Sincronización de Bases de Datos Móviles Basada en Servicios Web Bajo una Aproximación Multiplataforma

Jorge Alejandro Astudillo Gutiérrez
Gustavo Adolfo Álvarez Leyton

Director: Mag. Francisco Orlando Martínez Pabón
Co - Director: Dr. Gustavo Ramírez

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Telemática
Línea de Investigación de Servicios Avanzados de Telecomunicaciones
Popayán, Agosto de 2013

Sincronización de Bases de Datos Móviles Basada en Servicios Web Bajo una Aproximación Multiplataforma

Jorge Alejandro Astudillo Gutiérrez
Gustavo Adolfo Álvarez Leyton

Director:
Mag. Francisco Orlando Martínez Pabón
Co – Director:
Dr. Gustavo Ramírez

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Telemática
Línea de Investigación de Servicios Avanzados de Telecomunicaciones
Popayán, Agosto de 2013

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS.

1. INTRODUCCIÓN

1.1. PLANTEAMIENTO DEL PROBLEMA

La evolución de las tecnologías móviles en los últimos años ha facilitado su masificación a nivel global, al punto que hoy en día cuenta con más de cinco billones de usuarios alrededor del mundo, cerca del 80% de la población mundial [1]. Estos avances han motivado el aprovisionamiento de nuevas y sorprendentes características en dispositivos móviles como Smartphone y tabletas, junto con ellos las bases de datos móviles igualmente han experimentado una gran evolución siendo cada día más populares[2][3]. Gracias a esta tendencia se ha impulsado la creación y uso de una gran variedad de aplicaciones y servicios para todo tipo de usuarios, incluyendo el sector empresarial, donde los dispositivos móviles se están convirtiendo en una plataforma popular para las aplicaciones de negocios (e-bussines)[4][5], impulsando a las empresas a movilizar sus procesos y aplicaciones (m-bussines), permitiendo así el acceso, manipulación y gestión de la información empresarial desde cualquier lugar y dispositivo. La implementación de este tipo de soluciones optimiza los procesos en diferentes niveles de las organizaciones, reduce tiempos y costos, y aumenta la productividad y competitividad[6][7]. Sin embargo, las aplicaciones móviles empresariales presentan grandes desafíos a los desarrolladores como: la seguridad, la gestión de dispositivos y aplicaciones, las herramientas y plataformas de desarrollo, la variedad de dispositivos cliente y aplicaciones empresariales, y por último la integración con los sistemas existentes[8][9][10].

Según The Global Information Technology Report 2013[11] y el Network Readiness Index, que mide que tan bien preparados están los países para hacer uso de las tecnologías de información y comunicación, que impulsan la competitividad y el bienestar; en el sector de los negocios Colombia ocupa el lugar 77 entre 144, con un puntaje de 3.39 en una escala de 1 (peor) a 7 (mejor)[12][11], y en similares o peores circunstancias se encuentran otros países de la región. Evidenciando así, una brecha y a la vez el espacio para la investigación y el desarrollo de trabajos que conduzcan a facilitar la apropiación de este tipo de soluciones tecnológicas en el sector empresarial.

La popularidad de la computación móvil, se debe en gran parte a su capacidad para ofrecer información a los usuarios en cualquier momento y lugar [13], en este sentido, una de las características más importantes de las aplicaciones modernas, es la gestión de datos desde cualquier dispositivo [14], información que puede ser almacenada a través de bases de datos SQL, texto plano, XML y otros formatos populares. La movilidad de los usuarios y la portabilidad de los dispositivos representan nuevos problemas en la gestión de los datos[7] [8], ya que en un

entorno distribuido, estos datos pueden ser generados y modificados por los clientes incluso cuando están desconectados, para luego ser sincronizados a un sistema de información central [17].

De esta manera la computación móvil desconectada permite al usuario trabajar fuera de línea y posteriormente sincronizar sus datos y cambios locales con el servidor empresarial. Esta ofrece grandes beneficios y ventajas sobre las aplicaciones que necesitan de una conexión permanente a algún tipo de red como: en primera instancia reduce el consumo de batería del dispositivo[18], igualmente minimiza el costo por concepto de transmisión de datos sobre la red del operador [19], y además permite que las funciones de la aplicación pueden ser accedidas sin dependencia de una conexión de red, aumentando la autonomía, disponibilidad y el rendimiento de los clientes móviles[3].

Sin embargo cuando una aplicación soporta el trabajo desconectado o fuera de línea, se presentan contradicciones inevitables de información entre la base de datos central y la base de datos de los móviles [20]. La sincronización de datos es una acción automatizada, que permite a los datos replicados ser coherentes entre sí y permanecer actualizados [21]. De esta manera se resuelven las inconsistencias, se garantiza la integridad y se mantiene la misma versión de los datos entre múltiples dispositivos cliente y una o varias bases de datos centrales. En consecuencia, la sincronización de bases de datos móviles es uno de los temas esenciales en entornos de computación ubicua [22].

El gran interés que existe sobre la sincronización de bases de datos móviles desde hace varios años, ha convertido a las bases de datos y la gestión de datos móviles, en un área fértil de trabajo para investigadores[23] [24]. Sin embargo, las soluciones actuales presentan diversos inconvenientes como: el alto grado de acoplamiento entre sus componentes, generado entre otras razones, porque las soluciones dependen del tipo de servidor de base de datos, ya que utilizan meta-información específica[20]. En el mismo sentido, existe dependencia con respecto a la plataforma móvil usada e incluso algunas soluciones de sincronización son diseñadas específicamente para una aplicación, sistema operativo en concreto, protocolo de transporte o sincronización específico.

A las dificultades mencionadas, se le suma la alta fragmentación que existe en el mercado de los dispositivos móviles, lo cual promueve la falta de estándares consolidados. Respecto a la sincronización de bases de datos móviles, existe una iniciativa interesante de estandarización en SyncML, un protocolo abierto desarrollado por Open Mobile Alliance (OMA) para la sincronización de datos genéricos [25], que a pesar de ser una especificación muy completa y estar soportada por muchos fabricantes, no ha sido adoptada como un estándar de

facto. Igualmente, un gran número de soluciones de sincronización (algunas incorporan SyncML) concentran sus esfuerzos en funciones PIM (Personal Information Management), que se dedica específicamente a la implementación y estudio de las actividades realizadas en la gestión de datos personales[26].

No obstante, recientemente se ha dado paso a una nueva aproximación, para facilitar la sincronización de bases de datos móviles bajo estándares Web. Frameworks como RhoSync, un nuevo tipo de servidor de sincronización de código abierto para dispositivos móviles dirigido a aplicaciones empresariales [27], ofrecen la posibilidad de replantear la lógica de sincronización como servicio, específicamente a través de servicios Web; esto facilita su integración y despliegue en las arquitecturas modernas orientadas a servicios. Sin embargo existen pocos proyectos o trabajos que utilicen este enfoque, las soluciones libres ejecutan básicamente las tareas clásicas de sincronización como add, update, delete, pero no manejan claramente aspectos trascendentales como la resolución de conflictos y las políticas propias de sincronización, particulares a cada organización, ya que las necesidades varían en cada caso; por otro lado, varios de estos Framework's son propietarios, lo cual dificulta su extensión y adopción por parte de pequeñas y medianas empresas.

De acuerdo al contexto presentado anteriormente, el presente proyecto pretende resolver la siguiente pregunta de investigación: ¿Cómo facilitar el desarrollo de aplicaciones para la sincronización de bases de datos móviles, usando servicios Web bajo una aproximación multiplataforma?

1.2. OBJETIVOS

1.2.1. Objetivo general

Adaptar un Framework basado en servicios Web para la sincronización de bases de datos en dispositivos móviles usando una aproximación multiplataforma.

1.2.2. Objetivos específicos

- Determinar los lineamientos requeridos para la sincronización de bases de datos en dispositivos móviles usando una aproximación multiplataforma basada en servicios Web.
- Adaptar un Framework existente de sincronización de bases de datos en dispositivos móviles que usan servicios Web, de acuerdo a los lineamientos planteados.
- Evaluar el mecanismo de sincronización de datos y su implementación a través de un piloto desarrollado en el contexto de las tareas de recolección de información en campo de la empresa UTEN seccional Cauca.

1.3. APORTES DEL PROYECTO

El presente trabajo de grado pretende realizar los siguientes aportes:

- Definición de un conjunto de lineamientos para facilitar la sincronización de bases de datos móviles basada en servicios Web, bajo una orientación multiplataforma que simplifique el desarrollo de este tipo de aplicaciones y su despliegue e integración en una Arquitectura Orientada a Servicios.
- Adaptación de un Framework existente para la sincronización de bases de datos móviles por medio de servicios Web, basada en los lineamientos planteados.
- Piloto de una aplicación móvil de recolección de datos en campo en el contexto de la empresa UTEN Seccional Cauca.

1.4. ESTRUCTURA DEL TRABAJO DE GRADO

El contenido de la monografía se ha organizado en cinco capítulos como se muestra a continuación:

Capítulo 2: En este capítulo, se abordan las definiciones formales de conceptos claves para el entendimiento del proyecto realizado; se construye una base inicial de conocimiento sobre los temas directamente relacionados con el presente trabajado de grado y se presentan los trabajos relacionados correspondientes.

Capítulo 3: Se definen un conjunto de lineamientos para la construcción de aplicaciones de sincronización de bases de datos móviles basadas en servicios Web bajo una aproximación multiplataforma de acuerdo al contexto anterior y estructurados sobre la base de los productos, trabajos y definiciones de empresas líderes en el mercado de la informática.

Capítulo 4: Se desarrolla la adaptación de un Framework existente, ampliando sus funcionalidades y características; con el propósito de proveer las herramientas necesarias para facilitar y optimizar el desarrollo y despliegue de aplicaciones de sincronización de bases de datos basadas en servicios Web.

Capítulo 5: Se describe un caso de estudio en el marco de las labores de recolección de datos en campo de la empresa UTEN Seccional Cauca, se construye un piloto usando la adaptación desarrollada en el capítulo anterior y aplicando los lineamientos definidos en este trabajo, con el objetivo de hacer algunas pruebas al mecanismo de sincronización y los lineamientos definidos.

Capítulo 6: Se presentan los aportes y conclusiones del trabajo desarrollado y se plantean posibles trabajos futuros.

2. ESTADO DEL ARTE

2.1. Introducción

En este capítulo se presentan las definiciones y bases teóricas, necesarias para comprender la temática esencial del proyecto realizado. A continuación, se exponen los principales conceptos, en los que se fundamenta la propuesta de los lineamientos para la construcción de aplicaciones de sincronización de datos móviles, y posteriormente se presenta el estado actual de los proyectos, trabajos e iniciativas relacionadas con esta temática.

2.2. Contexto General

2.2.1. Bases de datos móviles

Las bases de datos móviles se pueden definir como una base datos portátil, físicamente separada del servidor central de base de datos, sin embargo esta base de datos tiene la propiedad de conectarse con el sitio central o maestro para intercambiar datos y mantenerse actualizada [28]; por lo que deben incluir un mecanismo de comunicación entre el servidor central y la base de datos del cliente móvil. Uno de los mecanismos usados para implementar bases de datos móviles de acuerdo con esta definición, es la replicación de datos móviles, que involucra el proceso de sincronización de datos.

2.2.2. Sincronización y Replicación de Bases de datos.

La replicación de datos consiste en mantener múltiples copias de información, llamadas réplicas, en equipos independientes, es una tecnología importante para los entornos distribuidos ya que permite una mayor disponibilidad y rendimiento[29][30]. Este conjunto de tecnologías están destinadas a la copia y distribución de datos y objetos de datos por medio de replicas, que pueden sincronizarce entre si para mantener la coherencia de la información. Permite distribuir datos entre diferentes ubicaciones y entre usuarios remotos o móviles mediante redes locales y/o de área extensa; por medio de conexiones de acceso telefónico, conexiones inalámbricas e Internet[31][32][33]. Los cambios aplicados en la réplica de cualquier sitio se capturan y se almacenan localmente antes de ser transmitidos y aplicados a cada uno de los sitios remotos restantes, este proceso es conocido como sincronización o reconciliación de los datos y se refiere al proceso de propagación de los cambios en los datos locales de una réplica por el entorno de replicación[34][35][36][37].

Entre las razones comunes para usar la replicación se destacan:

Disponibilidad: La replicación mejora la disponibilidad de las aplicaciones, ya que provee opciones alternativas de acceso a los datos. Si un sitio no se encuentra disponible, los usuarios pueden continuar consultando o actualizando desde las locaciones restantes o desde la réplica local.

Rendimiento: La replicación provee un rápido acceso local a los datos, que incrementa el rendimiento, balancea la actividad de actualizaciones y consultas sobre múltiples sitios, y no sobre uno solo. [38]

Computación Desconectada: Los sitios cliente con una copia o replica de una base de datos que puede ser completa o parcial, permiten a los usuarios trabajar sobre un subconjunto de la base de datos mientras están desconectados del servidor de base de datos central. Más tarde, cuando se establece una conexión, los usuarios pueden sincronizar los datos y mantener la coherencia entre los datos locales y remotos.

Reducción de la carga y Despliegue en masa: La replicación puede ser utilizada para distribuir datos sobre múltiples localizaciones regionales. Así, las aplicaciones pueden acceder a los datos en diferentes sitios en lugar de acceder a un servidor central.

2.2.2.1. Sitios de Replicación

En un entorno de replicación se manejan basicamente dos tipos de sitios, como se puede apreciar en la figura No1. El sitio maestro que aloja la base de datos central y los objetos de replicación principales que funciona como un control o gestor del entorno de replicación, y los sitios clientes que disponen de una copia replica local parcial o total de los objetos de replicación, estos sitios pueden tener la facultad de actualizar los datos o simplemente de consultarlos (esclavo). Es posible configurar entornos de replicación con múltiples sitios maestros o múltiples sitios clientes dependiendo las necesidades del entorno de sincronización en concreto.

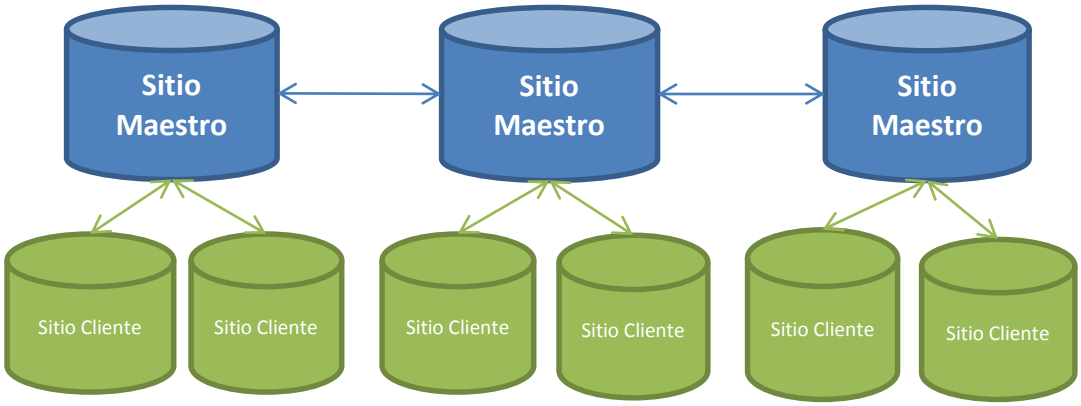


Figura No 1. Sitios en un entorno de replicación de datos

En general, existen dos modos básicos de sincronización de datos, la replicación asíncrona y la replicación síncrona:

2.2.2.2. Replicación Asíncrona u optimista y los conflictos de datos

La replicación asincrónica es el medio más común para implementar la replicación, con la replicación asincrónica, los cambios hechos en un sitio de replicación ocurren en un tiempo posterior en los otros sitios participantes del entorno de replicación, esto quiere decir que los cambios de un sitio se demoran en propagarse a otros permitiendo así la computación desconectada[39][40].

El hecho de usar la replicación asincrónica significa que se pueden presentar situaciones en las que varios usuarios intentan modificar un dato al mismo tiempo, es necesario establecer controles para impedir que las modificaciones de un usuario influyan negativamente en las de otros[41]. Para Microsoft ADO.NET [42], el sistema mediante el cual se controla lo que sucede en estas situaciones se denomina control de concurrencia y para Oracle Advanced Replication se denomina manejo de conflictos[43], en el presente trabajo se adoptada la definición de Oracle.

Por este motivo durante una sincronización de datos se debe tener especial cuidado antes de que los datos manipulados y enviados por un cliente sean escritos en el sitio destino maestro. Debido a que el registro objetivo también puede ser modificado desde otros clientes, y pueden haber cambiado los datos desde la última sincronización. Antes de escribir realmente la actualización en el sitio objetivo maestro, los conflictos de datos deben ser detectados y resueltos[44][45][46][47].

Los conflictos de datos más comunes que se pueden presentar en un entorno de replicación estan divididos en las siguientes categorías[48][49][50] :

A. Conflictos de Actualización: Se producen cuando los datos que se van a actualizar en el sitio maestro destino por un cliente A, han sido actualizados por otro cliente B después de la última sincronización del cliente A.

B. Conflictos de Eliminación: Se producen cuando los datos en el sitio maestro destino que se van a actualizar o a borrar por un cliente A, han sido eliminados por otro cliente B después de la última sincronización del cliente A.

C. Conflictos de Unicidad: Se producen cuando los datos que se van a insertar o actualizar por un cliente A, provocan una violación de una restricción de unicidad en el sitio maestro destino, como una clave primaria duplicada.

D. Conflictos de Orden: Se producen únicamente entre dos o más sitios maestros de un entorno de sincronización con múltiples sitios principales o maestros, sucede cuando estos tienen versiones diferentes de los datos e intentan sincronizarse entre sí.

E. Conflictos de negocio: Se producen cuando los datos son transmitidos al sitio maestro destino correctamente, pero violan la integridad general de la base de datos central.

2.2.2.3. Replicación Sincrónica o pesimista

Con la replicación sincrónica, los cambios hechos en un sitio ocurren inmediatamente en todos los sitios participantes del entorno de sincronización, una modificación de una tabla en un sitio replica requiere de una actualización inmediata en todos los sitios del entorno o al menos en el sitio maestro, esto significa que cada transacción puede involucrar a todos los sitios, bajo este enfoque solo puede existir una versión de los datos en todo el entorno de replicación.

Aunque se minimiza la posibilidad de conflictos de datos, cuando se utiliza la replicación Sincrónica se requiere un ambiente estable para operar correctamente y un mecanismo de comunicación permanente entre la réplica cliente y el sitio central. Si la comunicación falla no se procesaran correctamente las operaciones de sincronización.

2.2.3. Sincronización de bases de datos móviles

En la actualidad existe una fuerte y creciente demanda en muchas organizaciones de todo tipo a nivel mundial de ampliar la disponibilidad de los datos. En particular, la posibilidad de acceder e ingresar datos de negocio fuera de la oficina, en el terreno. Las aplicaciones móviles representan una fuerte alternativa para atender esta necesidad, ya que permiten a los empleados manipular la información directamente en el lugar y momento donde se ejecutan las labores organizacionales[51] por medio de diferentes dispositivos móviles. Por esta razón surgen las soluciones de sincronización de bases de datos móviles que permiten a las organizaciones movilizar su fuerza de trabajo para optimizar los procesos empresariales y la gestión de la información.

Existen dos enfoques básicos para el suministro de datos móviles "siempre disponibles": Se trata de un enfoque basado en la Web, utilizando aplicaciones que acceden directamente a los almacenes de datos en el servidor empresarial, estas aplicaciones requieren una conexión (ya sea a través de una red cableada o inalámbrica) para poder trabajar desde los entornos móviles.

Un segundo enfoque es de la computación conectada intermitentemente, que permite el trabajo en modo desconectado, combinando, bases de datos móviles y herramientas de sincronización de datos. Esta solución proporciona una infraestructura para la informática móvil que se está poniendo en marcha en muchos entornos de producción y ha demostrado su fiabilidad en muchas aplicaciones del mundo real.

La sincronización móvil de datos tiene como finalidad permitir la realización de tareas CRUD (Create, Read, Update, Delete) para bases de información distribuidas en las que sus datos se pueden manipular en diversos contextos. Así, diversos dispositivos cliente con diferentes características y funcionalidades modifican las bases de datos locales, para luego conectarse y centralizar la información de manera coherente con el sitio maestro o base de datos central por medio de un proceso de sincronización de datos[17]. Al respecto existe un amplio estudio, investigación y desarrollo de un tipo de sincronización en particular, que tiene que ver con el tratamiento de información personal (PIM)[52]. Este abarca entre otros temas: agendas personales, calendarios, tareas, correos electrónicos y dependiendo de la tecnología otros tipos de información[53][54].

Los fenómenos y tendencias de la computación móvil, la caracterizan ahora como una computación más dinámica y abierta; diferentes dispositivos pueden hacer uso de varias redes para interactuar con servidores centralizados o distribuidos. Además los dispositivos móviles son cada vez más potentes para el procesamiento de datos, PDA´s, Smart Phones, Pockets PC y Tablets hacen posible el procesamiento móvil masivo de información, y el uso de sus características y periféricos en procesos empresariales donde generalmente las funciones de las aplicaciones de datos están distribuidas [20] [55].

Así, con la llegada de la computación móvil, junto con el surgimiento de las bases de datos móviles que hacen posible las actividades de consulta, registro y edición de datos puedan ahora realizarse desde cualquier lugar a través de un dispositivo de mano [56], y las necesidades empresariales de realizar tareas colaborativas y de mantener la información en sus bases de datos actualizada, íntegra, coherente y disponible entre todos los puntos de acceso, dan paso a la sincronización de bases de datos móviles. Sus funciones y procedimientos resuelven todas las necesidades que se plantean alrededor de las tareas desarrolladas en ambientes distribuidos a través de dispositivos móviles, siempre teniendo en cuenta que las aplicaciones móviles empresariales deben cumplir con tres retos fundamentales: Garantizar que los datos sean siempre accesibles y convenientes, garantizar la convergencia de los datos usando métodos como la resolución de conflictos, y garantizar la seguridad y confiabilidad de la información.

En la Figura 2, se observa un ambiente distribuido integrado por elementos genéricos que conforman un escenario de sincronización de bases de datos móviles, donde se tiene un servidor encargado de manejar la conexión con la base de datos central, un servidor de sincronización y diferentes dispositivos clientes conectados a través de internet, los cuales manejan su propia réplica local de la base de datos central.

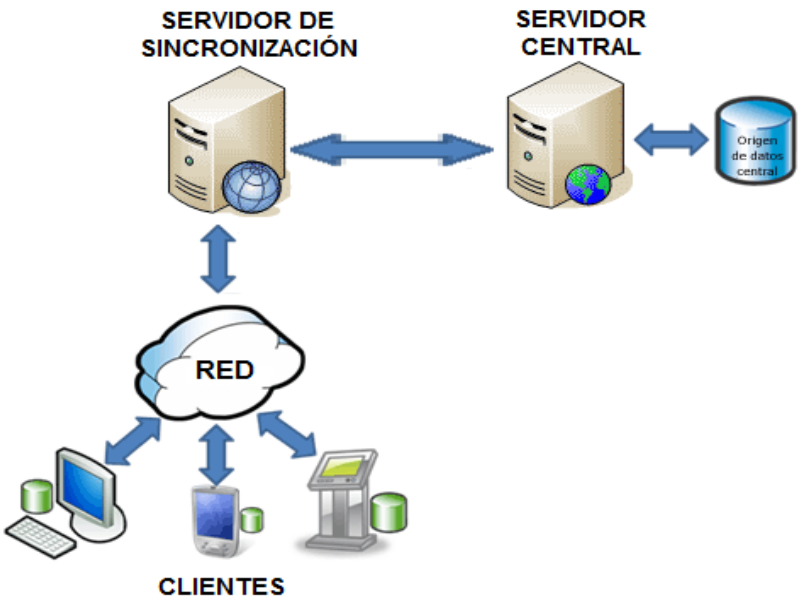


Figura No 2. Elementos genéricos en un sistema distribuido móvil - Fuente: [Propia].

2.2.3.1. Protocolo SyncML

Es importante destacar a Syncml, como el esfuerzo de estandarización para la sincronización de datos móviles. Es un Protocolo abierto para la sincronización de datos creado por OMA (Open Mobile Alliance). SyncML es una especificación basada en XML que brinda un marco común para la sincronización de datos, está diseñado para ser usado por dispositivos móviles que se conectan de forma intermitentemente a la red, también se puede utilizar para la sincronización punto a punto de datos. [57] Soporta varios protocolos de transporte, como, WAP, HTTP, OBEX y otros propietarios de comunicación inalámbrica, y permite también realizar operaciones de gestión de los dispositivos. [58]

El protocolo SyncML está descrito por un conjunto bien definido de mensajes que se transmiten entre las entidades que participan en una operación de sincronización de datos y se representan como un documento XML, que contiene los comandos del protocolo y meta-información, el protocolo está bien fundamentado en cuanto a las operaciones generales requeridas para la sincronización de bases de datos móviles, sin embargo, no es una especificación propiamente diseñada para trabajar con Servicios Web.

La aplicación directa más común de SyncML es el intercambio de información personal, referente a, contactos y calendario, entre dos dispositivos tales como un ordenador y un teléfono móvil. Para esto, se hace uso de formatos estándar para la sincronización de información personal, como vCard, para contactos e iCalendar, para calendarios y tareas. A pesar de esto el enfoque del estándar está ideado para permitir la sincronización de cualquier tipo de datos genéricos.

2.2.4. Sincronización de bases de datos móviles basada en servicios Web.

En una solución de sincronización de datos móviles existen muchas formas de almacenar los datos en cada cliente, diversos protocolos de comunicación para transferirlos, diferentes mecanismos de sincronización, varias plataformas móviles y empresariales, y diversas tecnologías de bases de datos. Esta heterogeneidad se convierte en un problema cada vez más complejo de manejar que implica sobrecostos en tiempo y recursos necesarios para desplegar un sistema de sincronización de este tipo.

Las soluciones y Framework's existentes ofrecen la posibilidad de sincronización bidireccional de datos, pero en la mayoría de casos presentan los inconvenientes descritos en el párrafo anterior y generalmente están estrechamente ligados a la tecnología de la base de datos central. Por lo tanto, si se requiere un cambio en la base de datos, como en la versión o en el motor mismo de base de datos, será también necesario modificar los clientes y/o la capa intermedia de conexión para que la sincronización se pueda seguir ejecutando[17]. Por esta razón estas soluciones presentan un alto grado de acoplamiento entre sus componentes, dependencia con la plataforma móvil y empresarial, dependencia con el protocolo de transporte y/o de sincronización, y en algunos casos mas complejos las mismas aplicaciones empresariales diseñan desde cero sus propios mecanismos para la sincronización de datos.

Como respuesta a este reto, una perspectiva SOA (Service oriented architecture), propone la implementación de servicios Web para el acceso a los datos centrales con fines de sincronización[17], replanteando entonces la lógica de la sincronización de datos como un servicio, específicamente como servicios Web.

Un servicio Web según [59] se puede definir como una aplicación accesible desde otras aplicaciones a través de internet, que permite intercambiar datos entre diferentes aplicaciones. Algunas de las principales ventajas que brinda el uso de servicios Web en el proceso de sincronización de bases de datos móviles son [17][60]:

- Interoperabilidad: El acceso a la información se da a través de las interfaces que los servicios web exponen y por lo tanto la lógica de negocio de datos es indiferente para quien accede a ellos, solo debe importar consumir la interface del servicio Web.
- Desacoplamiento: Los cambios que se hagan en el servidor empresarial o Back-End no se reflejaran en los clientes, siempre y cuando se conserve la misma estructura de las interfaces Web.
- Clientes Ligeros: Los clientes simplemente tienen la obligación de enviar y recibir la información que se ha actualizado y el servidor se encarga del resto de los procesos, como la lógica de la sincronización de datos.
- Agrupación de conexiones: Las conexiones ya no se manejaran directamente entre el motor de bases de datos y cada cliente, sino que el servicio web será el encargado gestionarlas, manejar las sesiones y permisos de los usuarios.
- Multiplataforma y Multilenguaje: Gracias al bajo grado de acoplamiento que genera entre los clientes y el servidor el uso de una perspectiva SOA, tanto en la capa empresarial como en la capa cliente se soportan aplicaciones que se ejecutan en diferentes plataformas y han sido desarrolladas con diferentes leguajes de programacion.

A continuación se mencionan tres trabajos que consideran la perspectiva basada en servicios Web para la sincronización de bases de datos móviles:

2.2.4.1. RhoSync

Rhosync es un Framework de sincronización perteneciente a la suite de productos Rhomobile, que permite la movilidad de las aplicaciones y la manipulación de datos en modo desconectado. Gracias al acceso a datos e información empresarial a través de servicios web, usando teléfonos inteligentes o tabletas[61]. La filosofía del servidor de sincronización que aplica Rhosync se enfoca en la utilización de las ventajas del SaaS (Software como Servicio) para reducir el acoplamiento con el acceso y conexión a la información disponible en los sistemas de bases de datos empresariales[62].

Los esfuerzos y la complejidad que enmarca un sistema de sincronización se reducen gracias a la utilización de Adaptadores Fuente, que podrán ser aprovechados para conectar por medio de servicios Web a los clientes móviles y los servicios expuestos por el Back-End. El adaptador Fuente es quien tiene la funcionalidad de recoger las instrucciones del cliente y hacer las operaciones adecuadas a través de los servicios Web. Se han definido un conjunto de operaciones en las que el desarrollador se concentrara y las cuales puede utilizar de acuerdo a sus necesidades particulares de sincronización[63].

Este Framework presenta varias ventajas para la construcción de las soluciones de sincronización, la primera se refiere al uso de un enfoque SOA (Service Oriented Architecture) en las conexiones hacia las bases de datos y servidores empresariales, específicamente su implementación a través de servicios Web. Esto se traduce en flexibilidad para la integración con diferentes sistemas de datos que puedan exponer interfaces de comunicación como servicios Web, y le brinda un alto grado de desacoplamiento entre sus diferentes componentes.

Otro aporte importante de este framework, está ligado al procedimiento de representación de los datos que se recogen en el servidor de sincronización, los datos son almacenados como campos OAV (object attribute value), que contribuyen al desacoplamiento con los motores de bases de datos. El Framewok utiliza una arquitectura dividida en tres capas, como se observa en la Figura No. 3.

Por último, el modelo de programación propuesto integra un mecanismo de reutilización, donde el desarrollador más que un trabajo de construcción hace es un esfuerzo de adaptación según las necesidades de su servicio o aplicación.

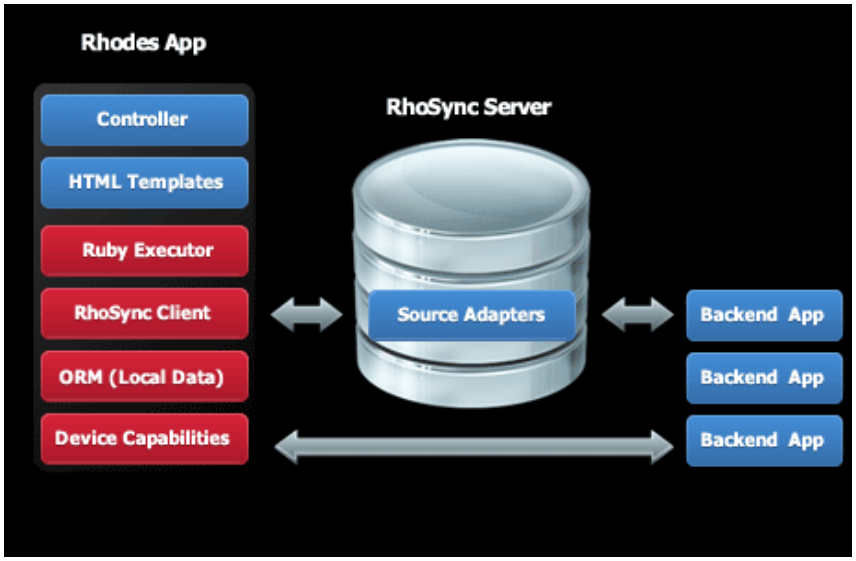


Figura No 3. Arquitectura de RhoSync [64]

2.2.4.2. RhoConect

RhoConnect es el primer servidor de una nueva categoría de trabajos de "integración" de aplicaciones móviles con servidores empresariales Back-End, desarrollado por Motorolla Solutions. Usando RhoConnect se simplifica drásticamente el desarrollo de la conexión a una aplicación Back-End, el servidor RhoConnect realiza todo el trabajo para permitir obtener los datos en el dispositivo móvil inclusive cuando está fuera de línea. Este Framework pretende eliminar

entre el 50 y el 80 por ciento del esfuerzo de desarrollo de aplicaciones que involucren la sincronización de datos[65].

