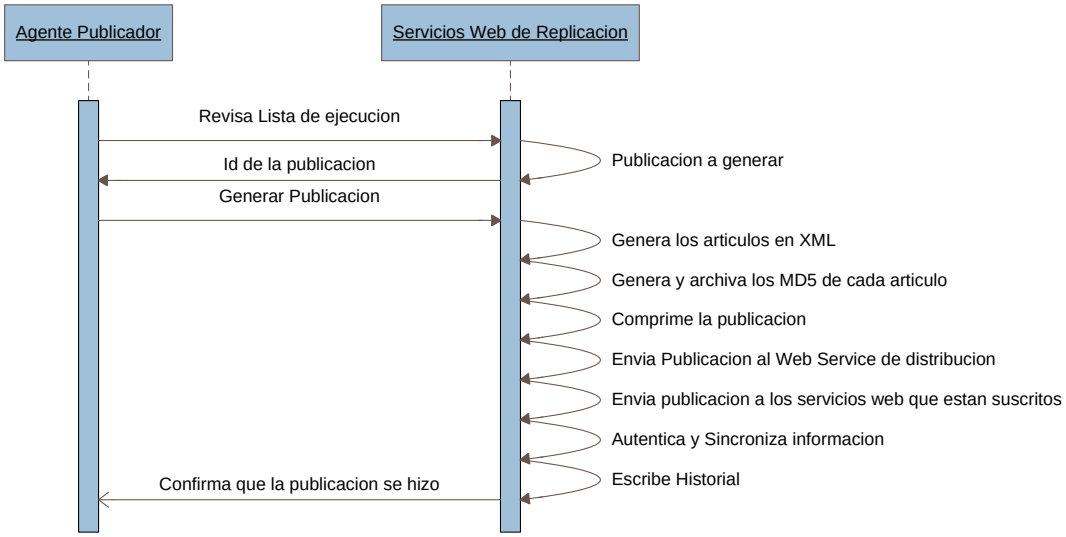

Figura 16 - Diagrama de secuencia – Generar publicación

Caso de Uso: *Definir programación*

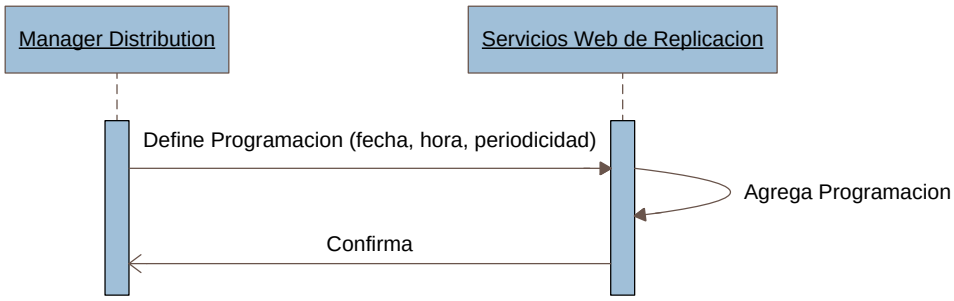

Figura 17 - Diagrama de secuencia – Definir Programación

Caso de Uso: *Registrar Servidor*

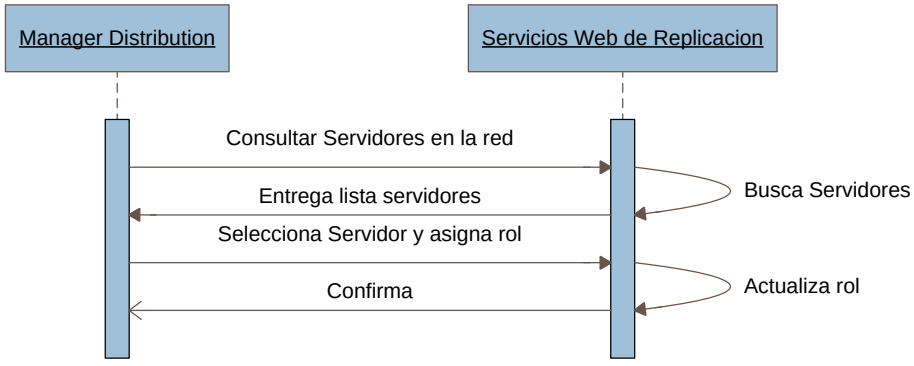

Figura 18 - Diagrama de secuencia – Registrar servidor

Caso de Uso: *Sincronizar Publicación*

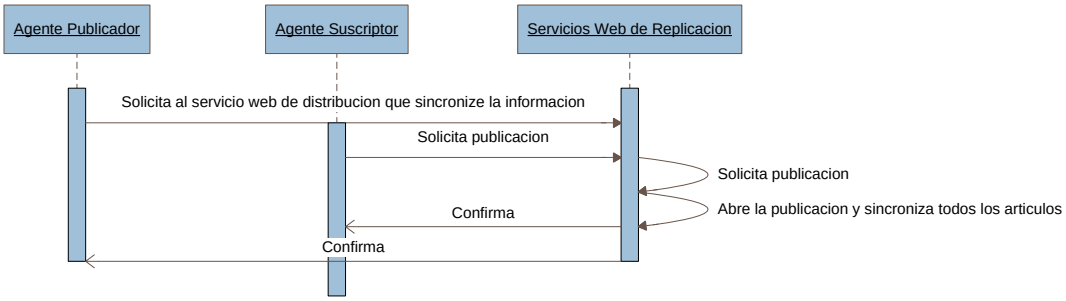

Figura 19 - Diagrama de secuencia – Sincronizar Publicación

Caso de Uso: *Gestionar Publicación*

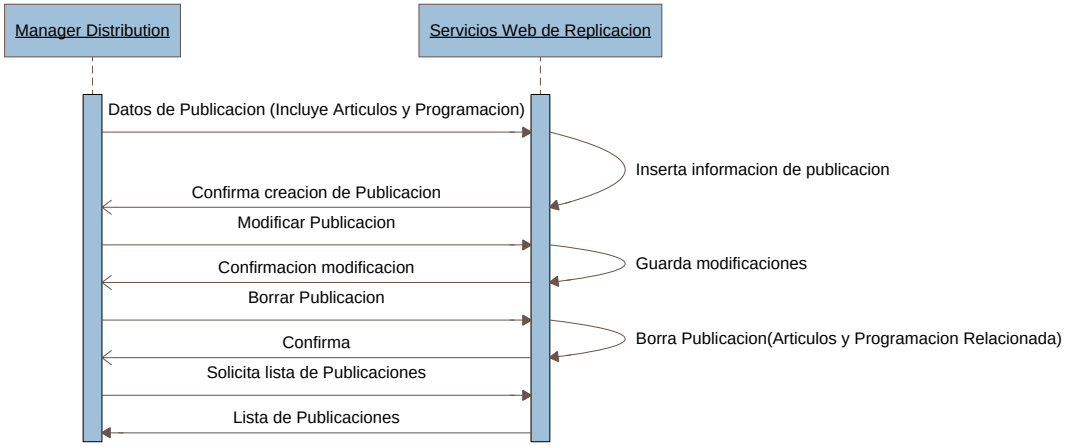

Figura 20 - Diagrama de secuencia – Gestionar Publicación

Caso de Uso: *Distribuir Publicación*

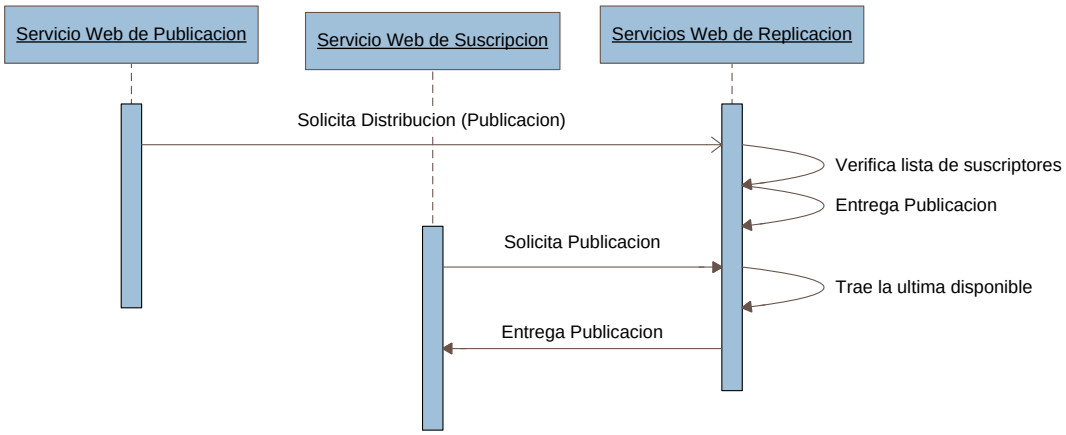

Figura 21 - Diagrama de secuencia – Distribuir Publicación

Caso de Uso: *Gestionar Suscripción*

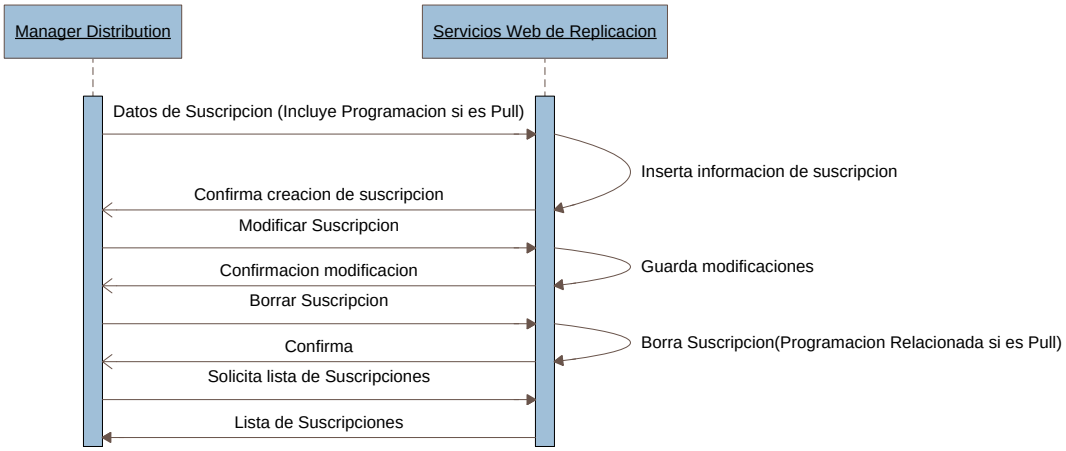

Figura 22 - Diagrama de secuencia – Gestionar Suscripción

8.3.1.4 MODELO CONCEPTUAL

Correspondiente a la fase de análisis y después de un entendimiento de las principales tareas de los servicios de distribución descritos se hace necesario un modelo conceptual que permita la visualización de los conceptos asociados a las tareas de distribución de las cuales será responsable el servicio Web.
La Figura 23 muestra el diagrama conceptual del sistema.

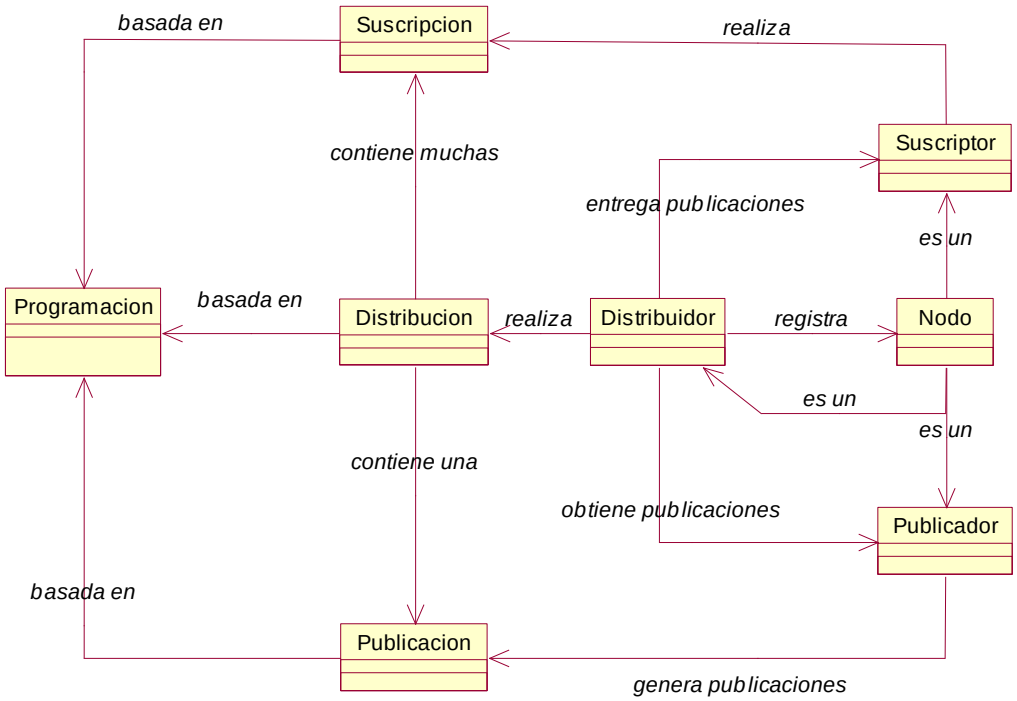

Figura 23 - Modelo conceptual de las tareas de distribución.

8.3.2 ARQUITECTURA

Para el desarrollo de la lógica de distribución fue necesario, en primer lugar, establecer una arquitectura que permita ver de qué manera se orientaría la solución y que responsabilidades estarían enfocadas en cada capa. Su arquitectura se ilustra en la Figura 24.

