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

ANEXOS

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

Anexo A. Adaptación: Análisis y diseño del Software.

Este anexo aborda en forma general las fases de análisis y diseño del software, descrito con los diagramas de casos de uso, clases y secuencia. Los diagramas se muestran a continuación.

a. Diagrama de casos de uso de la adaptación cRhoSync

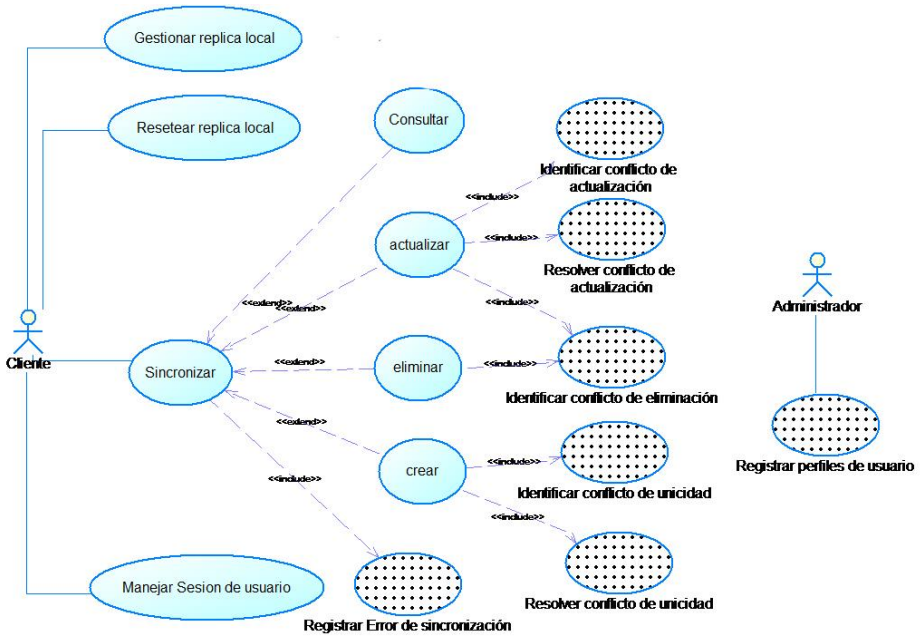

Figura 1. Diagrama de Casos de Uso de la Adaptación.

En el diagrama anterior podemos observar algunos casos de uso punteados, que representan las funcionalidades extra, agregadas para cumplir con los requerimientos de la adaptación.

b. Descripción detallada de los casos de uso esenciales

Nombre	Identificar Conflicto de Actualización
Iniciador	Caso de uso: Actualizar
Precondición	Petición de sincronización para una actualización.
Flujo Principal	a. El servidor de sincronización recibe los datos de una actualización para un registro específico. b. El sistema consulta en la base de datos de manejo de conflictos del servidor de sincronización, la fecha de la última modificación para este registro. c. Si la fecha de actualización que llega con la petición de actualización es diferente a la que tiene registrado el servidor, se pone en verdadera la bandera de identificación de conflicto de actualización.

Flujo Alternativo	c. Si la fecha de actualización que llega con la petición de actualización es igual a la que tiene registrado el servidor, se pone en falsa la bandera de identificación de conflicto de actualización.
Post Condiciones	- Si la bandera de identificación de este tipo de conflicto se puso en verdadera, se resolverá el conflicto según la configuración de resolución que tenga el usuario.
Excepciones	

Nombre	Resolver Conflicto de Actualización
Iniciador	Caso de uso: Actualizar.
Precondición	La bandera de conflicto de actualización tiene un valor verdadero.
Flujo Principal	a. El sistema verifica que configuración de resolución se tiene. b. El sistema ejecuta el proceso de resolución y se entrega una respuesta con un valor lógico. c. Si el valor lógico obtenido en el proceso de resolución es verdadero, indica que se debe enviar la actualización al Back-End o servidor central.
Flujo Alternativo	c. Si el valor lógico obtenido en el proceso de resolución es falso, indica que no se envía la actualización al Back-End o servidor central y además se debe informar al cliente, que se descartaran los cambios que se realizaron en ese registro.
Post Condiciones	- Terminado el proceso de resolución de un conflicto de actualización, las bases de datos del cliente que ha enviado la petición debe quedar convergente con los datos del registro central. - Si ha ocurrido una excepción en el proceso de resolución, se tendrá que registrar está en una bitácora o cola de error.
Excepciones	- Fallas en la conexión con el servicio de acceso a datos al servidor central. - Ningún método de resolución está configurado.