RhoConnect también funciona con sincronización continua que mantiene todos los datos de las aplicaciones back-end actualizados en los dispositivos cliente, cuando la información se cambia desde las aplicaciones back-end se sincroniza inmediatamente con todos los dispositivos, de manera que la información este disponible en los dispositivos. Los datos modificados en los dispositivos también se sincronizan por RhoConnect al sistema Back-End.

RhoConnect permite el uso de Rhodes para el desarrollo de la aplicación cliente, y adicionalmente también se puede utilizar desde aplicaciones escritas en Objective C, JAVA, Android o incluso de otros framwork's que utilicen JavaScript.

Un gran porcentaje del tiempo de desarrollo que se invierte en este tipo de aplicaciones se dedica a las conexiones a los sistemas Back-End. RhoConnect reduce considerablemente este esfuerzo por crear y gestionar esas conexiones.

RhoConnect, permite conectar simultáneamente una aplicación a muchas fuentes de datos fácilmente, como un back-end de gestión de relaciones con clientes (CRM), sistemas de planificación de recursos empresariales (ERP), servicios Web, bases de datos, entre otras[66]. En la figura No. 4 se puede apreciar la arquitectura del Framework.

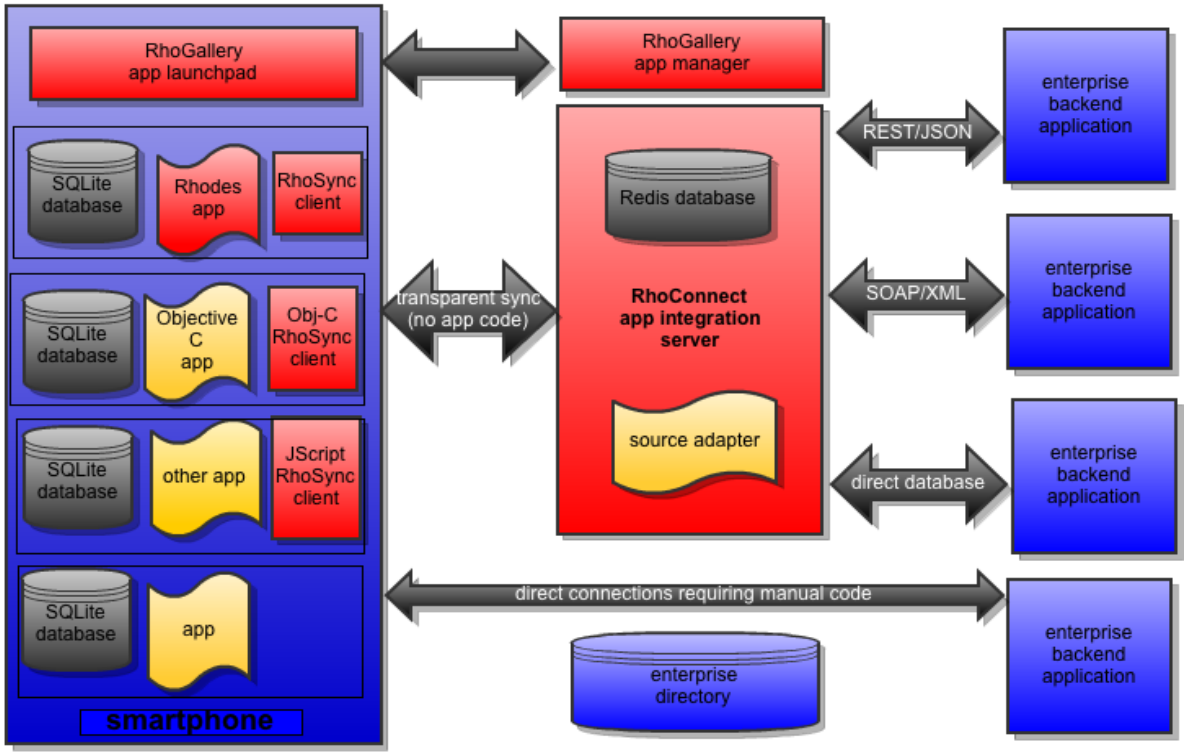


Figura No 4. Arquitectura de RhoConect [67]

2.2.4.3. Microsoft Sync Framework

Es una completa plataforma que le permite a los desarrolladores crear sus propios ambientes de sincronización, para que integren cualquier aplicación cliente con datos de cualquier almacén o base de datos, con el uso de un protocolo específico y a través de cualquier red. Microsoft Sync, permite el uso compartido de datos con múltiples topologías y permite el funcionamiento de los clientes en estados de desconexión, hace posible la colaboración y el acceso sin conexión para aplicaciones, servicios y dispositivos. Sync Framework incorpora las tecnologías y herramientas que habilitan la movilidad y el uso compartido de los datos.

Entre sus características integra la sincronización a través de servicios con otros tipos de almacenes de datos, y la detección y resolución de conflictos. En la Figura No. 5, se muestra la arquitectura de alto nivel del framework. En esta arquitectura se describe la interacción entre dos nodos o replicas, que es la forma en que siempre se produce la sincronización, independiente de la topología del sistema[68].

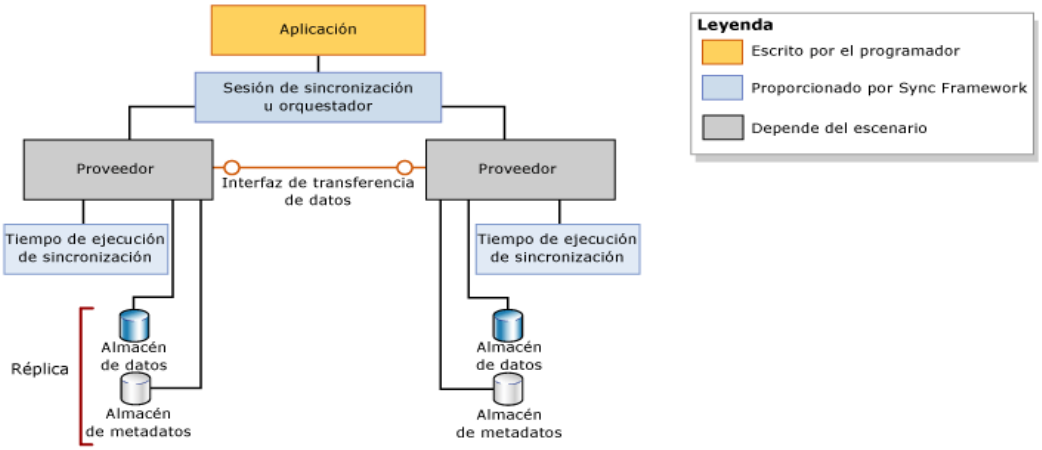


Figura No 5. Arquitectura de Microsoft Sync Framework [68]

2.3. Trabajos Relacionados.

2.3.1. SmartDOTS - Un framework para la sincronización eficiente de datos en dispositivos móviles

SmartDOTS (Smart Data Off The Spot) [55] es un Framework de sincronización para un ambiente empresarial nómada, donde existe una estructura de datos distribuidos a través de puntos de acceso inalámbrico que permite la conexión de dispositivos móviles a sus bases de datos en un núcleo fijo de equipos. El objetivo principal de este proyecto es permitir un acceso a datos eficiente y con una latencia mínima a través de la identificación de un perfil de dispositivo, red y tarea.

La replicación de almacenes de datos permitirá realizar tareas en un estado de desconexión del dispositivo, que resuelve uno de los principales inconvenientes en este ambiente de conexión.

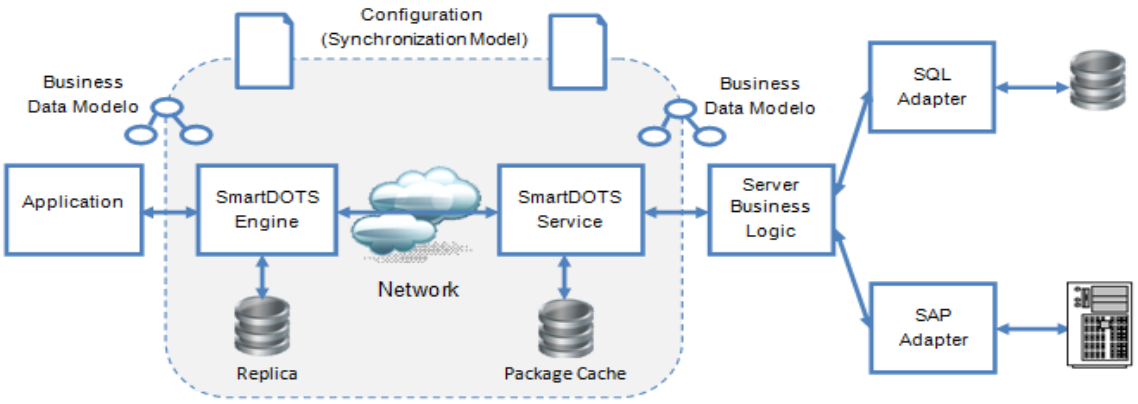


Figura No 6. Arquitectura de SmartDOTS - Fuente: Adaptado de [17].

La arquitectura manejada por este Framework ilustrada en la Figura No. 6 representa un aporte importante, ya que sus funcionalidades permiten por el lado del cliente una mejor fluidez de la información y por el lado de la capa de datos un modulo de adaptación para cada modelo especifico.

Desde el dispositivo móvil, las ventajas se evidencian con el manejo de una replica de datos, esto permite trabajar en modo de desconexion, favorable al comportamiento intermitente de las redes inalámbricas y al ahorro en el consumo de energia en el dispositivo. Por otro lado, en el servidor se plantea una caracteristica importante del Framework, que es la adaptacion de diferentes modelos de datos a uno orientado a objetos. Esto representa ventajas importantes en cuanto a portabilidad referente a DBMS y la capa de negocio con la que se puede contar ofrece productividad y mantenibilidad en el desarrollo de las soluciones.

El punto débil de este framework se ubica en el tratamiento de conflictos de las tareas de sincronización, ya que no se hace mención a algún método que pueda garantizar este proceso, aunque tenga en su arquitectura, dentro del middleware una capa de negocio que se podría aprovechar para este fin.

2.3.2. Arquitectura ligera de sincronización para bases de datos remotas en dispositivos Windows Mobile usando Servicios Web

Este trabajo describe una nueva arquitectura para sincronización de bases de datos, en la que se destacan tres componentes esenciales: La aplicación empresarial con la base de datos principal que expone los métodos de acceso a datos mediante servicios Web, usando XML para la representación de los datos.

Un mediador proxy en este caso llamado concentrador que actúa como un nivel intermedio entre una o más bases de datos empresariales, encargado de adaptar los formatos de datos para que puedan ser almacenados en el móvil o en la base de datos empresarial, realiza las operaciones de resolución de conflictos y puede determinar selectivamente que datos es necesario enviar en una operación de sincronización. El cliente por su parte cuenta con una réplica ligera de la base de datos central que permite al dispositivo trabajar fuera de línea y cuenta con una aplicación que se conecta con el concentrador para realizar las operaciones de sincronización con la base de datos empresarial, pueden conectarse mediante un canal por cable, una interfaz en serie y / o una interfaz de TCP/IP. [69]

El principal aporte de este trabajo es el uso de Servicios Web para la sincronización de bases de datos móviles, mediante una arquitectura que facilita la manera de acceder a una base de datos empresarial de cualquier tecnología, con un mediador que se encarga de ejecutar las operaciones de sincronización, que pueden ser accedidas desde los clientes móviles.

La debilidad de este trabajo es que soporta únicamente las plataformas de Windows, para la aplicación Web ASP.NET o PHP, en la lógica de sincronización usa el componente ADO.NET para las operaciones con XML y los clientes son Windows o Windows Phone, además de esto en el concentrador se aloja una réplica de la base de datos que es sincronizada con los clientes y la base de datos empresarial, lo que representa una desventaja en consumo de recurso y eficiencia frente al modelo en que la base de datos central es sincronizada directamente con la del dispositivo móvil.

2.3.3. Sync: un framework Java para Aplicaciones Móviles Colaborativas

Este trabajo presenta un framework de sincronización centrado en Java, por lo tanto maneja objetos como representación directa de los datos que se intercambian en los procesos de sincronización[19]. El desarrollo de aplicaciones bajo este modelo estará compuesto principalmente por tres componentes:

- Los almacenes de Sincronización (Sync Stores)
- Los Sincronizadores (synchronizers)
- Gestores o Manejadores de Almacén (store managers).

El primero es una colección persistente de datos con su respectiva representación Java, que provee acceso a través de un objeto interfaz llamado SyncStore. El segundo es el Sincronizador, este es el encargado de manejar las actualizaciones locales a enviar, recibir, y aplicar las que se obtengan desde otros dispositivos; este componente también es el encargado de manejar la comunicación para el intercambio de datos, y para esto implementa diferentes interfaces que le permiten

usar distintos medios de transporte y protocolos. El tercero es el encargado de administrar cada colección de manera local en cada dispositivo.

El marco incluye un conjunto de clases e interfaces que forman la implementación de las tiendas o almacenes de sincronización, y otro grupo de clases e interfaces que forman la implementación de los sincronizadores. Los dos grupos junto con el manejador de almacén, puede ser implementado de forma independiente uno de otro, y luego pueden ser incluidas implementaciones alternativas al Framework.

Existen dos principales aportes de este trabajo. El primero y más relevante se describe en la utilización como punto de partida del protocolo MDSS (Mobile Data Synchronization Service)[70] para la implementación del Framework, este protocolo expone un archivo XML para el intercambio de datos y de esta manera garantizar un proceso ligero muy necesario en el mundo móvil. El segundo aporte que se rescata, es el método de manejo de versiones de cada actualización de los objetos y la réplica en cada dispositivo, para garantizar la disponibilidad y la consistencia de los datos que se hace tan necesaria en sistemas distribuidos con un gran número de dispositivos y con volúmenes de información considerables.

Este trabajo considera tener un seguimiento continuo para cada dispositivo, manejando versiones de datos por cada cliente en el servidor de sincronización. Esto en un escenario empresarial con altos volúmenes de datos, podría ocasionar redundancia poco deseada en las representaciones de los datos persistentes.

2.3.4. Un algoritmo de sincronización de bases de datos móviles para la Computación Ubicua

Esta solución de sincronización, describe un algoritmo llamado SAMD (Synchronization Algorithms based on Message Digest) basado en consultas SQL nativas[20]. Los dispositivos móviles descargan repeticiones de la base de datos de un servidor cuando se conectan y luego pueden pasar a procesar varias tareas en modo desconectado, que permite un total grado de movilidad (característica fundamental de la Computación Ubicua).

El esquema de datos manejado en este trabajo contempla cuatro tablas, dos de ellas corresponden a las tablas de datos y las restantes a las tablas de resumen, que guardan los mensajes intercambiados en el proceso de sincronización. La actualización de las tablas de datos se hace solo cuando los campos MVD (valor del mensaje) en las tablas de resumen no coinciden.

El proceso de sincronización y la estructura de datos que lo respalda, resultan muy ventajosos en redes de baja capacidad donde es crítico el intercambio de un volumen considerable de datos, ya que aquí se considera sólo enviar información

que ha cambiado y necesita ser actualizada. Esto reduce considerablemente el tráfico de información, que representa un menor uso de las redes de datos y de energía en el dispositivo, lo que lo convierte en un proceso muy útil de emplear en ambientes empresariales.

Aunque se solucione el problema de dependencia de motores de bases de datos, sigue siendo un sistema acoplado, ya que cambios en las estructuras de datos necesitarían una restructuración en la implementación que se realice con este framework. Tampoco se habla de un manejo de conflictos que resultaría muy difícil porque no se tiene una capa intermedia que se pueda aprovechar para este fin.

2.3.5. Aplicación de control de versiones concurrentes a la sincronización de dispositivos móviles sin servidor

En este trabajo, se plantea una arquitectura sin servidor de sincronización, simplemente se tienen versiones concurrentes de las bases de datos en cada dispositivo con una carga mínima. Para poder realizar la sincronización y que los datos se mantengan actualizados, se agregan campos a cada representación de estos para que sea posible identificar a que objeto hace referencia esa versión y un propio identificador para esta[71].

Se plantea un mecanismo de detección de conflictos, usando campos en las versiones que identifiquen un estado y un identificador de la versión anterior. Con el campo que identifica al objeto y el que referencia la versión anterior, se comparan y si existen dos objetos con esos datos iguales, se produce un conflicto, y el estado pasa a describir la situación de dicho objeto.

El método de identificación de conflictos es muy interesante, al definir estados específicos que facilitan el siguiente paso que es la resolución de los mismos. Igualmente, las decisiones a tomar con la resolución son muy flexibles, ya que se almacenan diferentes versiones de los registros que pueden ser usadas para intervenir en este proceso. En la Figura 7 se muestra un ejemplo de detección y resolución de un conflicto aplicando el control de versiones concurrentes.

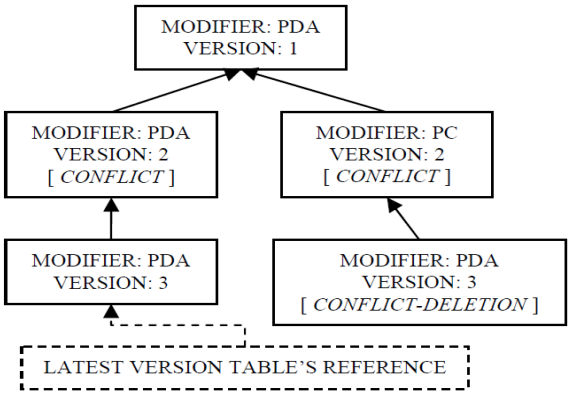


Figura No 7. Ejemplo de resolución de conflicto - Fuente [24]

La ausencia de un servidor de sincronización puede inferir en el tamaño de las bases de datos y la cantidad de información que se pueda manejar en los dispositivos por sus características de recursos limitados, además toda la lógica de sincronización y gestion de conexiones es cargada sobre los clientes y los servidores empresariales.

A continuación se mencionan algunos de los trabajos relacionados con el contexto que se propone en el presente proyecto: En [72] se presenta una estrategia de replicación de datos que permite mantener la consistencia de los datos replicados en sistemas de bases de datos móviles distribuidas a gran escala que implican un gran número de cambios, implementando una arquitectura de tres niveles, un protocolo de propagación de actualizaciones, y un método de la replicación; en [73] se propone un nuevo modelo de sincronización y replicación, para optimizar el tratamiento de conflictos de datos con un mecanismo de pre-procesamiento basado en los datos de usuario en cuestión y las transacciones relacionadas, este es ejecutado antes de la detección y resolución de conflictos en el modelo, ahorrando eficazmente los recursos de almacenamiento en el cliente móvil y el ancho de banda de las comunicaciones móviles, reduce las cargas de trabajo, disminuye la complejidad del algoritmo de detección y resolución de conflictos, y utiliza el protocolo SyncML. Otros trabajos como [74][75][76][77][78][79][80] abarcan conceptos importantes sobre la replicación de datos en temas como la identificación y resolución de conflictos, arquitecturas de referencia, protocolos de sincronización, seguridad, gestión de conexiones, optimización de operaciones, gestión de modelos de datos dinámicos y diferentes recomendaciones sobre las bases de datos móviles.

3. LINEAMIENTOS PARA LA CONSTRUCCIÓN DE APLICACIONES DE SINCRONIZACIÓN DE BASES DE DATOS MÓVILES BASADA EN SERVICIOS WEB

3.1. INTRODUCCIÓN

En este capítulo, se definen un conjunto de lineamientos para la construcción de aplicaciones de sincronización de bases de datos móviles basadas en servicios Web bajo una aproximación multiplataforma, tomando como referencia:

- Conceptos, recomendaciones y especificaciones, sobre sincronización de datos en entornos de escritorio, de algunas de las empresas con mayor reconocimiento a nivel mundial en el sector de las tecnologías de la información y las comunicaciones.
- El análisis realizado en los capítulos anteriores al estado actual del conocimiento referente a la sincronización de bases de datos móviles.
- La realimentación obtenida durante la implementación de la adaptación de un mecanismo de sincronización de bases de datos móviles basado en servicios Web, realizada en este trabajo.

Fijando este conjunto de lineamientos como punto de partida, se busca optimizar y facilitar el proceso de desarrollo de sistemas de sincronización de bases de datos móviles basadas en servicios Web, bajo una aproximación multiplataforma.

Algunas de las referencias más importantes:

a. Replicación avanzada - Oracle:

Una de las tecnologías de Oracle Para la sincronización de datos es la especificación (Oracle Advanced Replication) [81][82], que permite el despliegue de aplicaciones de replicación de bases de datos en entornos con conexiones intermitentes a la red empresarial [83]. Permite replicación bidireccional, múltiples sitios maestros, e incluye además soporte de mecanismos para la identificación y resolución de los conflictos de datos.

b. Microsoft Sync Framework y ADO.NET - Microsoft:

Es una plataforma de sincronización que permite la colaboración en modo desconectado entre aplicaciones, servicios y dispositivos. Los programadores pueden generar ecosistemas de sincronización que integran cualquier tipo de aplicación, cualquier dato, de cualquier almacén utilizando cualquier protocolo través de cualquier red. Sync Framework cuenta con tecnologías y herramientas que habilitan la movilidad, el intercambio y la captura de datos fuera de línea[84] [85].

c. InfoSphere Data Replication - IBM (International Business Machines):

Plataforma que permite integrar información de almacenes de datos heterogéneos en tiempo real, permite alta disponibilidad, migración de bases de datos, consolidación de aplicaciones, almacenamiento dinámico, transferencia de datos de alta velocidad manteniendo la integridad transaccional y la consistencia de los volúmenes de datos empresariales, brinda una detección robusta de conflictos, resolución y notificación entre otras características.

3.2. ESTRUCTURA GENERAL DE LOS LINEAMIENTOS PARA LA SINCRONIZACIÓN DE BASE DE DATOS MÓVILES BASADA EN SERVICIOS WEB BAJO UNA APROXIMACION MULTIPLATAFORMA

En la figura 8, se presenta la estructura general de los lineamientos planteados en este trabajo, divididos en lineamientos para la arquitectura general del sistema y lineamientos para la gestión de conexiones. A su vez los lineamientos para la arquitectura del sistema, se desglosan en: los lineamientos para las aplicaciones empresariales o Back-End, lineamientos para el servidor de sincronización de datos o mediador, lineamientos para las aplicaciones móviles clientes del sistema y los lineamientos para el tratamiento de conflictos.

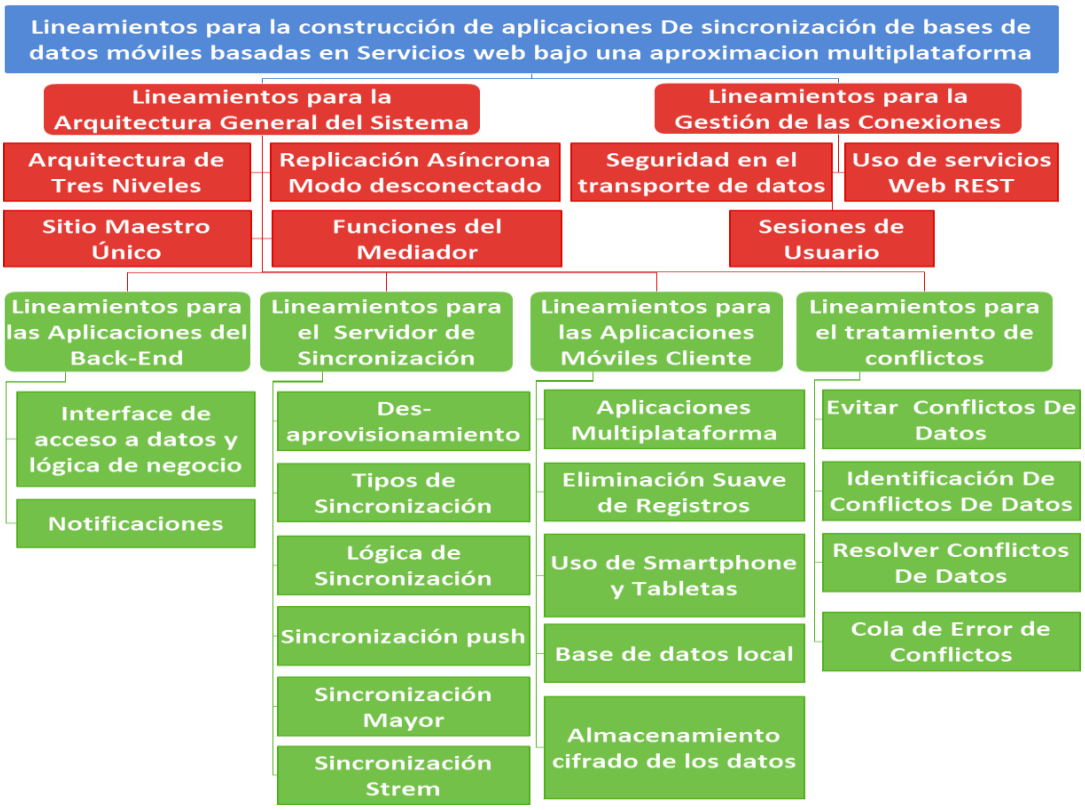


Figura No 8. Estructura general de los lineamientos para sincronización de bases de datos móviles basada en servicios Web.

3.3. LINEAMIENTOS PARA LA ARQUITECTURA GENERAL DEL SISTEMA

No existe consenso en una definición exacta para la arquitectura de software, algunos autores como en [86],señalan que el termino arquitectura se refiere a el diseño de más alto nivel del sistema; en [87] la arquitectura software se define como la estructura o estructuras del sistema que comprenden elementos de software, las propiedades de los elementos y las relaciones entre ellos; en [88] definen la arquitectura como la estructura de un software, el flujo de datos y la definición de las interacciones entre todos los elementos. En resumen, la arquitectura se refiere principalmente al diseño y especificación de la estructura general del sistema. La Figura No 9, describe la estructura de los lineamientos para la arquitectura general del sistema.

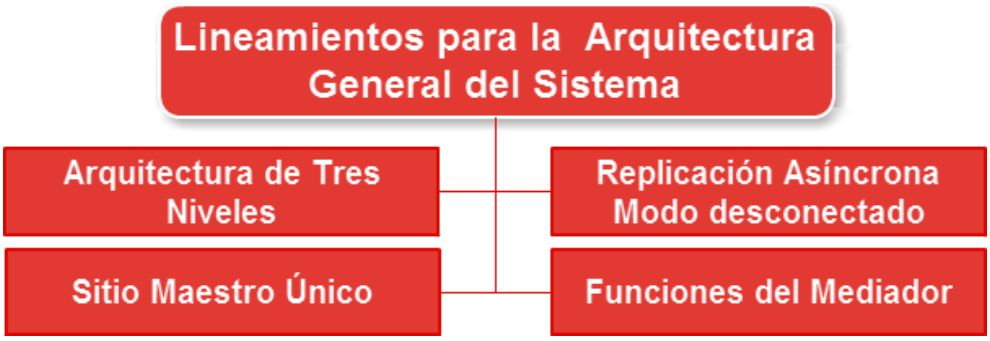


Figura No 9. Lineamientos para la arquitectura del sistema

3.3.1. Arquitectura de Tres Niveles

Se recomienda utilizar para un sistema de sincronización de bases de datos móviles, una arquitectura general dividida en tres capas, como se muestra en la Figura No 10:

Servidor Empresarial o central (Back-End): Se refiere a la aplicación empresarial que contiene la lógica del negocio, la base de datos central o sitio maestro y la persistencia a los datos.

Mediador: En este caso el mediador es el *servidor de sincronización,* que permite gestionar independientemente la lógica de la sincronización de los datos, entre los diferentes clientes móviles y el Servidor empresarial.

Cliente: Los clientes, son dispositivos móviles que acceden a los datos y lógica del negocio por medio del servidor de sincronización.

Al usar esta arquitectura se consigue separar la lógica de los clientes móviles, la lógica de la sincronización de datos, y la lógica de negocio de las aplicaciones

empresariales; con la ventaja de permitir un desarrollo en tres niveles. En caso de requerir un cambio, solo es necesario abordar el nivel requerido sin tener que realizar modificaciones de fondo en los otros niveles, igualmente, permite distribuir el trabajo en el desarrollo de las aplicaciones, con equipos especializados que se dediquen y especialicen al desarrollo de aplicaciones en cada uno de los niveles.

Figura No 10. Arquitectura de tres niveles

3.3.3. Funciones del Mediador

Con el fin de aligerar los clientes y disminuir el volumen de transacciones a la base de datos central, se considera una buena práctica, implementar los procesos complejos o lógica pesada en el mediador, además de toda la lógica de sincronización de datos. Algunos procesos relacionados con las funciones de esta capa son:

- Gestión de conexiones con el Back-end.
- Identificación y resolución de conflictos.
- Sesiones de usuario.
- Notificaciones a los clientes.
- Gestión de Dispositivos.

3.3.4. Sitio Maestro Único

En el sitio maestro, se aloja la base de datos principal que es replicada en los clientes; un sitio maestro es aquel que mantiene una copia completa de todos los objetos de datos de un grupo de replicación, mientras que los clientes móviles pueden contener todos o tan solo un subconjunto de las tablas y objetos de datos del sitio maestro[89].

Para la construcción de aplicaciones de sincronización de bases de datos móviles basada en servicios Web, se recomienda usar un solo sitio maestro o base de datos central para el ambiente de sincronización móvil, como se muestra en la figura No. 11, ya que de esta manera se reduce la posibilidad de datos divergentes, debido a que la resolución de conflictos se realiza para un sólo sitio, y además se elimina la posibilidad de conflictos de datos de orden que se pueden presentar en un entorno de replicación con múltiples sitios maestros.

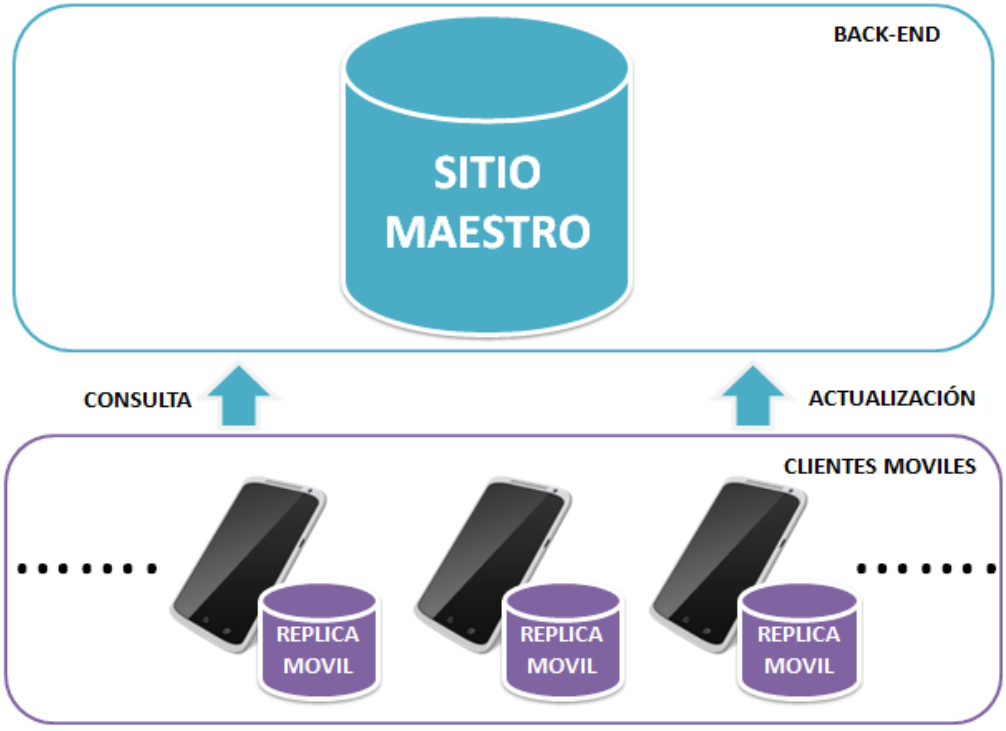


Figura No 11. Sitio maestro único

3.3.5. Trabajo en Modo desconectado - Replicación Asíncrona

En un entorno móvil, es muy común que los clientes necesiten acceder a una aplicación, en un momento o lugar donde no esté disponible una conexión a redes de datos móviles, por este motivo se debe implementar un mecanismo que permita acceder en modo desconectado a la aplicación, gestionar los datos locales y así permitir un modo de trabajo más autónomo y flexible como se muestra en la figura No. 12. El trabajo en modo desconectado, se convierte en un requisito clave para soportar las frecuentes conexiones y desconexiones de los clientes móviles[90].

Para soportar este requerimiento se debe implementar una replicación asíncrona, conocida también como replicación de almacenamiento. Esta consiste en capturar todos los cambios locales, almacenarlos en la réplica local de la base de datos

móvil, y en intervalos regulares de tiempo o a solicitud del cliente, propagar y aplicar estos cambios en el sitio remoto maestro, con un método de sincronización de datos. Al usar esta forma de replicación, es importante tener en cuenta que existe un período de tiempo antes de que todos los sitios consigan una convergencia de datos, a este periodo se le conoce como latencia.