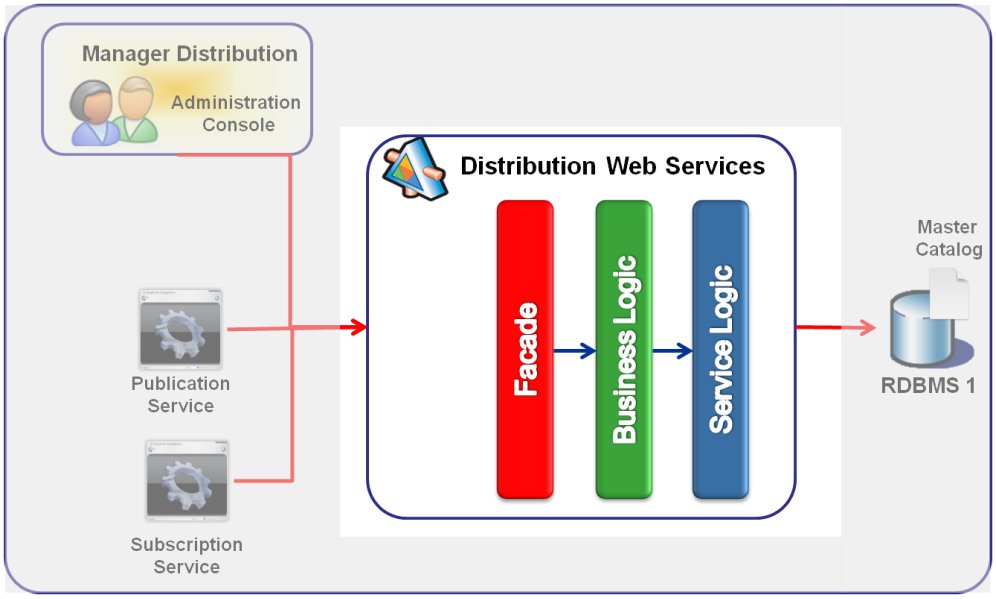

Figura 24 - Arquitectura de los servicios de replicación

Los servicios de replicación hacen uso de una sola fachada para el consumo de los servicios. Tanto la lógica de negocio como la lógica de servicios son clases implementadas y no un conjunto de servicios Web. La fachada es la única que expone los servicios necesarios para las tareas de la replicación. Como se ve la lógica de servicios se encarga de la comunicación con un bloque de aplicación encargado del acceso a datos. Este bloque de aplicación se especializa en realizar el acceso y las operaciones DML, DDL, SELECT y transacciones, sobre un motor de base de datos explicito. Este bloque de aplicación está construido en base a los patrones y prácticas fuertemente probadas y ampliamente utilizadas. La lógica de negocio contiene un conjunto de clases que soportan los servicios de replicación en sus tres tareas fundamentales de publicación, distribución y suscripción. De igual forma cada una de estas tareas tiene un servicio de programación para mantener la agenda de cada una de estas tareas, así como el soporte para las actividades solicitadas por los servicios Windows que monitorean el momento en el cual cada una de estas actividades debe ser realizada. La lógica de servicios además de comunicarse con el bloque de aplicación de acceso a datos, también es responsable de la comunicación con archivos XML como otro tipo de archivos. Gracias a la inclusión del bloque de aplicación para el acceso a datos, es posible contar con un libre acceso a cualquier motor de base de datos, de esta forma se asegura

que estos servicios de replicación pueden ser extendidos a otros motores de base de datos, ya que dentro de la lógica de negocio y el modelo relacional de datos se utiliza el estándar SQL99, omitiendo funciones propias de la sintaxis SQL de algún motor en especifico, por ejemplo Transact-SQL de SQL Server o PL-SQL de Oracle. A continuación se explican cada una de las capas que conforman los servicios web de replicación.

8.3.3 FACHADA

La fachada es la capa que expone todos los servicios referentes a la publicación, suscripción y distribución de instantáneas, así como sus respectivas agendas. Todos los servicios requieren de una autenticación para ser consumidos. Una vez realizada la autenticación frente al servicio, cada una de las funciones expuestas por éste, verifican que quien invoca dichos servicios este debidamente autenticado y autorizado frente al sistema. A continuación se muestra la Tabla 18 y la Tabla 19 que describe los aspectos más importantes de la Fachada:

Nombre	Descripción
myDatabase:	Objeto que contiene la información y métodos sobre la base de datos a la cual el servicio se está conectando. Ej. Nombre de la instancia, versión del motor de base de datos, bases de datos contenidas en esa instancia, etc.
myPublications:	Objeto que contiene toda la información referente a la publicación. Este objeto se requiere para invocar los métodos implementados en esta clase ubicada en la capa de negocios.
myArticles:	Objeto que contiene toda la información referente a los artículos de una respectiva publicación. Este objeto se requiere para invocar los métodos implementados en esta clase ubicada en la capa de negocios.
myProgramations:	Objeto que contiene la información referente a la programación de agenda de los servicios de publicación,

	distribución y suscripción. Este objeto se requiere para invocar los métodos implementados en esta clase ubicada en la capa de negocios.
mySuscriptions:	Objeto que contiene la información acerca de los suscriptores a una publicación en particular. Este objeto se requiere para invocar los métodos implementados en esta clase ubicada en la capa de negocios.
myDistributions:	Objeto que contiene la información referente a las distribuciones que se deben realizar a los suscriptores. Este objeto se requiere para invocar los métodos implementados en esta clase ubicada en la capa de negocios.
myServers:	Objeto que contiene la información acerca de los equipos que serán suscriptores de las publicaciones. Son necesarios para usar esta información en la lógica de distribución. Este objeto se requiere para invocar los métodos implementados en esta clase ubicada en la capa de negocios.

Tabla 18 - Objetos que permiten la exposición de métodos en la Fachada

Nombre	Descripción
[Database]Login	Este método se encarga de autenticar al usuario dentro de la base de datos. Por ser una aplicación que tiene acceso a todas las bases de datos, es pertinente que exista un usuario con los privilegios suficientes para realizar dicha tarea. Este método permite usar el resto de métodos que requieren de autenticación.
[Database]wsLogin	Este método se encarga de autenticar a quien lo solicite (Servicio Windows, Servicio Web) con un nombre de usuario y contraseña.
[Database]obtenerEn	Estos métodos se encargan de traer información

gines, Servers, BDs, Tables y Columnas	especifica de la base de datos y de los motores de base de datos soportados.
[Publication]Crear, Modificar, Borrar, listar Publicación	Son métodos que se encargan de la gestión de una publicación en particular. Estos métodos son los encargados de gestionar todos los datos para la publicación.
[Publication]Generar Publicacion	La generación de la publicación es la tarea primordial del publicador, este método se encarga de generar la instantánea de acuerdo a como se haya definido la publicación y sus respectivos artículos.
[Publication]crear, modificar, borrar, listar Plan de Ejecucion	Son métodos que sirven para la gestión de la ejecución de una publicación determinada en base a su programación inicial.
[Publication]escribe Historial	Este método se encarga de escribir la información del historial acerca de la actividad de una publicación.
[Publication] obtienePubsDataFromServer	Este método obtiene las publicaciones de un publicador en especial, especificado por su serverId.
[Publication] entregaPublicacionToPublicador	Este método se encarga de entregar una publicación generada dentro de un arreglo de bytes.
[Article]crear, borrar, modificar, listar Articulos	Métodos que se encargan de la gestión de artículos que hacen parte de una publicación en específico.
[Programation]crear, borrar, modificar Programaciones(Fija, Semanal, Mensual)	Métodos que se encargan de la gestión de los distintos tipos de programaciones de cualquiera de las tareas de replicación.
[Programation]FindNextExecution	Encuentra la siguiente fecha y hora de ejecución de una publicación en particular, basado en el tipo de

	programación de la misma. Esto para poder armar un plan de ejecución para cada publicación.
[Suscription]Synchronize	Este método sincroniza una publicación ya entregada dentro del equipo suscriptor.
[Suscription]addPullSuscription	Este método adiciona una suscripción por extracción (Pull) dentro de la tabla de suscripciones.
[Suscription]addPushSuscription	Este método adiciona una suscripción por inserción (Push) dentro de la tabla de suscripciones.
[Suscription] nuevoPlanEjecucionPullSus	Este método calcula un nuevo plan de ejecución para un suscripción por extracción (Pull)
[Suscription] recibePublicacionToSus	Este método ubica dentro de un suscriptor una publicación entregada por un distribuidor.
[Suscription] entregaPublicacionFromSus	Este método retorna una publicación en un arreglo de bytes al servicio que lo solicite.
[Suscription] auntenticaPublicacionEntregadaSus	Este método se encarga de descomprimir el archivo de publicación entregado por el distribuidor, extraer el contenido, leer el archivo de seguridad y sacar los respectivos MD5 a los archivos que están contenidos aquí para verificar que todos los archivos vienen bien desde el origen.
[Disitrbution]adicionaPublicador	Este método se encarga de registrar a los servidores que son publicadores dentro de la tabla que controla los publicadores a los que este distribuidor debe atender.
[Distribution] adicionaSuscriptor	Este método se encarga de registrar una distribución a la que este distribuidor se encargue de entregarle la publicación.
[Distribution]recibePublicacion	Este método se encarga de guardar una publicación enviada en un arreglo de bytes.
[Distribution]entrega	Este método se encarga de entregar una publicación

Publicacion	localizada dentro del equipo distribuidor y retorna el archivo solicitado en un arreglo de bytes.
[Server]adiciona, modifica, borra y lista Servidores	Estos métodos se encargan de gestionar la información de un servidor dentro de la red de distribución, como su Url, su nombre, el rol que desempeña (publicador, distribuidor, suscriptor), etc.
[Server]getUsrPwd	Este método se encarga de obtener el nombre de usuario y contraseña de un servidor registrado en el servidor.

Tabla 19 - Descripción de los métodos más importantes expuestos en la Fachada

8.3.4 LÓGICA DEL NEGOCIO

Esta capa es la encargada de contener todas las clases necesarias para la correcta ejecución de los métodos expuestos en la Fachada. Dentro del desarrollo de cada una de estas clases se encuentran métodos que si bien no están expuestos en la Fachada son de vital importancia para la correcta ejecución de los métodos primarios. Esta capa y cada una de las clases modeladas aquí, consumen la lógica del servicio necesaria para poder llevar a cabo con éxito las tareas correspondientes a la lógica por la cual fueron creadas. Esta característica hace de esta capa responsable únicamente de la lógica de replicación y deja la responsabilidad de persistencia y recuperación de datos a la capa de servicios.

A continuación las siguientes tablas describen los campos y métodos más importantes de cada una de las clases implementadas en la capa de Lógica del Negocio.

Clase: Article	
myDataAccessServices	**Tipo:** [ServiceLayer.]DataAccess **Descripción:** Proveer acceso a los métodos para realizar operaciones sobre la base de datos.
Métodos	

Crear, borrar, modificar, Artículos	**Retorna:** Los tres métodos retornan un tipo booleano (bool) dependiendo del resultado de la operación (true/false). **Descripción:** Estos métodos de gestión de artículos son indispensables para la gestión de una Publicación completa. Por concepto el artículo hace parte de la Publicación y esta última puede contener uno o muchos artículos.

Tabla 20 - Descripción de la clase Artículo

Clase: DatabaseInfo	
Engine, server, user, password	**Tipo:** (string) Cadena de texto **Descripción:** Estos campos guardan la información referente a la instancia de motor de base de datos a la cual se conecta en un servidor específico.
myDataAccess	**Tipo:** [ServiceLayer.]DataAccess **Descripción:** Proveer acceso a los métodos para realizar operaciones sobre la base de datos.
myXMLServices	**Tipo:** [ServiceLayer.]XMLFileAccess **Descripción:** Proveer acceso a los métodos para realizar operaciones sobre archivos XML.
Métodos	
Login	**Retorna:** bool **Descripción:** Método que se encarga de autenticar al usuario en el servidor especificado. Este método es indispensable para la autenticación de las aplicaciones que consuman el servicio.
wsLogin	**Retorna:** bool **Descripción:** Método que se encarga de autenticar a un servicio que envié el nombre de usuario y contraseña del servidor. El motor de base de datos y el nombre de la instancia la obtiene de la configuración

	local del servidor.
Obtener Bases de Datos, Tablas y Columnas	**Retornan:** DataSet **Descripción:** Obtienen las bases de datos de la instancia determinada a la cual se está autenticando en el método Login. La obtención de tablas y columnas se hacen respecto a una base de datos específica y una tabla específica, respectivamente.

Tabla 21 - Descripción de la clase DatabaseInfo

Clase: Distributions	
myFileAccessServices	**Tipo:** [ServiceLayer.]FileAccess **Descripción:** Proveer acceso a los métodos que manejan archivos.
myDataAccessServices	**Tipo:** DataAccess **Descripción:** Proveer acceso a los métodos correspondientes a operaciones con la base de datos.
Métodos	
entregaPublicaicon	**Retorna:** byte[] **Descripción:** Método que entrega una archivo especificado por su folder y nombre de archivo.
recibePublicacion	**Retorna:** bool **Descripción:** Método que recibe un archivo en un arreglo de bytes y lo ubica en un folder especificado.
adicionaPublicador	**Retorna:** bool **Descripción:** Adiciona la información acerca de un publicador que será distribuida por este distribuidor.
adicionaSuscriptor	**Retorna:** bool **Descripción:** Adiciona un suscriptor asociado a una publicación que este distribuidor distribuye.
listarPublicadoresSusc	**Retorna:** DataSet

riptores	**Descripción:** Lista la información de una publicación y los suscriptores asociados a esta.

Tabla 22 - Descripción de la clase Distributions

Clase: Programation	
engine **database** **server**	**Tipo:** string **Descripción:** Estos campos son utilizados para guardar información del servidor en el cual se está realizando la programación.
myDataAccessServices	**Tipo:** [ServiceLayer.]DataAccess **Descripción:** Proveer acceso a los métodos para realizar operaciones sobre la base de datos.
Métodos	
Crea, modifica y borra Programaciones	**Retornan:** bool **Descripción:** Estos métodos y sus respectivas sobrecargas se encargan de realizar la gestión de los diferentes tipos de programación. Cada sobrecarga está asociada a un tipo de programación (Fija, diaria, semanal, mensual), conservando el mismo concepto de gestión de la programación pero especializándose en cada uno de los tipos de programación.
Listar Programaciones	**Retornan:** DataSet **Descripción:** Son 4 métodos que se encargan de listar todas las programaciones o bien listar las programaciones, diarias, semanales y/o mensuales.
FindNextExecution	**Retorna:** DataSet **Descripción:** Encuentra la siguiente ejecución de una programación en particular. Este método se hace necesario dado que las programaciones pueden tener una periodicidad definida por el administrador.

findNextSunday **findLastMonday**	**Retornan:** DateTime **Descripción:** Métodos que dan soporte al cálculo de las siguientes ejecuciones así encuentran el inicio y fin de una semana en la cual se debe realizar una ejecución de una actividad determinada.
verifyDaySelected **verificaHora**	**Retornan:** bool **Descripción:** Métodos utilizados para calcular la fecha exacta de una siguiente ejecución dependiendo del tipo de periodicidad de una programación determinada.
estableceEjecucion	**Retorna:** DataSet **Descripción:** Establece la periodicidad de la ejecución de la siguiente actividad de acuerdo al tipo de finalización de la programación.
convertDay	**Retorna:** string **Descripción:** Convierte el día de la semana seleccionado al idioma en el cual está la maquina en la que la aplicación se instala.

Tabla 23 - Descripción de la clase Programation

Clase: Suscription	
myDataAccessServices	**Tipo:** [ServiceLayer.]DataAccess **Descripción:** Proveer acceso a los métodos para realizar operaciones sobre la base de datos.
myFileAccessServices	**Tipo:** [ServiceLayer.]FileAccess **Descripción:** Proveer acceso a todos los métodos relacionados al manejo de archivos.
myXMLFileServices	**Tipo:** [ServiceLayer.]XMLFileAccess **Descripción:** Proveer acceso a los métodos para realizar operaciones sobre archivos XML.
myProgramation	**Tipo:** [BusinessLayer.]Programation **Descripción:** Provee acceso a los métodos asociados a

	una programación.
Métodos	
Synchronize	**Retorna:** bool **Descripción:** Sincroniza una publicación completa en la base de datos destino.
addSuscription	**Retorna:** bool **Descripcion**: Adiciona una suscripción de tipo push o pull (2 Sobrecargas).
nuevoPlanEjecucionPullSus	**Retorna:** bool **Descripcion**: Genera un nuevo plan de ejecución para una suscripción por extracción (Pull).
recibePublicacion	**Retorna:** bool **Descripcion**: Guarda una publicación enviada por un servicio de distribución.
auntenticaPublicacionEntregada	**Retorna:** bool **Descripcion**: Descomprime una publicación entregada, carga en memoria el archivo de seguridad y saca MD5 a los demás archivos para comprobar que su estado no se haya alterado

Tabla 24 - Descripción de la clase Suscription

Clase: Server	
myDataAccessServices	**Tipo:** [ServiceLayer.]DataAccess **Descripción:** Proveer acceso a los métodos para realizar operaciones sobre la base de datos.
Métodos	
addServer, modifyServer, deleteServer, listarServers	**Retorna:** bool **Descripción:** Metodos que gestionan la información de un servidor.
getUsrPwdServer	**Retorna:** string[] **Descripción:** Retorna el ususario y contraseña de un servidor registrado

dentro del servidor.