Nombre	Identificar Conflicto de Eliminación.
Iniciador	Caso de uso: Actualizar o eliminar.
Precondición	Petición de sincronización para una actualización o una eliminación.
Flujo Principal	a. El sistema consulta en la base de datos de manejo de conflictos del servidor de sincronización, para verificar si el identificador del registro a borrar o actualizar, existe. b. Si el identificador ya no existe, se pone en verdadera la bandera de identificación de conflicto de eliminación. c. Si el proceso es una eliminación simplemente se debe notificar al cliente que el registro ya había sido eliminado antes, si el proceso es una actualización se debe notificar al cliente y enviar una petición de eliminación de este registro hacia el cliente móvil.
Flujo Alternativo	b. Si el identificador existe, se deja la bandera de identificación de este conflicto en falsa, para que se continúe con el proceso de eliminación o actualización.

Post Condiciones	-Al finalizar el proceso eliminación, las bases de datos del cliente y servidor central, deben quedar convergentes. -Como los conflictos de eliminación no se pueden resolver, se registra esta acción en una bitácora o cola de error.
Excepciones	

Nombre	Identificar Conflicto de Unicidad.
Iniciador	Caso de uso: Crear
Precondición	Petición de sincronización para una creación.
Flujo Principal	a. El servidor de sincronización recibe los datos de una creación, para un registro específico al servidor de sincronización. b. El sistema consulta en la base de datos de manejo de conflictos del servidor de sincronización, para verificar si el identificador del nuevo registro ya está siendo usado. c. Si el identificador ya está siendo usado, se pone en verdadera la bandera de identificación de conflicto de unicidad.
Flujo Alternativo	c. Si el identificador del nuevo registro no está siendo usado, la bandera de identificación de conflicto sigue en su valor falso.
Post Condiciones	- Si la bandera de identificación de este tipo de conflicto se puso en verdadera, se resolverá el conflicto según la configuración de resolución que tenga el usuario.
Excepciones	

Nombre	Bitácora de Error.
Iniciador	Caso de uso: Sincronizar
Precondición	Que hayan ocurrido errores en la resolución de un conflicto.
Flujo Principal	a. El servidor de sincronización procesa los errores de la sincronización, y verifica los casos de conflictos que no pudieron ser resueltos. b. Se registran los errores identificados como conflictos no resueltos en la base de datos de manejo de conflictos.
Flujo Alternativo	
Post Condiciones	
Excepciones	

Nombre	Resolver Conflicto Unicidad
Iniciador	Caso de uso: crear
Precondición	La bandera de conflicto de unicidad tiene un valor lógico verdadero.
Flujo Principal	a. El servidor de sincronización verifica la configuración de resolución que se tiene. b. Una vez identificado el método de resolución que se tiene configurado, se ejecuta su proceso de resolución y se obtiene una respuesta con modificaciones en los datos de la petición y con un valor lógico, que indica si se debe seguir con el proceso de sincronización. c. Si el valor lógico obtenido en el proceso de resolución es verdadero, indica que se debe enviar la actualización al Back-End o servidor central con los datos modificados por método de resolución.
Flujo Alternativo	c. Si el valor lógico obtenido en el proceso de resolución es falso, indica que no se envía la actualización al Back-End o servidor central.
Post Condiciones	- Terminado el proceso de resolución de conflicto de unicidad, las bases de datos del cliente que ha enviado la petición debe quedar convergente con los datos del registro central. - Si ha ocurrido una excepción en el proceso de resolución, se tendrá que registrar está en una bitácora o cola de error.
Excepciones	- Ningún método de resolución ha sido configurado. - Problemas de conexión con el servidor Central.

c. Diagrama de clases de la adaptación.