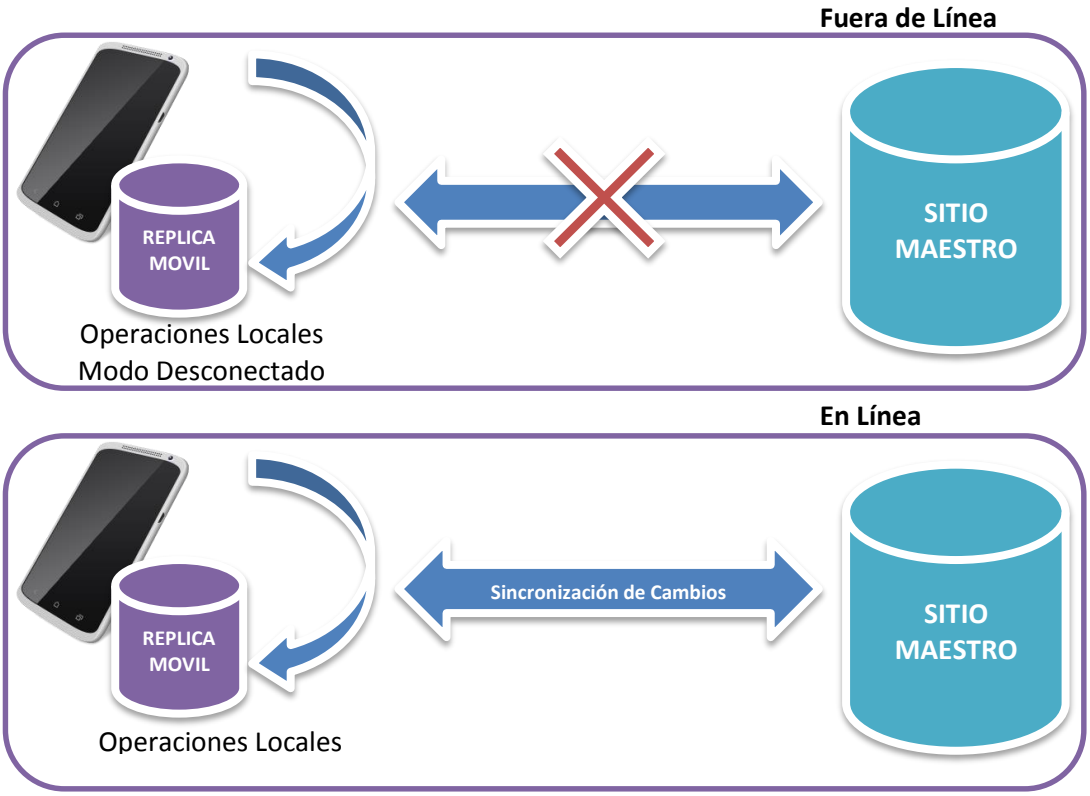


Figura No 12. Trabajo en modo desconectado – replicación asíncrona

3.4. LINEAMIENTOS PARA LA GESTIÓN DE LAS CONEXIONES

La conexión, se refiere a los enlaces o relaciones entre los distintos componentes de la arquitectura general del sistema de sincronización de bases de datos móviles basada en servicios Web. La Figura No. 13, describe la estructura de los lineamientos para la gestión de las conexiones.

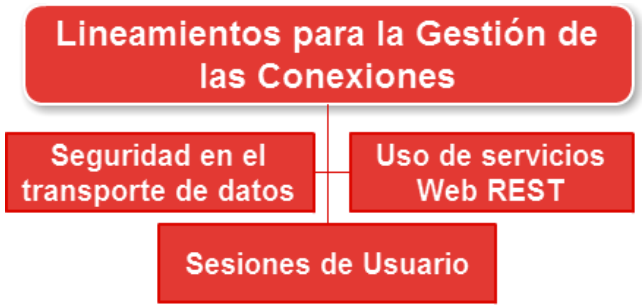


Figura No 13. Lineamientos para la gestión de conexiones

3.4.1. Seguridad en el transporte de datos

La seguridad en los servicios de internet, es un concepto muy amplio que puede abarcar muchos factores y diferentes tratamientos. En este caso, se abordará la seguridad desde el punto de vista de la capacidad para transmitir información, sin que esta pueda ser modificada, accedida o descifrada por alguien diferente a su destinatario. La seguridad siempre debe ser una consideración importante, ya que el rápido desarrollo de las redes en términos de cantidad de conexiones, usuarios, aplicaciones y servicios que se ofrecen, las hace más complejas y por lo tanto, más vulnerables a diversos tipos de ataques[91].

La implementación y el correcto funcionamiento de un mecanismo de seguridad para proteger la información a intercambiar en un entorno de sincronización de datos móviles, es de gran importancia debido a que los datos manipulados son sensibles para la organización y pueden ser confidenciales.

En este sentido, se recomienda utilizar un protocolo seguro para la transferencia de información como HTTPS, que implementa una conexión cifrada de datos basada en los protocolos criptográficos SSL/TLS[92][93], para asegurar las conexiones HTTP a través de Internet, creando un canal bidireccional cifrado para la comunicación entre un servidor y un cliente, que protege la información transmitida del espionaje, la manipulación y falsificación [94][95]. Está diseñado para proporcionar canales orientados hacia la seguridad, usados principalmente en aplicaciones que involucran datos potencialmente sensibles como: en el comercio electrónico, la banca móvil, el correo electrónico y para transacciones en sistemas de información empresarial[96] .

3.4.2. Uso de servicios Web REST

REST, Representational State Transfer, es un estilo arquitectónico para sistemas distribuidos de hipermedia como World Wide Web. REST, es un estilo híbrido derivado de varios de los estilos arquitectónicos basados en la red[97], no es una norma, protocolo o una solución, y establece los principios de los sistemas distribuidos en red[98].

Definido en primer lugar por Roy T. Fielding, en su tesis doctoral, como un conjunto coordinado de restricciones arquitectónicas, que indican las funciones o características de los elementos y las relaciones permitidas entre esos elementos. Cada restricción mencionada trae algunas propiedades arquitectónicas, como la escalabilidad y el bajo acoplamiento, claves para el éxito de la Web. De esta manera REST constituye un buen método para la construcción de servicios Web[99].

Trabajos como [100] [101] [102], concluyen que los servicios Web REST, son más eficientes en el transporte de datos, en términos de la utilización de ancho de banda de red, durante la transmisión de las solicitudes de servicio a través de Internet, la latencia de ida y vuelta durante estas solicitudes, y además brinda un mayor rendimiento y tiempos de respuesta más bajos.

Da acuerdo con estas conclusiones y con el objetivo de optimizar el sistema de sincronización de bases de datos móviles basado en servicios Web, se recomienda usar servicios Web REST y el formato JSON para la representación de los datos, ya que JSON es implementado con mayor frecuencia en los servicios Web REST y un documento con esta representación, es más liviano que un XML para la misma cantidad de información. Cabe resaltar que estas características, tienen más relevancia cuando hablamos del procesamiento de información en dispositivos móviles, que aún tienen limitaciones en el procesamiento de grandes volúmenes de información, a comparación con equipos de escritorio o laptops.

3.4.3. Manejo de Sesión para la Sincronización

Para garantizar la integridad y seguridad de los datos, y la información del sistema, se recomienda implementar un método de autenticación y gestión de las sesiones de los usuarios, que usan los clientes móviles para poder acceder a los servicios de sincronización de datos. Este manejo de sesión, debe realizarse en el Servidor de Sincronización, ya que involucra procedimientos con alta demanda de procesamiento o lógica compleja.

3.5. LINEAMIENTOS PARA EL DESARROLLO DE LAS APLICACIONES DEL BACK-END

Los lineamientos para el desarrollo de las aplicaciones Back-End, constituyen una guía sobre las características generales, que debe tener una aplicación empresarial o Back-End, para integrarse correctamente en un entorno de sincronización de datos móviles basado en servicios web. La Figura No. 14, describe la estructura de los lineamientos para desarrollo de las aplicaciones del Back-End.

Figura No 14. Lineamientos para las aplicaciones del Back-End

3.5.1. Interface de acceso a datos y lógica de negocio

El Back-End, debe exponer una o varias interfaces que permitan el acceso por parte de los clientes a la lógica de negocio y los datos de las aplicaciones empresariales. De acuerdo al enfoque del trabajo y a lo planteado en la sección 3.4.2, estas interfaces deben utilizar servicios Web REST.

De esta manera se facilita y desacopla la conexión remota a los datos, y lógica del negocio, permitiendo que las aplicaciones Back-End, sean desarrolladas en cualquier lenguaje de programación, ejecutadas sobre cualquier plataforma e implementen cualquier motor de bases de datos. Igualmente, proporciona mayor flexibilidad e independencia frente a las aplicaciones cliente y las aplicaciones de la capa media, brinda una mayor escalabilidad e interoperabilidad a las soluciones, y facilita su integración con las modernas Arquitecturas Orientadas a Servicios[103].

3.5.2. Notificaciones

El servidor empresarial, debe exponer un mecanismo de notificación, que permita informar al servidor de sincronización y a los dispositivos clientes cuando se presentan eventos como: problemas en la inserción de datos, en las consultas o procedimientos ejecutados, y problemas de disponibilidad del sistema Back-End. De esta forma, los clientes podrán recibir las novedades y enterarse de los sucesos que ocurran durante una operación de sincronización de datos.

3.6. LINEAMIENTOS PARA EL DESARROLLO DE LAS APLICACIONES MÓVILES CLIENTE

Los lineamientos para el desarrollo de las aplicaciones móviles cliente, constituyen una guía sobre las características generales, que debe tener una aplicación móvil cliente para integrarse correctamente y consumir los servicios de un entorno de sincronización de datos móviles basado en servicios web. La Figura No. 15, describe la estructura de los lineamientos para desarrollo de las aplicaciones móviles cliente.

Figura No 15. Lineamientos para las aplicaciones móviles cliente

3.6.1. Aplicaciones Multiplataforma

Como consecuencia de la aparición y proliferación de los distintos sistemas operativos móviles y diferentes características hardware entre dispositivos, los desarrolladores se enfrentan al problema de la fragmentación, definida como la imposibilidad de poder desarrollar una aplicación y que esta se pueda ejecutar en distintos dispositivos[104]. Esto provoca que los desarrolladores de aplicaciones móviles, se tengan que enfrentar a una multitud de desafíos, que complican en gran medida el proceso de desarrollo; como el hecho de que cada plataforma tiene sus herramientas y API's específicas, o que cada plataforma hace uso de distintos lenguajes de programación. Por esta razón una alternativa para facilitar y optimizar el trabajo, es el desarrollo multiplataforma de aplicaciones móviles[105].

Actualmente existen diferentes Framework's de desarrollo, que permiten programar en un único lenguaje para obtener aplicaciones que pueden ser ejecutadas en dispositivos con sistemas operativos diferentes, en los cuales es una tendencia generalizada, el uso de estándares Web como HTML5, JavaScript y CSS. Algunas de las plataformas más populares son PhoneGap, RhoDes, Appcelerator Titanium [106].

3.6.2. Uso de Smartphone y Tabletas

Al momento de implementar el sistema de sincronización de bases de datos móviles basado en servicios Web, es importante tener en cuenta para la selección de los dispositivos clientes, que los Smartphone o las tabletas permiten más prestaciones y características; brindan un mejor rendimiento, soportan las nuevas plataformas móviles y disponen de periféricos como GPS, cámara fotográfica y de video, Conexiones Wi-Fi y Bluetooth que aportan un valor agregado a la aplicación

de sincronización de datos móviles y a sus posibles aplicaciones en el mercado empresarial.

Dispositivos Endurecidos: A modo de sugerencia, es importante considerar el uso de dispositivos endurecidos, fabricados especialmente para ser utilizados en entonos empresariales, en donde las labores se ejecutan en condiciones adversas por diferentes factores. En estos casos, los dispositivos endurecidos brindan una mayor cobertura de garantía y una mayor durabilidad frente a los dispositivos diseñados para el uso personal. Un dispositivo endurecido, está diseñado para funcionar de forma fiable, en condiciones adversas como la exposición al agua, polvo, temperaturas, altitudes extremas, humedad, rayones, vibraciones y golpes. En muchos casos se hace referencia al estándar militar MIL-STD-810[105], que establece los criterios de durabilidad del material electrónico en el ejército norteamericano. Los dispositivos que cumplen con este estándar, son capaces de trabajar bajo condiciones extremas; las características más comunes que definen a un dispositivo robusto o endurecido son:

- **Resistencia al agua y al polvo:** Esta se mide gracias al estándar IP, este estándar caracteriza los equipos en función de la estanqueidad de los mismos a la entrada de agua y/o polvo en base a dos dígitos (IPXX). El primero hace referencia al nivel de protección frente al polvo y el segundo frente al agua. Así un equipo IP67 presenta máxima protección para polvo y agua, es decir no debe permitir entrada de polvo bajo ninguna condición y debe soportar la inmersión en agua durante 30 minutos a 1 metro de profundidad. Como regla general a mayor IP mayor grado de protección frente a polvo y agua tendrá el dispositivo.

- **Temperaturas extremas y humedad:** Los equipos robustos funcionan en condiciones óptimas en rangos de temperaturas donde otros equipos no lo harían. Los equipos robustos pueden llegar a operar en rangos de temperatura entre -20ºC y 70ºC.
- **Vibración y golpes:** Los equipos robustos están preparados para soportar repetidas caídas de hasta 3 metros de altura sobre suelo de hormigón. Dentro de las características está también la de aguantar vibraciones derivadas de su uso en lugares expuestos a movimientos repetitivos.
- **Pantalla:** Si su propósito es usar el dispositivo en exteriores a plena luz del día, debe tener en cuenta que la pantalla sea de alto contraste, con ello podrá utilizar su dispositivo en las condiciones de luminosidad exterior más extremas.
- **Las baterías:** Son un elemento importante a tener en cuenta, de forma general estos dispositivos poseen baterías de alta duración. En algunos

equipos se presentan dos baterías que puede ser desconectadas para su recarga en caliente (con el equipo encendido) [108].

Estas entre otras características contribuyen a incrementar la vida útil de los dispositivos dentro de este tipo de entornos empresariales.

3.6.3. Base de datos local

El móvil, debe contar con una réplica de la base de datos central, que será usada en el proceso de sincronización. Esta base de datos, permite almacenar los cambios locales y sincronizarlos posteriormente con el servidor empresarial; de esta manera se soporta el trabajo en modo desconectado, que requieren los clientes móviles. Es importante considerar el desarrollo multiplataforma descrito en la sección 3.6.1, debido a que contribuye a mitigar los problemas de compatibilidad, que se puedan presentar entre los diferentes motores de bases de datos, ya que las bases de datos del cliente pueden ser de diferentes fabricantes, usar diferentes plataformas e incluso tener versiones diferentes para un mismo motor.

3.6.4. Almacenamiento cifrado de los datos

Una de las principales preocupaciones de las organizaciones, en este tipo de aplicaciones, está relacionada con la seguridad de la información; en este caso es importante considerar la protección de los datos que se alojen en los clientes móviles, de esta manera, si el dispositivo se pierde o es hurtado, se protegerá la información confidencial de la organización almacenada en él.

Una de las soluciones recomendadas, consiste en implementar un método de cifrado para almacenar los datos en la réplica local del dispositivo móvil. En este sentido, la criptografía se encarga del estudio de los algoritmos, protocolos y sistemas que se utilizan para brindar seguridad a las comunicaciones, a la información y a las entidades que se comunican[109], permite diseñar funciones capaces de transformar mensajes legibles a mensajes cifrados, de tal manera que esta transformación (cifrar) y su transformación inversa (descifrar), sólo pueden ser factibles con el conocimiento de una o más llaves.

Existen dos grandes grupos en la criptografía, una de llave secreta o simétrica y la criptografía de llave pública o asimétrica[110], para los cuales existen diversos algoritmos de cifrado como el DES, Triple DES, RC5, IDEA, AES para el cifrado simétrico y Diffie-Hellman para el cifrado asimétrico; algunos de estos algoritmos permiten grandes niveles de seguridad gracias a su complejidad, ya que para poder descifrar la información en un ataque, es necesario disponer de una gran cantidad de tiempo y recursos[93].

En el caso de la protección de las bases de datos móviles, se pueden implementar mecanismos que permitan almacenar solo información cifrada en la base de datos; de manera que esta solo pueda ser consultada y modificada por un usuario que disponga de la llave, si un intruso logra acceder por algún medio a la base de datos, no podrá comprender la información almacenada y le tomara una gran cantidad de tiempo descifrarla, tiempo para el cual, la información habrá perdido su valor.

3.6.5. Manejo de sesión en las aplicaciones cliente

Es un hecho, que las aplicaciones deben usar un servicio de autenticación para poder acceder a los servicios de sincronización; pero además de esto, es importante considerar un mecanismo de autenticación para hacer uso de la aplicación cliente tanto en el modo Online como en el modo Offline. Con el objetivo de brindar más seguridad en la aplicación y a los datos almacenados en el dispositivo móvil, se describen a continuación algunas alternativas que los desarrolladores pueden elegir:

3.6.5.1. Usuario local

Este método, consiste en implementar un modo de autenticación local de usuario en el dispositivo móvil. Este mecanismo, debe permitir al usuario acceder a la aplicación cliente tanto en el modo desconectado como en el modo conectado; de manera que se debe implementar por medio de la base de datos local del dispositivo o una alternativa similar.

3.6.5.2. Token de autenticación

Esta alternativa, consiste en implementar un método de autenticación, que permita a los usuarios conectarse al menos una vez al servidor de sincronización, para obtener un Token que le permitirá el acceso a la aplicación durante la jornada de trabajo y durante las posibles desconexiones al sistema u operaciones en modo desconectado. Este mecanismo, permite al usuario acceder a la aplicación cliente tanto en el modo desconectado como en el modo conectado; solo que la primera autenticación de la jornada se debe realizar en línea, de manera que se debe usar un mecanismo de gestión de usuario en el servidor de sincronización o aplicación Back-End.

3.6.5.3. Libre de uso

Con esta alternativa, los usuarios no deberán hacer ningún tipo de autenticación para utilizar la aplicación en modo desconectado o conectado; solo se autenticaran para usar los servicios de sincronización del servidor, es decir que solo se tendrá acceso a la información local.

3.7. LINEAMIENTOS PARA EL SERVIDOR DE SINCRONIZACIÓN

Los lineamientos para el servidor de sincronización, constituyen una guía sobre las características generales, que este debe cumplir para soportar correctamente un entorno de sincronización de datos móviles basado en servicios web. La Figura No. 16 describe la estructura de los lineamientos para el servidor de sincronización.

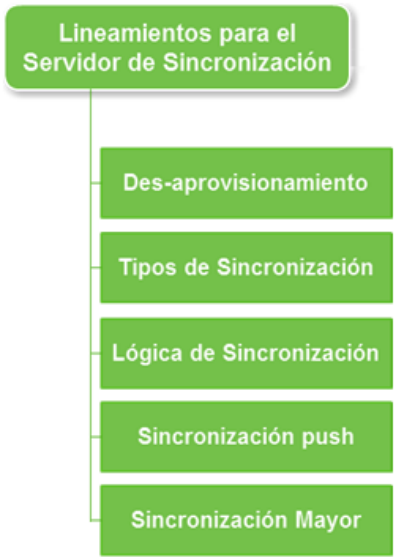


Figura No 16. Lineamientos para el servidor de sincronización.

3.7.1. Des-aprovisionamiento

Con el objetivo de velar por la seguridad de la información, almacenada en los dispositivos móviles cliente, se debe establecer un mecanismo de des-aprovisionamiento de datos y aplicaciones, que puede ser útil en casos de robos o pérdidas de los dispositivos clientes o despido de trabajadores. Para este propósito, es importante contar con una plataforma de gestión de usuarios y dispositivos, que permita eliminar los datos y aplicaciones de los dispositivos clientes que presenten alguno de los inconvenientes descritos.

3.7.2. Tipos de sincronización

Para evitar el consumo excesivo de recursos del dispositivo y de la red de comunicaciones, se considera una buena práctica definir diferentes métodos de sincronización, que no involucren una transmisión total de los datos. Algunos métodos son:

- *Sincronización parcial:* Consiste en intercambiar en una operación de sincronización, únicamente los registros que han sufrido cambios en el

cliente móvil y de estos registros enviar solo las columnas que han sido modificadas localmente. A continuación una tabla que ilustra este proceso:

TIEMPO	ACCION	SITIO MAESTRO	CLIENTE MOVIL
1	Convergencia de datos, tres registros en la tabla.	R1={id:1, altura:173, peso:59} R2={id:2, altura:185, peso:80} R3={id:3, altura:165, peso:54}	R1={id:1, altura:173, peso:59} R2={id:2, altura:185, peso:80} R3={id:3, altura:165, peso:54}
2	El cliente móvil ***actualiza*** dos registros.	R1={id:1, altura:173, peso:59} R2={id:2, altura:185, peso:80} R3={id:3, altura:165, peso:54}	R1={id:1, altura:***170***, peso:59} R2={id:2, altura:185, peso:80} R3={id:3, altura:***190***, peso:***94***}
3	El cliente móvil sincroniza sus datos.	Mensaje de actualización: {R1={id:1, altura=***170***}, R3{id:3, altura:***190***, peso:***94*** }} *"solo se envían los registros y atributos que cambiaron"*	
4	Sincronización exitosa, convergencia de datos.	R1={id:1, altura:170, peso:59} R2={id:2, altura:185, peso:80} R3={id:3, altura:190, peso:94}	R1={id:1, altura:170, peso:59} R2={id:2, altura:185, peso:80} R3={id:3, altura:190, peso:94}

Tabla 1. Sincronización parcial.

- *Sincronizaciones parciales o en una sola dirección:* Se envían los datos solo desde el Servidor al móvil o viceversa. Por ejemplo, cuando se enciende un dispositivo, este mecanismo solo debe enviar la información actualizada desde el servidor central hasta el cliente móvil, para que actualice todos sus datos; puede ser útil hacerlo al inicio de la jornada laboral.

3.7.3. Lógica de sincronización (API)

El servidor de sincronización, debe exponer una API con los métodos predefinidos para facilitar la implementación de los sistemas de sincronización móvil. Estos métodos deben ejecutar las operaciones CRUD de datos, facilitar las conexiones, manejar los conflictos de datos, gestionar la recuperación de datos, la extracción de la carga útil y la representación de los mismos para el almacenamiento en los dispositivos. Para esto, el servidor de sincronización, debe tener un conjunto amplio de bibliotecas y herramientas que permitan la implementación y extensión de estas operaciones.

3.7.4. Sincronización push

La Sincronización push, es un tipo de sincronización de datos que envía los cambios ejecutados en el sitio maestro a los cliente móviles. El servidor de sincronización, envía estas modificaciones a los usuarios que necesitan datos actualizados, y de esta manera se evita la obsolescencia en la versión de los datos almacenados y la divergencia de datos en los dispositivos móviles más críticos del sistema. Además, gracias a este mecanismo se puede enviar información respecto a cambios en los modelos de datos, o notificaciones a los clientes móviles.

Este mecanismo, remplaza métodos como el de consultas frecuentes del dispositivo cliente al sistema central (Encuesta), para mantener una versión de datos actualizada. Éste, cumple la misma función, pero incrementa el consumo de energía en el dispositivo, aumenta la carga de la Red y produce gastos innecesarios en servicios de telecomunicaciones.

3.7.5. Sincronización mayor

En casos en que el entorno de sincronización y los modelos de negocio presenten grandes cantidades de datos (más de 10MB), tales como catálogos de productos o listas de cuentas de los clientes; se debe implementar un método de Sincronización bulk o “Mayor”, que optimice la comunicación, las conexiones y la forma de representar los datos, con el objetivo de reducir el volumen de la información a intercambiar y los tiempos de respuesta en las operaciones de sincronización. Por ejemplo, cuando se requiere enviar a todos los clientes móviles, la nueva versión de un catálogo de productos con una gran cantidad de información; en lugar de usar una transmisión de texto, es preferible crear una base de datos móvil con la información, comprimirla y enviarla a todos los clientes para optimizar el proceso de transmisión.

3.8. LINEAMIENTOS PARA EL TRATAMIENTO DE CONFLICTOS DE DATOS

En un entorno de sincronización de datos móviles basado en servicios Web, en donde se permite la manipulación de datos en modo desconectado y las actualizaciones bidireccionales por parte de los clientes móviles, con cambios almacenados localmente en el dispositivo, es inevitable, que se presenten inconsistencias como los conflictos de datos al momento de una sincronización. Para dar un tratamiento adecuado a estos conflictos, se deben implementar mecanismos para evitar, identificar, resolver y registrar los conflictos de datos posibles en un entorno de replicación. La lógica para el tratamiento de conflictos, debe ser implementada en el servidor de sincronización, ya que es el encargado de procesar las operaciones de sincronización. En la Figura 17, se presenta la estructura de los lineamientos para el tratamiento de conflictos de datos.

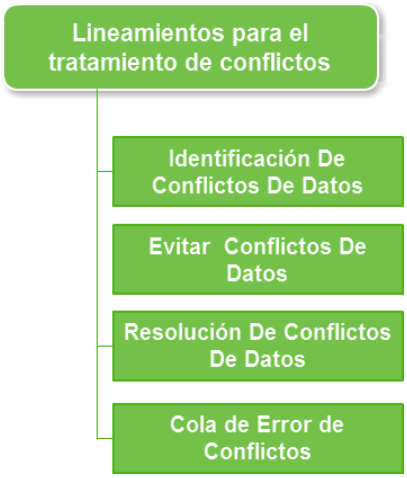


Figura No 17. Lineamientos para el tratamiento de conflictos

3.8.1. Identificación de conflictos de datos replicados

En un entorno de replicación de datos, es muy posible que se presenten conflictos de datos entre las diferentes versiones que puedan tener los clientes móviles de cada registro. Estos conflictos de datos, pueden ocasionar perdida de información sensible para la organización y divergencia entre las versiones de los datos de cada cliente. Con el objetivo de resolver estos problemas, una solución de sincronización de bases de datos móviles, debe incluir un método, que permita identificar los diferentes tipos de conflictos de datos que se puedan presentar.

A partir de las características del entorno de sincronización específico, se deben identificar qué tipos de conflictos se pueden presentar. A continuación, los mecanismos para identificar los tipos de conflictos más comunes en un sistema de sincronización móvil [43]:

3.8.1.1. Identificación de conflictos de actualización

Se presentan cuando en dos o más réplicas de la base de datos central, se modifica el mismo registro aproximadamente al mismo tiempo, o sin tener una versión convergente de los datos. De esta manera al intentar sincronizar los datos, se presenta un conflicto de actualización, debido a que la versión del registro del sitio maestro, cambió desde la última vez que uno de los clientes móviles la consultó [42]. La siguiente tabla describe el proceso de identificación de un conflicto de actualización.

TIEMPO	ACCIÓN	SITIO MAESTRO	CLIENTE A	CLIENTE B
1	Convergencia de datos, en todos los sitios el valor de x=9	2	2	2
2	Cliente A actualiza a x=5	2	5	2
3	Cliente B actualiza a x=3	2	5	3
4	Cliente A sincroniza exitosamente los datos con el Sitio Maestro.	5	5	3
5	Cliente B envía una solicitud de sincronización de datos.	5	5	3
6	Conflicto de actualización ya que el valor de x en el sitio maestro cambio desde la última vez que el Cliente B lo leyó.	Conflicto de Actualización en x	5	3

Tabla 2. Identificación de un conflicto de actualización.

3.8.1.2. Identificación de conflictos de unicidad

Se presentan cuando una inserción viola la restricción de unicidad de una fila, ya que por ejemplo, se pueden introducir dos registros con la misma clave primaria en distintos sitios y al mismo tiempo. La siguiente tabla describe el proceso de identificación de un conflicto de unicidad.

TIEMPO	ACCIÓN	SITIO MAESTRO	CLIENTE A	CLIENTE B
1	Cliente A crea un registro R1 con clave primaria PK=12		R1={PK=12…}	
2	Cliente B crea un registro R2 con clave primaria PK=12		R1={PK=12…}	R2={PK=12…}
3	Cliente A Sincroniza exitosamente sus datos con el Sitio Maestro.	R1={PK=12 …}	R1={PK=12…}	R2={PK=12…}

4	Cliente B envía solicitud de sincronización de datos.	R1={PK=12 …}	R1={PK=12…}	R2={PK=12…}
5	Conflicto de unicidad detectado ya en el sitio maestro ya existe un registro R1 con clave primaria igual a 12	Conflicto de Unicidad	R1={PK=12…}	R1={PK=12…}

Tabla 3. Identificación de un conflicto de unicidad.

3.8.1.3. Identificación de conflictos de eliminación

Se puede presentar cuando un registro es borrado en un sitio, se completa la operación exitosamente en una sincronización, y luego en otro sitio es borrado o actualizado el mismo registro. De esta manera, cuando el último cliente realiza la sincronización, el registro ya no existe para ser borrado o actualizado, porque otro cliente lo eliminó antes. La siguiente tabla describe el proceso de identificación de un conflicto de eliminación.

TIEMPO	ACCIÓN	SITIO MAESTRO	CLIENTE A	CLIENTE B
1	Convergencia de datos entre todos los sitios del entorno de sincronización.	R1{PK=12…}	R1{PK=12...}	R1{PK=12…}
2	Cliente A elimina el registro R1 con clave primaria PK=12	R1{PK=12…}		R1{PK=12…}
3	Cliente B actualiza el registro R1 con clave primaria PK=12	R1{PK=12…}		R1*{PK=12…}
4	Cliente A Sincroniza exitosamente sus datos con el Sitio Maestro.			R1*{PK=12…}
5	Cliente B envía solicitud de sincronización de datos.			R1*{PK=12…}
6	Conflicto de eliminación detectado ya en el sitio maestro no existe un registro R1 con clave primaria igual a 12	Conflicto de Unicidad		R1*{PK=12…}

Tabla 4. Identificación de un conflicto de eliminación.

3.8.2. Evitar conflictos de datos replicados

La mejor práctica en un entorno de sincronización es evitar los conflictos de datos que se puedan presentar, a continuación se definen algunas estrategias para evitar determinados tipos de conflictos en un entorno de sincronización móvil de datos, ilustradas en la Figura No. 18.

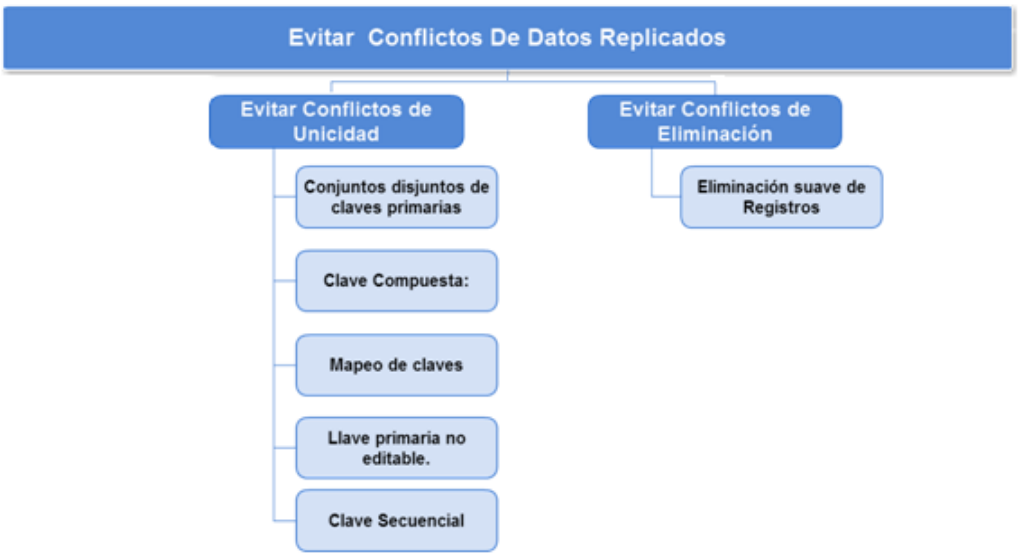


Figura No 18. Métodos para evitar conflictos de datos replicados

3.8.2.1. Evitar conflictos de unicidad

Existen diferentes alternativas para evitar conflictos de unicidad, en entornos de sincronización de datos móviles basada en servicios Web. Algunas de estas son:

a. Conjuntos disjuntos de claves primarias:

Generar conjuntos disjuntos de números secuenciales, en cada cliente móvil para usarlos como clave primaria. Con este método, se inhibe la posibilidad de un conflicto de unicidad durante una operación crear. Sin embargo, es importante tener en cuenta que la implementación y administración de este método es compleja si aumenta el número de clientes móviles.

b. Clave compuesta:

Agregar un prefijo que identifique al cliente móvil, como parte de una clave compuesta. Mediante este mecanismo, se inhibe la posibilidad de un conflicto de unicidad durante una operación crear, ya que cada clave primaria será única en cada dispositivo móvil y en el entorno de sincronización de datos. Una consideración importante que se debe tener al implementar este método, es el tipo de dato de la clave primaria, ya que si este es tipo numérico no se pueden anexar caracteres a la clave primaria compuesta.

c. Mapeo de claves

El mapeo de claves primarias para cada registro en el servidor de sincronización, consiste en identificar cada una de las claves primarias, con una clave global y su respectiva clave equivalente local, en el dispositivo móvil. Este método,

incrementa complejidad a la implementación del servidor de sincronización de datos, ya que se debe proveer el mecanismo para mapear y almacenar las claves primarias.

d. Llave primaria no editable (Actualización).