Tabla 25 - Descripción de la clase Server

Clase: Publication	
myArticles	**Tipo:** [BusinessLayer.]Article **Descripción:** Este objeto es necesario para invocar a los métodos necesarios para realizar las tareas de gestión de una publicación. El objeto articulo es necesario para representarlo en el contexto de una publicación y poder llevar a cabo sus tareas de la mejor manera.
myPrograma tion	**Tipo:** [BusinessLayer.]Programation **Descripción:** Este objeto se utiliza para invocar las tareas de programación de las publicaciones.
myDataAccessServ ices	**Tipo:** [ServiceLayer.]DataAccess **Descripción:** Proveer acceso a los métodos para realizar operaciones sobre la base de datos.
myXMLFileServices	**Tipo:** [ServiceLayer.]XMLFileAccess **Descripción:** Proveer acceso a los métodos para realizar operaciones sobre archivos XML.
Métodos	
Crear, borrar y modificar Publicación	**Retornan:** bool **Descripción:** Estos métodos son utilizados para realizar la gestión de la publicación. Cada uno de estos métodos realiza invocaciones internas a los métodos de programación y artículos para conservar el significado de dichas actividades en el contexto de una publicación.
Crear, borrar y modificar planes de ejecución	**Retornan:** bool **Descripción:** Estos métodos de gestión de planes de ejecución se encargan de invocar métodos de la clase programación para establecer las ejecuciones a realizar en un tiempo futuro, basándose en la programación

	inicial y una periodicidad establecida.
Generar Publicación	**Retorna:** bool **Descripción:** Este método se encarga de fabricar la publicación con sus respectivos artículos.
escribeHistorial	**Retorna:** bool **Descripción:** Este método se encarga de escribir el historial de la actividad de una publicación.
entregaPublicacion	**Retorna:** byte[] **Descripción:** Este método se encarga de pasar un archivo de publicación en un arreglo de bytes.

Tabla 26 - Descripción de la clase Publication

8.3.5 LÓGICA DEL SERVICIO

Esta capa es la encargada de realizar todas las tareas en cuanto a servicios de acceso a bases de datos, gestión de documentos XML y manejo de archivos en general. A continuación se describen las clases implementadas para brindar los servicios necesarios que se requieren para el manejo de archivos e información en bases de datos.

Clase: DataAccess	
myDb	**Tipo:** [Microsoft.Practices.EnterpriseLibrary.Data.Sql.] SqlDatabase **Descripción:** Este objeto es indispensable para la representación de la base de datos y el uso del DataAccess Application Block. Al igual que esta clase se encuentra una clase OraDataBase que representa a una base de datos Oracle. Estos objetos son utilizados para enviar sentencias DML, DDL y SELECT a las respectivas bases de datos que el bloque de aplicación manejará.
myXMLServices	**Tipo:** [ServiceLayer.]XMLFileAccess **Descripción:** Este objeto se requiere para manipular

	archivos XML. Tarea necesaria en el método de Login para recuperar la información referente a la configuración en donde se almacenan datos como el nombre de la instancia, nombre del motor de base de datos a usar, entre otros datos útiles a la hora de realizar la tarea de autenticarse frente al sistema.
Métodos	
SQLLogin **OraLogin**	**Retornan:** bool **Descripción:** Métodos utilizados para la autenticación de un usuario al sistema usando en estos casos el motor de base de datos SQL Server o bien Oracle.
Ejecutar Select	**Retorna:** DataSet **Descripción:** Método utilizado para ejecutar una sentencia Select y devolver un conjunto de datos.
ejecutarDML	**Retorna:** int **Descripción:** Método utilizado para ejecutar operaciones del tipo INSERT, UPDATE, DELETE. Este método posee una sobrecarga haciendo posible la ejecución de estas sentencias a través de comandos o bien a través de cadenas.
ejecutarEscalar	**Retorna:** object **Descripción:** Método utilizado para ejecutar sentencias SELECT que únicamente devuelven un solo resultado y no un conjunto de resultados.
evitaSQLInject	**Retorna:** string **Descripción:** Método utilizado para evitar SQL Injection, un ataque típico al momento de concatenar campos a sentencias SQL a través de cadenas.

Tabla 27 - Descripción de la clase DataAccess

Clase: XMLFileAccess	
Métodos	
ObtenerXML	**Retorna:** DataSet

	Descripción: Método utilizado para la obtención de un archivo XML desde una ruta especificada y cargarlo a memoria.
escribirXML	**Retorna:** bool **Descripción:** Método utilizado para escribir un archivo XML a una ruta especificada.

Tabla 28 - Descripción de la clase XMLFileAccess

Clase: FileAccess **Métodos**	
Compress	**Retorna:** void **Descripción:** Método utilizado para realizar la compresión de un folder a través de la ruta del folder y se debe especificar una ruta destino en donde se ubicara el archivo en formato comprimido.
Decompress	**Retorna:** void **Descripción:** Método utilizado para descomprimir un archivo.
ApplyMD5	**Retorna:** String **Descripción:** Método utilizado para generar el MD5 de un archivo en particular.
SaveFile	**Retorna:** bool **Descripción:** Método utilizado para salvar el contenido de un stream dentro de un archivo.
OpenFile	**Retorna:** Stream **Descripción:** Método utilizado para recuperar la información de un archivo y obtenerla en un arreglo de bytes.
uploadFile	**Retorna:** void **Descripción:** Escribe una archivo que viene en un arreglo de bytes, en una ruta especificada.
downloadFile	**Retorna:** byte[] **Descripción:** Retorna un arreglo de bytes que

	representa un archivo especificado cuando se invoca el método.

Tabla 29 - Descripción de la clase FileAccess

8.4 PROGRAMACION DE SERVICIOS WINDOWS

Los servicios Windows dentro de Manager Distribution, tienen la importante tarea de ejecutar en el momento programado por el administrador de base de datos cada una de las tareas pertinentes a la replicación por instantáneas. Cada uno de los servicios se encarga de una tarea específica de la siguiente manera:

Publicador: Al iniciar se autentica frente a los servicios Web expuestos por el servidor, después de autenticarse consulta las publicaciones que debe generar en ese día. Este servicio se activa cada 5 o 10 segundos para verificar si hay algún cambio en las programaciones realizadas a las publicaciones a realizar. Apenas el servicio detecta que debe generar una publicación, la genera con todos sus artículos. Posteriormente debe sacar un MD5 a todos los artículos y guardar esa información en un archivo dentro del mismo folder de la publicación. Posteriormente comprime el archivo y pasa el archivo comprimido al servicio web de distribución. Este a su vez busca en las tablas relacionadas en el CMR verificando a que suscriptores debe entregar esta publicación. El siguiente paso es que el servicio web de distribución pasa el archivo a los servicios web de suscripción de cada uno de los suscriptores por inserción.

Suscriptor: Este servicio debe realizar la tarea de subir los datos en los suscriptores de las publicaciones que estén listas para ser sincronizadas. Este servicio también está pendiente de una programación en particular en caso de tener un tipo de suscripción por extracción (Pull).

Estos servicios Windows tienen la siguiente arquitectura. Ver Figura 25:

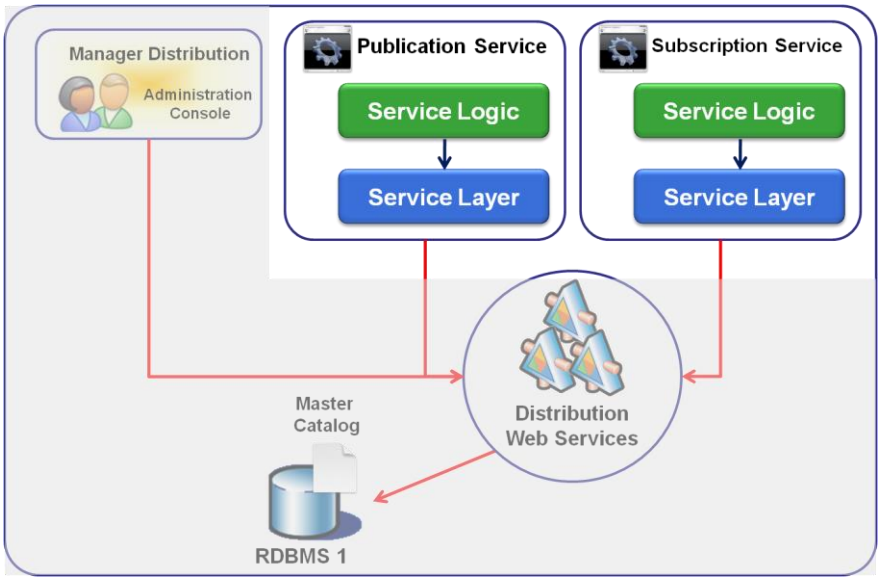

Figura 25 - Arquitectura Servicios Windows

A continuación se explican cada una de las capas que conforman los servicios Windows de replicación.

8.4.1 LOGICA DEL SERVICIO (SERVICE LOGIC)

Dentro de la lógica del servicio, se establece la creación de métodos que ayuden a ejecutar de una manera correcta el servicio que se está programando. Este servicio implementa métodos que usan un ProxyFacade implementando el patrón Singleton para la invocación de los servicios Web. Este patrón se aplica para garantizar la existencia de un objeto global valido y garantizar que solamente una instancia de la clase sea creada. A continuación la Tabla 30 describe la clase PublicationService.

Clase: PublicationService	
myProxyFacadeServices	**Tipo:** [ServiceLayer.]ProxyFacade **Descripción:** Objeto necesario para invocar los métodos desde el servicio Web. Este objeto sirve como único objeto para la invocación de métodos del servicio Web.

myTimer	**Tipo:** [System.Timers.]Timer **Descripción:** Objeto necesario para la ejecución de la actividad de monitoreo de las programaciones de las publicaciones, suscripciones o distribuciones para generarlas en el momento justo.
dsPublications	**Tipo:** DataSet **Descripción:** Objeto necesario para realizar almacenar las publicaciones a realizar en ese día determinado.
year, month, day, hour, minute	**Tipo:** int **Descripción:** Variables necesarias para la comparación de las programaciones obtenidas en dsPublications con el tiempo y fecha actual.
Métodos	
OnStart	**Retorna:** Nada **Descripción:** Se autentica frente al sistema e inicia el timer.
OnStop	**Retorna:** Nada **Descripción:** Deshabilita el timer.
reviewListPublications	**Retorna:** Nada **Descripción:** Revisa la lista de publicaciones que se cargo en memoria y si hay publicaciones, suscripciones y distribuciones, respectivamente, para generar, de ser así, las genera.
OnElapsedTime	**Retorna:** Nada **Descripción:** Invoca a los métodos para cargar las ejecuciones del día actual y revisa si tiene que hacer alguna publicación.

Tabla 30 - Descripción de la clase PublicationService

8.4.2 CAPA DE SERVICIOS (SERVICE LAYER)

Esta capa es la encargada de proveer acceso a los métodos expuestos por el servicio Web que serán utilizados por cada uno de los servicios. Dado que son tres servicios diferentes cada uno de ellos tendrá un ProxyFacade diferente, que es la clase que está disponible para invocar estos métodos en cada uno de los servicios. La Tabla 31 describe la clase ProxyFacade de en el caso del servicio de publicación, se asume que en los otros servicios se tendrán los mismos métodos obviamente aplicados a cada uno de los casos de distribución y suscripción.

Clase: ProxyFacade	
myWSDistribution	**Tipo:** [PublicationService.wsDistribution] Facade **Descripción:** Objeto que se requiere para la invocación de los métodos en el servicio Web.
Métodos	
Instance	**Retorna:** ProxyFacade **Descripción:** Método necesario para implementar el patrón Singleton.
[Database]Autenticación	**Retorna:** bool **Descripción:** Expone la autenticación que realiza el servicio Web.
[Publication]listaEjecucionesDiaHoy	**Retorna:** DataSet **Descripción:** Expone el método que lista las ejecuciones para la fecha y hora actual.
[Publication]generarPublicacion	**Retorna:** bool **Descripción:** Expone el método que genera la publicación.
[Publication]borrarPublicacion	**Retorna:** bool **Descripción:** Expone el método que borra la publicación.
[Publication]borrarEjecucion	**Retorna:** bool **Descripción:** Expone el método que borra la ejecución.

[Publication]escribeHistorial	**Retorna:** bool **Descripción:** Expone el método que escribe en el historial.
[Publication]entregaPublicacionFromPublicador	**Retorna:** bytes[] **Descripción:** Expone el método que entrega una publicación.
[Distribution]recibePublicacion	**Retorna:** bool **Descripción:** Expone el método que recibe una publicación
[Distribution]entregaPublicaicon	**Retorna:** bytes[] **Descripción:** Expone el método que entrega una publicación.
[Distribution]listarPublicadoresSuscriptores	**Retorna:** DataSet **Descripción:** Expone el método que lista los suscriptores de una publicación especifica a realizar por el distribuidor.
[Subscription]recibePublicacionToSuscription	**Retorna:** bool **Descripción:** Expone el método que recibe una publicación de un distribuidor
[Subscription]autenticaPublicacionEntregadaSus	**Retorna:** bool **Descripción:** Expone el método que autentica si una publicación es valida o no.
SynchronizeSubscription	**Retorna:** bool **Descripción:** Expone el método que sincroniza una publicación

Tabla 31 - Descripción de la clase ProxyFacade del servicio de Publicación

8.5 APLICACIÓN WINDOWS MANAGER DISTRIBUTION

Esta aplicación es la más importante para el administrador de base de datos y para un correcto aprovechamiento de los servicios Web programados para las tareas de replicación. Manager Distribution es una consola de administración

de las tareas de publicación, distribución y suscripción de un servidor específico que a través de una interfaz agradable de administración facilita la labor del mismo para llevar a cabo dichas tareas. A continuación se muestra a alto nivel como está conformada la aplicación (Ver Figura 26).