En la figura 2 se puede observar el diagrama de clases de la adaptación, donde se encuentran las clases que ya incluía el framework para el manejo de la sincronización, así como las agregadas para los procesos de detección, resolución y registro de conflictos. También se detallan las modificaciones hechas a las clases existentes, principalmente el trabajo realizado en SourceSync, donde se modificaron los métodos de procesamiento de las operaciones CUD y los métodos agregados para el registro en la bitácora de error.

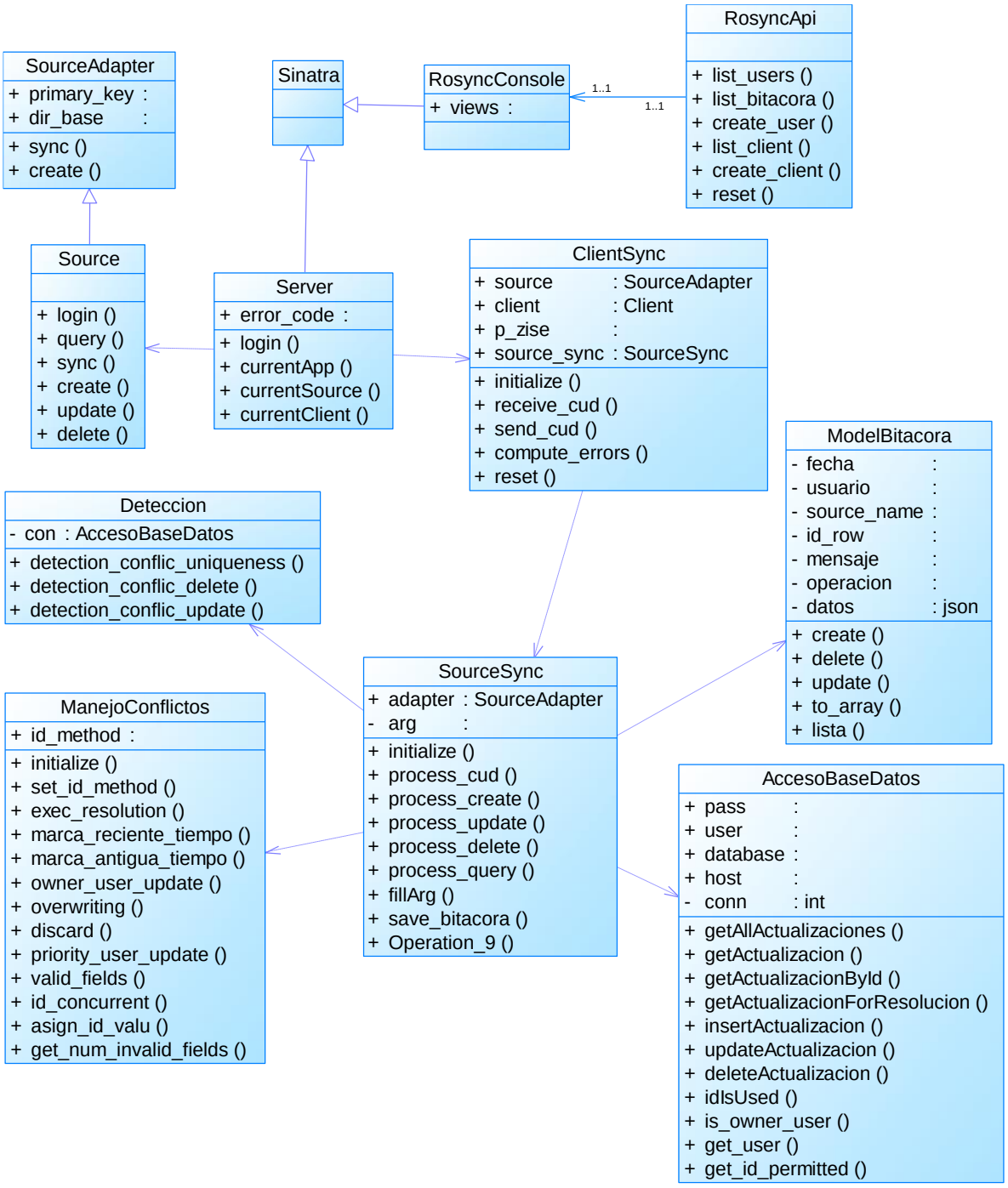