Para evitar conflictos de unicidad en una actualización, es recomendable que la clave primaría de los registros a sincronizar no sea editable. La razón más importante para implementar este mecanismo, esta en que la clave primaria es la identificación global del registro y una modificación a ésta no solo generaría un posible conflicto de unicidad, sino que puede resultar en información cruzada entre registros e inconsistencia en la base de datos.

e. Clave secuencial:

Este método, consiste en implementar en el servidor Back-End y específicamente en la base de datos maestra, una clave primaria auto-incremental o secuencial para cada registro. De esta manera, no habrá posibilidad de que se presenten conflictos de datos en los dispositivos móviles. Es importante tener en cuenta, que desde el dispositivo móvil no se tendrá acceso a la clave primaria de cada registro, y no se podrá definir su valor en una operación crear, actualizar o eliminar; por lo que se debe proveer un método de identificación aleatorio, para cada registro proveniente de los clientes.

3.8.2.2. Evitar conflictos de Eliminación – Eliminacion suave.

Es recomendable evitar al máximo los conflictos de eliminación, ya que una vez se presenta este tipo de conflicto, es complejo implementar un método para resolverlo. Una estrategia útil para evitar estos conflictos, consiste en implementar un mecanismo de borrado suave de los registros; para implementar este método, es necesario borrar el registro solamente en el cliente móvil, y a su vez en el sistema empresarial o Back-End no eliminar filas mediante comandos DELETE directamente; en su lugar, la aplicación debe marcar las filas a eliminar y periódicamente purgar las filas, para eliminar los registros que estén marcados. Gracias a este método también se evitará que se eliminen datos críticos del sistema en el servidor central[89].

3.8.2.3. Evitar conflictos de actualización

En un modelo de propiedad compartida de datos, los conflictos de actualización no pueden evitarse en todos los casos, por esta razón se debe entender exactamente qué tipos de conflictos de replicación son posibles y luego configurar el sistema para resolver estos conflictos cuando se producen.

3.8.3. Métodos para la resolución de conflictos

En el caso que no se pueda evitar y se identifique un conflicto de datos, se debe disponer de mecanismos para resolver dichos conflictos. En la figura 19, se presenta la estructura de los mecanismos para resolver los conflictos de datos.

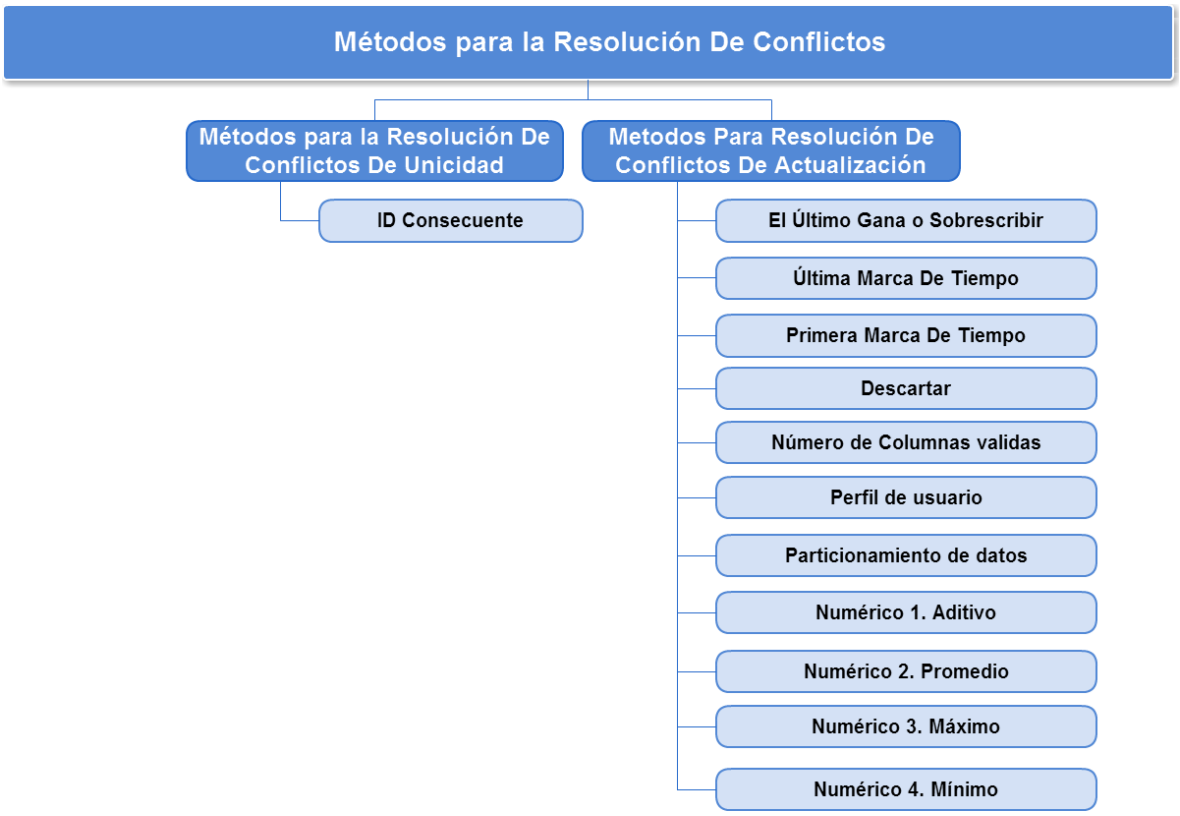


Figura No 19. Métodos para la resolución de conflictos de datos

3.8.3.1. Métodos para la Resolución de Conflictos de Unicidad

a. ID Consecuente

El método ID consecuente, permite resolver un conflicto de unicidad. Cuando un conflicto de unicidad es detectado, se consulta cual es el último ID registrado para esa tabla y se asigna al nuevo registro el ID Consecuente.

Entornos de Destino: Este método, es útil cuando la aplicación de los clientes móviles permite a los usuarios ingresar el valor de la clave primaria.

Mecanismos de Apoyo: No hay necesidad de mecanismos de apoyo adicionales para el método ID Consecuente, en la resolución de conflictos.

A continuación una tabla que explica el funcionamiento del método:

TIEMPO	ACCIÓN	SITIO MAESTRO	CLIENTE A	CLIENTE B
1	Cliente A crea un registro R1 con clave primaria PK=3		R1={PK=3…}	
2	Cliente B crea un registro R2 con clave primaria PK=12		R1={PK=3…}	R2={PK=3…}
3	Cliente A Sincroniza exitosamente sus datos con el Sitio Maestro.	R1={PK=3…}	R1={PK=3…}	R2={PK=3…}
4	Cliente B envía solicitud de sincronización de datos.	R1={PK=3…}	R1={PK=3…}	R2={PK=3…}
6	Conflicto de unicidad detectado ya en el sitio maestro ya existe un registro R1 con clave primaria igual a 12	Conflicto de Unicidad	R1={PK=3…}	R2={PK=3…}
7	El servidor de sincronización consulta cual es el último ID registrado R1 {PK=3} y asigna al nuevo registro el ID consecuente R2-{PK=4}	R1={PK=3…} R2={PK=4}	R1={PK=3…}	R2={PK=4…}
8	Conflicto de unicidad resuelto por método de ID consecuente.	R1={PK=3…} R2={PK=4}	R1={PK=3…}	R2={PK=4…}

Tabla 5. Método de resolución por ID Consecuente.

3.8.3.2. Métodos para Resolución de Conflictos de Actualización

A continuación se describen los métodos de resolución, para los conflictos de actualización. Algunos de ellos son definidos por Oracle[89] y adaptados para un entorno móvil de sincronización de datos, con las características expuestas anteriormente:

b. El Último Gana o Sobre-Escritura

El método El Ultimo Gana o Sobrescribir, resuelve un conflicto, reemplazando el valor del servidor central con el nuevo valor originado desde el cliente móvil. Lo que se hace al detectar el conflicto, es utilizar el valor del cliente que hizo la sincronización de datos más recientemente y descartar la versión existente en el Sitio maestro.

Entornos de Destino: Está diseñado para ser utilizado por un solo sitio maestro y múltiples clientes móviles. Por ejemplo, si en el entorno de sincronización, se tiene un sitio maestro único utilizado principalmente para las consultas, estadísticas e

informes y la gran mayoría de las actualizaciones son realizadas en los sitios cliente; entonces es conveniente seleccionar el método de sobre-escritura. También, es útil si la principal preocupación es la convergencia de datos y no hay una regla de negocio en particular para la selección de una actualización sobre la otra.

Funcionamiento: Luego de detectar un conflicto de este tipo, el valor de origen del cliente móvil se inserta en el sitio maestro, dando prioridad al cliente móvil que más recientemente ha ejecutado una operación de sincronización. Debido el método para la resolución de conflictos de sobre-escritura, asegura la convergencia de datos para entornos de replicación que tienen un solo sitio maestro con cualquier número de clientes móviles, es ideal para entornos de despliegue masivo. A continuación se muestra una tabla que permite ilustrar mejor su funcionamiento:

TIEMPO	ACCIÓN	SITIO MAESTRO	CLIENTE A	CLIENTE B
1	Convergencia de datos, en todos los sitios el valor de x=9	2	2	2
2	Cliente A actualiza a x=5	2	5	2
3	Cliente B actualiza a x=3	2	5	3
4	Cliente A sincroniza exitosamente los datos con el Sitio Maestro.	5	5	3
5	Cliente B envía una solicitud de sincronización de datos.	5	5	3
6	Conflicto de actualización ya que el valor de x en el sitio maestro cambio desde la última vez que el Cliente B lo leyó.	Conflicto de Actualización en x	5	3
7	Como cliente B envió la solicitud de sincronización luego que el cliente A (Tiempo 5>Tiempo 4) se elige la versión del cliente B.	3	5	3
8	Cliente A sincroniza los datos exitosamente.	3	3	3
9	Convergencia de datos , en todos los sitios el valor de x=3	3	3	3

Tabla 6. Método de resolución por Sobre-escritura.

Mecanismos de Apoyo: No es necesario implementar mecanismos de apoyo adicionales para el método de resolución de conflictos de sobre-escritura.

c. Última Marca de Tiempo

El método de marca de tiempo más reciente, resuelve un conflicto de actualización, basado en la marca de tiempo del registro enviada por los clientes móviles. Esta marca de tiempo, debe estar implementada por medio de una columna con la fecha y hora, correspondiente a la última actualización en el móvil.

Implementación: Para utilizar el método de marca de tiempo, debe designar una columna en la tabla greplicada de tipo DATE. Esta columna será actualizada con el tiempo local del sistema cada vez que en una aplicación cliente se edita, modifica o actualiza cualquier registro.

Funcionamiento: En caso de presentarse el conflicto, se debe identificar el cliente, con el valor de marca de tiempo más reciente en su tabla, entonces el sistema aceptará esta actualización y descartará las otras. Se debe designar un método de respaldo, como el de prioridad del sitio, que se llamará en caso de que los sitios que intentan hacer la actualización, tengan la misma marca de tiempo. A continuación se muestra una tabla que permite ilustrar mejor su funcionamiento:

TIEMPO	ACCIÓN	SITIO MAESTRO	CLIENTE A	CLIENTE B
1	Convergencia de datos, en todos los sitios el valor de x=9	2	2	2
2	Cliente A actualiza a x=5	2	5	2
3	Cliente B actualiza a x=3	2	5	3
4	Cliente B sincroniza exitosamente los datos con el Sitio Maestro.	3	5	3
5	Cliente A envía una solicitud de sincronización de datos.	3	5	3
6	Conflicto de actualización ya que el valor de x en el sitio maestro cambio desde la última vez que el Cliente A lo leyó.	Conflicto de Actualización en x	5	3
7	Como cliente B actualizo x más recientemente (Tiempo 3>Tiempo 2) se elige la versión del cliente B.	3	5	3
8	Cliente A sincroniza los datos exitosamente.	3	3	3
9	Convergencia de datos , en todos los sitios el valor de x=3	3	3	3

Tabla 7. Método de resolución por Última Marca de Tiempo.

Consideraciones: Al utilizar la resolución de marca de tiempo, se debe considerar cuidadosamente, cómo es medido el tiempo por los diferentes dispositivos clientes. Por ejemplo, si un entorno de replicación, cruza zonas horarias, las

aplicaciones que utilizan el sistema deben convertir todas las marcas de tiempo a una zona común, como (GMT).

Además, si dos sitios en un sistema, no tienen sus relojes sincronizados razonablemente bien, las comparaciones de marca de tiempo podrían no ser lo suficientemente precisas para satisfacer las necesidades de aplicación.

Mecanismos De Apoyo: Si existen dos actualizaciones con la misma marca de tiempo, se puede resolver el conflicto con un método de respaldo, como prioridad de usuario.

d. Primera Marca de Tiempo

Es similar al método de marca de tiempo más reciente. Este método que utiliza la marca de tiempo más antigua, resuelve un conflicto basado en la marca de tiempo de la actualización. Esta marca de tiempo, debe estar implementada por medio de una columna con la fecha y hora de cuando se produjo la última actualización.

Implementación: Para utilizar el método de marca de tiempo, se debe designar una columna en la tabla replicada de tipo DATE. Esta columna será actualizada con el tiempo local del sistema, cada vez que en una aplicación cliente se edita, modifica o actualiza cualquier registro.

Funcionamiento: En caso de presentarse el conflicto, se debe identificar la versión del dato proveniente del cliente, con el valor de marca de tiempo más antigua en su tabla; entonces el sistema aceptara esta actualización y descartara las otras. Se debe designar un método de respaldo, como el de prioridad del sitio, que se llamará, en caso de que los dos sitios que intentan hacer la actualización tengan la misma marca de tiempo.

Consideraciones: Al utilizar la resolución de marca de tiempo, se debe considerar cuidadosamente cómo es medido el tiempo por los diferentes dispositivos clientes. Por ejemplo, si un entorno de replicación cruza zonas horarias, las aplicaciones que utilizan el sistema deben convertir todas las marcas de tiempo a una zona común, como (GMT).

Además, si dos sitios en un sistema no tienen sus relojes sincronizados razonablemente bien, las comparaciones de la marca de tiempo, podrían no ser lo suficientemente precisas para satisfacer las necesidades de aplicación.

Mecanismos de Apoyo: Si existen dos actualizaciones con la misma marca de tiempo, se puede resolver el conflicto con un método de respaldo como prioridad de usuario.

e. Descartar

El método de descarte, hace caso omiso de los valores del sitio de origen, este ignora el nuevo valor generado en el cliente móvil y conserva el valor en el sitio maestro. Este método está diseñado para ser utilizado por un solo sitio maestro y múltiples clientes móviles.

El método de descarte es útil si:

- La principal preocupación es la convergencia de datos.
- No existe una regla negocio en particular para la selección de una actualización sobre otra.

Entornos de Destino: El método de resolución de conflictos por descarte, es adecuado para un modelo de despliegue a gran escala, que tiene un solo sitio maestro con un gran número de clientes móviles.

Funcionamiento: Si se detecta un conflicto de actualización, el valor de origen del cliente móvil que ejecuta la última sincronización, se pasa por alto, dando prioridad a los clientes móviles que se sincronizaron en primer lugar.

Mecanismos de Apoyo: No es necesario implementar mecanismos de apoyo adicionales, para el método de resolución de conflictos por descarte.

f. Número de Columnas validas

El método Número de Columnas validas, valora el número de columnas validas originadas en una actualización, desde el cliente móvil. Este método ignora la actualización del cliente móvil que tenga un menor número de columnas validas, está diseñado para ser utilizado por un solo sitio maestro y múltiples clientes móviles.

Entornos de Destino: El método de resolución de conflictos por número de columnas validas, es el más adecuado, para un modelo de despliegue en el que se desea tener las versiones de datos más completas generadas desde los diferentes clientes móviles. Es recomendado para un entorno de sincronización de datos, enfocado a la recolección de datos en campo, ya que da prioridad a las versiones de un registro más completas.

Funcionamiento: Si se detecta un conflicto de actualización, se compara el número de columnas validas de cada una de las actualizaciones de los clientes y el sitio maestro. De esta manera se pasa por alto los registros más incompletos y se registra en el sitio maestro la versión con más información valida.

Mecanismos de Apoyo: Si existen dos actualizaciones con el mismo número de columnas validas, se puede resolver el conflicto con un método de respaldo, como la marca de tiempo o por prioridad del perfil de usuario.

g. Perfil de usuario

El método perfil de usuario, resuelve un conflicto de actualización de acuerdo a la prioridad o rango de un cliente móvil, en un determinado entorno de replicación. Se debe definir una columna de prioridad, que permita asignar un rango de a cada uno de los usuarios del sistema y facilite al servidor de sincronización, resolver los conflictos de actualización, con base a la prioridad definida. La prioridad de usuario, puede ser útil si un cliente móvil o un grupo de ellos, son considerados como los más opcionados a tener la información más precisa.

Entornos de Destino: El método de resolución de conflictos basado en la prioridad de usuario, puede implementarse en un entorno donde exista flujo de trabajo entre diferentes actores; se puede garantizar la convergencia de datos en un entorno de sincronización con un solo sitio maestro y varios clientes móviles.

Implementación: Una columna debe ser designada para almacenar la información de la prioridad del cliente móvil, y se deben definir las prioridades para cada uno de los usuarios del entorno de sincronización.

Funcionamiento: Si se detecta un conflicto de actualización, se compara la prioridad del usuario de cada una de las actualizaciones provenientes de los clientes y se pasa por alto el registro generado por el usuario con una menor prioridad y se registra en el sitio maestro la versión generada por el usuario que presenta una mayor prioridad o rango.

Mecanismos de Apoyo: Si existen clientes móviles con la misma prioridad de usuario, se puede resolver el conflicto con un método de respaldo como la marca de tiempo.

h. Particionamiento de datos

Este método, evalúa el cliente móvil que realizo la actualización, dando prioridad al usuario que creó el registro en una primera instancia, sobre los usuarios restantes.

Entornos de Destino: Este método de resolución de conflictos, es recomendado para un entorno de sincronización, enfocado a la recolección de datos en campo, que da prioridad al usuario que creo cada uno de los registros del sistema y que pueda ser la persona que más información tiene sobre el registro.

Funcionamiento: Si se detecta un conflicto de actualización, se comparan los usuarios que generan cada una de las actualizaciones de los clientes, de esta manera, se pasa por alto el registro que haya sido modificado por un usuario diferente al que creó el registro y se guarda en el sitio maestro la versión del registro modificada por el mismo usuario que creó el campo.

Mecanismos de Apoyo: Si existen dos actualizaciones con usuarios diferentes al que creó el registro inicialmente, se puede resolver el conflicto con un método de respaldo como la marca de tiempo.

i. Numérico 1. Aditivo

El método aditivo, trabaja con tablas en las que desde los clientes móviles solo es actualizable un dato, correspondiente a una columna numérica única para cada registro. Si surge un conflicto, en lugar de elegir un valor sobre otro, entonces la diferencia de los dos valores se añade al valor actual.

Entornos de Destino: El método de resolución de conflictos aditivo, está diseñado para conservar los datos en lugar de elegir los datos más adecuados. Este método, podría ser útil en un entorno financiero, donde los depósitos y retiros ocurren con tanta frecuencia que pueden surgir conflictos con un balance, es importante para conservar los datos en lugar de elegir un valor sobre otro.

El método de resolución de conflictos aditivo, ofrece convergencia para cualquier sitio maestro y múltiples clientes móviles. Puede ser utilizado en entornos, donde los clientes móviles solo pueden modificar una columna numérica de todo el registro, como por ejemplo en una aplicación de manejo de inventario, donde el único campo susceptible a actualización por el usuario, es un valor numérico que corresponde a la cantidad disponible de determinado elemento en almacén.

Funcionamiento: El método aditivo agrega la diferencia entre los valores antiguos y nuevos en los clientes móviles, para registrar el valor actual en el sitio maestro, de acuerdo con la siguiente fórmula:

Valor actual = valor actual + (nuevo valor - valor viejo)

Mecanismos de Apoyo: No es necesario implementar mecanismos de apoyo adicionales para el método de resolución de conflictos aditivo.

j. Numérico 2. Promedio

Al igual que el método aditivo, el método de promedio trabaja con tablas, en las que desde los clientes móviles solo es editable un único dato o columna numérica, para cada registro de la tabla. En lugar de añadir la diferencia con el valor actual,

el método de promedio, resuelve el conflicto mediante el cálculo de la media de los valores actuales y el nuevo valor.

Entornos de Destino: El método de la media, puede ser implementado en un entorno de despliegue a gran escala, con un solo sitio maestro y cualquier número de clientes móviles. El método de la media podría ser útil para aplicaciones científicas, que prefieren la media de dos valores antes que elegir un valor sobre otro (por ejemplo, para calcular la temperatura media o peso)

Funcionamiento: El método de resolución de conflictos por promedio, promedia el valor de la columna en el sitio maestro con el valor que llega desde el cliente móvil.

Valor actual = (valor actual + nuevo valor) / 2

Mecanismos de Apoyo: No es necesario implementar mecanismos de apoyo adicionales, para el método de resolución de conflictos por promedio.

k. Numérico 3. Máximo

Cuando la replicación, detecta un conflicto de actualización con un registro y llama el método de resolución de conflicto por valor máximo, se compara el nuevo valor enviado desde el cliente móvil, con el valor actual del sitio maestro de la única columna editable en el registro.

Entornos de Destino: Este método de resolución de conflictos por valor máximo, es adecuado si la columna única numérica del sitio maestro que se utiliza para resolver el conflicto, siempre está aumentando en todos los clientes móviles, entonces este método garantiza la convergencia de datos con un sitio maestro y múltiples clientes móviles.

Funcionamiento: Se elige el valor de la columna mayor entre las dos actualizaciones, si el nuevo valor de la columna designada, es mayor que el valor actual, el valor de la columna del cliente móvil se aplica en el sitio maestro. Si el nuevo valor de la columna designada es menor que el valor actual, entonces el conflicto se resuelve dejando los valores anteriores del sitio maestro.

Si los dos valores para la columna designada son los mismos (por ejemplo, si la columna designada no causa el conflicto), entonces el conflicto no se resuelve, y el valor de la columna numérica permanece sin cambios.

Mecanismos de Apoyo: No es necesario implementar mecanismos de apoyo adicionales para el método de resolución de conflictos por máximo valor.

l. Numérico 4. Mínimo

Cuando la replicación detecta un conflicto con un registro y llama el método de resolución de conflicto por valor mínimo, se compara el nuevo valor enviado desde el cliente móvil, con el valor actual del sitio maestro para la única columna editable del registro.

Entornos de Destino: Si ha definido el método de resolución de conflictos mínimo y la columna única numérica del sitio maestro que se utiliza para resolver el conflicto, siempre está disminuyendo en todos los clientes móviles, este método garantiza la convergencia de datos con un sitio maestro y múltiples clientes móviles.

Funcionamiento: Se elige el valor de la columna minima entre las dos actualizaciones, si el nuevo valor de la columna designada, es menor que el valor actual, el valor de la columna del cliente móvil se aplica en el sitio maestro. Si el nuevo valor de la columna designada es mayor que el valor actual, entonces el conflicto se resuelve dejando los valores del sitio maestro.

Si los dos valores para la columna designada son los mismos (por ejemplo, si la columna designada no causa el conflicto), entonces el conflicto no se resuelve, y el valor de la columna numérica permanece sin cambios.

Mecanismos de Apoyo: No es necesario implementar mecanismos de apoyo adicionales, para el método de resolución de conflictos por mínimo valor.

3.8.3.3. Métodos Personalizados

El entorno de sincronización, debe permitir a los usuarios, definir sus propios métodos para la resolución de conflictos de datos, ya que los métodos mencionados anteriormente son genéricos y pueden existir casos en los que para el entorno de sincronización, ninguno de estos métodos sea adecuado y se cuente con una regla de negocio que permita resolver estos, de una mejor manera.

3.8.4. Registro de los conflictos o Cola de Error

Se debe implementar una cola de error, que permita registrar información sobre los conflictos que ocasionaron errores y no se pudieron resolver, de tal manera que un administrador del sistema pueda visualizar los errores y solucionar los problemas manualmente. Se pueden presentar estas situaciones en el caso en que un método de resolución de conflictos falle, o por el contrario en el caso en que no se haya definido ningún método para dar tratamiento al conflicto encontrado.

3.9. ARQUITECTURA GENERAL DE UN SISTEMA DE SINCRONIZACION

A continuación, en la Figura No. 20 se presenta la arquitectura general de un sistema de sincronización de bases de datos móviles orientado a servicios Web bajo una aproximación multiplataforma, basada en los lineamientos planteados a lo largo de este capítulo.

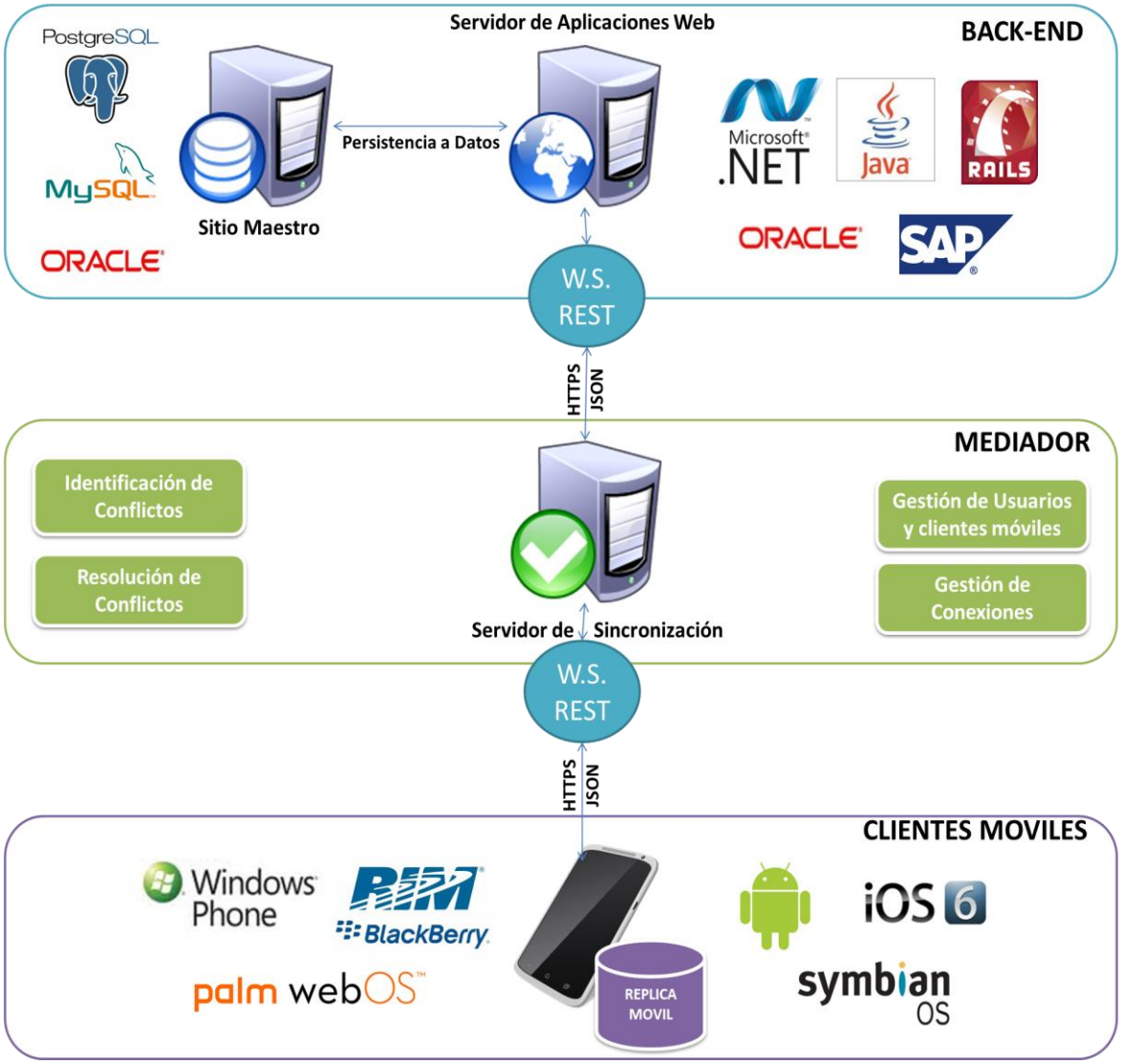


Figura No 20. Arquitectura general de un sistema de sincronización de bases de datos móviles basado en servicios Web

4. ADAPTACIÓN DE UN FRAMEWORK DE SINCRONIZACIÓN DE BASES DE DATOS MOVILES BASADO EN SERVICIOS WEB

El objetivo de este capítulo es describir el proceso de selección y adaptación de un Framework existente para la construcción de aplicaciones de sincronización de base de datos móviles basadas en servicios Web, tomando como referencia los lineamientos definidos en el capítulo anterior. Con la implementación de esta adaptación se pretende brindar las herramientas necesarias para facilitar y optimizar el desarrollo y despliegue de aplicaciones en entornos de sincronización de datos móviles basadas en servicios Web bajo una aproximación multiplataforma.

La Figura No. 21, describe el proceso de la adaptación. En la etapa de selección se muestran los candidatos y sus características más relevantes para elegir el Framework que se adaptará. A continuación se hace un diagnóstico del Framework seleccionado, realizando una confrontación con los lineamientos planteados en el capítulo anterior. En la siguiente etapa, se definen las características a modificar basadas en el diagnóstico y finalmente se ejecuta la adaptación.

Para mayor detalle del proceso de adaptación del framework se puede consultar el anexo A, que corresponde al análisis y diseño del software en forma general.

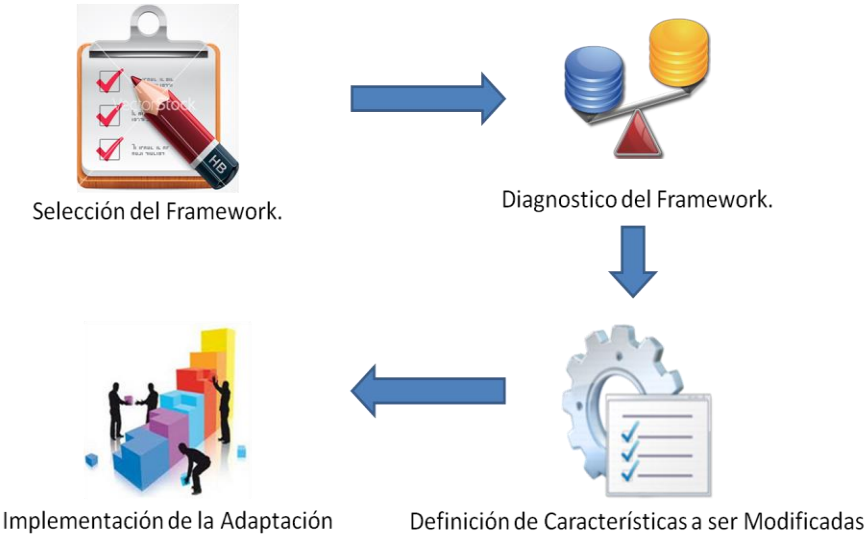


Figura No 21. Proceso de adaptación.

4.1. SELECCIÓN DEL FRAMEWORK

De acuerdo con los lineamientos, una solución de sincronización de bases de datos móviles, debe contemplar una arquitectura dividida en tres capas fundamentales: Back-End, Servidor de Sincronización y aplicaciones cliente.

4.1.1. Servidores Empresariales - BackEnd

Teniendo en cuenta los lineamientos planteados, una solución de sincronización de bases de datos móviles debe ser orientada a servicios Web y debe intercambiar datos utilizando tecnologías como REST o SOAP.