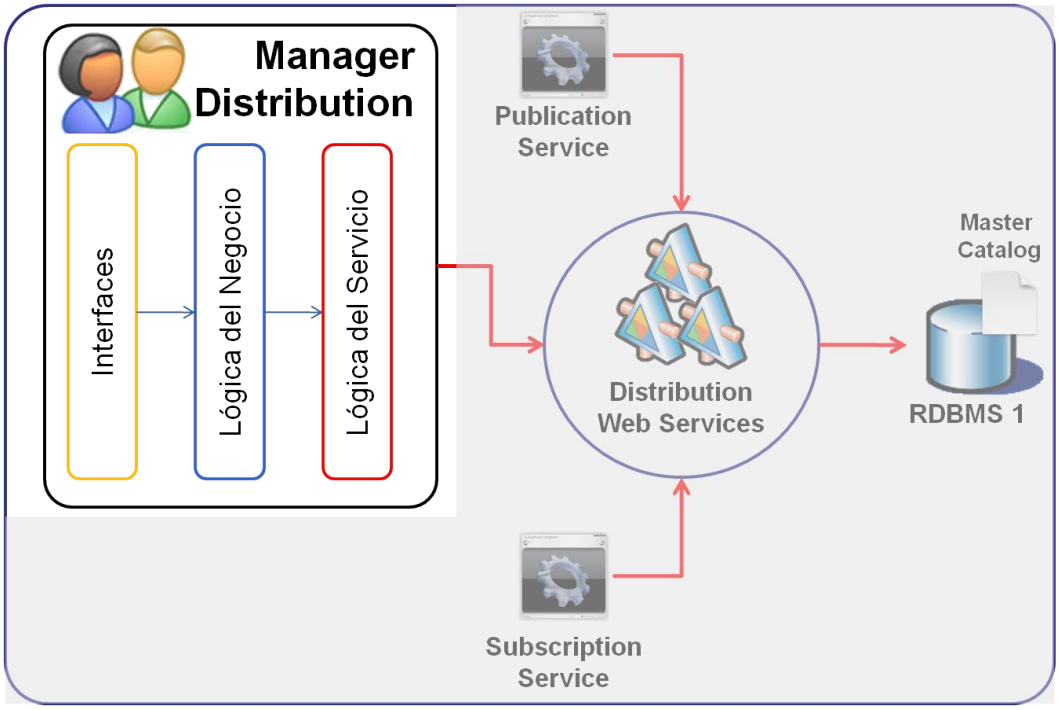

Figura 26 - Arquitectura Manager Distribution Windows Application

A continuación se describen cada una de las capas para la construcción de la consola de administración de Manager Distribution.

8.5.1 INTERFACES – CAPA DE PRESENTACION

Esta capa se encarga de manejar la lógica de presentación al usuario final. A través de interfaces agradables se permite el uso de esta herramienta para la programación de servicios de replicación de manera independiente al motor de base de datos. Esta aplicación está basada en una interfaz MDI (Multiple Document Interface) donde la ventana principal cuenta con un menú que contiene todas las actividades que puede realizar un administrador. A continuación se lista que tipo de acciones puede realizar un administrador sobre la aplicación.

- Nueva Publicación
- Ver Publicaciones
- Nueva Suscripción
- Definir Rol
- Suscribir Servidor

A continuación se describe en la Tabla 32 el propósito de cada una de estas funciones.

Nombre Función	Descripción
Nueva Publicación	Responsable de crear una nueva publicación con sus respectivos artículos, permitiendo al usuario seleccionar de manera sencilla la base de datos, la tabla y el tipo de filtro que desea aplicarle a un artículo en particular de la publicación.
Ver Publicaciones	Muestra al administrador las publicaciones disponibles que ya han sido programadas por cada base de datos. Además permite modificar su definición y/o programación de las mismas y de igual forma también borrarlas.
Nueva Suscripción	Responsable de crear una nueva suscripción a las publicaciones existentes. Una nueva suscripción implica, seleccionar la publicación a la cual se desea suscribir, y programar en qué momento se desea sincronizar dicha información.
Definir Rol	Define el rol de un servidor que ya este registrado dentro de la lista de servidores registrados para ese servidor.
Suscribir Servidor	Suscribe un nuevo servidor dentro de la red de distribución, definiendo los datos mas importantes para ese equipo, tales como url, nombre de usuario y contraseña de acceso, y el rol que desempeña en la red de dsitribucion.

Tabla 32 - Descripción de las principales tareas de Manager Distribution

La capa de interfaz no involucra ningún tipo de lógica de negocio, solamente se encarga de la presentación de los datos, la única lógica involucrada es en la validación de datos que se debe realizar para enviar información correcta hacia el servidor.

Las interfaces que acompañan el diseño de esta aplicación se describen a continuación, describiendo el curso normal de los eventos en cada uno de los casos. A continuación se presentan las actividades más importantes del sistema y su respectiva interfaz de usuario.

8.5.1.1 ENTRADA AL SISTEMA – LOGIN

Acción de los actores	Acción del Sistema
1. Este caso de uso comienza cuando un usuario inicia la aplicación Manager Distribution (Ver Figura 27)	
2. El sistema despliega esta ventana donde el usuario selecciona el motor de base de datos, y digita el nombre de la instancia, nombre de usuario y contraseña y se conecta.	3. El sistema se conecta a los servicios Web y envía la información. 4. El servicio Web se conecta con la instancia de base de datos del motor seleccionado. 5. El sistema despliega el menú principal de la aplicación si la conexión es exitosa.

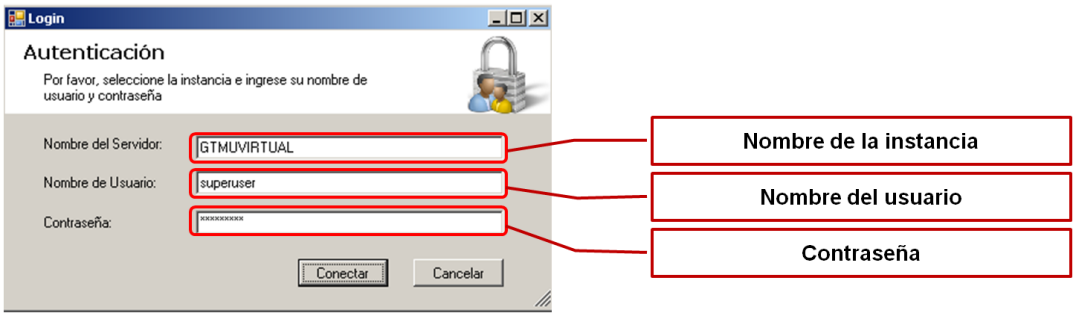

Figura 27 - Caso de Uso Real - Login

8.5.1.2 MENU PRINCIPAL (MDI)

El menú principal de la aplicación está dispuesto como un documento de interfaz múltiple (MDI Multiple Document Interface) y este agrupa una serie de opciones disponibles para el administrador que se ilustran en la Figura 28.

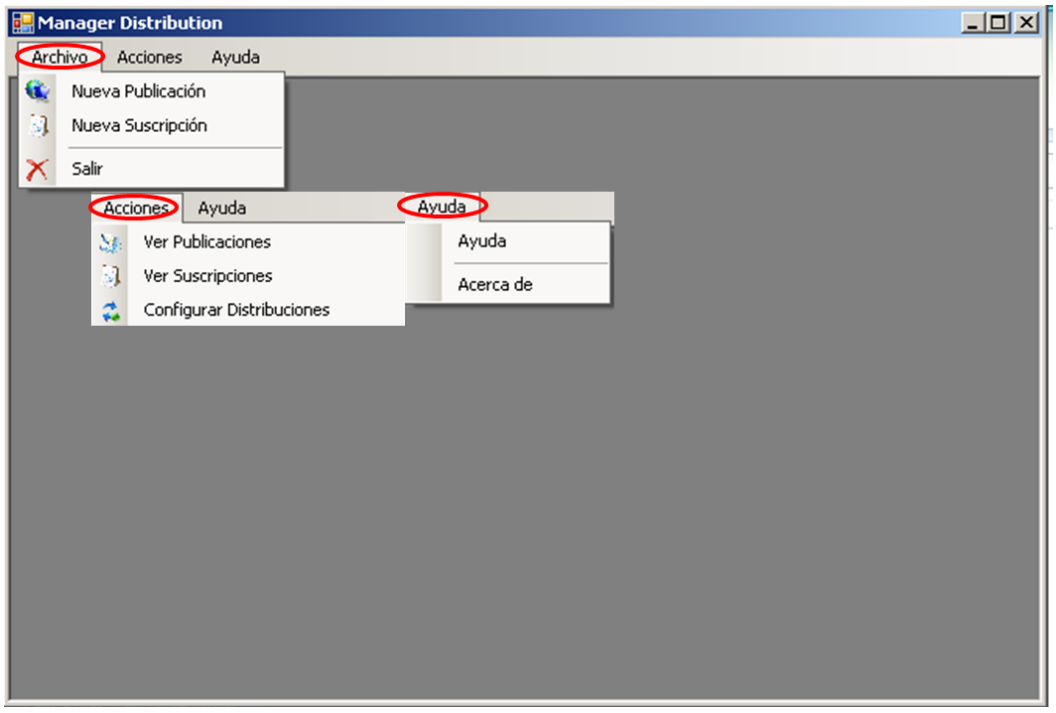

Figura 28 - Menú Principal – Manager Distribution

8.5.1.3 NUEVA PUBLICACION

Acción de los actores	Acción del Sistema
1. Este caso de uso inicia cuando el usuario selecciona la opción Nueva Publicación del menú principal.	
	2. El sistema muestra una ventana (Ver Figura 29) que lo guiará paso a paso en el proceso de creación de una publicación.
3. El usuario selecciona la base de datos y la tabla que quiere replicar, da click en siguiente.	4. Pasa a la siguiente pestaña solicitando datos de la publicación y su programación.
5. El usuario digita los datos de la publicación y da click en el botón de Programación (Ver Figura 30).	6. Se muestra la ventana de programación.
7. El usuario selecciona la programación que desea y prosigue con el paso siguiente.	
8. El usuario debe digitar los datos solicitados por la creación del artículo y seleccionar una base de datos, una tabla de la misma y seleccionar columnas que desea para el artículo, de igual forma puede seleccionar una condición para el artículo que desea (Ver Figura 31).	
9. El usuario debe seleccionar un distribuidor para la publicación. Si el distribuidor es remoto, se despliega la lista de servidores disponibles para esta publicación.	

Despues de seleccionar uno de los servidores disponibles, se debe establecer una contraseña que permite controlar las suscripciones a esta publicación. (Ver Figura 32) 10. El usuario puede agregar más de un artículo a la publicación.	11. El sistema presenta un resumen de la publicación y dispone la confirmación por parte del usuario para la creación de la publicación.(Ver Figura 33)