Figura 2. Diagrama de clases de la adaptación.

d. Diagramas de secuencia de la adaptación.

La lógica que se usa para la detección y resolución de los conflictos de datos esta descrita en los diagramas de secuencia que se presentan a continuación. Estos detallan los casos en que ocurre un conflicto y es solucionado correctamente o detectado solamente para su respectivo registro, en el caso de los de eliminación.

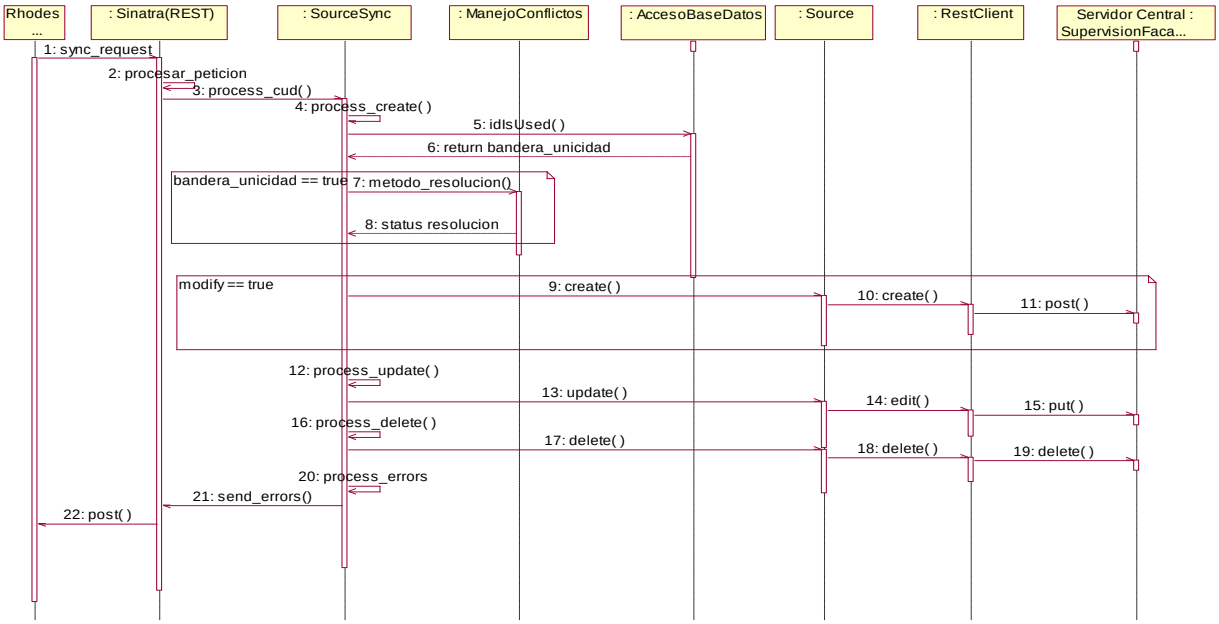

Figura 3. Diagrama de secuencia identificación y resolución de conflictos en una creación.