En este orden de ideas, es indiferente el lenguaje de programación y la plataforma de software que se utilice para desarrollar las aplicaciones de esta capa[61]. Igualmente es indiferente el motor o tecnología de base de datos que se utilice; en teoría el requisito único y fundamental que debe cumplir una aplicación de Back-End para integrarse adecuadamente a este tipo de entornos de sincronización, es exponer servicios Web para el acceso a los datos y su lógica de negocio. Adicionalmente es recomendable exponer métodos para enviar notificaciones a los clientes o al servidor de sincronización, sobre las operaciones que se realicen en el acceso a los datos.

4.1.2. Mediador - Servidor de Sincronización

Para la selección del Framework en la capa del servidor de sincronización, se compararon tres proyectos existentes: RhoSync, RhoConnect y Microsoft Sync Framework mencionados en el capítulo 2.

4.1.2.1. Criterios de Selección del Framework de Sincronización

A continuación se presentan las características principales consideradas en la evaluación realizada a los Framework´s:

Software Libre: Este criterio de evaluación se refiere al tipo de licencia bajo la cual se distribuye el Framework de desarrollo, cumple con esta condición el software que tenga algún tipo de licencia de software libre o gratuito.

Código Abierto: Hace referencia a las condiciones de acceso y modificación de los códigos fuentes del programa. Es una característica fundamental para poder realizar la adaptación.

Entorno de desarrollo: Este criterio evalúa si el Framework cuenta con un Entorno Integrado de Desarrollo (IDE), que facilite la creación, compilación y depuración de las aplicaciones de sincronización.

Lenguaje de programación: Se refiere al lenguaje de programación que deben usar los desarrolladores para crear aplicaciones usando el Framework.

Sistemas Operativos compatibles: Se refiere a la compatibilidad del Framework con los diferentes sistemas operativos disponibles en el mercado.

4.1.2.2. Selección del Framework de Sincronización

En la Tabla No. 8, se presenta el resultado de la confrontación entre las características y funcionalidades definidas en los criterios de evaluación y cada uno de los Framework's considerados.

FRAMEWORK / CRITERIO	**RhoSync**	RhoConect	Microsoft Sync Framework
Código Abierto	SI	NO	NO
Software Libre	GNU General Public License	Comercial	Libre / Comercial
Entorno de desarrollo	No Tiene	Rhostudio	VisualStudio
Lenguaje de programación	RUBY	RUBY	C# - VB.NET
Sistemas Operativos compatibles	Linux, Mac, Windows	Linux, Mac, Windows	Windows

Tabla 8. Características generales de los Framework´s de sincronización.

De acuerdo con los requerimientos del trabajo y con el objetivo de realizar la adaptación de un Framework existente para la sincronización de bases de datos móviles basado en servicios Web, se da prioridad al criterio de selección referente al Código Abierto y en segunda estancia a la licencia bajo la cual se distribuye el software. En este orden, el Framework seleccionado es **RhoSync,** ya que es el único con código abierto y además se distribuye con licencia GNU (General Public License), características que facilitan su extensión y adopción.

4.1.3. Aplicaciones Cliente

Al usar una aproximación orientada a servicios Web, en teoría existe independencia de la plataforma móvil en la que se desarrollen las aplicaciones móviles clientes, siempre y cuando éstas sean capaces de consumir los servicios Web expuestos por el servidor de sincronización. Además, los lineamientos recomiendan el desarrollo de aplicaciones móviles multiplataforma, por estas razones se presentan a continuación algunos Framework's que permiten el desarrollo de este tipo de aplicaciones móviles:

Rhodes: Es un Framework para la construcción aplicaciones móviles multiplataforma, éstas aplicaciones están optimizadas para interactuar con aplicaciones empresariales Back-End[111]. También están diseñadas para trabajar con datos localmente, utilizando bases de datos como SQLite o HSQL, que se pueden sincronizar con una base de datos central. Rhodes está disponible para desarrollo de aplicaciones en plataformas como: iPhone, Research in Motion (Blackberry), Windows Mobile y Android[61].

Phone Gap: PhoneGap es un framework de código abierto para el desarrollo de aplicaciones móviles, desarrollado por Nitobi Software bajo la licencia Massachusetts Institute of Technology (MIT). Esto significa que los desarrolladores y las empresas pueden crear aplicaciones gratuitas, comerciales, con código abierto y con otras posibles combinaciones de licencias[111]. Útil para la creación de aplicaciones multiplataforma móviles, utilizando tecnologías web modernas, como HTML5, CSS3 y JavaScript empleando la funcionalidad de los SDK [112] [113].

Appcelerator Titanium: Es una plataforma abierta y extensible para el desarrollo y la creación de aplicaciones nativas en diferentes dispositivos móviles y sistemas operativos, incluyendo iOS, Android, Windows y RIM, así como híbridos y HTML5[114]. Incluye un SDK de código abierto con más de 5.000 API's de dispositivos y sistemas operativos móviles, y permite desarrollar en un potente IDE basado en Eclipse[115]. Titanium, proporciona una API independiente de la plataforma para acceder a los componentes nativos y simplifica el proceso de desarrollo móvil ya que permite a los desarrolladores rápidamente construir, probar, empaquetar y publicar aplicaciones utilizando JavaScript[116].

4.1.3.1. Selección del Framework para el desarrollo de aplicaciones clientes

El criterio más importante al seleccionar el Framework para el desarrollo de las aplicaciones clientes móviles, es la compatibilidad con el Framework escogido en la capa del mediador; en este caso **RhoSync**. Por esta razón se eligió a **RhoDes v.3.3.1**, ya que dispone de integración automática con las aplicaciones desarrolladas en RhoSync, facilitando el desarrollo y despliegue de las aplicaciones. La Tabla No. 9, describe las características y funcionalidades principales de RhoDes:

Característica	**RhoDes**
Licencia	MIT
Código abierto	SI
Entorno de desarrollo	RhoStudio
Lenguaje de programación	HTML, Javascript, CSS, Ruby
Sistemas Operativos compatibles	Linux, Mac, Windows
Sistemas Operativos Móviles Soportados	iPhone, Windows Mobile, BlackBerry, Android
Aplicación	Web/Nativa
MVC	SI
Desarrollo Multiplaforma	SI
API de acceso a datos.	API RHOM (Sqlite/Hsql)
Rhosync Client.	Basado en Asynchttp-API [117]

Tabla 9. Características Generales de Rhodes.

La Figura No. 22, presenta los componentes principales del Framework RhoDes para el desarrollo de aplicaciones móviles multiplataforma.

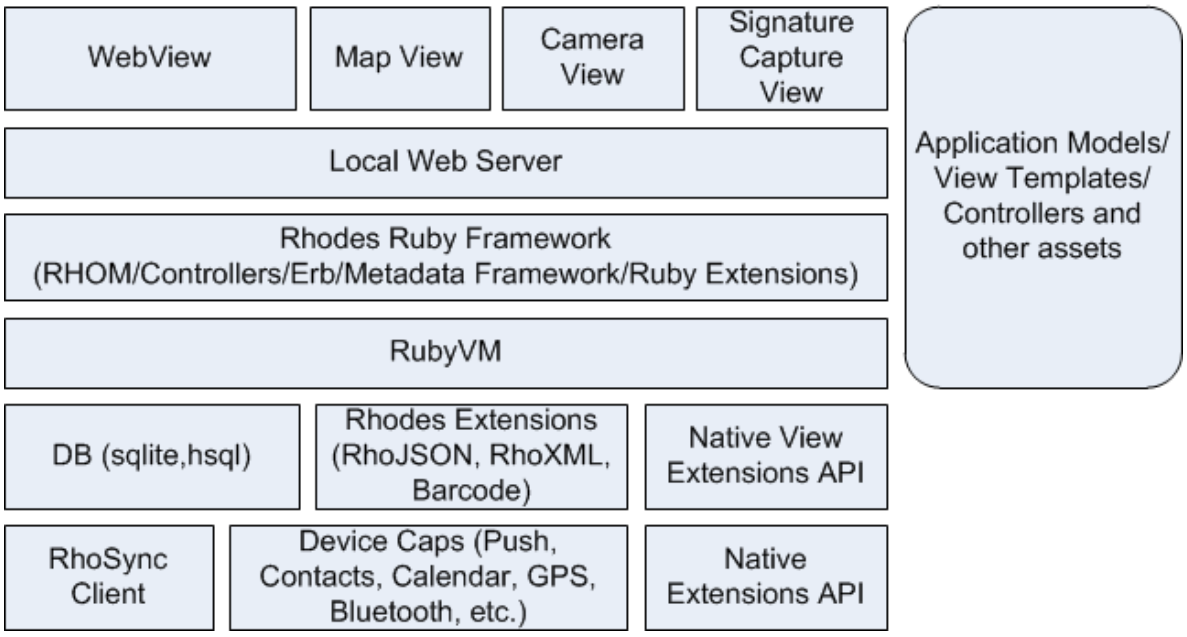


Figura No 22. Componentes Generales de Rhodes [118]

4.1.4. SOLUCIÓN DE SINCRONIZACIÓN SELECCIONADA

De acuerdo con las consideraciones y la evaluación realizada, los Framework´s y tecnologías que serán usadas en este trabajo para el desarrollo de aplicaciones de sincronización de datos móviles basadas en servicios Web bajo una aproximación multiplataforma, son los siguientes:

1. *Back-End:* En teoría cualquier Back-End que exponga servicios Web para el acceso a la lógica de negocio y los datos. Sin embargo cabe aclarar que en este trabajo se utilizaron servicios REST desarrollados en JAVA y Ruby obteniendo una integración adecuada con el servidor de sincronización, pero dada las características de los servicios y la flexibilidad de las aplicaciones de sincronización, sería interesante probar aplicaciones con tecnologías como .NET, Oracle, Sap, Sugar, entre otras capaces de exponer servicios Web.
2. *Mediador o servidor de Sincronización:* RhoSync v 2.1.17
3. *Aplicaciones Clientes:* RhoDes v 3.3.1

En la Figura No. 23, se ilustra el framework o tecnología de desarrollo elegido en cada una de las capas definidas por la arquitectura general del sistema en el capítulo anterior.

Figura No 23. Tecnologías elegidas para cada capa.

4.2. DIAGNOSTICO DEL FRAMEWORK

Debido a la dificultad de encontrar un Framework, para el desarrollo de aplicaciones de sincronización de datos basadas en servicios Web bajo una aproximación multiplataforma, que cumpla completamente con los lineamientos planteados en el capítulo anterior, es necesario realizar una adaptación al Framework seleccionado.

En esta sección se realiza un diagnóstico del Framework con el propósito de identificar sus fortalezas, falencias y definir los requisitos de la adaptación. Para realizar este diagnóstico se tomó como referencia la documentación y algunas pruebas funcionales realizadas al Framework.

4.2.1. TABLA RESUMEN DE LA EVALUACION

En la Tabla No. 10, se presenta un resumen de la confrontación de los lineamientos definidos con las características del Framework seleccionado. Como se puede apreciar, se evidencia una debilidad fuerte en el tratamiento de los conflictos de datos en el servidor de sincronización, por esta razón este trabajo se centrará en el tratamiento de éstos.

La información consignada en la Tabla No 10, es resultado del proceso realizado en el Anexo B.

LINEAMIENTO		PRESENTE	
		SI	NO
Arquitectura General Del Sistema	Arquitectura de Tres Niveles def tres niveles	X	
	Funciones del Mediador		X(*)
	Sitio Maestro Único	X	
	Replicación Asíncrona – Modo desconectado	X	
Gestión De Las Conexiones	Seguridad en el transporte de datos	X	
	Uso de servicios Web REST	X	
	Manejo de Sesión para la Sincronización	X	
Aplicaciones Del Back-End	Interface de acceso a datos y lógica de negocio		
	Notificaciones		
Aplicaciones Multiplataforma	Desarrollo Multiplataforma	X	
	Uso de Smartphone y Tabletas	X	
	Base de datos local	X	
	Almacenamiento cifrado de los datos	X	
	Manejo de Sesión		X (**)
Servidor De Sincronización	Des-aprovisionamiento		X
	Tipos de Sincronización	X	
	API de Sincronización	X	
	Sincronización push	X	
	Sincronización Mayor	X	
	Sincronización Strem	X	
Tratamiento de conflictos de datos	Identificación De Conflictos		X
	Metodos para evitar Conflictos		X
	Métodos de Resolución de conflictos		X
	Registro de los conflictos o Cola de Error		X

Tabla 10. Resumen de la evaluación.

*Cumple solo algunas funciones.

**Depende del método que se requiera implementar en el entorno de sincronización.

4.3. DEFINICION DE LOS REQUISITOS DE ADAPTACIÓN

Con base en el diagnóstico del Framework seleccionado, se presentan a continuación los requisitos de la adaptación que se debe realizar al Framework. Destacando que el presente trabajo y el proceso de adaptación se enfocan principalmente en el tratamiento de conflictos. El des-aprovisionamiento de datos y aplicaciones, no se contempla porque no hace parte del alcance del trabajo definido y se encuentra disponible como un servicio de pago de la suite Rhomobile, denominado RhoGallery. En cuanto al manejo de sesión en los clientes móviles, depende de cada solución determinar que método usar y su respectiva implementación en las aplicaciones cliente depende del desarrollador.

A continuación se presenta la lista de requerimientos de la adaptación.

Requerimiento 1: Implementar un mecanismo que contribuya a evitar los posibles conflictos de sincronización de datos.
Requerimiento 2: Implementar un mecanismo que permita identificar conflictos de datos en el momento en que éstos se presenten en un escenario determinado.
Requerimiento 3: Implementar un mecanismo que permita dar un tratamiento a los conflictos de datos y resolverlos de acuerdo a una política de resolución previamente establecida.
Requerimiento 4: Implementar una cola de error que permita registrar los conflictos de datos que se presentan y los conflictos que por alguna razón no se pueden resolver automáticamente, de esta manera un usuario administrador puede monitorear la ocurrencia de conflictos, consultar que política fue aplicada y si el conflicto fue resuelto de la mejor manera.
Requerimiento 5: Implementar un entorno de desarrollo para facilitar la creación, depuración y despliegue de las aplicaciones Rhosync.

En el proceso de selección del Framework, se evidenció que una de las debilidades de RhoSync, es la ausencia de un entorno integrado de desarrollo; por esta razón y con el objetivo de brindar las herramientas necesarias para facilitar el proceso de desarrollo de aplicaciones de sincronización de datos móviles basadas en servicios Web, en el presente trabajo y adicional al proceso de adaptación, se implementó un entorno integrado de desarrollo para el Framework adaptado.

4.4. IMPLEMENTACIÓN DE LA ADAPTACIÓN DEL FRAMEWORK SELECCIONADO

A partir de los requisitos planteados, se implementa una adaptación a RhoSync denominada cRhosync y algunas modificaciones necesarias a RhoDes, que tienen que ver con el procesamiento de las respuestas del servidor en un proceso de sincronización. La adaptación comprende principalmente la implementación de métodos para la detección, resolución y registro de conflictos, así como la implementación de un entorno de desarrollo para la creación y gestión de los proyectos cRhoSync.

En esta sección se presenta un esquema de la adaptación, los principales diagramas UML que permiten aclarar e identificar las modificaciones hechas al Framework, el proceso de creación y las funcionalidades extra incluidas en un Plugin implementado para el entorno de desarrollo Eclipse. Posteriormente se muestra el procedimiento con algunos casos de prueba en escenarios de conflicto, las respuestas obtenidas de la adaptación, y una tabla de resumen con las características modificadas.

4.4.1. Esquema de la adaptación.

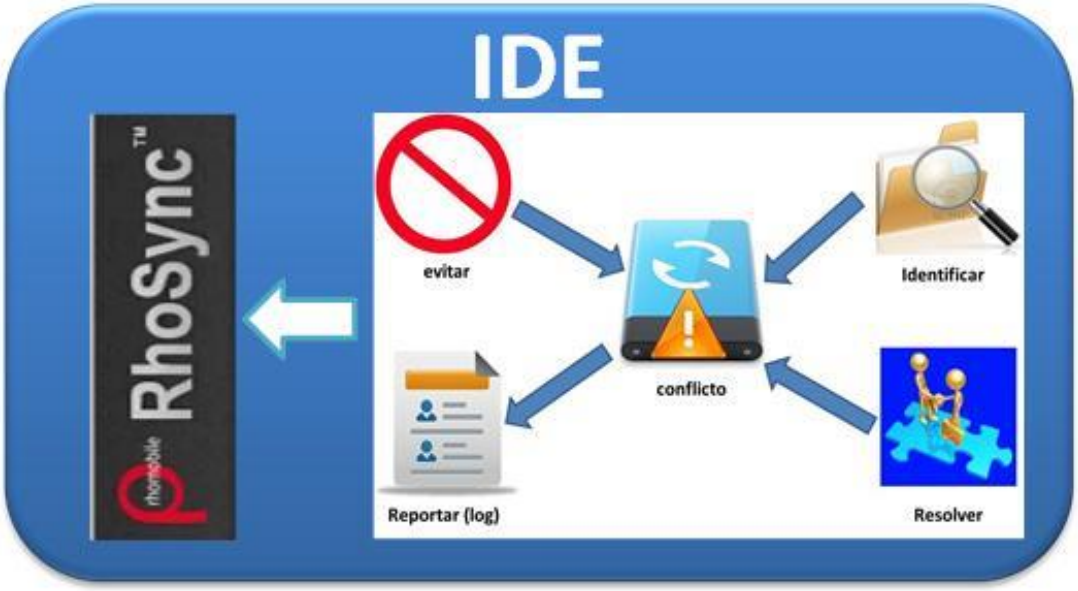


Figura No 24. Esquema de la adaptación.

La Figura No. 24, muestra el esquema general de la adaptación, en donde se puede identificar las principales funcionalidades relacionadas con evitar, identificar, resolver y registrar los conflictos de datos. Estas funcionalidades deberán ser extendidas al Framework RhoSync 2.1.17, que es la última versión estable desarrollada por Rhomobile. Luego de la modificación del Framework, el entorno de desarrollo integrado, facilitará la creación, depuración y despliegue de las aplicaciones de sincronización de bases de datos móviles, basadas en servicios Web. A continuación se describen generalmente las principales características de la adaptación.

***a. Identificación de Conflictos*:** En la adaptación, se implementaron métodos para la identificación de conflictos de unicidad, actualización y eliminación, mediante una base de datos presente en el servidor de sincronización que permite identificar los conflictos de datos en cada uno de los modelos o “tablas” replicadas.

***b. Resolución Conflictos*:** La adaptación cuenta con métodos para la resolución de conflictos de unicidad y actualización:

- Conflictos de unicidad: Id Consecuente.
- Conflictos de actualización: Ultima y mas receinte marca de tiempo, descarte, sobrescribir, número de columnas validas, particionamiento de datos, perfil de usuarios y los numéricos.
- Los conflictos de eliminación no pueden ser resueltos, en estos casos se registra el conflicto en Log y se notifica al cliente.

c. Evitar Conflictos: El trabajo cuenta con mecanismos para evitar conflictos de eliminación y unicidad, en el caso de los conflictos de eliminación se dispone del método de borrado suave. Los conflictos de actualización no se pueden evitar.

***d. Registro de Conflictos*:** En caso de detectar un conflicto de datos y que por alguna razón no pueda ser resuelto, como es el caso de los de eliminación, se registrará en el log o bitácora de error de manera que un administrador del sistema pueda resolver manualmente estos conflictos.

4.4.2. Modelado de la adaptación.

4.4.2.1. Diagramas de Casos de uso.

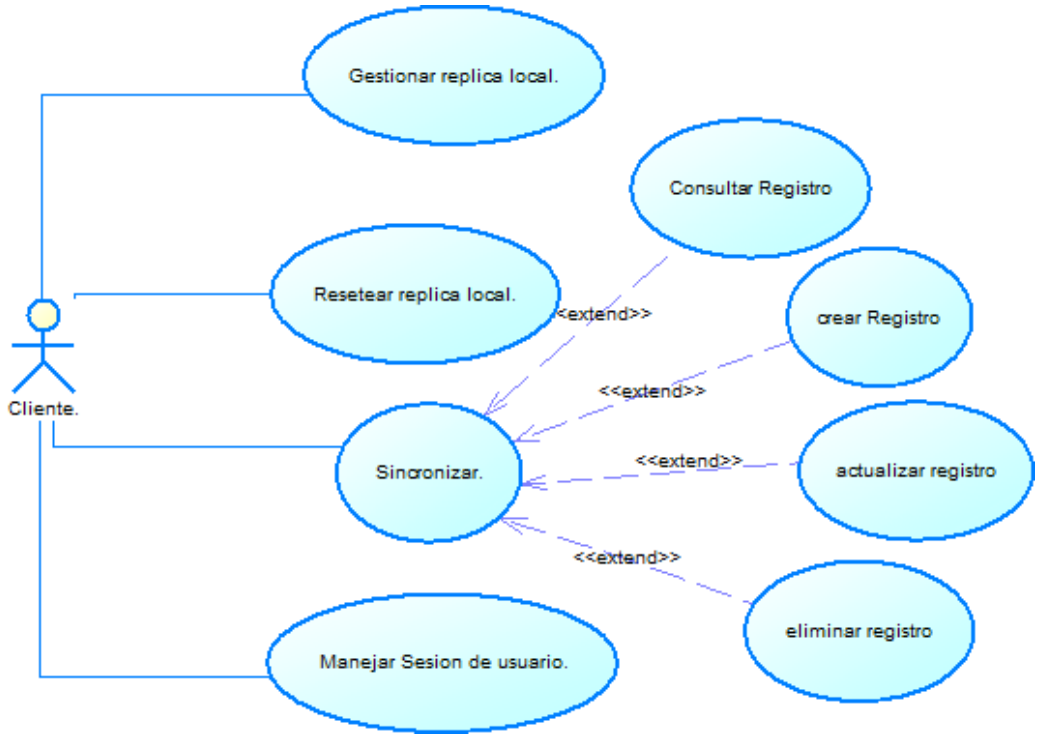


Figura No 25. Diagrama de casos de uso general de una aplicación de sincronización desarrollada con Rhosync 2.1.17 y Rhodes 3.3.1.

La Figura No. 25, es la representación del diagrama de casos de uso de una aplicación desarrollada con Rhosync 2.1.17 y Rhodes 3.3.1 sin la adaptación. Como se aprecia en el diagrama, la principal función es la sincronización de las diferentes operaciones que se pueden realizar en la base de datos local. También se identifican dos funcionalidades más, que permiten resetear la base de datos local del móvil y el manejo de sesión, necesario para ejecutar las operaciones de sincronización.

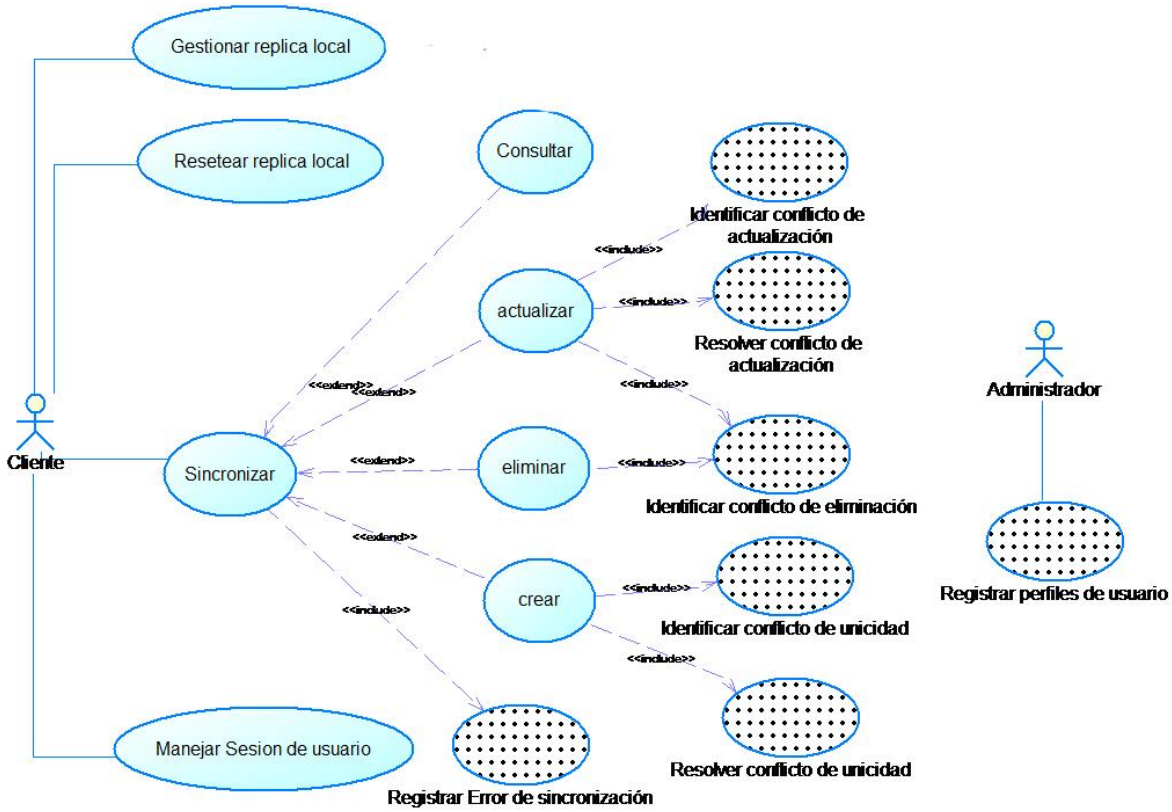


Figura No 26. Diagrama de casos de uso de la adaptación del Framework de Sincronización.

Cliente: Corresponde a un usuario del sistema que accederá a través de un dispositivo móvil.

Administrador: Este realiza tareas de administración en el servidor de sincronización, encargado principalmente del registro y del manejo de perfiles de usuario.

La Figura No. 26, representa el diagrama de casos de uso de la adaptación del Framework de sincronización deniminada cRhoSync. Los casos de uso con relleno punteado, definen las funcionalidades extra agregadas en el proceso de adaptación, con el propósito de cumplir con los requerimientos planteados. Estas funcionalidades permiten dar un tratamiento efectivo a los conflictos de datos que se puedan presentar durante la sincronización en la ejecución de alguna de las operaciones CRUD (Create, Update y Delete).

La descripción detallada de los casos de uso necesarios para la adaptación, se encuentra en el anexo A. Sin embargo, se hace una aclaración importante respecto al mecanismo de borrado suave de registros, que permite evitar el conflicto de eliminación; y por otro lado, para los conflictos de actualización que

por su naturaleza no se pueden evitar, se han definido múltiples métodos de resolución mencionados anteriormente.

4.4.2.2. Diagramas de componentes.

El diagrama que se muestra en la Figura No. 27, recoge los componentes más importantes que hacen parte de una aplicación generada con el framework Rhosync 2.1.17. El principal componente es *sinatra*, que es un DSL (Lenguaje Especifico de Dominio) escrito en Ruby, libre y de código abierto para el desarrollo de aplicaciones web con un mínimo esfuerzo [119]. Este permite recoger las peticiones de los clientes móviles a través del componente *server*, y las peticiones provenientes de la interfaz de administración hacia el componente *RhosyncConsole*.

Cuando se despliega una aplicación Rhosync, son cargados todos los componentes *source* que haya creado el desarrollador. El componente *ClientSync* se encarga de manejar todas las peticiones de entrada y salida, delegando a *SourceSync* para procesar cada operación CRUD que trae la petición, y ejecutar la lógica de uso de los servicios del servidor central descritos en los componentes *source*. Finalmente se retornan las respuestas del proceso de sincronización a través del mismo camino de llegada.

Por lo tanto la adaptación para el manejo de conflictos está concentrada en los procesos que se ejecutan en el módulo *SourceSync*.

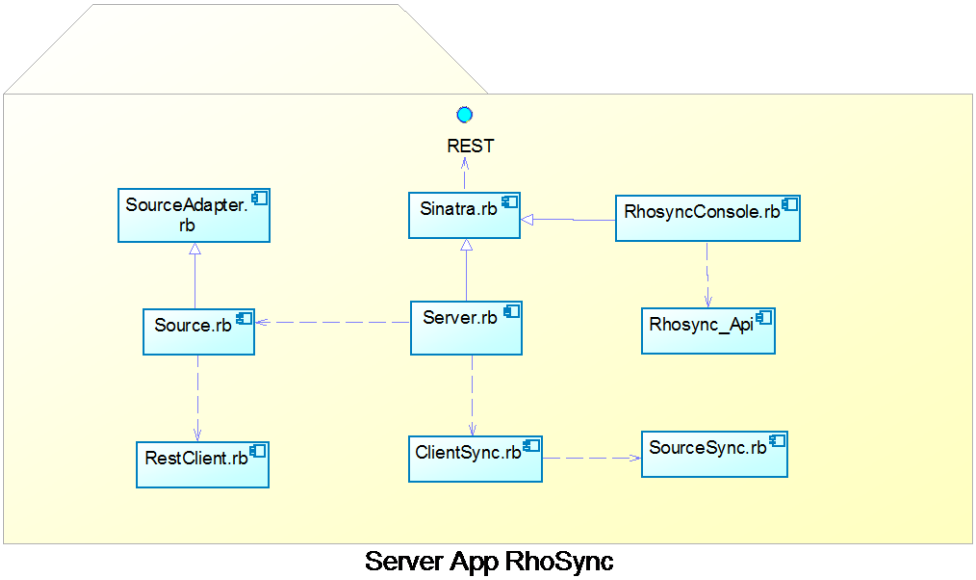


Figura No 27. Diagrama General de Componentes de una aplicación de sincronización desarrollada con Rhosync 2.1.17 y Rhodes 3.3.1.

Luego de realizar la adaptación al Framework, se obtiene un diagrama de componentes como el que se observa en la Figura No. 28. Los componentes en negro representan las modificaciones, y los punteados las adiciones.

Principalmente el trabajo se concentró en incluir los procedimientos de detección, resolución y registro de los conflictos de datos en el componente *SourceSync*, donde se procesa cada operación CRUD que llega con una petición de sincronización.

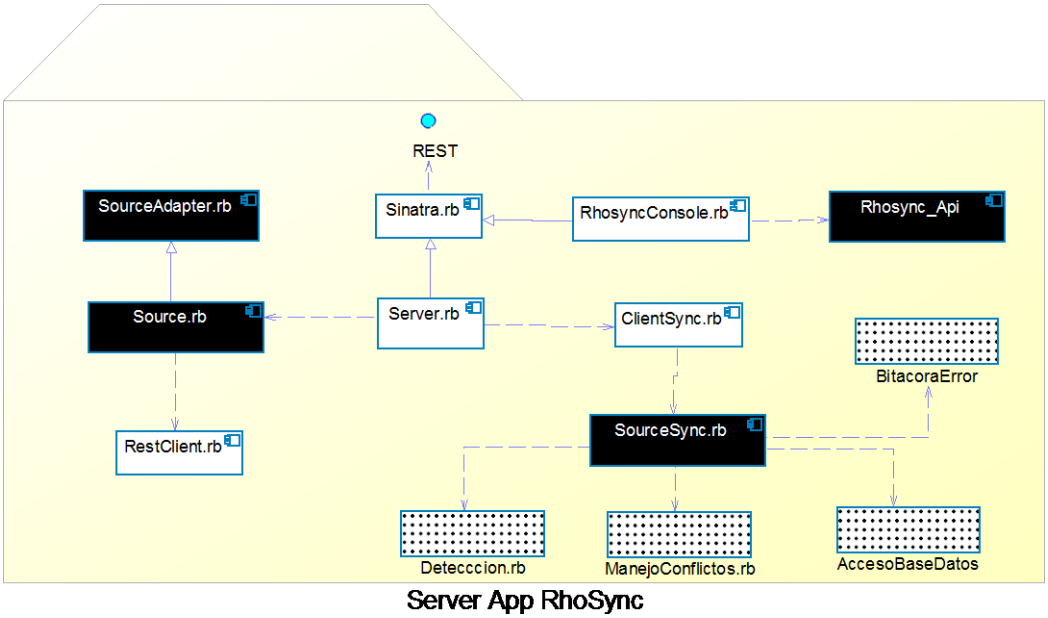


Figura No 28. Diagrama General de Componentes de una aplicación de sincronización desarrollada con el Framework Adaptado.

Las modificaciones hechas a los componentes *SourceAdapter* y *Source* (Figura 28), fueron necesarias para soportar el funcionamiento de los mecanismos de detección y resolución de conflictos, y se realizaron adicionando los parámetros necesarios en los métodos de tratamiento de conflictos. La modificación en la consola de Rhosync, se centró en la gestión de los perfiles de los usuarios móviles y la adición de una vista de la consola de errores de sincronización.

El componente *AccesoBaseDatos*, está encargado de la gestión de una tabla de datos Mysql, donde se registra y consulta la información para el tratamiento de conflictos. Esta tabla, es un resumen de las operaciones de sincronización específicas para cada registro y adaptador o fuente de datos; se almacenan el último usuario que lo modificó o creó, y las marcas de tiempo del cliente y del servidor.