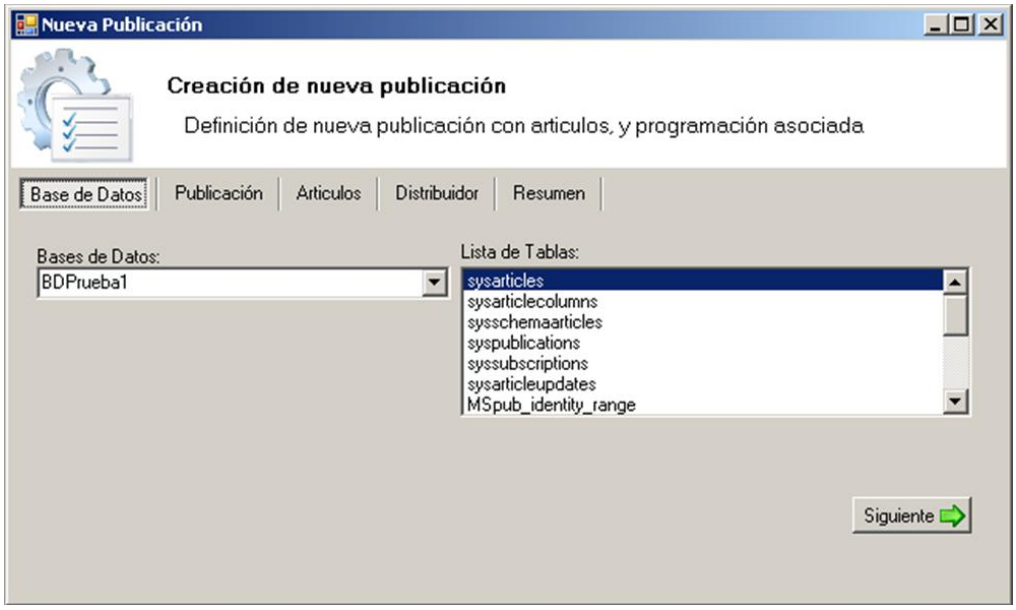

Figura 29 - Interfaz nueva publicación

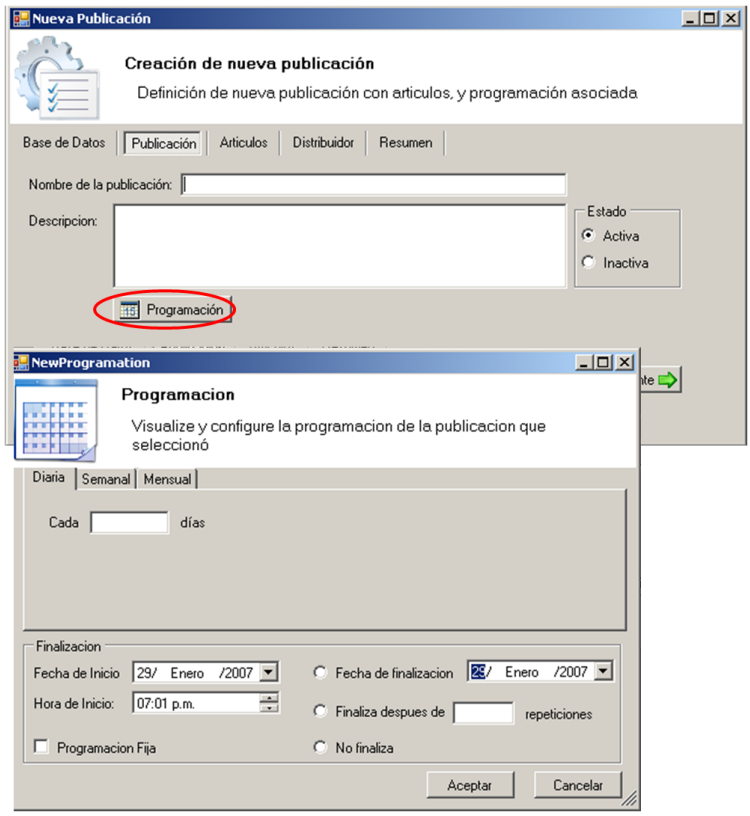

Figura 30 - Interfaz datos de publicación y su programación

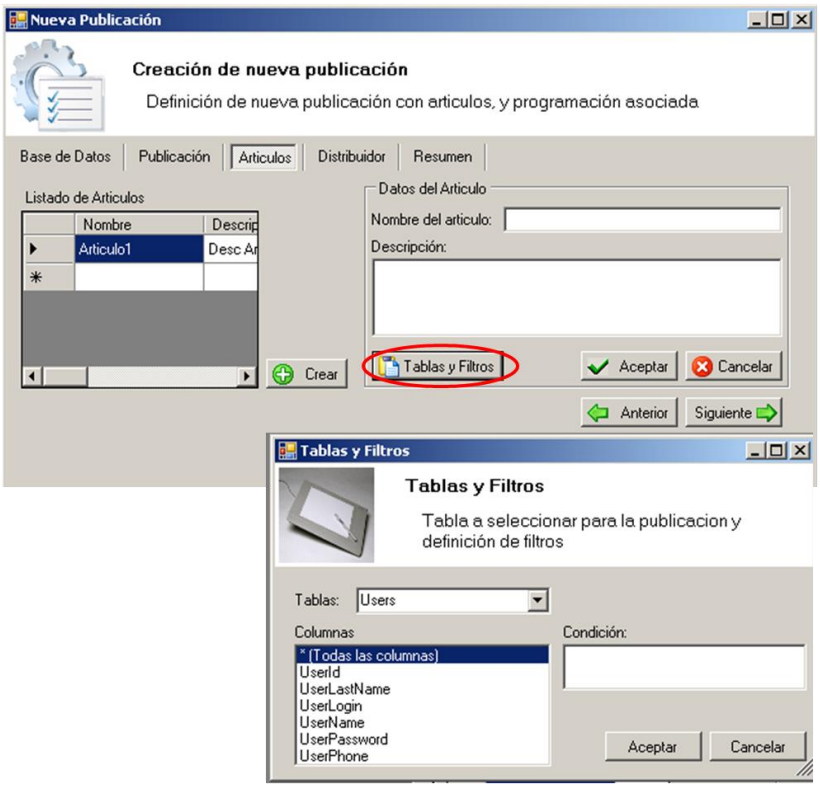

Figura 31 - Interfaz datos de artículos de publicaciones

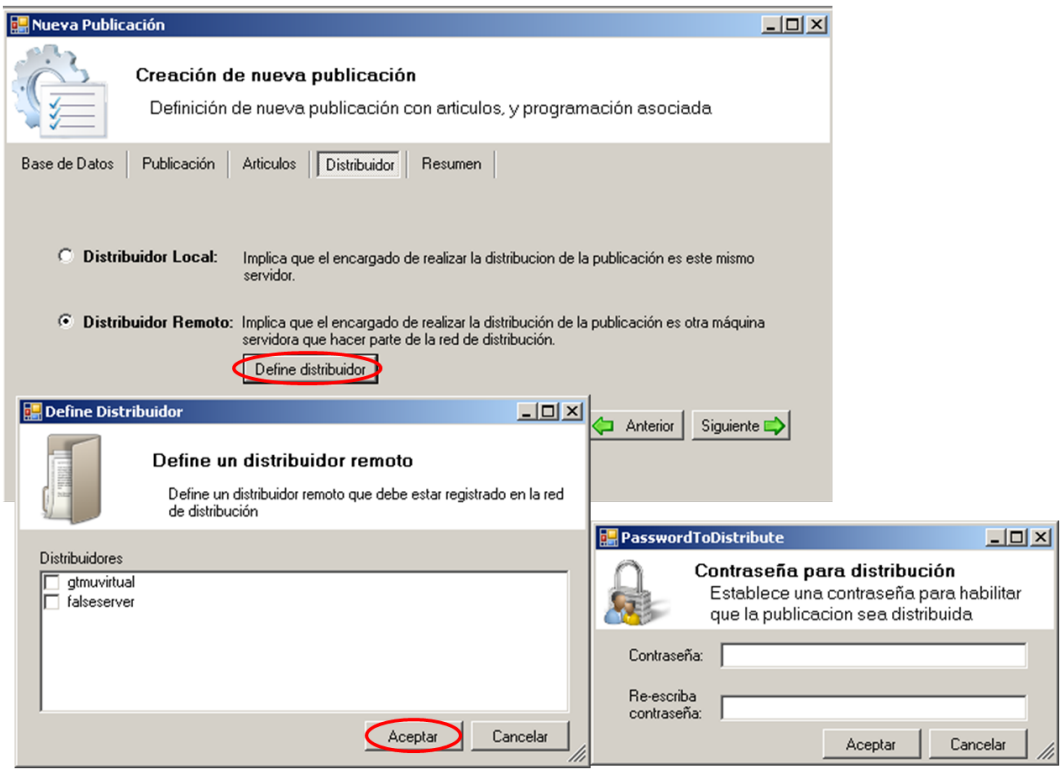

Figura 32 - Interfaz definición del distribuidor.

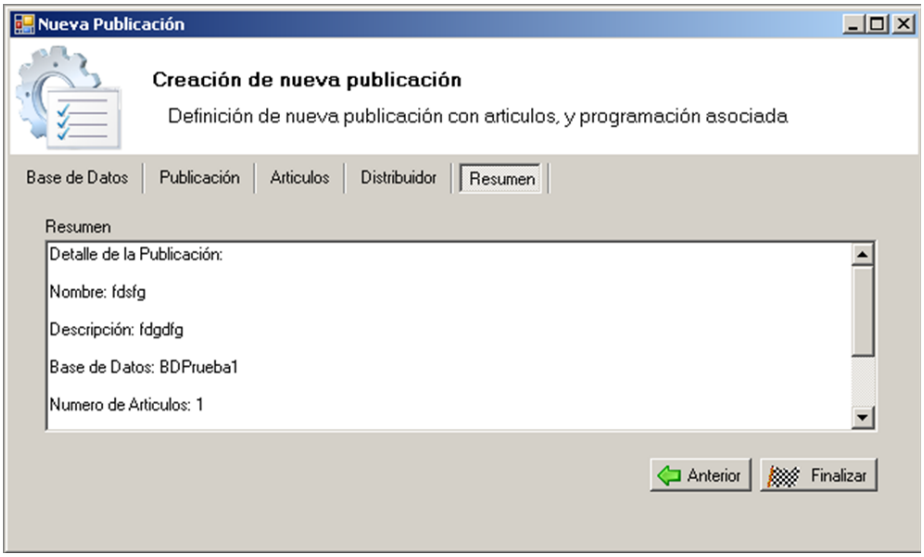

Figura 33 - Interfaz Resumen datos de la Publicación

8.5.1.4 VER PUBLICACIONES

Acción de los actores	Acción del Sistema
1. Este caso de uso inicia cuando el usuario selecciona Ver Publicaciones del menú principal.	
	2. El sistema le muestra la ventana donde se muestran las bases de datos de la instancia a la cual está conectado y las publicaciones disponibles por cada una de ellas.(Ver Figura 34)
3. El usuario puede Borrar la publicación o ver en detalle la misma.	
4. El usuario selecciona Ver Detalle.	
	5. El sistema muestra una ventana con los datos de la publicación. (Ver Figura 35)
6. En esta ventana se pueden visualizar los artículos y la programación del mismo. (Ver Figura 36)	
7. El usuario selecciona Borrar	
	8. La publicación seleccionada es borrada.

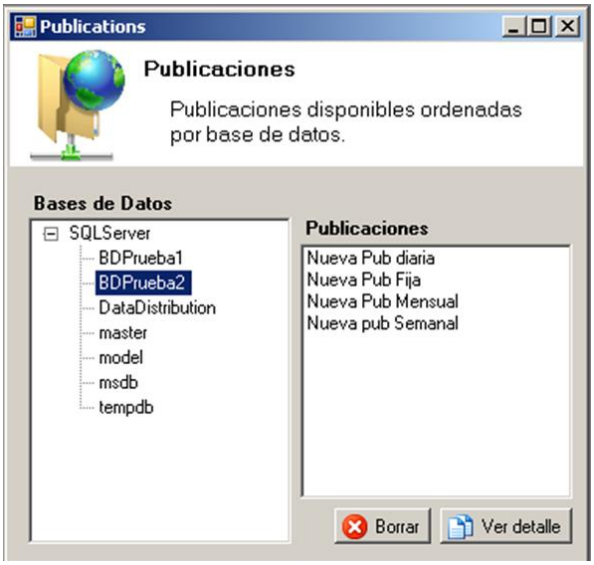

Figura 34 - Interfaz Ver Publicaciones

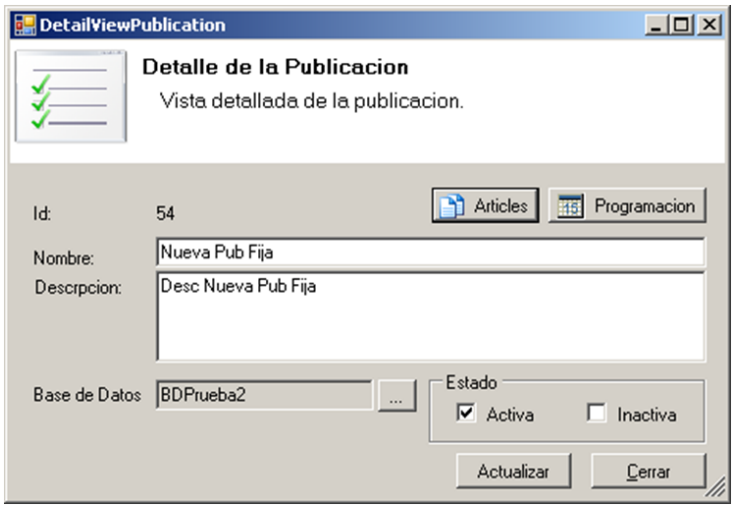

Figura 35 - Interfaz Detalle de la Publicación

Figura 36 - Interfaces Modificación de Artículos y Programación

8.5.1.5 NUEVA SUSCRIPCION

Acción de los actores	Acción del Sistema
1. Este caso de uso inicia cuando el usuario selecciona Nueva Suscripción del menú Archivo.	
	2. El sistema le muestra la ventana donde se muestran los servidores registrados como distribuidores (Ver Figura 37).
3. El usuario selecciona un distribuidor y da click en siguiente.	
	4. El sistema consulta en el servidor seleccionado las publicaciones disponibles (Ver Figura 38).
5. El usuario debe seleccionar ahora el tipo de suscripción a realizar. Si	

selecciona suscripción por extracción debe definir una programación (Ver Figura 39). 6. Ahora el usuario debe seleccionar la base de datos de suscripción en la cual se sincronizaran los datos y el tipo de sincronización (Ver Figura 40) 7. El usuario debe ingresar la contraseña que le permite suscribirse a esa publicación (Ver Figura 41). 8. Finalmente se muestra un resumen de la suscripción (Ver Figura 42). Para crear la suscripción el usuario debe hacer click en Finalizar.	

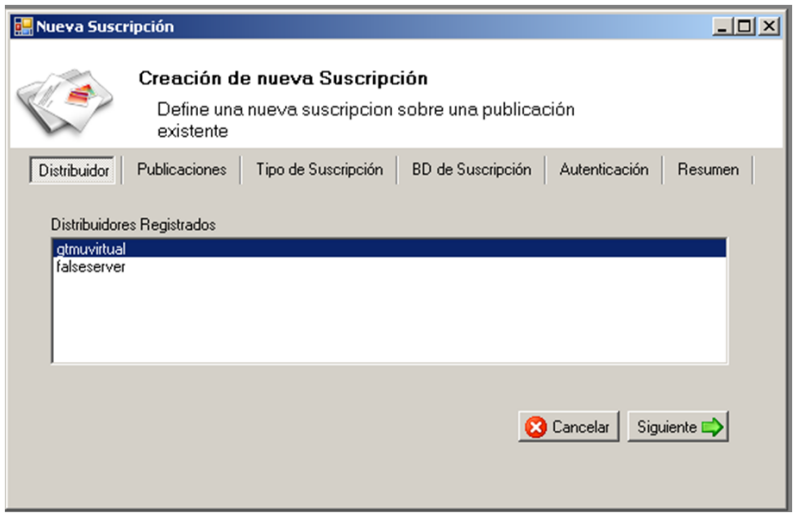

Figura 37 - Interfaz, selección de distribuidor

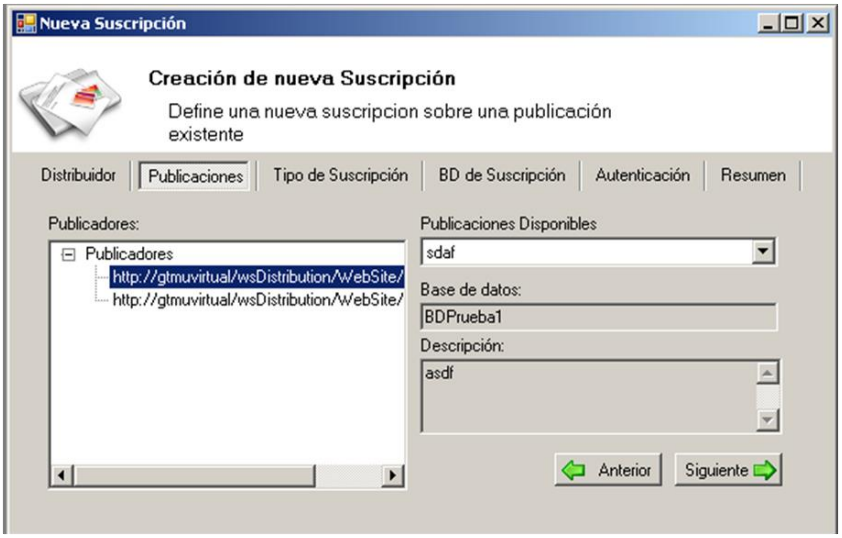

Figura 38 - Interfaz Selección de la publicación a suscribir

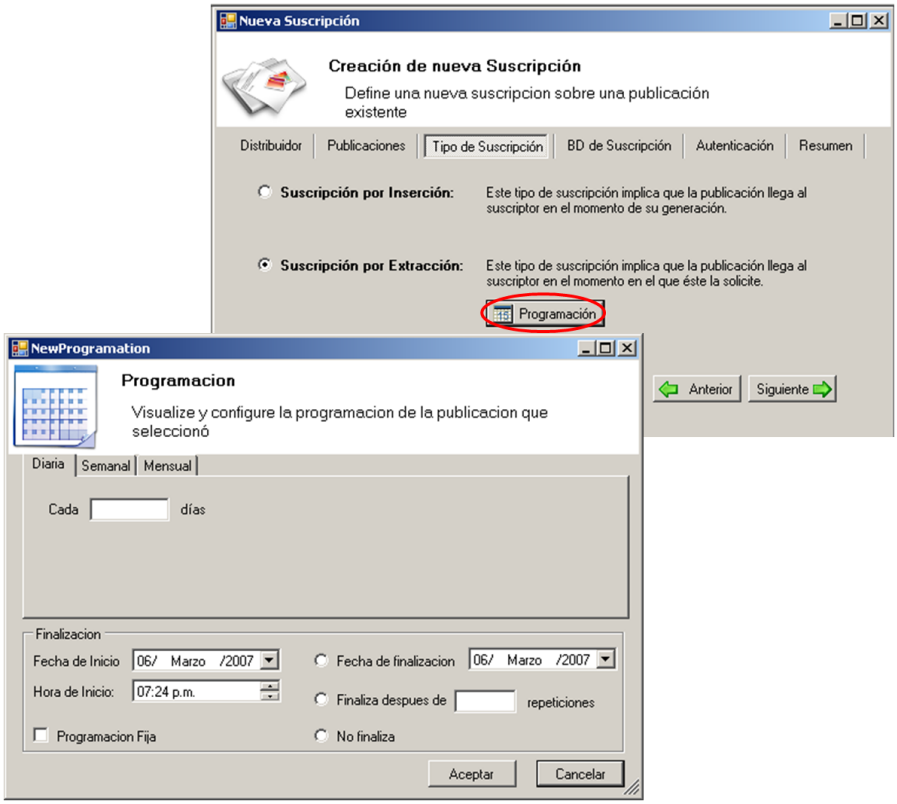

Figura 39 - Interfaz Selección del tipo de suscripción – Muestra programación en caso de seleccionar Suscripción por extracción.