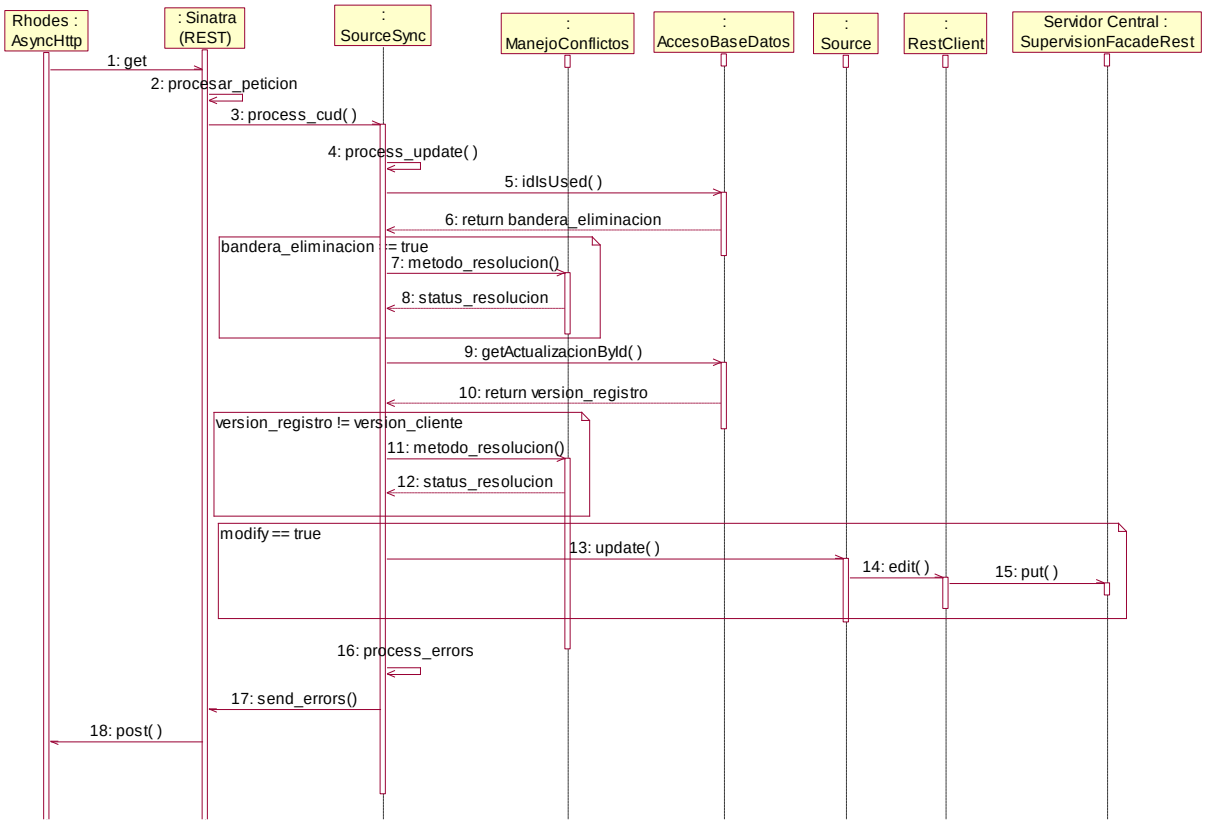

Figura 4. Diagrama de secuencia, Identificación y resolución de conflictos en una Actualización