De ahí que para el proceso de identificación y resolución de los conflictos, es necesario conocer las marcas de tiempo de la última y anterior actualización o

modificación en el cliente, por esta razón se realizó una modificación en el Framework Rhodes para que las aplicaciones cliente envíen estos datos y de esta manera estén disponibles al momento de la sincronización. De esta forma se puede determinar, si la versión anterior que tiene el móvil coincide con la versión actual del registro en la base de datos para el manejo de conflictos; existirá un conflicto de actualización, si las versiones no coinciden.

En la Tabla No. 11, se ilustra cómo funciona el mecanismo para la identificación de conflictos de actualización implementado. Las columnas describen el momento en que se ejecuta la acción, la descripción de la acción, la versión del dato en el sitio maestro, la versión actual y anterior del dato en cada cliente móvil y por último la versión del dato que se almacena en la tabla para el manejo de conflictos respectivamente.

TIEMPO	ACCIÓN	SITIO MAESTRO	CLIENTE A	CLIENTE B	TABLA DE CONFLICTO
1	Convergencia de datos en el registro R=1	R=1	R(anterior)=1 R(actual)=1	R(anterior)=1 R(actual)=1	R=1
2	El cliente A modifica el registro a R=2	R=1	R(anterior)=1 R(actual)=2	R(anterior)=1 R(actual)=1	R=1
3	El cliente A sincroniza exitosamente el Registro R=2, ya que la versión anterior del cliente A coincide con la versión de la tabla para detectar conflictos.	R=2	R(anterior)=2 R(actual)=2	R(anterior)=1 R(actual)=2	R=2
4	El Cliente B modifica el registro a R=3	R=2	R(anterior)=2 R(actual)=2	R(anterior)=1 R(actual)=3	R=2
5	Sincroniza y se detecta un conflicto de actualización en R, debido a que la versión anterior de R en el cliente (B) es R=1 y no coincide con la versión de la tabla para detectar conflictos.	R=2	R(anterior)=2 R(actual)=2	R(anterior)=1 R(actual)=3	R=2 Conflicto de datos detectado.

Tabla 11. Identificación del conflicto de actualización.

Para la identificación del conflicto de unicidad y eliminación, se utiliza la misma tabla de la base de datos de manejo de conflictos, donde se comprueba la existencia o no de la clave primaria del registro. Para el caso de los conflictos de unicidad, se hace esta comprobación en una creación; si el identificador del registro a crear ya existe, se presenta un conflicto de unicidad. La detección de los conflictos de eliminación, se hace para peticiones de actualización o eliminación; antes de cada una de estas operaciones, se verifica si el registro existe en la tabla de manejo de conflictos; si no se encuentra, un conflicto de eliminación es detectado.

En la Tabla No 12, se ilustra cómo funciona el mecanismo para la identificación de conflictos de eliminación implementado. La información de las columnas es similar a la Tabla No 11, sólo que ahora en cada replica de datos se tiene en cuenta la llave primaria de cada registro.

TIEMPO	ACCIÓN	SITIO MAESTRO	CLIENTE A	CLIENTE B	TABLA DE CONFLICTO
1	Convergencia de datos en el registro R=1	R=(PK = 1)	R=(PK = 1)	R=(PK = 1)	R=(PK = 1)
2	El cliente A elimina el registro a con PK=1	R=(PK = 1)		R=(PK = 1)	R=(PK = 1)
3	El cliente A sincroniza exitosamente la eliminación.			R=(PK = 1)	
4	El Cliente B modifica o elimina el registro con PK=1			R_1=(PK = 1)	
5	Cuando el cliente B sincronice la eliminación o modificación del registro con PK=1, se presenta un conflicto de eliminación por que el registro no existe sino en su base local.				

Tabla 12. Identificación de conflictos de eliminación.

En la Tabla No 13, se ilustra cómo funciona el mecanismo para la identificación de conflictos de unicidad implementado. La información de las columnas es similar a la tabla anterior.

TIEMPO	ACCIÓN	SITIO MAESTRO	CLIENTE A	CLIENTE B	TABLA DE CONFLICTO
1	No existe un registro especifico en ninguna replica.				
2	El cliente A crea un registro con PK=1		R_A=(PK = 1)		
3	El cliente A sincroniza exitosamente la creación.	R_A=(PK = 1)	R_A=(PK = 1)		R_A=(PK = 1)
4	El cliente B crea un registro con PK=1			R_B=(PK = 1)	
5	Cuando el cliente B sincronice la creación del registro con PK=1, se presenta un conflicto de unicidad por que el registro ya existe en la base de datos central.				

Tabla 13. Identificación de conflictos de unicidad.

En el siguiente capitulo se describen las pruebas realizadas a la adaptación, con el animo de verificar la funcionalidad de los métodos de identificación y resolución de conflictos implementados.

4.4.3. Plug-in para Rhosync.

Una de las falencias identificadas en la construcción de aplicaciones con RhoSync v 2.1.17, era la ausencia de un entorno de desarrollo (IDE). La suite de Rhomobile provee un plugin para eclipse llamado RhoStudio, que permite la creación de aplicaciones para otros productos como RhoConnect y RhoDes. Usando a RhoStudio como punto de partida y gracias a que es una aplicación de código abierto disponible en el repositorio github bajo una licencia (MIT), se modificó este plugin de eclipse para que soporte la creación, depuración y despliegue de aplicaciones RhoSync.

Con RhoSync 2.1.17, la creación de aplicaciones se realiza a través de la ejecución manual de comandos expuestos por la gema ruby Templater, que permite generar el esquema de una aplicación RhoSync desde la interfaz de comandos a partir de unas plantillas Ruby On Rails. Debido a esto se modificaron las plantillas de generación de los adaptadores fuente, para que incluyeran las características necesarias para la adaptación.

Las características del plugin para el desarrollo de aplicaciones cRhoSync, son:

a. Creación de proyectos cRhoSync a través de un asistente.
b. Creación de adaptadores fuente a través de un asistente.
c. Editar y depurar códigos ruby.
d. Despliegue de aplicaciones cRhoSync.

Para mayor detalle de las características del plugin, se puede consultar el anexo D

4.4.4. Resumen de la adaptación.

La Tabla No 14, resume las características de la adaptación cRhoSync y se confrontan con las de la versión 2.1.17 de RhoSync, como punto de partida.

Framework / **Características**	**RhoSync V. 2.1.17**	**cRhoSync**
Evitar conflictos de datos.	Ningún método.	Recomendaciones para evitar conflictos de eliminación y, recomendaciones para evitar conflictos de unicidad.
Identificación de Conflictos de datos.	Ningún método de identificación.	Identificación de conflictos de eliminación, unicidad y actualización.
Resolución de Conflictos de datos.	Ningún método de resolución.	Métodos de resolución de conflictos de unicidad y actualización.
Identificación de operaciones exitosas en el back-end.	Solo se reportan y procesan los errores que ocurren en el servidor de sincronización.	El desarrollador adapta según el comportamiento del servicio, las respuestas después de la ejecución de los métodos de creación, actualización y eliminación de los adaptadores fuente.
Log o bitácora de errores de sincronización.	No se hace un registro de los errores que ocurren en las peticiones.	Se registra las excepciones ocurridas en el procesamiento de las peticiones de sincronización, como los conflictos de datos.
IDE (Entorno de Desarrollo Integrado)	Creación de proyectos a través de Interfaz de comandos.	Plugin RhoStudio extendido para el manejo de proyectos RhoSync.

Tabla 14. Resumen de la Adaptación.

5. CONSTRUCCIÓN DE UN PILOTO PARA LA VALIDACIÓN DEL MECANISMO Y LOS LINEAMIENTOS

En el presente capítulo, se presenta un caso de estudio con el objetivo de ejemplificar el uso y aplicación de los lineamientos planteados y el Framework adaptado cRhoSync para la construcción de aplicaciones de sincronización de bases de datos móviles basadas en servicios Web. El escenario establecido, está definido en las labores de recolección de datos en campo ejecutadas por la empresa UTEN Seccional Cauca. La figura 29, describe el proceso de desarrollo que se utilizó para la implementación del prototipo.

Para mayor detalle del proceso de construcción del piloto, se puede consultar el anexo E, que corresponde al análisis y diseño del software en forma general.

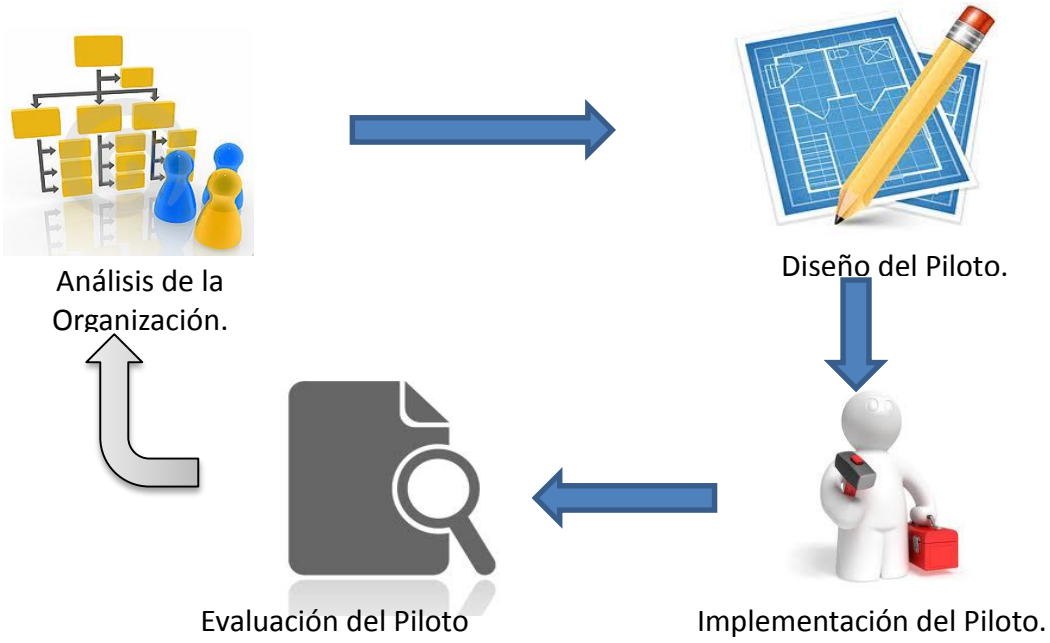


Figura No 29. Proceso de desarrollo del piloto

5.1. Caso de estudio

UTEN Seccional Cauca, es una organización prestadora de servicios en la industria energética nacional, actualmente encargada de ejecutar las actividades de operación y mantenimiento del sistema eléctrico en el departamento del Cauca, bajo la modalidad de contrato colectivo sindical suscrito con Cedelca S.A. y la Compañía energética de Occidente, en los procesos de generación, distribución y comercialización de energía eléctrica. En cada uno de estos procesos, diariamente se ejecutan diferentes tipos de labores operativas en todo el departamento, por lo general en entornos móviles y en algunos casos en sitios sin acceso a una red de datos.

5.2. Análisis de la organización

5.2.1. Entrevista de diagnóstico

Con el propósito de determinar los requerimientos y las necesidades tecnológicas para el tratamiento de datos en campo fue realizada una entrevista con algunos funcionarios de la empresa UTEN como se muestra en la figura No. 30; como resultado de este dialogo se generó el siguiente diagnóstico:

Actualmente el registro y control de las actividades desarrolladas por los cerca de mil trabajadores en terreno, se realiza mediante formularios diligenciados manualmente, que posteriormente son llevados a la oficina operativa donde pueden ser digitados en un sistema de información central o en tablas de datos. La empresa no cuenta con un sistema de información único que facilite el registro, el control, almacenamiento y la organización de la información operativa, generada diariamente en campo.

Bajo el modelo actual de operación se presentan diferentes dificultades en procesos como:

1. Recopilación de Datos: Formularios mal diligenciados por los trabajadores con campos incompletos o con información incoherente, además se usa y desperdicia gran cantidad de papel, lo que incrementa la huella ambiental de la organización.

2. *Transporte de la información:* Dificultad para conservar la integridad y seguridad de la información y tardanzas en el transporte de los formularios o actas físicas hasta la oficina operativa.

3. Digitación de la información: Inconvenientes con la interpretación de los datos recolectados, que implican errores de digitación y disminuye la confiabilidad de la información.

4. Registro y almacenamiento de la información: Dificultad para registrar y almacenar la información, debido a que no se dispone de un sistema de información único y adecuado que permita realizar el proceso de gestión de la información.

Por las razones mencionadas anteriormente, actualmente se presentan dificultades en la gestión de la información al interior de la empresa, lo que genera sobrecostos y tardanzas en la toma de decisiones en procesos de más alto nivel. En este sentido, existe la necesidad de un sistema que permita optimizar el proceso de recolección y manipulación de datos en campo; además de un registro y control sistematizado más confiable y organizado de la información operativa.

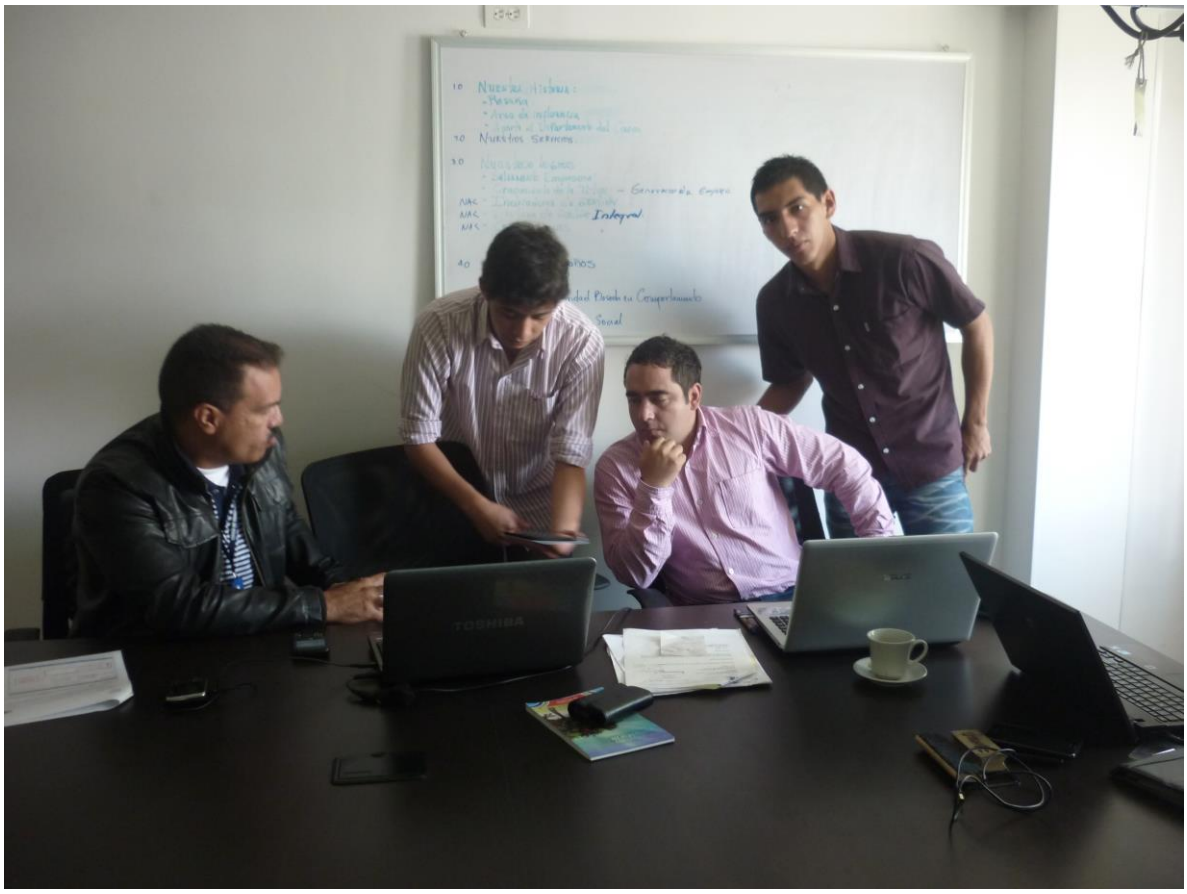


Figura No 30. Entrevista de diagnostico

5.2.2. Modelado de negocio y su descripción.

En la Figuras No 31-35 se ilustra el modelado del negocio.

a. Descripción de Actores.

Usuario: Se refiere al suscriptor del servicio de energía eléctrica.

Técnico Electricista: Es quien ejecuta las diferentes labores operativas de la organización en campo.

Supervisor: Encargado de supervisar a los técnicos, en aspectos como la calidad en la ejecución de las labores, herramientas, materiales y equipos.

Digitador: Encargado de digitar las actas, formularios e informes generados por los supervisores y técnicos electricistas en campo.

Coordinador de Proceso: Es el responsable de cada proceso de la organización y es quien dirige la ejecución de las labores, la medición de calidad, productividad y cálculo de indicadores de gestión.

b. Modelo de la Organización.

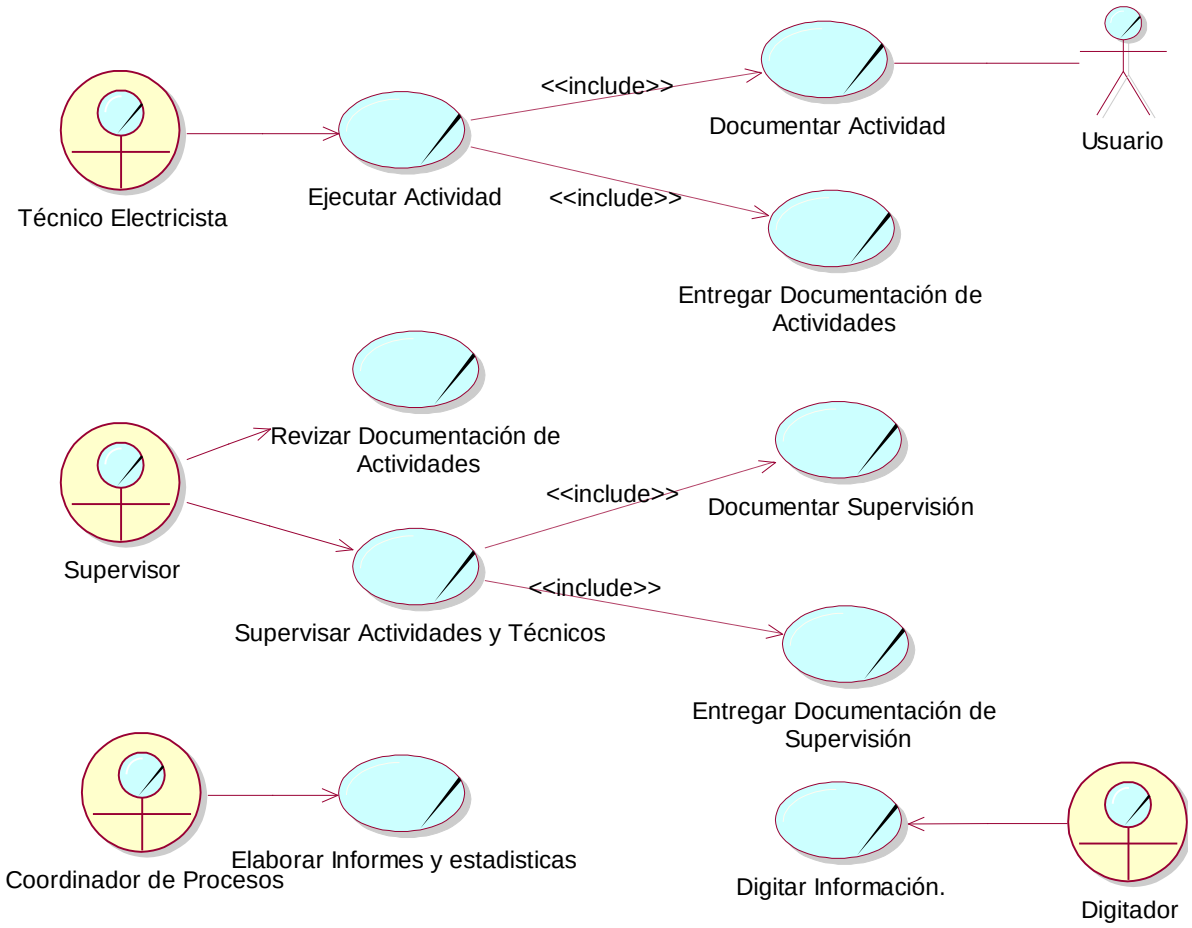


Figura No 31. Diagrama de casos de uso del Negocio.

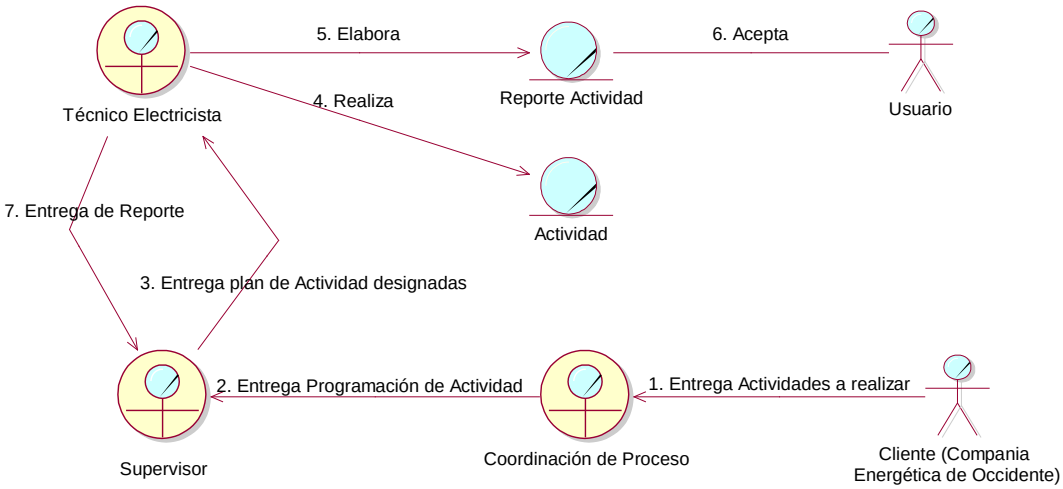


Figura No 32. Modelo objeto del negocio. Ejecutar Actividad.

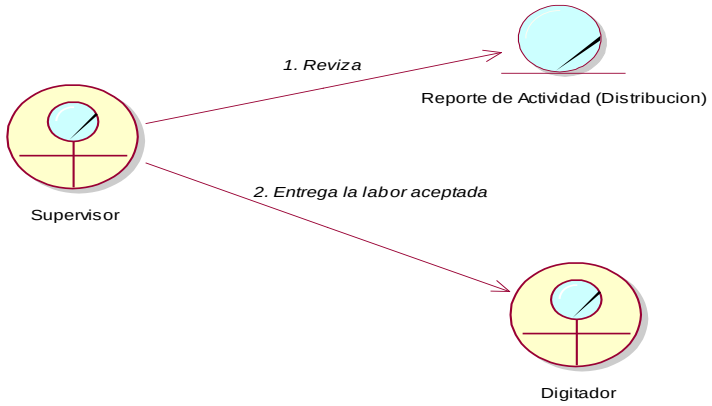


Figura No 33. Modelo objeto del negocio, Supervisar actividades y técnicos.

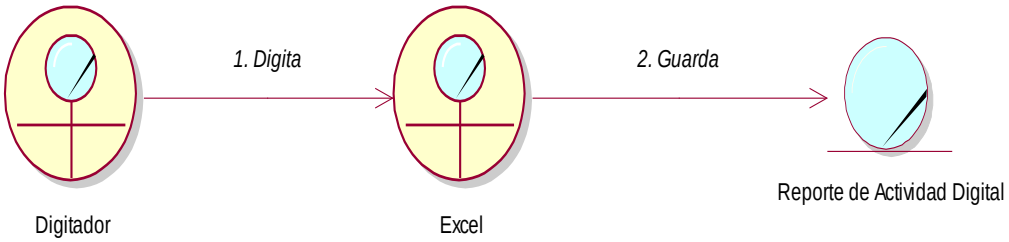


Figura No 34. Modelo Objeto del Negocio. Revisar Documentación y Digitar Información.

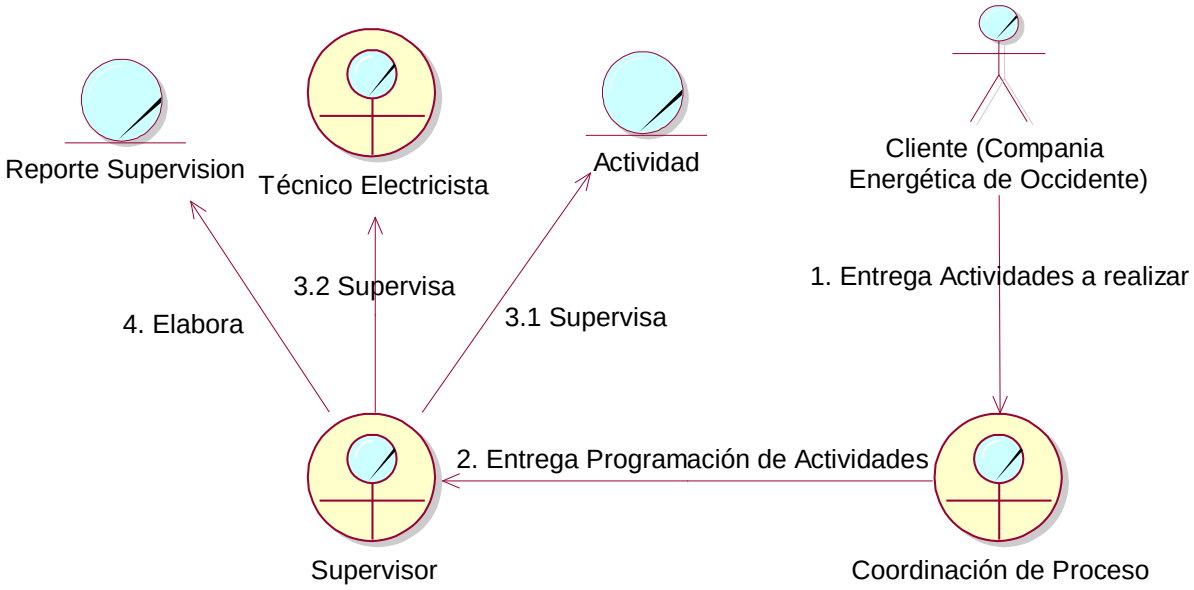


Figura No 35. Modelo Objeto del Negocio. Supervisar Actividades y Técnicos.

5.3. DISEÑO Y MODELADO DEL PILOTO

5.3.1. Requerimientos funcionales y no funcionales

Con base en el diagnóstico realizado en la empresa se plantearon los siguientes requerimientos para el piloto:

Requerimiento 1: Construir una solución que permita a los trabajadores de la organización documentar y recoger la información relacionada con la ejecución de sus labores en campo por medio de dispositivos móviles.

Requerimiento 2: La información recogida en campo debe ser almacenada y sincronizada con un servidor de datos central en la empresa.

Requerimiento 3: Se debe permitir a los trabajadores, operar en modo desconectado, debido a la escasa cobertura de redes móviles en algunas regiones del departamento y con el ánimo de disminuir costos innecesarios en el transporte de la información.

5.3.2. Descripción del piloto

a. Descripción de Actores.

   Usuario: Se refiere al suscriptor del servicio de energía eléctrica.

   Técnico Electricista: Es quien ejecuta las diferentes labores operativas de la organización en campo.

   Supervisor: Encargado de supervisar a los técnicos, en aspectos como la calidad en la ejecución de las labores, herramientas, materiales y equipos; y registra las actividades de supervisión en un dispositivo móvil.

En la Figura No. 36 se muestra el modelo de casos de uso del sistema.

b. Modelo de Casos de Uso del Sistema.

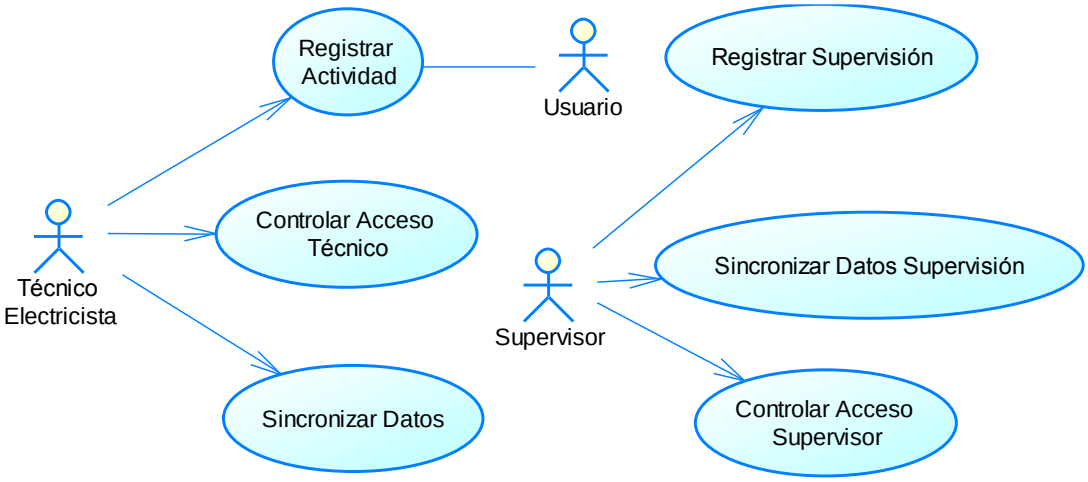


Figura No 36. Modelo de Casos de Uso del Sistema.

Para la construcción del piloto y teniendo en cuenta los alcances del trabajo de grado, así como las recomendaciones surgidas de la entrevista de diagnóstico, el piloto se realizó en el marco de las tareas del Supervisor, debido a la prioridad de sus actividades de control para la organización. Para más detalles de los casos de uso del piloto, consultar el Anexo E.

5.3.3. Arquitectura de Referencia

A continuación se describe, la arquitectura de referencia implementada en el piloto. En la figura No. 37, se ilustra el diagrama de despliegue.

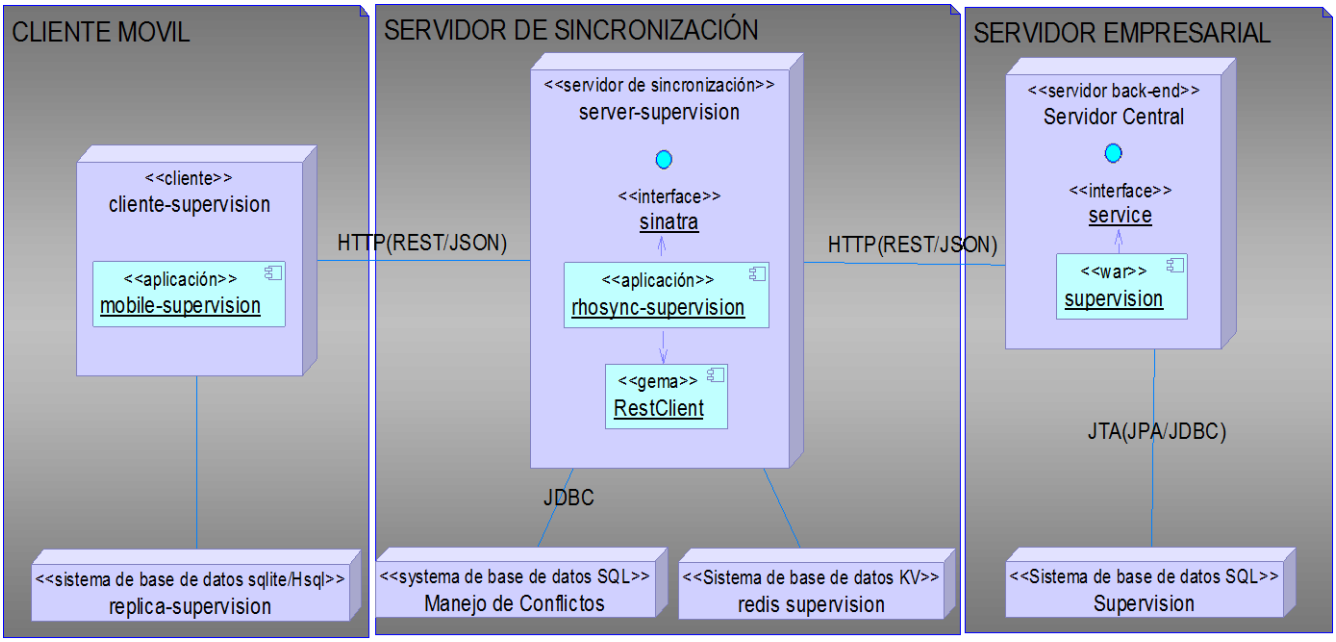


Figura No 37. Diagrama de Despliegue del piloto.