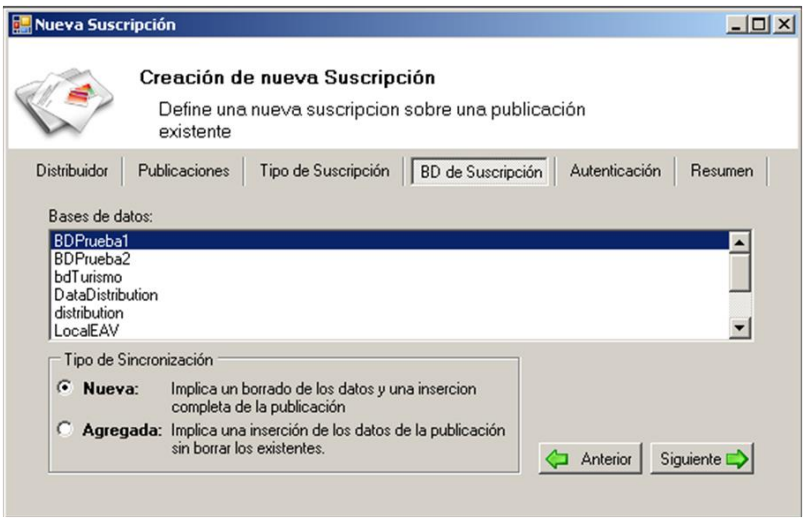

Figura 40 - Interfaz Selección de la base de datos de suscripción y tipo de sincronización.

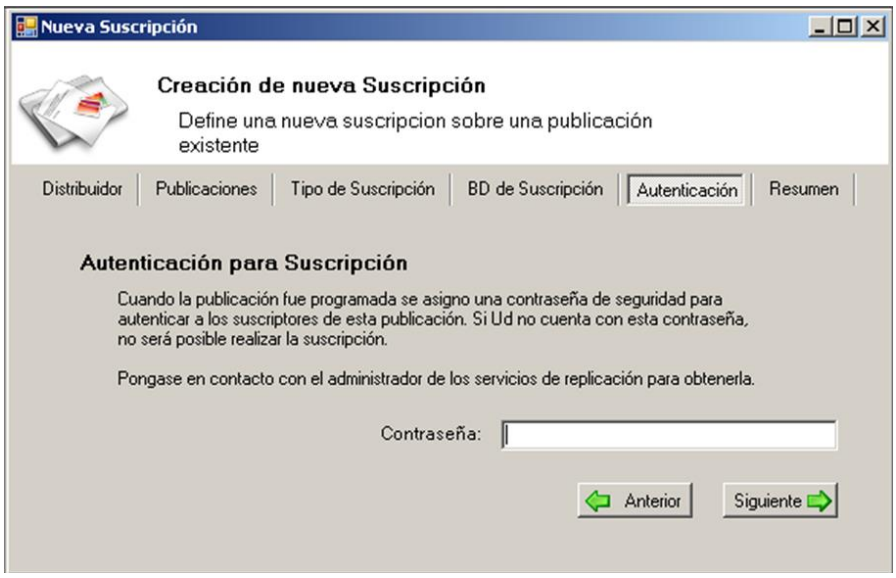

Figura 41 - Interfaz Autenticación para habilitar la suscripción.

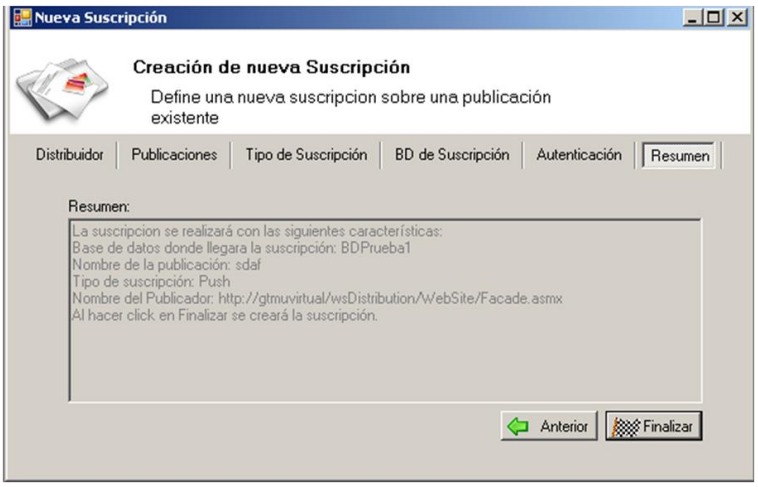

Figura 42 - Interfaz Resumen de la nueva suscripción.

8.5.2 LOGICA DEL NEGOCIO

Esta capa es la encargada de preparar la información para el consumo de su funcionalidad a través de la capa de servicios. Dicha funcionalidad se encuentra en los servicios Web que esta aplicación consume. Por esta razón todo lo que hacen las clases de esta capa es invocar a la capa de servicios, quien tiene la responsabilidad de dar acceso a la funcionalidad expuesta por los servicios Web. Las clases que están definidas dentro de esta lógica del negocio son las mismas definidas para la Lógica del Negocio de los servicios Web, solamente que estos permiten consumir los servicios expuestos a través de un llamado entre capas. Existen claro está, algunas tareas que se requieren antes de invocar los métodos que ya están implementados en el servicio Web.

8.5.3 LÓGICA DEL SERVICIO

Esta capa es la encargada de proveer servicios que se solicitan a través de la capa de Negocios. Dado que toda la lógica del negocio hace invocaciones a métodos del Web Service. De igual forma provee servicios para el tratamiento de archivos XML dentro de la aplicación Windows. Esta capa de servicios dado que accede a la misma lógica de servicios que tienen los servicios Windows es desarrollada implementando el patrón Singleton, esto para garantizar que

solamente se utilizará un solo objeto único global, para la invocación de métodos al servicio Web. Con este patrón se puede mantener la autenticación con el servicio válida porque de esta forma se conserva la misma sesión. Al igual que la capa de negocios de esta aplicación lo que se ve aqui es la invocación a métodos ya desarrollados dentro del servicio Web de distribución.

CAPITULO IV: EASY TOUR GUIDE 2007

Este capítulo está orientado a describir el desarrollo de la herramienta SIG para turismo denominada Easy Tour Guide 2007. Aquí se describe su proceso de desarrollo, el concepto de solución planteado y su impacto en un posible mercado en el ambiente Colombiano.

9 CONCEPTO DE SOLUCION EASY TOUR GUIDE

Easy Tour Guide 2007 tiene como objetivo facilitar el acceso a información turística de una localidad de manera sencilla, práctica y sobre todo utilizando las últimas tecnologías disponibles. La idea es hacer inclusión de las TIC's a la industria del turismo en Colombia y más explícitamente al departamento del Cauca y la ciudad de Popayán. Esta propuesta fue planteada a CINTEL (Centro de Investigaciones de las Telecomunicaciones) y a Movistar (Telefonica) en una iniciativa denominada PROTEUS[2] en el cual una de los objetivos del programa era la inclusión de tecnologías en la industria y fomento del turismo. Easy Tour Guide 2007 fue tomada dentro de las 12 propuestas más llamativas brindando apoyo y soporte para la realización del proyecto. En la primera fase del proyecto por petición de Movistar y CINTEL fue desarrollado para dispositivos de baja gama. En donde se planteo el desarrollo de un sistema para el turismo de la ciudad de Popayán sin incluir la parte del SIG.

El componente de sistema de información geográfico (SIG) dentro de este proyecto de investigación, sin ser menos importantes que el resto del trabajo fue omitido por las siguientes razones:

1. Una alternativa estudiada fue el uso de los servicios web de Mappoint, pero estos, limitan al usuario al uso de mapas pre establecidos por Microsoft y

[2] PROTEUS. Programa de fomento al desarrollo de la industria de las telecomunicaciones móviles e inalámbricas en Colombia. Disponible en Web: http://www.cintel.org.co/noticintel/noticia.php3?nt=3876

entre los disponibles NO se encuentra en la actualidad ninguno en detalle de ciudades de Colombia.

2. Otra alternativa estudiada fue la de Autodesk MapGuide, que posee un servidor y visor de mapas muy completo, útil y gratuito. Desafortunadamente, este visor no es funcional en Pocket PC y no se logró enmascarar su funcionalidad para servirla a través de un servicio web.
3. Actualmente, los visores para Pocket PC de mapas están en desarrollo y los actuales no cuentan con la posibilidad de interactuar o aprovechar la funcionalidad de estos sistemas georeferenciados.
4. La idea principal de este trabajo no son los sistemas de información geográfica y por este motivo no se abordo la idea de implementar un visor para Pocket, además porque esto rebasaba totalmente los alcances del proyecto. Si bien el tema está planteado dentro del anteproyecto y la monografía, la relevancia de tener servicios de SIG no le agregan al modelo de replicación absolutamente nada. Luego no se vio la necesidad de extender el trabajo con este tema.

A continuación se muestra la primera versión de la aplicación así como su versión para dispositivos de gama alta los cuales soportan contenido gráfico mucho más avanzado.

9.1 EASY TOUR GUIDE 2006

Easy Tour Guide 2006 hace parte de la primera fase del proyecto y se desarrollo siguiendo los siguientes requerimientos: Acceso para teléfonos de gamma baja y servicios de consulta acerca de información turística. En base a estas peticiones se realizó el siguiente diagrama de casos de uso. (Ver Figura 43)

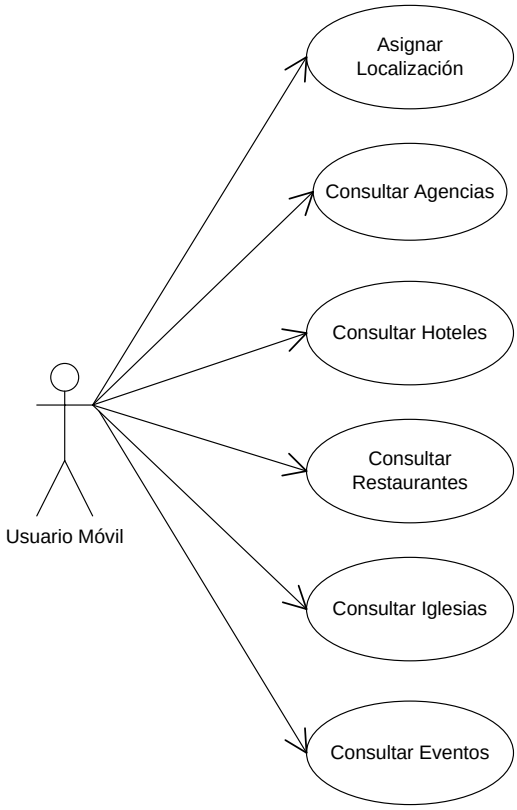

Figura 43 - Diagrama de Casos de Uso Easy Tour Guide 2006

Los casos de uso presentados serán llevados a cabo por el sistema. La descripción explicita del detalle de la construcción de este sistema no es el principal fin de este documento, por esto esta parte no se especifica a detalle. A continuación se muestra un diagrama conceptual de la aplicación y su segundo ciclo de desarrollo para la versión para Pocket PC.

9.1.1 DIAGRAMA CONCEPTUAL

A continuación se muestran los conceptos involucrados en el modelamiento del sistema. Lo que muestra la Figura 44 es la relación entre los conceptos que serán representados por el sistema.

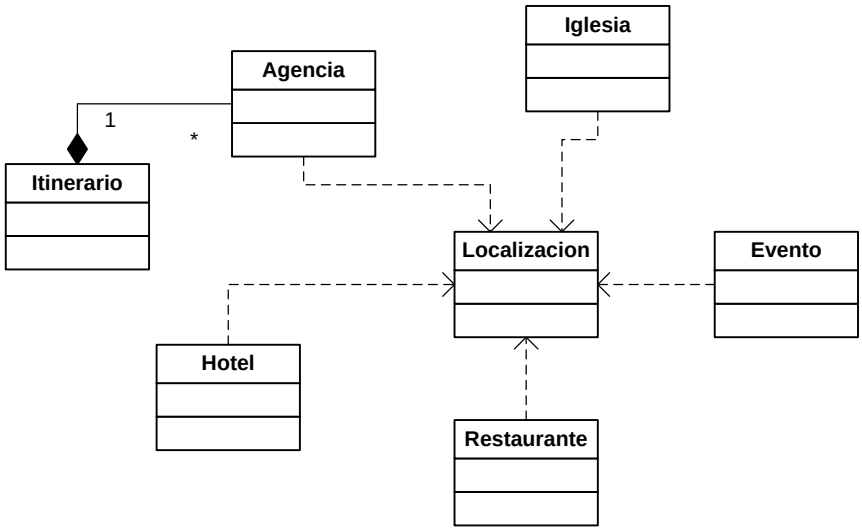

Figura 44 - Modelo conceptual – Easy Tour Guide 2006

Como se describe en los casos de uso así como se puede visualizar en el modelo conceptual de la aplicación, esta misma está diseñada para prestar servicios de información basados en la localización del usuario. A continuación se muestra la arquitectura de la aplicación usada para esta versión de Easy Tour Guide.

9.1.2 ARQUITECTURA

La siguiente arquitectura no pretende desglosar a fondo cada una de las capas, sino mostrar a un alto nivel como está estructurada la aplicación (Ver Figura 45).

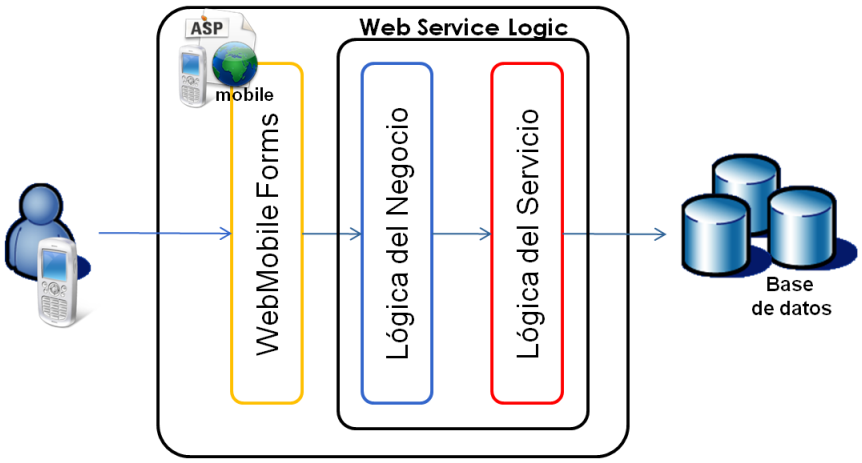

Figura 45 - Arquitectura Easy Tour Guide 2006

Por tratarse de una aplicación Web Mobile, esta hace uso de la lógica de los servicios Web de manera directa sin necesidad de referencias. En caso de que esta lógica se desee utilizar para el desarrollo de otra aplicación puede hacer referencia a los servicios Web en donde reside la lógica de esta aplicación. Esto presenta la posibilidad de poder cambiar de interfaz y poder programar otro tipo de aplicaciones que consume esta misma lógica de negocio para ser presentada por otro dispositivo.

Como resultado final se tiene una aplicación para teléfonos celulares de baja gamma asequible a través de internet en la URL: http://spar.unicauca.edu.co/mSIG/Application y que se puede ver como lo muestra la Figura 46

Figura 46 - Interfaces de Easy Tour Guide 2006

Después de algunas pruebas de usuario final realizadas por el señor Cristian Mejía de Movistar y Edwin Monroy, Gerente de Proyectos de CINTEL, se dieron las siguientes recomendaciones.

- Uso de iconografía usando formatos de imagen de baja resolución.