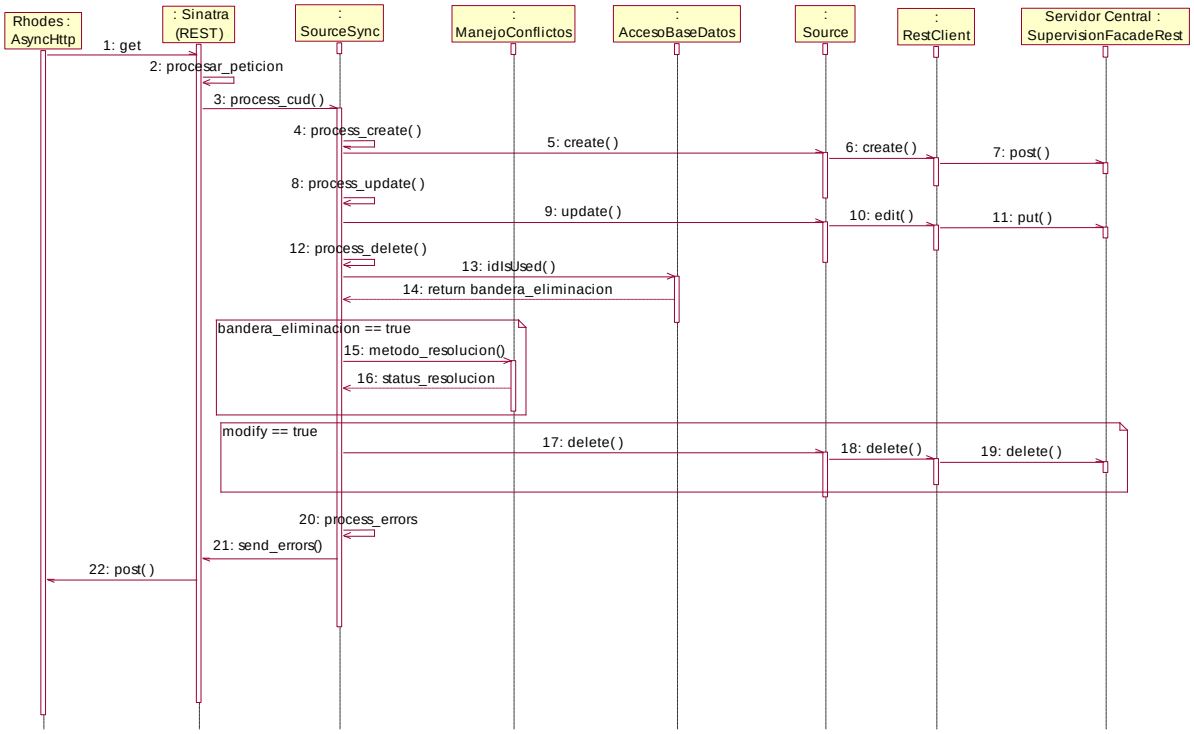

Figura 5. Diagrama de secuencia, Identificación de conflictos en una Eliminación.

Anexo B. Evalucaión del Framework seleccionado.

En este anexo se realiza la confrontación de los lineamientos definidos, con el Framework seleccionado. Para realizar este diagnóstico se tomó como referencia la documentación y algunas pruebas funcionales realizadas al Framework.

1. Evaluación de los lineamientos para la arquitectura general del sistema

a. Arquitectura de Tres Niveles:

El Framework seleccionado cumple con este lineamiento, ya que se divide en tres niveles: Back-End: Aplicación con interfaces Web para el acceso a los datos y logica de negocio, Servidor de Sincronizacion: RhoSync, Aplicaiones Cliente: Rhodes.

b. Funciones del Mediador:

El Framework seleccionado cumple parcialmente con este lineamiento, debido a que RhoSync dispone de metodos para la gestión de conexiones con el Back-end, maneja las sesiones de usuario y permite la gestion de dispositivos, pero no dispone de metodos para Identificación y resolución de conflictos, ni de un mecanismo para el manejo de notificaciones.

c. Sitio Maestro Único:

El Framework seleccionado cumple con este lineamiento, ya que esta diseñado para que las aplicaciones clientes desarrolladas con Rhodes, se conecten a un solo servidor de sincronizacion RhoSync y el servidor de sincronizacion se conecta a un solo servidor de aplicaciones empresariales Back-End.

d. Replicación Asíncrona – Modo desconectado:

El Framework seleccionado cumple con este lineamiento, debido a que Rhosync y Rhodes permiten el trabajo en modo desconectado, y la sincronizacion de datos por medio de un conjunto predeterminado de metodos de consulta, actualizacion, creacion y eliminacion de registros, alojados en los llamados adaptadores fuente.