Como se ilustra en la Figura No. 37, se tiene una arquitectura en tres niveles, cliente, servidor de sincronización y servidor empresarial. En el cliente, se tiene dos nodos principales, que corresponden a la aplicación y la réplica de la base de datos; en el servidor de sincronización se observa tres nodos, el servidor de sincronización, la base de datos para el manejo de conflictos y la base de datos redis, que maneja la información completa de los procesos de sincronización; y el finalmente, en el servidor empresarial, se tienen dos nodos principales, que corresponden al servidor de aplicaciones que despliega el servicio de acceso al otro nodo que es la base de datos central.

5.3.4. Componentes de la arquitectura

A continuación, en la Figura No. 38, se muestra, un diagrama con los módulos del piloto implementado, y las herramientas utilizadas para la implementación de la arquitectura.

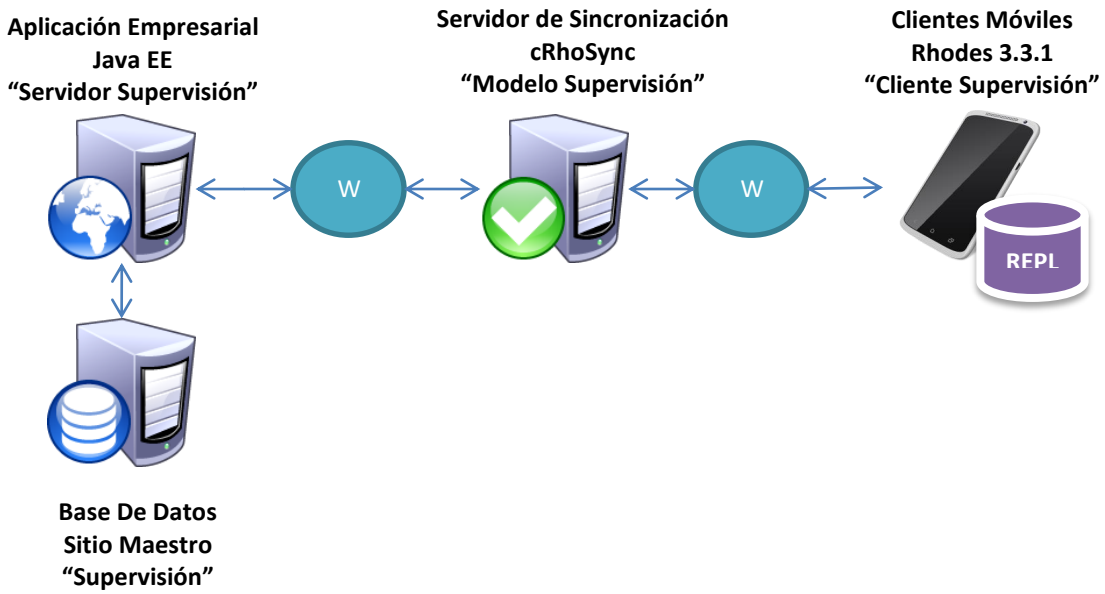
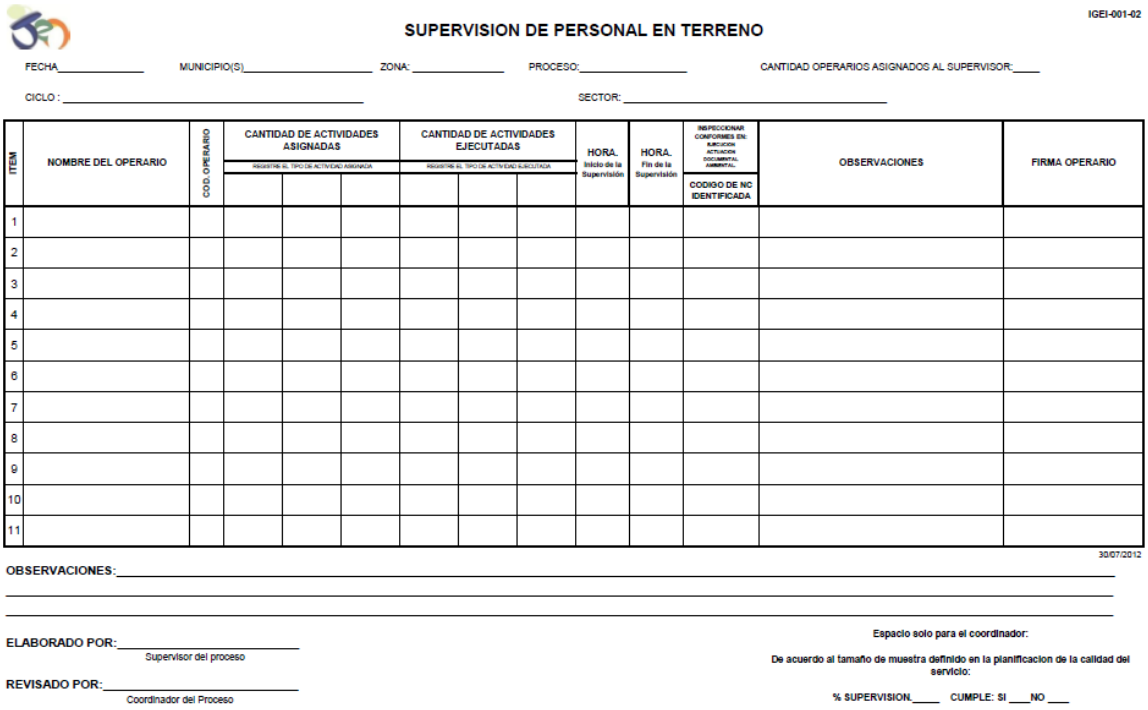


Figura No 38. Componentes de la Arquitectura del Piloto

5.3.5. Descripción de la Aplicación

De acuerdo con el modelo de negocio de la empresa y su Sistema Integrado de Gestión, se determinó conjuntamente con los representantes de la organización implementar para el piloto uno de los formularios utilizados por los Supervisores para recolectar información en campo denominado “IGEI-001-02 Supervisión de Personal en Terreno”; ver Figura No. 39.



IGEI-001-02

SUPERVISION DE PERSONAL EN TERRENO

FECHA_____ MUNICIPIO(S)_____ ZONA: _____ PROCESO:_____ CANTIDAD OPERARIOS ASIGNADOS AL SUPERVISOR:____

CICLO : _____ SECTOR: _____

ITEM	NOMBRE DEL OPERARIO	COD. OPERARIO	CANTIDAD DE ACTIVIDADES ASIGNADAS			CANTIDAD DE ACTIVIDADES EJECUTADAS			HORA. Inicio de la Supervisión	HORA. Fin de la Supervisión	INSPECCIONAR CONFORMES EN: EJECUCION ACTUACION DOCUMENTAL AMBIENTAL. CODIGO DE NC IDENTIFICADA	OBSERVACIONES	FIRMA OPERARIO
1													
2													
3													
4													
5													
6													
7													
8													
9													
10													
11													

30/07/2012

OBSERVACIONES:_____

ELABORADO POR:_____
Supervisor del proceso

REVISADO POR:_____
Coordinador del Proceso

Espacio solo para el coordinador:

De acuerdo al tamaño de muestra definido en la planificacion de la calidad del servicio:

% SUPERVISION____ CUMPLE: SI ___NO ___

Figura No 39. Formato IGEI-001-02 Supervisión de Personal en Terreno

El piloto es desplegado usando la arquitectura anteriormente expuesta, la aplicación móvil “Cliente Supervisión” permite a los clientes móviles manipular y sincronizar desde su réplica móvil la base de datos “Supervisión” y la tabla denominada “Supervisión de Personal”, alojada en el Back-End por medio de la aplicación “Server Supervisión” en el servidor de sincronización. La Figura No. 40 muestra una de las interfaces de la aplicación móvil.

Figura No 40. Aplicación cliente móvil.

En este caso, y de acuerdo con los lineamientos planteados y el entorno de sincronización del caso de estudio son implementados los siguientes métodos para el tratamiento de conflictos de datos:

5.3.5.1. Conflictos de Unicidad:

Desde el cliente móvil se permite ingresar la clave primaria de la tabla en una operación (Crear), en una operación actualizar no se permite modificar la clave primaria ya que podría afectar la convergencia de los datos. En caso de presentarse un conflicto en una operación crear, este será resuelto mediante el método de **Id Consecuente**, de esta forma se pretende garantizar la convergencia de datos frente a este conflicto.

5.3.5.2. Conflictos de Actualización

Debido a que el entorno de sincronización presenta una propiedad compartida de datos y se pueden presentar conflictos de actualización, se establece como método de resolución de conflictos el de **Última marca de tiempo**. De esta manera se pretende garantizar convergencia de datos frente a los posibles conflictos de este tipo.

5.3.5.3. Conflictos de Eliminación

Es implementado el método de borrado suave para evitar la presencia de conflictos de eliminación, dicha implementación se realiza en el servidor empresarial Back-End. De esta manera se pretende garantizar convergencia de datos frente a los posibles conflictos de este tipo.

5.3.5.4. Registro de Conflictos

Si por algún motivo los conflictos de datos no pueden ser resueltos, serán registrados en la bitácora o log de errores; la Figura No. 41, ilustra la consola o bitácora de errores de sincronización de la aplicación.

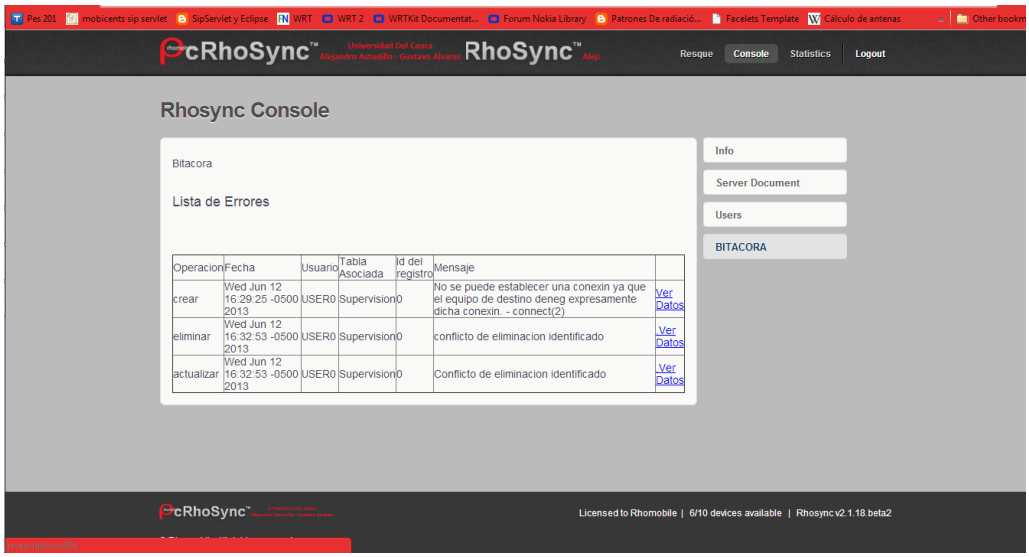


Figura No 41. Bitácora de error.

5.4. PRUEBAS FUNCIONALES

A continuación se presenta el proceso de pruebas funcionales realizadas al piloto y los resultados obtenidos, las pruebas consistieron en generar diferentes tipos de conflictos desde dos clientes móviles, para determinar de qué manera la aplicación y la adaptación cRhoSync resuelve los conflictos de datos. La Figura No. 42 fue capturada el dia de las pruebas funcionales.

Figura No 42. Pruebas funcionales.

Las tablas resumen que se presentan a continuación ilustran el comportamiento del piloto desarrollado utilizando la adaptación cRhosync, cuando se presenta algún conflicto de datos. En sus columnas se registra: El momento en que se ejecuta cada acción, la descripción de la acción, que sucede con los datos en el

sitio maestro y en las réplicas de los dispositivos móviles y la columna *"Tabla de Conflictos cRhoSync"* que describe el estado de los registros en cRhoSync para el manejo de conflictos.

5.4.1. Conflictos de Unicidad:

Se genera un conflicto de unicidad entre dos clientes móviles en una operación CREAR REGISTRO, en la Tabla No. 15 se presenta el resultado que permite determinar el tratamiento, que la adaptación cRhoSync da a un conflicto de Unicidad.

TIEMPO	ACCIÓN	SITIO MAESTRO	CLIENTE A	CLIENTE B	Tabla de Conflictos cRhoSync
1	Cliente A crea un registro R1 con clave primaria PK=1		R1={PK=1…} (Offline)	(Offline)	
2	Cliente B crea un segundo registro R2 con la misma clave primaria PK=1		R1={PK=1…} (Offline)	R2={PK=1…} (Offline)	
3	Cliente A Sincroniza exitosamente sus datos con el sitio maestro.	R1={PK=1…}	R1={PK=1…} (Online)	R2={PK=1…} (Offline)	R1={PK=1…}
4	Cliente B envía solicitud de sincronización de datos.	R1={PK=1…}	R1={PK=1…} (Offline)	R2={PK=1…} (Online)	R1={PK=1…}
5	El servidor de sincronización detecta un conflicto de unicidad, debido a que ya existe un registro con la clave primaria igual a 1 en la tabla de conflictos de RhoSync.	R1={PK=1…}	R1={PK=1…} (Offline)	R2={PK=1…} (Online)	R1={PK=1…}
6	El servidor de sincronización resuelve el conflicto de acuerdo al método definido para este fin (En este caso ID consecuente).	R1={PK=1…}	R1={PK=1…} (Offline)	R2={PK=1…} (Online)	R1={PK=1…} R2={PK=2…}
7	El servidor de sincronización inserta el registro en el sitio	R1={PK=1…} R2={PK=2…}	R1={PK=1…} (Offline)	R1={PK=1…} R2={PK=2…} (Offline)	R1={PK=1…} R2={PK=2…}

	central e informa al cliente de los cambios realizados. El cliente A se Sincronizara en la siguiente conexión.				

Tabla 15. Proceso de un conflicto de unicidad de cRhoSync.

Como se puede apreciar en la Tabla No. 15, la adaptación cRhoSync detecta el conflicto de unicidad gracias a la tabla para manejo de conflictos; una vez es detectado se resuelve por el método definido, en este caso (ID Consecuente); los cambios son transmitidos al servidor empresarial y al cliente móvil.

5.4.2. Conflictos de Actualización:

Se genera un conflicto de actualización entre dos clientes móviles en una operación MODIFICAR REGISTRO, haciendo cambios aproximadamente al mismo tiempo a un registro existente. En la Tabla No. 16 se presenta el resultado que permite determinar el tratamiento que la adaptación cRhoSync, da a un conflicto de Actualización.

TIEMPO	ACCIÓN	SITIO MAESTRO	CLIENTE A	CLIENTE B	Tabla de Conflictos cRhoSync
1	Convergencia de datos, en todos los sitios en el valor: R1= Carlos	R1= Carlos	R1= Carlos (Offline)	R1= Carlos (Offline)	R1= Carlos
2	Cliente A actualiza a R1= Juan	R1= Carlos	R1= Juan (Offline)	R1= Carlos (Offline)	R1= Carlos
3	Cliente B actualiza a R1 = Pedro	R1= Carlos	R1= Juan (Offline)	R1 = Pedro (Offline)	R1= Carlos
4	Cliente A sincroniza exitosamente los datos con el Sitio Maestro.	R1= Juan	R1= Juan (Online)	R1 = Pedro (Offline)	R1 = Juan
5	Cliente B envía una solicitud de sincronización de datos con el Sitio Maestro. Como la versión anterior del cliente B (R1=Carlos) no coincide con la versión actual de la tabla de conflictos (R1=Juan) se	R1 = Juan	R1= Juan (Offline)	R1 = Pedro (Online)	R1 = Juan

	detecta un conflicto de actualización.				
6	El servidor de sincronización resuelve el conflicto con el método de resolución definido, en este caso (Ultima marca de tiempo) como T3>T2 gana el registro modificado por el cliente B.	R1 = Pedro	R1= Juan (Offline)	R1 = Pedro (Online)	R1 = Pedro

Tabla 16. Proceso de un conflicto de actualización.

Como se puede apreciar en la Tabla No. 16, la adaptación cRhoSync detecta el conflicto de actualización gracias a la tabla para manejo de conflictos, una vez es detectado, éste se resuelve por el método definido, en este caso (Ultima marca de tiempo). Los cambios son transmitidos al servidor empresarial y al cliente móvil.

5.4.3. Conflictos de Eliminación:

Debido a que los conflictos de eliminación no se pueden resolver, se implementó un método para evitar conflictos llamado borrado suave de registros, tal como se mencionó en el capítulo de lineamientos en la sección 3.8.3.2. Cuando un conflicto de eliminación es detectado se notifica al cliente y se registra en la bitácora de errores implementada en esta adaptación.

5.5. PRUEBAS DE DESEMPEÑO PROTOTIPO

A continuación se presenta el proceso de pruebas realizadas al piloto y los resultados obtenidos, que permiten determinar el rendimiento de una aplicación desarrollada con cRhoSync en términos de tiempos de respuesta y ancho de banda consumido.

Las pruebas se realizaron con dos objetivos principales:

- Demostrar que el proceso de adaptación del Framework de sincronización, no influyó drásticamente en los tiempos de respuesta de un servicio de sincronización desarrollado con cRhosync.
- Realizar un análisis comparativo del ancho de banda consumido, entre la representación con XML y Json para los datos intercambiados entre el servidor de sincronización y el Back-End.

Para cumplir con los objetivos anteriores, se ejecutaron dos pruebas, la primera consta de un proceso de carga al servidor de sincronización desde unos clientes simulados con JMeter, comparando cRhosync y Rhosync; y la segunda prueba, compara el ancho de banda utilizado por la conexión entre un servicio de

sincronización cRhoSync y el Back-End, haciendo una comparación en la representación de los datos usando xml y json.

5.5.1. Especificaciones técnicas

Los resultados de las pruebas de rendimiento están condicionados por las prestaciones de los equipos donde se ejecuta el servidor de sincronización y el servidor empresarial (Back-End). En este caso, los equipos empleados para desplegar el plan de pruebas cuentan con las siguientes especificaciones técnicas, ilustradas en la Tabla No. 17.

Servidor de Sincronización.	
Microprocesador	Intel Core i5 2.67GHz
Memoria RAM	4.0 GB
Disco duro	500 GB
Sistema Operativo	Windows 7 Ultimate.

Servidor de Back-End.	
Microprocesador	Intel Core i7 2.67GHz
Memoria RAM	8.0 GB
Disco duro	500 GB
Sistema Operativo	Windows 7.

Tabla 17. Especifiaciones técnicas de los equipos de pruebas

5.5.2. Prueba 1 – Carga al servidor de sincronización.

5.5.2.1. Proceso de pruebas.

Para realizar las pruebas de carga, se utilizó un plug-in desarrollado con el Framework Junit para generar el tráfico de los clientes móviles en JMeter. Las peticiones de los clientes son solicitudes de consulta (query), debido a que en esta operación se intercambia la mayor cantidad de datos. Para la realización de todas las pruebas se usara la misma cantidad de datos alojados en la base central.

Como se ilustra en la Figura No. 43, cada prueba consta de dos componentes principales, en el primero se realiza la autenticación de los clientes para obtener las credenciales necesarias en las operaciones con el servidor y en segunda instancia se realizan las solicitudes de sincronización con la operación consulta (query).

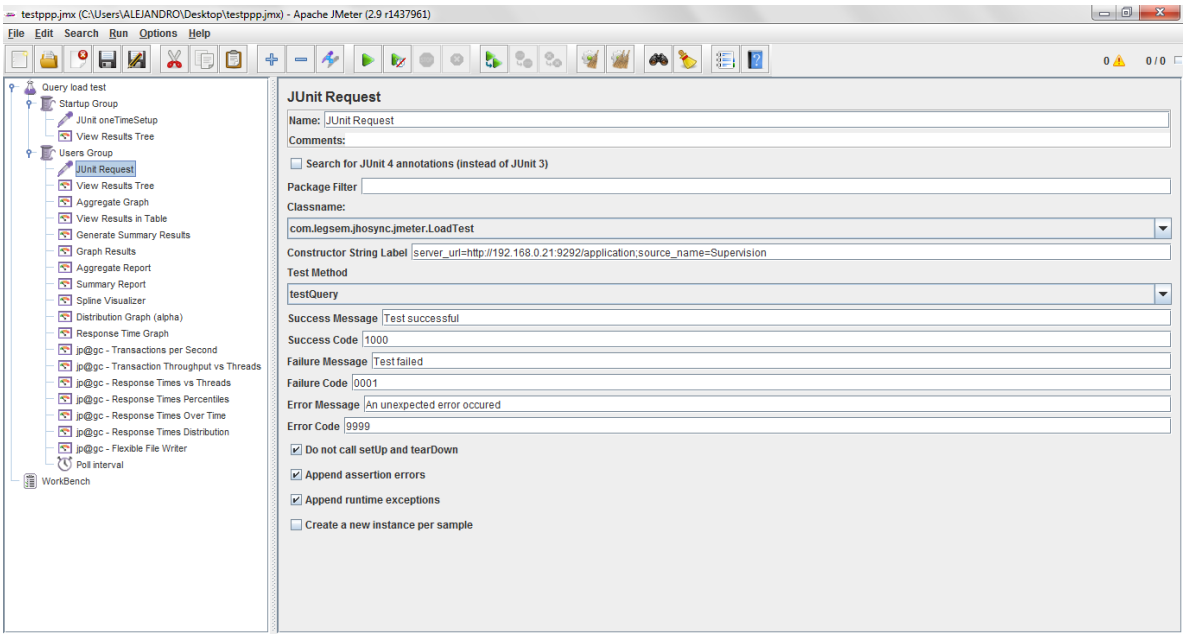


Figura No 43. Configuración Jmeter, prueba de rendimiento.

5.5.2.2. Escenarios de pruebas

Se contemplan dos escenarios de pruebas mencionados a continuación:

Escenario1: Conformado por el Framework RhoSync v 2.1.17 sin adaptaciones.
Escenario2: Conformado por el Framework adaptado cRhoSync.

Contextos:

En un escenario de sincronización de datos móviles, donde los usuarios presentan desconexiones frecuentes a la red, se recomienda definir el intervalo de sondeo con el cual el cliente consulta los datos del servidor empresarial. Para ésto se deben tener dos consideraciones: En primer lugar el intervalo de sondeo no debe saturar la red, ni el servidor de sincronización con solicitudes; en segundo lugar, debe garantizar una convergencia en los datos y una latencia mínima. En el caso de estudio se considera un intervalo de sondeo de cinco minutos, valor por defecto del servidor de sincronización RhoSync, con el que se obtuvieron buenos resultados.

Para cada contexto se realizaron tres repeticiones, los resultados consignados a continuación

Contexto1: 250 usuarios.
Contexto2: 500 usuarios.
Contexto3: 1000 usuarios.

5.5.2.3. Resultados obtenidos

Los tiempos de repuesta obtenidos en las pruebas, se resumen en las tablas presentadas a continuación, que consignan la información específica para cada contexto y escenario.

Los valores de media, mínimo, máximo, desviación estándar son valores en ms (milisegundos) y el rendimiento representa las peticiones por segundo.

Resultados para el Escenario 1.

No. Prueba	No. De muestras	Media	Mínimo	Máximo	Desviación estándar.	Rendimiento
1	250	58	35	168	18,06215934	0,836355364
2	250	58	36	417	28,21839088	0,836260244
3	250	58	35	163	19,45749213	0,836341374
PROMEDIO	250	58	35	249	21,91268078	0,836318994

Tabla 18. Resumen contexto 1, Escenario 1.

No. Prueba	No. de muestras	Media	Mínimo	Máximo	Desviación estándar.	Rendimiento
1	492	59	29	415	29,302394	1,64275984
2	492	60	29	415	25,0664125	1,38544522
3	492	64	31	148	17,0863262	1,64374774
PROMEDIO	492	61	30	326	23,8183776	1,5573176

Tabla 19. Resumen contexto 2, Escenario 1.

No. Prueba	No. de muestras	Media	Mínimo	Máximo	Desviación estándar.	Rendimiento
1	961	68	29	250	31,2010728	3,20326927
2	968	68	29	223	22,0609196	3,22817315
3	964	68	29	251	32,5842182	3,21817132
PROMEDIO	964	68	29	241	28,6154035	3,21653791

Tabla 20. Resumen contexto 3, Escenario 1.

Resultados para el Escenario 2.

No. Prueba	No. de muestras	Media	Mínimo	Máximo	Desviación estándar.	Rendimiento
1	250	73	41	234	23,2412478	0,83600297
2	250	72	54	155	16,4646258	0,8361204
3	250	73	47	313	23,7327322	0,83651768
PROMEDIO	250	73	47	234	21,1462019	0,83621368

Tabla 21. Resumen contexto 1, Escenario 2.

No. Prueba	No. de muestras	Media	Mínimo	Máximo	Desviación estándar.	Rendimiento
1	490	78	38	390	30,09438498	1,635295688
2	490	77	33	1579	75,07286312	1,636207605
3	490	76	32	405	29,86213498	1,63422126
PROMEDIO	490	77	34	791	45,00979436	1,635241518

Tabla 22. Resumen contexto 2, Escenario 2.

No. Prueba	No. de muestras	Media	Mínimo	Máximo	Desviación estándar.	Rendimiento
1	950	93	31	763	67,76188834	3,16682501
2	965	71	32	458	27,84143178	3,21636647
3	957	81	32	988	86,27088991	3,18975545
PROMEDIO	957	82	32	736	60,62473668	3,19098231

Tabla 23. Resumen contexto 3, Escenario 2.

Una observación importante que se debe hacer de los resultados consignados en las tablas para ambos escenarios, es que el rendimiento es similar para los dos casos, esto significa que la cantidad de peticiones por segundo que puede atender el servidor de sincronización desarrollado con cRhosync, es similar a la que puede atender un servicio desarrollado con Rhosync v. 2.1.17; incluso para el caso del contexto 2, se tiene un mejor resultado de rendimiento con cRhosync.

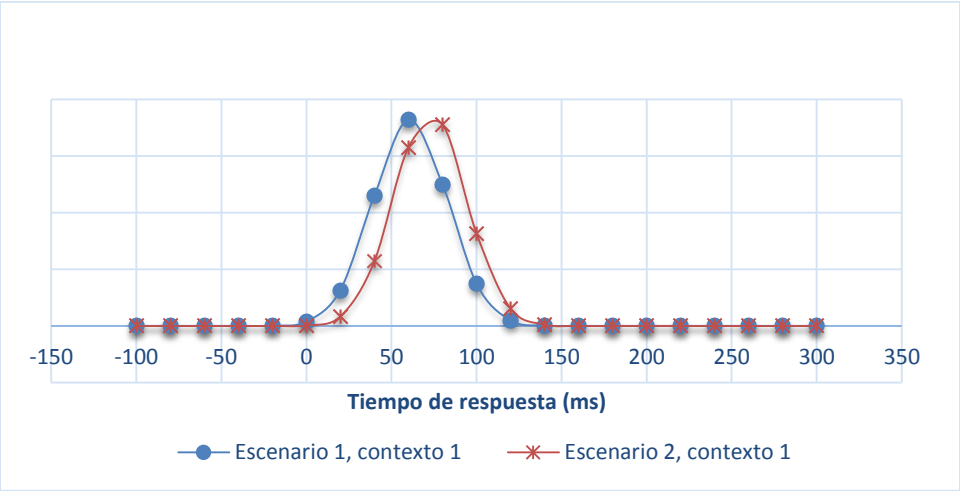


Figura No 44. Distribución de probabilidad de tiempos de respuesta, contexto 1.

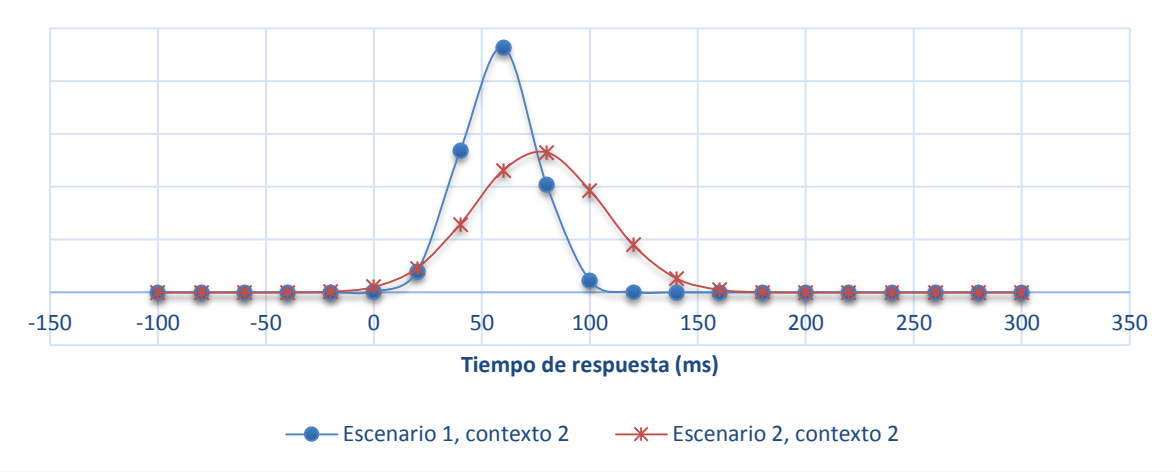


Figura No 45. Distribución de probabilidad de tiempos de respuesta, contexto 2.

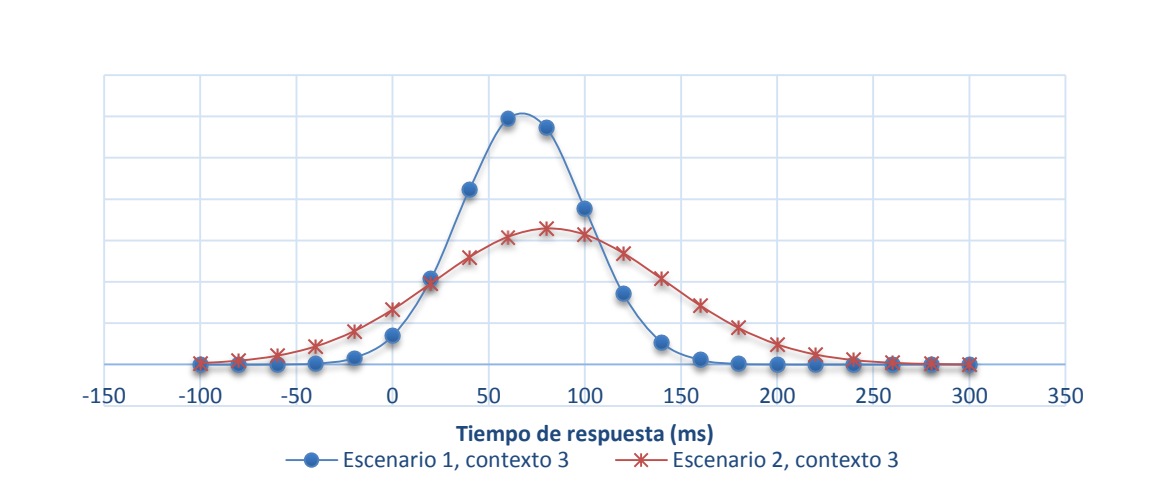


Figura No 46. Distribución de probabilidad de tiempos de respuesta, contexto 3.

De las tres gráficas anteriores se puede observar, que existe una relación directamente proporcional entre el número de usuarios y los tiempos de respuesta, a medida que el número de usuarios aumenta se presenta una mayor probabilidad de que sean más largos con cRhosync. Esto se debe principalmente al proceso que se hace en una consulta para que los datos lleven la marca de tiempo almacenada en la base de datos para el tratamiento de conflictos.

Otra tendencia en las gráficas y que también se puede observar en las tablas, es el mayor aumento en la desviación estándar para el caso de cRhosync (Escenario 2); esto significa que los tiempos de respuesta están más dispersos y que tienen probabilidades de ser más grandes.

Sin embargo las diferencias no son tan altas, ya que la mayor se presenta entre las medias del Escenario 2 (Promedios), con un valor de 16 ms, y haciendo una comparación con el valor obtenido en los servicios de sincronización, específicamente con el tratamiento de conflictos, éste representa una buena relación costo-beneficio.

5.5.3. Prueba 2 – Ancho de banda.

5.5.3.1. Proceso de pruebas.

Para realizar las pruebas del ancho de banda usado por los datos representados con Json y XML, se utilizó JMeter para consumir el servico Rest del Back-End; el equipo usado para esta tarea, fue el mismo donde se desplegó el servidor de sincronización. Al igual que en la prueba anterior, se usó el método de consulta, por representar el mayor volumen de datos intercambiados.

5.5.3.2. Escenarios de pruebas

Se contemplan dos escenarios de pruebas mencionados a continuación:

Escenario1: Consultas al servicio Rest del Servidor Central, usando XML para la representación de los datos.
Escenario2: Consultas al servicio Rest del Servidor Central, usando Json para la representación de los datos.