- Uso de textos que remplacen las imágenes que no puedan ser desplegadas.
- Servicios orientados a la necesidad del cliente sin tener que fijar localización.
- Habilitar la opción de llamadas en los resultados obtenidos por consultas de Agencias, Hoteles, Restaurantes y/o Eventos.

9.2 EASY TOUR GUIDE 2007

Esta nueva versión está diseñada para mejorar las características y servicios ofrecidos en Easy Tour Guide 2006, incluyendo mapas geo referenciados de información turística, imágenes y nuevos servicios para brindarle al usuario. A continuación el diagrama de casos de uso (ver Figura 47) muestra los nuevos servicios.

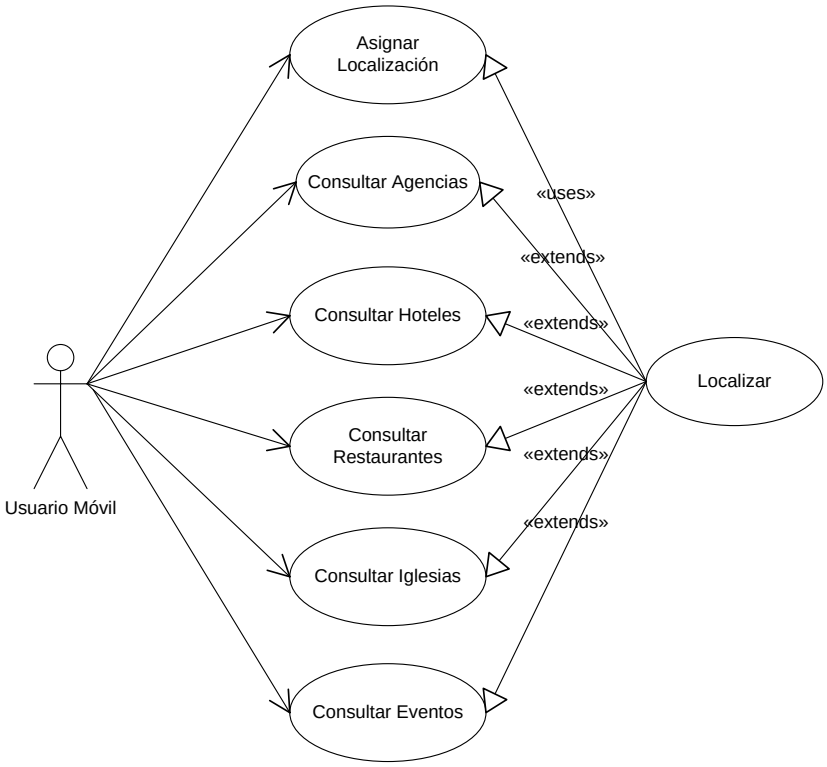

Figura 47 - Diagrama de casos de uso – Easy Tour Guide 2007

Si bien no se muestran grandes cambios, los servicios y el dispositivo a utilizar si da la posibilidad de mejorar los servicios actuales. Por utilizar una Pocket PC

para el despliegue de la aplicación se permite tener una interfaz mejorada que facilite el uso del aplicativo.

A continuación la Figura 48 muestra la arquitectura que presenta el uso de la misma arquitectura que Easy Tour Guide 2006, pero con una capa de interfaz diferente y utilizando referencias a Web Services que son los que siguen teniendo la lógica de la aplicación. Es decir, la estructura de los servicios Web se mantiene pero se presenta una nueva capa de Fachada para exponer los servicios que serán consumidos por la aplicación Pocket PC.

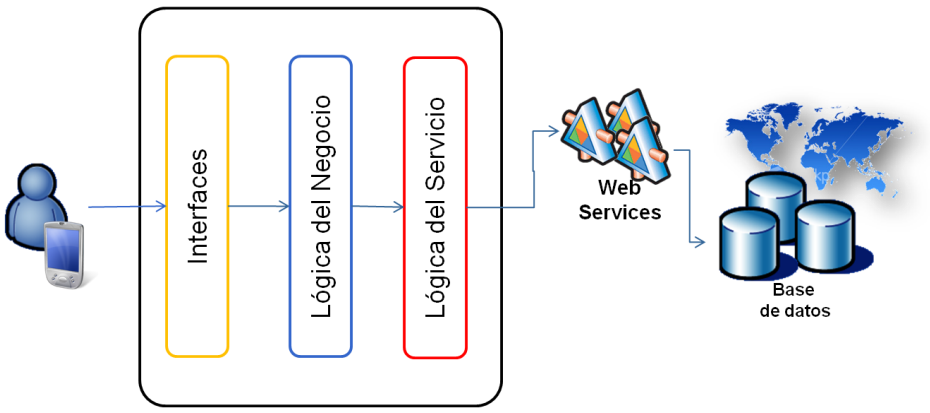

Figura 48 - Arquitectura Easy Tour Guide 2007

Si bien esta aplicación usa mapas para la presentación de la información no presta servicios que normalmente se encuentran en un sistema de información geográfico, como presentar las rutas más cortas entre puntos, o localización a través de sistema de posicionamiento global (GPS). Si bien el enfoque de este trabajo pretende solucionar una necesidad actual que es la de brindar un servicio más amigable de consulta de sitios turísticos de una localidad específica, en este caso Popayán, la idea de esta solución es mostrar como el uso de este aplicativo con información replicada en diferentes motores de bases de datos se hace posible sin tener que asignar esta responsabilidad a un motor de base de datos especifico. En el siguiente capítulo se explica de mejor manera como esta aplicación tendrá una replicación de datos independiente del motor de base de datos.

El objetivo de esta implementación fue demostrar como el uso de Manager Distribution, los servicios Windows y los servicios Web para replicación, hicieron posible la replicación de datos de Easy Tour Guide de manera que esta tarea se haga de manera independiente del motor de base de datos y de la aplicación en particular. Para llegar a este resultado fue necesario cumplir con ciertas características en cuanto al diseño de las tablas a ser replicadas, para la que el resultado de la replicación sea óptimo.

10 LINEAMIENTOS PARA EL DISEÑO DE TABLAS PARA REPLICACION

Dado que nos encontramos con diferentes motores de base de datos, es necesario que las tablas a replicar sean mapeados a tipos de datos que se puedan soportar por diferentes motores de bases de datos sin causar pérdida de información. La Tabla 33 muestra los tipos de datos soportados por los motores SQL Server y Oracle.

Tipos de datos SQL Server y Oracle[16]	
Int	timestamp
binary	char
smallint	datetime
tinyint	float
Bit	image
decimal	real
numeric	smalldatetime
money	text
smallmoney	varchar
varbinary	

Tabla 33 - Tipos de datos soportados por Oracle y SQL Server

Con el objetivo de replicar, la definición de los campos de las tablas deben estar en lo posible definidos en los tipos de datos especificados en la Tabla 33.

Para las tablas definidas dentro de las bases de datos que reciben publicaciones deben estar definidas con los mismos campos que está definida la tabla publicada pero sin restricciones de llave primaria o de campo único o bien llaves foráneas. Esto dado que cuando se selecciona un tipo de suscripción que adiciona datos y no borra los existentes, se pueden presentar problemas de inconsistencia. Esto no significa que las tablas que hacen parte de una publicación deban estar definidas sin llave primaria o sin restricciones de unicidad en los campos, solo las tablas que son parte de la suscripción son las que tienen este tipo de restricción, claro está solo cuando el tipo de suscripción obliga a conservar los datos que han sido sincronizados anteriormente.

Seleccione bien que tabla va a replicar. Hacer una mala selección al respecto puede provocar fallas de rendimiento e inconsistencia de datos. Una buena planeación de la programación de publicaciones así como sus distribuciones y suscripciones son la clave de un buen rendimiento y una alta disponibilidad.

Para la selección de la tabla se debe tener en cuenta también el tipo de replicación a usar. Por favor diríjase al capítulo II sección 4 en donde se explican los tipos de replicación existentes.

La periodicidad de la publicación debe ajustarse a las necesidades del contexto en cual esa información se requiera en otros servidores de base de datos. Generalmente la replicación por instantáneas demanda poca actualización de datos, puesto que esta operación es costosa a nivel de red. Además el seleccionar este tipo de replicación implica que los datos no cambian con frecuencia, dado que para esto existen modelos de replicación que se ajustan de mejor manera a constantes cambios de información.

La suscripción a una publicación debe realizarse teniendo en cuenta los tiempos de publicación. Esto porque en caso de realizar una suscripción por extracción (Pull) que contenga una programación que se anticipe a la programación asignada a la publicación genera datos obsoletos o bien reporte de errores (en caso de que sea la primera publicación y la suscripción solicite primero la publicación, sin que esta este disponible.)

Balance de cargas al elegir los servidores que serán responsables de cada rol dentro de la red de distribución. Si bien existen esquemas en donde se usan distribuidores locales, el hecho de tener un alto volumen de publicación y

suscripción puede provocar que el servicio de replicación y de eficiencia de operaciones en la base de datos se vea afectado. De igual forma cuando se asignan distribuidores remotos y el volumen de publicación tiene intervalos de repetición continuos, se puede provocar un bloqueo en la red dado la generación de publicaciones se basa en mensajes entre los servidores con alto volumen de transferencia de datos.

11 MANAGER DISTRIBUTION EN EASY TOUR GUIDE

Easy Tour Guide es un sistema para el turismo que maneja información acerca de una localidad en particular. Dentro del modelo relacional de la base de datos de Easy Tour Guide existen tablas de alto nivel de transacción así como otras que prácticamente son de solo lectura. Estas tablas son las propicias para la programación de replicación de datos, esto por las características del tipo de replicación implementado (Instantáneas). La Figura 49 se muestra el modelo relacional de la base de datos de Easy Tour Guide.

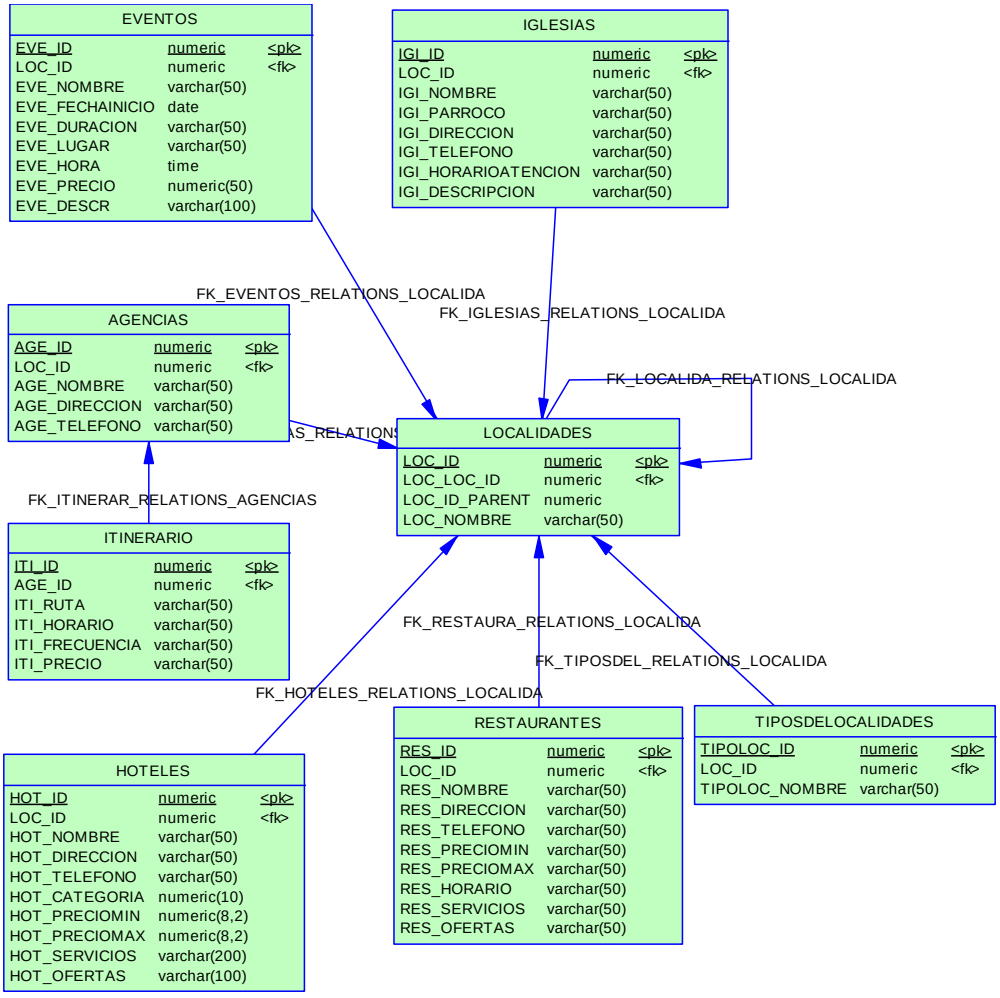

Figura 49 - Modelo de base de datos EasyTour Guide 2007

La tabla que fue seleccionada para la replicación fue Localidades, esto puesto que es una tabla con poco nivel de cambio y que prácticamente es de solo lectura, cumpliendo así una de las condiciones para seleccionar replicación por instantáneas. Por otro lado la definición de los tipos de datos son soportados por todos los motores de base de datos.

Se procede entonces a programar la publicación definiendo una nueva publicación para la tabla localidades y tipos de localidad. En la Figura 50 y Figura 51 se ve el detalle de esta publicación, que fue generada con dos artículos (localidades y tipo_de_localidad respectivamente), con un intervalo de repetición del 1er día de cada 2 meses, programación que no finaliza. Para

este caso el distribuidor elegido es el distribuidor local. Es decir que en el mismo equipo en donde se realiza la publicación se realizarán los servicios de distribución para esa publicación en particular.