Para cada escenario se realizaron tres repeticiones, con 120 usuarios a intervalo de medio segundo entre peticiones y sin repeticiones en las consultas.

5.5.3.3. Resultados obtenidos

Resultados para el escenario 1.

	Ancho de Banda (Kb/s)	Tamaño de la respuesta (bytes)
prueba 1	10,76498745	5473
prueba 2	10,76932562	5473
prueba 3	10,77077246	5473
promedio	10,76836184	5473

Tabla 24. Resumen de resultados para consultas con xml.

Resultados para el escenario 2.

	Ancho de Banda (Kb/s)	Tamaño de la respuesta (bytes)
prueba 1	7,035894675	3575
prueba 2	7,036249181	3575
prueba 3	7,036485539	3575
promedio	7,036209798	3575

Tabla 25. Resumen de resultados para consultas con Json.

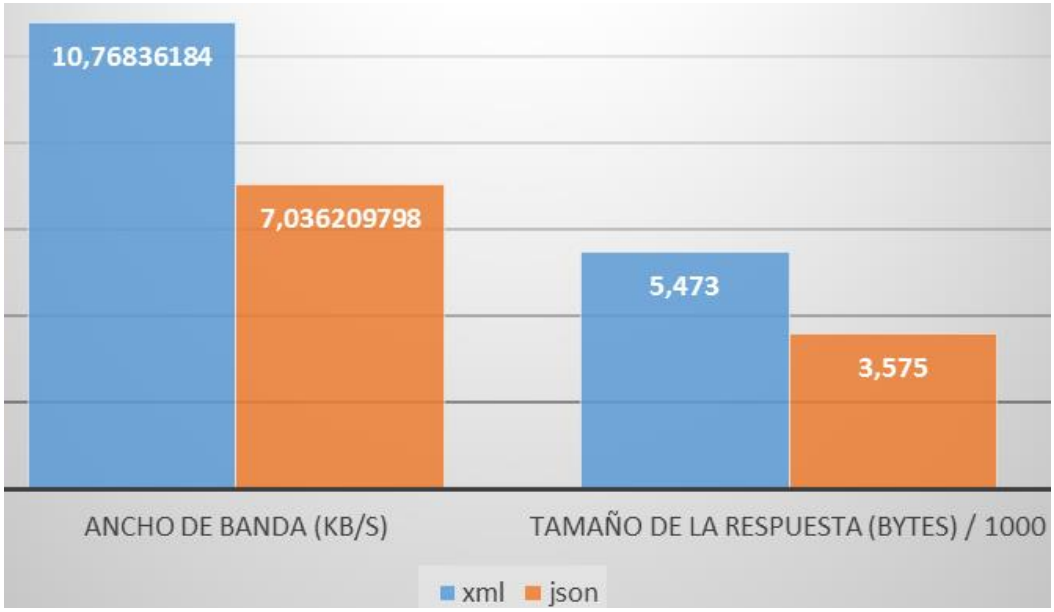


Figura No 47. Resumen prueba ancho de banda.

Las Tablas 24 y 25, y la Figura No. 47, resumen los resultados de las pruebas descritas en esta sección, de donde se observa la ventaja de usar json, ya que presenta un menor consumo de ancho de banda. De los datos consignados en las tablas, se puede calcular que tanto el tamaño de las respuestas y el ancho de banda consumido con Json, es aproximadamente un 34,7% menor que el representado por los datos en xml.

6. CONCLUSIONES Y TRABAJO FUTURO

6.1. APORTES

Entre las principales contribuciones de este trabajo de grado se destacan las siguientes:

- Base de conocimiento alrededor de los conceptos asociados a la sincronización de bases de datos móviles basada en servicios Web.
- Lineamientos para facilitar el desarrollo de aplicaciones de sincronización de bases de datos móviles basada en servicios Web bajo una orientación multiplataforma, apoyados en definiciones de empresas líderes en el sector las tecnologías de la información y las telecomunicaciones, y la retroalimentación obtenida en el desarrollo de la adaptación.
- Evaluación de funcionalidad y desempeño del Framework RhoSync para construcción de aplicaciones de sincronización de datos.
- Adaptacion de un Framework de código abierto y licencia GPLv3 para la sincronización de bases de datos móviles por medio de servicios Web, basado en los lineamientos planteados. Adaptación realizada en el Framework RhoSync llamada cRhoSync, disponible en GitHub.
- Mecanismos para evitar, identificar, resolver y registrar conflictos de datos, en aplicaciones de sincronización de bases de datos móviles basada en servicios Web bajo una orientación multiplataforma.
- Entorno de desarrollo para el Framework cRhoSync, diseñado con el objetivo de facilitar el desarrollo de aplicaciones de sincronización de bases de datos móviles basadas en servicios Web bajo una orientación multiplataforma (Disponible en GitHub).
- Evaluacion de la funcionalidad de los métodos para evitar, identificar y resolver conflictos propuestos e implementados en el trabajo.
- Piloto de una aplicación móvil de recolección de datos en campo, en el contexto de las actividades ejecutadas por la empresa UTEN Seccional Cauca.

6.2. PUBLICACIONES

Articulo “Sincronización de Bases de Datos Móviles Basada en Servicios Web Bajo una Aproximación Multiplataforma”. Publicado en: V Congreso Internacional de Computación y Telecomunicaciones COMTEL, 2013, Lima, Perú.

6.3. CONCLUSIONES

- Los lineamientos de sincronización definidos en este trabajo representan una guía para optimizar y facilitar la construcción de sistemas de sincronización de datos en ambientes móviles basados en servicios web, no sólo con el uso del framework cRhoSync, sino para la implementación de este tipo de servicios con otras características y haciendo uso de diversas tecnologías. Los lineamientos birndan una clara orientación en el proceso de desarrollo de este tipo de soluciones, fijando los requeriemientos y características clave con las que se deben contar y algunas recomendaciónes para implementarlas.
- En un sistema de sincronización de datos móviles que permita la manipulación de datos fuera de línea, es importante contar con mecanismos de detección y resolución de conflictos de datos, con el objetivo de evitar inconsistencias y pérdidas de información. En el desarrollo de este trabajo, se evidenció que la implementación de este tipo de mecanismos, garantiza un tratamiento de información confiable y consistente.
- La configuración de los métodos de resolución de conflictos de datos, depende directamente de las características particulares de cada entorno de sincronización; por esta razón en el presente trabajo, se definen diferentes métodos para la resolución de estos, de tal manera que un usuario administrador pueda definir el proceso de tratamiento en cada escenario o desarrollar un mecanismo propio que se adapte a las condiciones particulares.
- El uso de un mediador en una arquitectura de sincronización de datos móviles, facilita el despliegue de este tipo de soluciones, ya que permite separar la lógica de negocio, de la lógica de sincronización; además posibilita la implementación de procesos específicos, como la gestión de usuarios, conexiones, el tratamiento de conflictos, que posibilita tener clientes más ligeros.
- El uso de servicios Web Rest con datos en formato Json, optimiza el consumo de ancho de banda en la transferencia de información, reduce los tiempos de respuesta y además el proceso de interpretación de los datos se vuelve menos complejo, por lo tanto, sistemas como Rhosync que hacen uso de estas tecnologías, son muy eficientes en estos aspectos, sobre todo si se trata de un ambiente con clientes móviles.
- Los sistemas de sincronización de bases de datos en ambientes móviles, plantean diferentes retos, en términos de seguridad de los datos, fragmentación de las plataformas móviles, heterogeneidad en las fuentes de datos, el acceso y los conflictos de datos en ambientes fuera de línea. En este sentido, es importante definir y seguir políticas como las planteadas en los lineamientos de este trabajo.

- En el piloto se pusieron a prueba los métodos de resolución de conflictos, obteniendo resultados muy satisfactorios, por la eficiencia y funcionalidad de los procesos que ejecutaron. Esto representa un valor importante para un sistema de sincronización, ya que se pueden manejar las inconsistencias de los registros, decidiendo que método se adapta a cada necesidad, con el fin de mantener siempre la integridad de la información.
- La configuración realizada en el piloto, se ejecutó de acuerdo a las necesidades de la empresa UTEN, incluso se obtuvo una realimentación al implementar un método de resolución que cumpliera con los requerimientos específicos de UTEN (Número de columnas válidas); y el sistema de sincronización se adaptó correctamente al contexto de las actividades de la empresa, cumpliendo totalmente con los requerimientos de las tareas de supervisión.
- Dado que el piloto se desarrolló y probó bajo un contexto real en el sector productivo, se puede afirmar que el resultado del trabajo de grado puede ser aplicado más allá de los objetivos académicos.

6.4. TRABAJO FUTURO

- Actualmente RhoSync, sólo brinda integración automática con aplicaciones cliente desarrolladas en RhoDes, sería interesante adaptar el Framework para soportar aplicaciones cliente nativas, o desarrolladas en otros Framework's multiplataforma.

- Por el momento, la Suite de Rhomobile cuenta con el módulo RhoGalery para tareas de des-aprovisionamiento de datos y aplicaciones, sin embargo se distribuye bajo licencia comercial. Por lo tanto sería interesante construir un mecanismo que permita esta funcionalidad en las aplicaciones cliente, teniendo en cuenta los requerimientos de seguridad de las aplicaciones empresariales.

- Uno de los puntos clave para la construcción de un sistema de sincronización móvil basado en servicios web, es tener en cuenta el entorno donde se va a desplegar, esto hace necesario considerar factores como el número de clientes móviles, los tipos de información que se recogen en campo, la necesidad o no de oportunidad de los registros, posibles conflictos de datos, entre otros. Por esta razón, sería de mucha ayuda la construcción de una herramienta que permita parametrizar un entorno de sincronización, y obtener así información que ayude en una implementación adecuada del servicio de sincronización.

7. REFERENCIAS

[1] M. Lennighan, «Number of phones exceeds population of world», 20-may-2011. [En línea]. Disponible en: http://www.totaltele.com/view.aspx?ID=464922. [Accedido: 13-mar-2012].

[2] A. B. Waluyo, B. Srinivasan, y D. Taniar, «Research in mobile database query optimization and processing», *Mobile Information Systems*, vol. 1, n.º 4, pp. 225-252, ene. 2005.

[3] Mi-Seon Choi, Young-Kuk Kim, y Juno Chang, «Transaction-centric split synchronization mechanism for mobile e-business applications», en *International Workshop on Data Engineering Issues in E-Commerce, 2005. Proceedings*, 2005, pp. 112- 118.

[4] Y. Natchetoi, V. Kaufman, y A. Shapiro, «Service-oriented architecture for mobile applications», en *Proceedings of the 1st international workshop on Software architectures and mobility*, New York, NY, USA, 2008, pp. 27–32.

[5] *E-business 2.0: Roadmap for Success*. Addison-Wesley Professional, 2001.

[6] S. Mall, T. Stefanov, y S. Stadelman, *Mobilizing Your Enterprise with SAP*. SAP Press, 2012.

[7] H. H., Eric Lai, Lori Jo Piquet y Sybase, *Enterprise Mobility Guide 2011*. Sybase.

[8] *Yankee Guide to Successfully Deploying Enterprise Mobile Applications pdf free ebook download*. .

[9] C. Kleisath y O. Rege, *The Business Value Roadmap to Mobilized Software Solutions*. .

[10] R. C. Basole, «The Emergence of the Mobile Enterprise: A Value-Driven Perspective», en *Management of Mobile Business, 2007. ICMB 2007. International Conference on the*, 2007, pp. 41-41.

[11] «The Global Information Technology Report 2013», *The World Economic Forum*. [En línea]. Disponible en: http://reports.weforum.org/global-information-technology-report-2013/. [Accedido: 20-abr-2013].

[12] Beñat Bilbao-Osorio, Soumitra Dutta, y Bruno Lanvin, «Insight Report The Global Information Technology Report 2013 Growth and Jobs in a Hyperconnected World», WORLD ECONOMIC FORUM, 2013.

[13] Byung-Yun Lee, Tae-Wan Kim, Dae-Woong Kim, y Hoon Choi, «Data synchronization protocol in mobile computing environment using yncML», en *High Speed Networks and Multimedia Communications 5th IEEE International Conference on*, 2002, pp. 133- 137.

[14] M. Satyanarayanan, «Accessing information on demand at any location. Mobile information access», *IEEE Personal Communications*, vol. 3, n.º 1, pp. 26-33, feb. 1996.

[15] T. Imielinski y B. R. Badrinath, «Mobile wireless computing: challenges in data management», *Commun. ACM*, vol. 37, n.º 10, pp. 18–28, oct. 1994.

[16] S. K. Madria y S. S. Bhowdrick, «Mobile data management», *IEEE Potentials*, vol. 20, n.º 4, pp. 11-15, 2001.

[17] E. Jamil, «SOA in Asynchronous Many-to-One Heterogeneous Bi-Directional Data Synchronization for Mission Critical Applications», *WeDoWebSphere, Jul*, 2009.

[18] T. E. Starner, «Powerful change part 1: batteries and possible alternatives for the mobile market», *IEEE Pervasive Computing*, vol. 2, n.º 4, pp. 86- 88, dic. 2003.
[19] N. Cohen, «A Java Framework for Mobile Data Synchronization», en *Cooperative Information Systems*, vol. 1901, P. Scheuermann y O. Etzion, Eds. Springer Berlin / Heidelberg, 2000, pp. 287-298.
[20] Mi-Young Choi, Eun-Ae Cho, Dae-Ha Park, Jong-Youn Bae, Chang-Joo Moon, y Doo-Kwon Baik, «A Synchronization Algorithm of Mobile Database for Ubiquitous Computing», en *Fifth International Joint Conference on INC, IMS and IDC, 2009. NCM '09*, 2009, pp. 416-419.
[21] A. Laganà, *Computational science and its applications: ICCSA 2004, international conference, Assisi, Italy, May 14-17, 2004 : proceedings*. Springer, 2004.
[22] M.-S. Choi y Young-Guk Kim, «Introduction of mobile database and research status», *Journal of Database Research*, vol. 17, n.º 3, pp. 3-16.
[23] Budiarto, S. Nishio, y M. Tsukamoto, «Data management issues in mobile and peer-to-peer environments», *Data & Knowledge Engineering*, vol. 41, n.º 2-3, pp. 183-204, jun. 2002.
[24] D. Barbara, «Mobile computing and databases-a survey», *IEEE Transactions on Knowledge and Data Engineering*, vol. 11, n.º 1, pp. 108-117, 1999.
[25] U. Hansmann, R. M. Mettala, A. Purakayastha, y P. Thompson, *SyncML: synchronizing and managing your mobile data*. Prentice Hall Professional, 2003.
[26] W. Jones, «Personal Information Management», *Annual Review of Information Science and Technology*, vol. 41, n.º 1, pp. 453-504, ene. 2007.
[27] S. Allen, V. Graupera, y L. Lundrigan, *Pro Smartphone Cross-Platform Development: IPhone, Blackberry, Windows Mobile and Android Development and Distribution*. Apress, 2010.
[28] J. H. Abawajy, M. M. Deris, y M. Omar, «A novel data replication and management protocol for mobile computing systems», *Mobile Information Systems*, vol. 2, n.º 1, pp. 3-19, ene. 2006.
[29] Y. Saito y M. Shapiro, «Optimistic replication», *ACM Comput. Surv.*, vol. 37, n.º 1, pp. 42–81, mar. 2005.
[30] J.-L. Li y J. Li, «Data synchronization protocol in mobile computing environment using SyncML and Huffman coding», en *2012 International Conference on Wavelet Active Media Technology and Information Processing (ICWAMTIP)*, 2012, pp. 260-262.
[31] «Replicación de SQL Server». [En línea]. Disponible en: http://msdn.microsoft.com/es-es/library/ms151198.aspx. [Accedido: 18-abr-2013].
[32] N. Preguiça, M. Shapiro, y C. Matheson, «Semantics-Based Reconciliation for Collaborative and Mobile Environments», en *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, R. Meersman, Z. Tari, y D. C. Schmidt, Eds. Springer Berlin Heidelberg, 2003, pp. 38-55.

[33] J. Cho y H. Garcia-Molina, «Synchronizing a database to Improve Freshness», 1999. [En línea]. Disponible en: http://ilpubs.stanford.edu:8090/396/. [Accedido: 06-jul-2013].
[34] «Sincronizar datos». [En línea]. Disponible en: http://msdn.microsoft.com/es-es/library/ms151793.aspx. [Accedido: 18-abr-2013].
[35] M. Wiesmann y A. Schiper, «Comparison of database replication techniques based on total order broadcast», *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, n.º 4, pp. 551-566, 2005.
[36] E. Cecchet, G. Candea, y A. Ailamaki, «Middleware-based database replication: the gaps between theory and practice», en *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, New York, NY, USA, 2008, pp. 739–752.
[37] M. Wiesmann, F. Pedone, A. Schiper, B. Kemme, y G. Alonso, «Understanding replication in databases and distributed systems», en *20th International Conference on Distributed Computing Systems, 2000. Proceedings*, 2000, pp. 464-474.
[38] «Introduction to Advanced Replication». [En línea]. Disponible en: http://docs.oracle.com/cd/E14072_01/server.112/e10706/repoverview.htm#i19713. [Accedido: 18-abr-2013].
[39] N. Schiper, R. Schmidt, y F. Pedone, «Optimistic Algorithms for Partial Database Replication», en *Principles of Distributed Systems*, M. M. A. A. Shvartsman, Ed. Springer Berlin Heidelberg, 2006, pp. 81-93.
[40] R. G. Guy, G. J. Popek, T. W. Page, y Jr, «Consistency Algorithms for Optimistic Replication», en *In Proceedings of the First International Conference on Network Protocols. IEEE*, 1993.
[41] «Control de concurrencia en ADO.NET». [En línea]. Disponible en: http://msdn.microsoft.com/es-es/library/22hhd212(v=vs.80).aspx. [Accedido: 19-abr-2013].
[42] «Introducción a la concurrencia de datos en ADO.NET». [En línea]. Disponible en: http://msdn.microsoft.com/es-es/library/cs6hb8k4(v=vs.80).aspx. [Accedido: 03-abr-2013].
[43] «Conflict Resolution Concepts and Architecture». [En línea]. Disponible en: http://docs.oracle.com/cd/E11882_01/server.112/e10706/repconflicts.htm#i26888. [Accedido: 03-abr-2013].
[44] «Master-Master Replication». [En línea]. Disponible en: http://msdn.microsoft.com/en-us/library/ff649910.aspx. [Accedido: 19-abr-2013].
[45] F. D. Munoz-Escoi, J. Pla-Civera, M. I. Ruiz-Fuertes, L. Irun-Briz, H. Decker, J. E. Armendariz-Inigo, y J. R. G. de Mendivil, «Managing Transaction Conflicts in Middleware-based Database Replication Architectures», en *25th IEEE Symposium on Reliable Distributed Systems, 2006. SRDS '06*, 2006, pp. 401-420.
[46] Jesús López Méndez, «Gestión de concurrencia en ADO.NET en DNM+». [En línea]. Disponible en: http://www.dnmplus.net/articulos/gestion-de-concurrencia-en-ado-net.aspx. [Accedido: 19-abr-2013].

[47] «DB2 Universal Database». [En línea]. Disponible en: http://publib.boulder.ibm.com/infocenter/db2luw/v8/index.jsp?topic=/com.ibm.db2.ii.doc/admin/cqrsbo10.htm. [Accedido: 19-abr-2013].
[48] Joseph Debuzna - Product Manager, Oracle y Bharath Aleti - Software Development Manager, Oracle, «Best Practices for Conflict Detection and Resolution in Oracle GoldenGate for Active/Active», presentado en Oracle OpenWorld.
[49] F. D. Borda Luciani, E. O. Gagliardi, y G. Hernández Peñalver, «Sincronización de datos entre aplicaciones sobre redes móviles», presentado en VIII Workshop de Investigadores en Ciencias de la Computación, 2006.
[50] John Garmany y Robert Freeman, «Multi-Master Replication Conflict Avoidance and Resolution». Rampant TechPress., 2004.
[51] Chris Kleisath y Ojas Rege, «The Developer's Roadmap», en *MOBILIZED SOFTWARE SOLUTIONS*, .
[52] W. Jones, «Personal Information Management», *Annual Review of Information Science and Technology*, vol. 41, n.º 1, pp. 453–504, 2007.
[53] X. L. Dong y A. Halevy, «A platform for personal information management and integration», en *Proceedings of VLDB 2005 PhD Workshop*, 2005, p. 26.
[54] S. Dumais, E. Cutrell, J. Cadiz, G. Jancke, R. Sarin, y D. C. Robbins, «Stuff I've seen: a system for personal information retrieval and re-use», en *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, New York, NY, USA, 2003, pp. 72–79.
[55] W. Kurschl, S. Mitsch, y R. Prokop, «SmartDOTS - A Framework for Efficient Data Synchronization on Mobile Devices», en *Third International Conference on Information Technology: New Generations, 2006. ITNG 2006*, 2006, pp. 300-305.
[56] P. D. Fernández y L. A. G. Moreno, «Evolución de las Bases de Datos: de Fijas a Móviles».
[57] Open Mobile Alliance, *SyncML Representation Protocol*, 1.0.1 ed. 2001.
[58] H. Mei y J. Lukkien, «A remote personal device management framework based on SyncML DM specifications», en *Proceedings of the 6th international conference on Mobile data management*, New York, NY, USA, 2005, pp. 185–191.
[59] Christensen, E., Curbera, F., Meredith, G., y Weerawarana, S., «Web Service Definition Language (WSDL)». [En línea]. Disponible en: http://www.w3.org/TR/wsdl. [Accedido: 26-mar-2013].
[60] Ejaz Jamil, «Emerging SOA Appliances». .
[61] A. Nalwaya, *Rhomobile Beginner's Guide*. Packt Publishing Ltd, 2011.
[62] S. Allen, V. Graupera, y L. Lundrigan, *Pro Smartphone Cross-Platform Development: iPhone, Blackberry, Windows Mobile and Android Development and Distribution*, 1.ª ed. Apress, 2010.
[63] «Rhomobile | RhoSync Introduction». [En línea]. Disponible en: http://docs.rhomobile.com/rhosync/introduction. [Accedido: 25-mar-2013].
[64] Michael Vizard, «Finding the Middle Ground for Mobile Computing | Blogs | ITBusinessEdge.com». [En línea]. Disponible en:

http://www.itbusinessedge.com/cm/blogs/vizard/finding-the-middle-ground-for-mobile-computing/?cs=46302. [Accedido: 25-mar-2013].

[65] «RhoConnect - Motorola Solutions USA». [En línea]. Disponible en: http://www.motorola.com/Business/US-EN/Business+Product+and+Services/Software+and+Applications/RhoMobile+Suite/RhoConnect. [Accedido: 25-mar-2013].

[66] Motorola Solutions, «Develop Applications for the Next Generation of Business Mobility with RhoMobile Suite from Motorola Solutions | Reuters». [En línea]. Disponible en: http://www.reuters.com/article/2012/05/31/idUS118022+31-May-2012+BW20120531. [Accedido: 25-mar-2013].

[67] «How RhoSync Makes Smartphone App Integration Easier - Rhomobile». [En línea]. Disponible en: http://www.rhomobile.com/blog/rhosync-for-easy-smartphone-app-integration/. [Accedido: 25-mar-2013].

[68] «Microsoft Sync Framework». [En línea]. Disponible en: http://msdn.microsoft.com/es-es/library/bb902854.aspx. [Accedido: 25-mar-2013].

[69] Krishnaa G, «Light Weight Architecture for Remote DB Synchronization on Windows Mobile Devices using Web Services», *SRM University*, may 2008.

[70] M Butricio, N Cohen, J Givler, A Mohindra, A Purakayastha, D Shea, J Cheng, D Clare, G Fisher, R Scott, Y Sun, M Wone, y Q Zondervan, «Enterprise data access from mobile computers: an and to end story.», presentado en Workshop on research issues in data Eng, San Diego, California, 2000, pp. 9-16.

[71] H. Larkin, «Applying Concurrent Versioning to Serverless Mobile Device Synchronisation», en *6th IEEE/ACIS International Conference on Computer and Information Science, 2007. ICIS 2007*, 2007, pp. 157-162.

[72] A. Ashraf, P. Dominic, D. Durai, A. Azween, y I. Hamidah, «A New Optimistic Replication Strategy for Large-scale Mobile Distributed Database Systems», *International Journal of database management systems*, vol. 2, n.º 4, pp. 86–105, 2010.

[73] M. Jin, X. Zhou, J. Zhou, X. Gao, y C. Gong, «Strategy of Conflict Preprocessing and Reconciliation for Mobile Databases», en *Wireless Communications, Networking and Mobile Computing, 2008. WiCOM'08. 4th International Conference on*, 2008, pp. 1–6.

[74] S. C. Choi y H. Y. Youn, «Dynamic hybrid replication effectively combining tree and grid topology», *J. Supercomput.*, vol. 59, n.º 3, pp. 1289–1311, mar. 2012.

[75] Y. Xia y A. (Sumi) Helal, «A dynamic data/currency protocol for mobile database design and reconfiguration», en *Proceedings of the 2003 ACM symposium on Applied computing*, New York, NY, USA, 2003, pp. 550–556.

[76] D. R. Selvarani y T. N. Ravi, «A survey on data and transaction management in mobile databases», *arXiv:1211.5418*, nov. 2012.

[77] Z. Ding, X. Meng, y S. Wang, «A novel conflict detection and resolution strategy based on TLRSP in replicated mobile database systems», en *Seventh International Conference on Database Systems for Advanced Applications, 2001. Proceedings*, 2001, pp. 234-240.

[78] Y. Saito y M. Shapiro, «Replication: Optimistic Approaches», 2002.
[79] V. . Zuikevi\vciūt\.e y F. Pedone, «Conflict-aware load-balancing techniques for database replication», en *Proceedings of the 2008 ACM symposium on Applied computing*, New York, NY, USA, 2008, pp. 2169–2173.
[80] P. Castro, F. Giraud, R. Konuru, A. Purakayastha, y D. Yeh, «A Programming Framework for Mobilizing Enterprise Applications», en *Proceedings of the Sixth IEEE Workshop on Mobile Computing Systems and Applications*, Washington, DC, USA, 2004, pp. 196–205.
[81] «History | Oracle Corporation». [En línea]. Disponible en: http://www.oracle.com/us/corporate/history/index.html. [Accedido: 01-may-2013].
[82] «Oracle Database 11g Features: Data Replication & Integration». [En línea]. Disponible en: http://www.oracle.com/technetwork/database/features/data-integration/index.html. [Accedido: 30-abr-2013].
[83] «About Oracle | Company Information | Oracle». [En línea]. Disponible en: http://www.oracle.com/us/corporate/index.html. [Accedido: 01-may-2013].
[84] «About Microsoft: Your Potential. Our Passion.» [En línea]. Disponible en: http://www.microsoft.com/about/en/us/default.aspx. [Accedido: 02-may-2013].
[85] «Our Commitment to Our Customers». [En línea]. Disponible en: http://www.microsoft.com/about/companyinformation/ourbusinesses/en/us/business.aspx. [Accedido: 02-may-2013].
[86] E. J. Braude, *Software engineering: an object-oriented perspective*. John Wiley & Sons, Inc., 2000.
[87] L. Bass, P. Clements, y R. Kazman, *Software architecture in practice*. Addison-Wesley Professional, 2012.
[88] A. Rollings y D. Morris, «Game architecture and design: a new edition», 2003.
[89] Ted Burroughs y Randy Urbano, *Oracle9i Advanced Replication, Release 2 (9.2)*. 2002.
[90] J. Holliday, D. Agrawal, y A. E. Abbadi, «Disconnection Modes for Mobile Databases», *Wireless Networks*, vol. 8, n.º 4, pp. 391-402, jul. 2002.
[91] H. Hu, W. Guo, B. Zhang, y X. Chen, «A method of security measurement of the network data transmission», en *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*, 2005, p. 8 pp.-.
[92] «RFC 5246 - The Transport Layer Security (TLS) Protocol Version 1.2». [En línea]. Disponible en: http://tools.ietf.org/html/rfc5246. [Accedido: 28-may-2013].
[93] W. Stallings, *Fundamentos de seguridad en redes: aplicaciones y estándares*. Pearson Educación, 2003.
[94] T. Chomsiri, «HTTPS Hacking Protection», en *21st International Conference on Advanced Information Networking and Applications Workshops, 2007, AINAW '07*, 2007, vol. 1, pp. 590-594.
[95] «HTTPS | Electronic Frontier Foundation». [En línea]. Disponible en: https://www.eff.org/pages/https. [Accedido: 28-may-2013].
[96] E. Rescorla, «HTTP Over TLS». [En línea]. Disponible en: http://tools.ietf.org/html/rfc2818#page-2. [Accedido: 02-abr-2013].

[97] R. T. Fielding, «Architectural styles and the design of network-based software architectures», University of California, 2000.
[98] L. Zhang, «RESTful Web Services», en *Web Services, Architecture Seminar, University of Helsink, Department of Computer Science*, 2004.
[99] X. Feng, J. Shen, y Y. Fan, «REST: An alternative to RPC for Web services architecture», en *First International Conference on Future Information Networks, 2009. ICFIN 2009*, 2009, pp. 7-10.
[100] G. Mulligan y D. Gracanin, «A comparison of SOAP and REST implementations of a service based interaction independence middleware framework», en *Simulation Conference (WSC), Proceedings of the 2009 Winter*, 2009, pp. 1423-1432.
[101] T. Aihkisalo y T. Paaso, «Latencies of Service Invocation and Processing of the REST and SOAP Web Service Interfaces», en *2012 IEEE Eighth World Congress on Services (SERVICES)*, 2012, pp. 100-107.
[102] P. Kumar, S. Ahuja, K. Umapathy, y Z. Prodanoff, «Comparing Performance of Web Service Interaction Styles: SOAP vs. REST», *Journal of Information Systems Applied Research*, vol. 6, n.º 1, p. 4, 2013.
[103] T. Erl, *Service-oriented architecture*. Prentice Hall Englewood Cliffs, 2004.
[104] D. C. Rajapakse, «Fragmentation of Mobile Applications», *MobileFragmentation. info*, 2008.
[105] J. NARANJO SANHERMELANDO, «Entornos de desarrollo multiplataforma para clientes móviles de aprendizaje ubicuo», 2012.
[106] J Berlana Hernández, «xAPP : XML Application Platform : framework de desarrollo multiplataforma basado en la nube», Universidad Carlos III de Madrid, 2012.
[107] «MIL-STD-810G DOD Test Method Standard: Environmental Engineering Considerations and Laboratory Tests - std810g.pdf». .
[108] «Dispositivos robustos o rugerizados para uso en trabajos de campo | Quadralia, S.L.» [En línea]. Disponible en: http://www.quadralia.com/en/dispositivos-robustos-o-rugerizados/#.UVttolcZHbN. [Accedido: 02-abr-2013].
[109] J. P. Franco y M. A. S. López, *Criptografía digital: fundamentos y aplicaciones*. Universidad de Zaragoza, 1998.
[110] P. C. Gil, *Introducción a la Criptografía*. Ra-ma, 1996.
[111] M. Palmieri, I. Singh, y A. Cicchetti, «Comparison of cross-platform mobile development tools», en *2012 16th International Conference on Intelligence in Next Generation Networks (ICIN)*, 2012, pp. 179-186.
[112] P. Smutny, «Mobile development tools and cross-platform solutions», en *Carpathian Control Conference (ICCC), 2012 13th International*, 2012, pp. 653-656.
[113] Adam M. Christ, «Bridging the Mobile App Gap», *Sigma Journal: Inside the Digital Ecosystem*, vol. 11, n.º 1, pp. 27-32, oct. 2011.
[114] Robert Smith, «Whitepaper: Native vs HTML5 Mobile App Development», 22-ago-2012.
[115] Sudheer Raju, «10 of Best Cross-Platform Mobile Development Tools - Tools Journal», 12-mar-2011. [En línea]. Disponible en:

http://www.toolsjournal.com/item/157-10-of-best-cross-platform-mobile-development-tools. [Accedido: 25-mar-2013].
[116] «Titanium Mobile Application Development | Appcelerator Inc.» [En línea]. Disponible en: http://www.appcelerator.com/platform/titanium-platform/. [Accedido: 25-mar-2013].
[117] Rhomobile, «Rhomobile | Rho::AsyncHttp API», *Rho::AsyncHttp API*. [En línea]. Disponible en: http://docs.rhomobile.com/rhodesapi/asynchttp-api. [Accedido: 06-may-2013].
[118] «Rhomobile | Rhodes». [En línea]. Disponible en: http://docs.rhomobile.com/rhodes/introduction. [Accedido: 25-mar-2013].
[119] Blake Mizerany, «Sinatra». [En línea]. Disponible en: http://www.sinatrarb.com/. [Accedido: 07-may-2013].