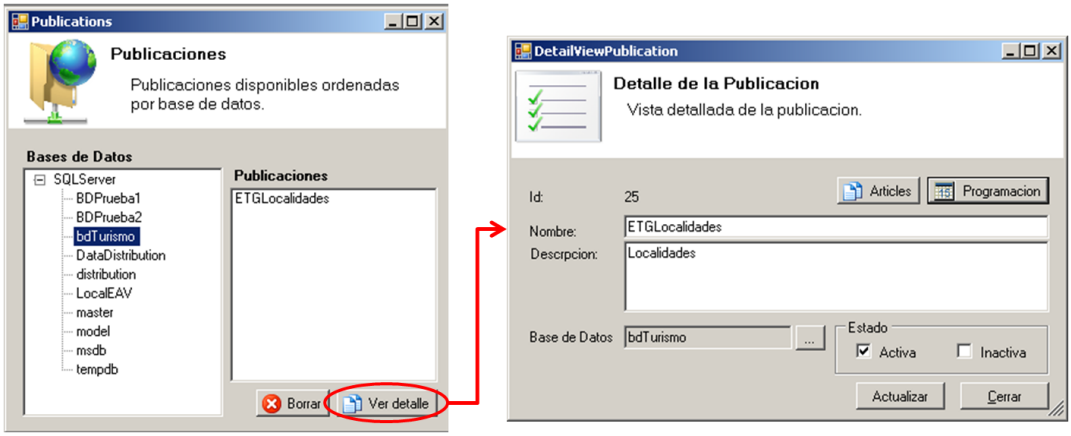

Figura 50 - Detalle publicación generada para la tabla Localidades

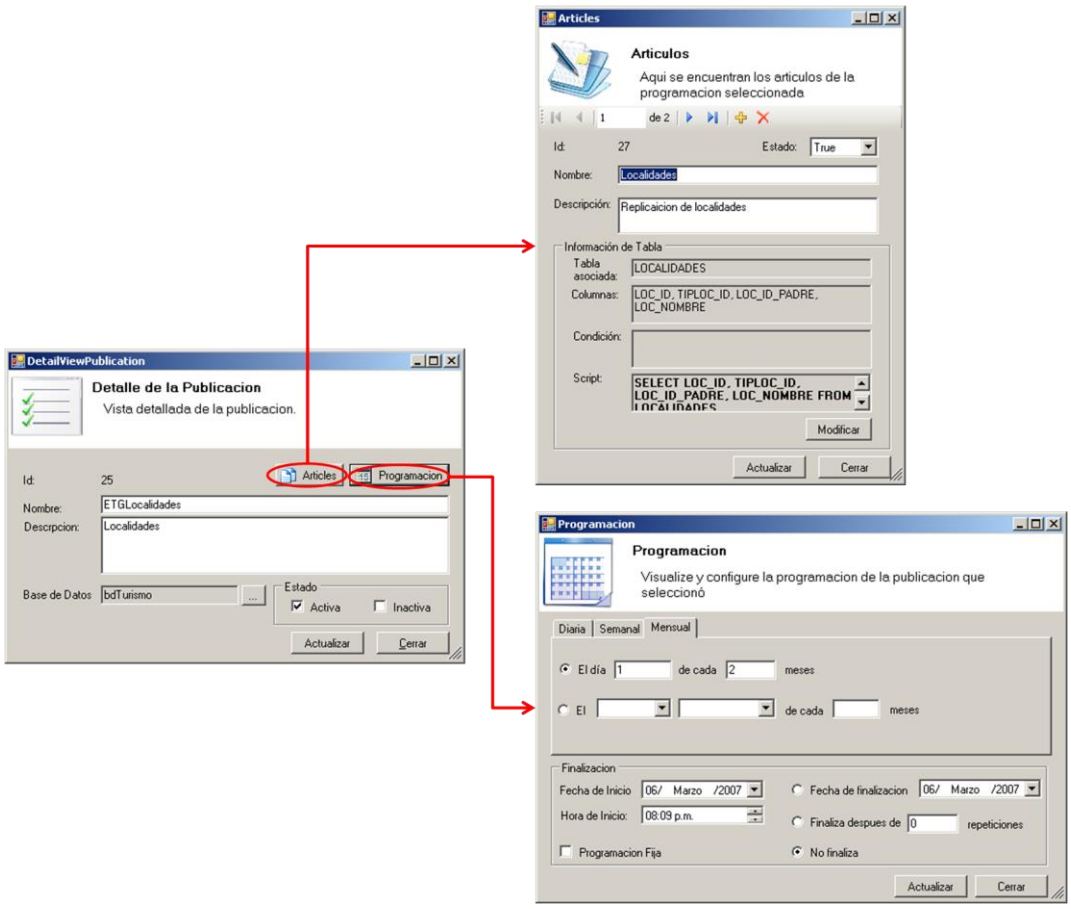

Figura 51 - Detalle de artículos y programación de la publicación generada para Localidades.

11.1 AGREGACION DEL SERVICIO WEB DE PRUEBA

Dentro de la arquitectura del sistema Easy Tour Guide, se tiene basada su arquitectura en servicios Web que son consumidos por dispositivos móviles.

En la capa de servicios del sistema, se hace necesario realizar un cambio, para que realice consultas distribuidas en diferentes servidores. Lo ideal en esta parte seria contar con un servicio que se encargue de la realización de balance de carga y permita asignar consultas distribuidas de acuerdo al nivel de consulta a un servidor en particular, pero esto no se contempla como objetivo de este trabajo de grado, por lo que se diseño un servicio sencillo que realiza

consultas de los datos de localidades de manera aleatoria en los dos servidores en donde se encuentran los datos replicados. La Figura 52 muestra como se integra este servicio dentro de la arquitectura de ETG.

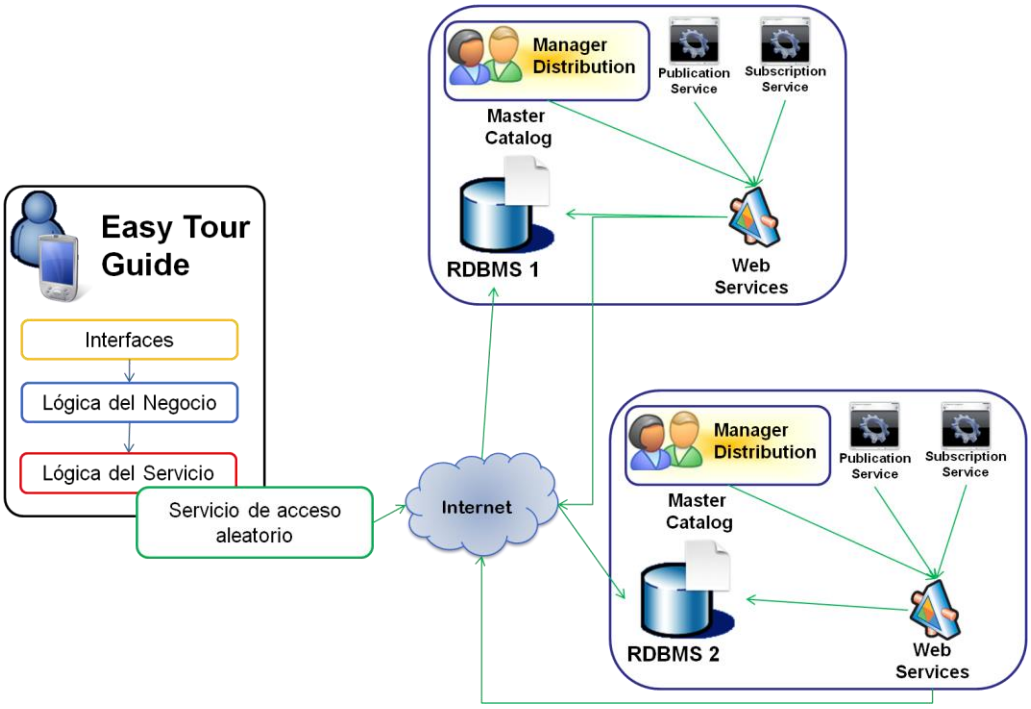

Figura 52 - Sistema de replicación implantado en EasyTour Guide

Una vez el servicio de acceso aleatorio a los datos se integra a la capa de servicios de Easy Tour Guide (o de cualquier aplicativo), todas las consultas asociadas a la tabla o las tablas replicadas se realizan en diferentes motores de base de datos. Con esto se demuestra como el acceso a la información en diferentes motores de base de datos es transparente para el aplicativo que usa el servicio de acceso aleatorio y como a través de los servicios de replicación realizados por Manager Distribution, permiten que la misma información esté disponible en contextos diferentes y pueda ser replicada de manera independiente al motor de base de datos.

De manera general, Easy Tour Guide 2007 se implemento usando una arquitectura convencional de tres capas que usa servicios Web para el consumo y tratamiento de la información turística de un lugar en particular.

Para probar que Manager Distribution funciona en particular en esta aplicación, se siguieron los siguientes pasos:

1. Generación de una publicación que incluya objetos de la base de datos de turismo implementada en el motor de base de datos SQL Server, en otra base de datos implementada en un motor de base de datos Postgres.
2. Generación de una suscripción a la publicación creada en el primer punto, que permita la replicación de la información expuesta en la publicación se pueda trasladar al servidor desde el cual se está generando la suscripción.
3. Agregación de un modulo de acceso aleatorio a las bases de datos involucradas en los puntos uno y dos. Este modulo hace consultas a la tabla replicada sobre cualquiera de los dos motores de base de datos. Cabe notar que este modulo abre una nueva oportunidad para quienes deseen experimentar sobre consultas distribuidas con la creación de un componente (Bloque de aplicación) que implemente alguna de las estrategias existentes en el balance de carga de un servidor, etc.
4. Finalmente se ve como la aplicación hace petición de información a las diferentes de bases de datos obteniendo los mismos resultados.

CAPITULO VI: CONCLUSIONES Y TRABAJO FUTURO

En esta parte final se presentan las conclusiones relacionadas con el uso de un sistema de replicación de datos independiente del motor de base de datos y de la aplicación, además de las conclusiones propias del trabajo de investigación. Por otro lado también se muestran algunas conclusiones acerca de la aplicación de prueba y su impacto en la sociedad. Por último se presentan las actividades futuras a desarrollar por parte del investigador y unas recomendaciones para otros investigadores que se motiven a realizar trabajos en esta área.

12 CONCLUSIONES

- La realización de un modelo relacional abierto para soportar replicación por instantáneas en múltiples motores bases de datos es el primer paso para la construcción de servicios independientes de replicación que permitan la integración de sistemas con motores de base de datos heterogéneos.

- Exponer los servicios que soportan la replicación permite que otros aplicativos los usen para personalizar sus servicios de replicación.

- La utilización de servicios independientes que integren diversas plataformas y sistemas, es una tendencia global que permite ver hoy en día más productos e investigaciones orientadas a la creación de estándares que permitan la interoperabilidad de sistemas ofreciendo herramientas fáciles de ajustar a modelos ya existentes.

- El éxito de la realización de replicación de datos dentro de un sistema no solo depende de la posibilidad de hacerla de manera dependiente o independiente del motor de base de datos, sino de un análisis detallado sobre los motivos que llevan a un administrador de base de datos hacer la replicación de información. Dado el caso se decida hacer la replicación de información, se requiere analizar el tipo de replicación a realizar, las

características de los objetos a replicar según el tipo de replicación y finalmente el periodo de repetición con el cual los datos deben replicarse a sus suscriptores.

- La creación de sistemas para el turismo en Colombia es un mercado que está en auge, tal como lo mostró el reconocimiento que hizo Movistar y CINTEL al sistema de turismo Easy Tour Guide como uno de los proyectos de grado más sobresalientes en el área del turismo en su convocatoria de proyectos realizada en el año 2005.

- Está visto que las empresas Colombianas y mundiales están dispuestas a generar investigación a través de productos que sean generados desde la academia. Es indispensable que los actuales trabajos de grado y como tal la profesión de ingeniero de sistemas este directamente ligado a las necesidades cambiantes de las empresas. Gracias a la experiencia con Easy Tour Guide en donde se tuvo la oportunidad de recibir la retroalimentación de Telefónica no solo acerca del proyecto como tal sino también de lo que este tipo de empresas buscan en los estudiantes de último semestre, concluí que se debe iniciar una labor mancomunada entre universidad y empresa como experiencia obligatoria en la formación integral del Ingeniero de Sistemas del mañana.

13 TRABAJO FUTURO

- Para la Universidad del Cauca, es claro que se abre una nueva área de interés dentro del grupo de investigación en Tecnologías de la Información respecto a la replicación de datos y bases de datos distribuidas (en especial el desarrollo de la computación en Grid). Luego el estudio de otros tipos de replicación, el manejo de transacciones distribuidas y todos los conceptos asociados a la construcción de servicios que ofrezcan interoperabilidad entre los sistemas existentes son un punto de estudio con gran auge en el mundo. De igual forma y para entrar más en detalle en el tema de replicación, es posible también realizar estudios respecto a nuevos algoritmos y tipos de replicación que permitan tener mayor eficiencia y menos restricciones para implementar servicios de distribución de datos.

- Dentro de la implantación de este servicio de replicación en particular se resaltó la necesidad de tener un componente de balance de carga que se pueda integrar a la capa de servicios y permita la realización eficiente de consultas distribuidas. Esto además de volver al sistema mucho más rápido, permite desarrollar en estos sistemas alta disponibilidad.

- La construcción de sistemas para el turismo es un mercado el cual se debe seguir explotando. La integración de varias áreas del conocimiento aplicado a una problemática real como la inclusión de tecnologías de la información en áreas como el turismo permiten potencializar una de las economías base de nuestra sociedad.

BIBLIOGRAFÍA

[1]. MSDN Library. Microsoft Corporation. Smart Client Definition. Smart Client Developer Center [en línea]. Publicado en septiembre de 2004. Disponible en Web: http://msdn.microsoft.com/smartclient/understanding/definition/default.aspx

[2]. Open GIS Consortium. Buehler Kurt. OpenGIS Reference Model [en línea]. Referente Number OGC 03-040. Publicado el 3 de marzo de 2003. Disponible en Web: http://www.opengeospatial.org/specs/?page=orm

[3]. DATE, C. J. Introducción a los sistemas de bases de datos: Bases de datos distribuidas. Septima Edición. Prentice Hall. Mexico, 2001. Pag. 651.

[4]. BAKKEN, David E. Middleware: Definición. Encyclopedia of Distributed Computing, Kluwer Academic Press, 2003.

[5]. Microsoft Official Course. 2072A - Microsoft Corporation. Administering a Microsoft SQL Server 2000 Database. Microsoft Learning, 2003.

[6]. MSDN Library. Microsoft Corporation. Types of Replication [en línea]. Disponible en Web: http://msdn2.microsoft.com/en-us/library/aa179414(SQL.80).aspx

[7]. SILBERSCHATZ, Abraham, KORTH, Henry F. y SUDARSHAN S., "Fundamentos de bases de datos", 3a. edición, McGraw-Hill/Interamericana de España, España, 1998.

[8]. MSDN Library. Microsoft Corporation. No-Touch Deployment in the .NET Framework [en línea]. Disponible en Web: http://msdn2.microsoft.com/en-us/library/aa289511(VS.71).aspx

[9]. DISEGNI, Carolina. Tecnología e Internet fomentan el desarrollo mundial del turismo. Diario La Tercera (Chile), 9 de Agosto de 2004. Disponible en Web: http://www.turistel.cl/noticia_turismo/pag/internet.htm

[10]. CARMONA Alvaro. MONSALVE Jhon Jairo. Sistemas de Información Geográficos [en línea]. Publicado el 7 de diciembre de 1999. Disponible en Web: http://www.monografias.com/trabajos/gis/gis.shtml

[11]. GUEVARA Tinoco Roberto. Definición y algunas aplicaciones de los sistemas de información geográfica [en línea]. Publicado el 6 de febrero de

2004. Disponible en Web: http://www.monografias.com/trabajos14/informageogra/informageogra.shtml

[12]. GALEANO Rafael E. SIG Institucional [CD-ROM]. En capacitación en SIG Institucional. Popayán Universidad del Cauca 2003.

[13]. W3C. Extensible Markup Language (XML) 1.0 (Fourth Edition) [en línea]. Publicado el 16 de Agosto de 2006 y editado el 29 de Septiembre de 2006. Disponible en Web: http://www.w3.org/TR/2006/REC-xml-20060816

[14]. Computer Desktop Encyclopedia. Web Services [en línea]. Publicado en 2002. Disponible en Web: http://computing-dictionary.thefreedictionary.com/Web+services

[15]. Enterprise Library. Patterns and Practices. The enterprise library applications blocks [en línea]. Disponible en Web: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnpag2/html/entlib.asp

[16]. GULUTZAN Peter. Standard SQL [en línea]. Publicado en Septiembre de 2002. Disponible en Web: http://www.dbazine.com/db2/db2-disarticles/gulutzan3