2. Evaluación de los lineamientos para la Gestión De Las Conexiones

a. Seguridad en el transporte de datos:

El Framework seleccionado cumple con este lineamiento, ya que RhoSync permite utilizar HTTPS para las conexiones con las aplicaciones cliente y el servidor empresarial o Back-End.

b. Uso de servicios Web REST:

El Framework seleccionado cumple con este lineamiento, debido a que RhoSync es capas de consumir servicios Web Rest expuestos por las aplicaiones Back-End. Por otra parte RhoSync se puede comunicar con las aplicaciones cliente moviles desarrolladas con Rhodes por medio de servicios Web Rest. En ambos casos la informacion a intercambiar es representada mediante JSON, aunque para las consultas a los servicios empresariales pueden ser hechas para otras tecnologías como SOAP y representaciones en xml, según la documentación de Rhosycn.

c. Manejo de Sesión para la Sincronización:

El Framework seleccionado cumple con este lineamiento, ya que para poder realizar el proceso de sincronizacion de datos, RhoSync exige a los clientes moviles iniciar sesión, esta sesion la maneja el propio RhoSync mediante una base de datos que usa el modelo OAV y permite administrar los usuarios y dispositivos moviles asociados mediante una consola de administracion Web.

3. Evaluación de los lineamientos para las Aplicaciones Del Back-End

a. Interface de acceso a datos y lógica de negocio:

El Framework seleccionado cumple con este lineamiento, debido a que en teoria, RhoSync puede conectarce a cualquier tipo de aplicaion Back-End, capas de exponer servicios Web para el acceso a la logica de negocio y a los datos.

b. Notificaciones:

El Framework seleccionado cumple con este lineamiento, debido a que la interface expuesta por RhoSync es adaptable para recibir las notificaciones del Back-End y enviar mensajes a los clientes moviles.

4. Evaluación de los lineamientos para las Aplicaciones cliente moviles

a. Desarrollo Multiplataforma:

El Framework seleccionado cumple con este lineamiento, gracias a que RhoDes permite el desarrollo multiplataforma, especificamente en las plataformas moviles: iPhone, Windows Mobile, BlackBerry y Android.

b. Uso de Smartphone y Tabletas:

El Framework seleccionado cumple con este lineamiento, ya que RhoDes solo permite trabajar con plataformas moviles que se ejecutan sobre Smartphones y

Tabletas, depende de cada solucion particular elegir la plataforma y el tipo de dispositivo a usar.

c. Base de datos local:

El Framework seleccionado cumple con este lineamiento, ya que RhoDes dispone de las librerías necesarias para manejar bases de datos locales, Sqlite y Hsql, de esta manera la persistencia a los datos es transparente para el desarrollador de aplicaciones móviles cliente.

d. Almacenamiento cifrado de los datos:

El Framework seleccionado cumple con este lineamiento, debido a que RhoDes tiene un método de almacenamiento cifrado de datos en la base de datos local, el cual se activa en el modelo de cada tabla mediante una variable tipo bandera.

e. Manejo de Sesión

El Framework seleccionado, permite implementar cualquiera de los metodos propuestos en los lineamientos para el manejo de sesion, depende de cada desarrollador y de cada entorno de sincronizacion, que opcion elegir.

5. Evaluación de los lineamientos para el Servidor De Sincronización

a. Des-aprovisionamiento:

El Framework seleccionado, no cumple con este lineamiento debido a que RhoSync no dispone de mecanismos preconfigurados para gestionar el des-aprovisionamiento de datos y aplicaciones presentes en los dispositivos moviles cliente.

b. Tipos de Sincronización

El Framework seleccionado cumple con este lineamiento; debido a que Rhodes y RhoSync, disponen de métodos pre-configurados que permiten elegir entre diferentes alternativas de sincronización, similares a los definidos en los lineamientos.

c. API de Sincronización

El Framework seleccionado cumple con este lineamiento, debido a que Rhodes y RhoSync cuentan con librerías y métodos pre-configurados para tareas como la sincronización de datos, gestión de conexiones, manipulación de la base de datos local, uso de las interfaces nativas de los dispositivos; entre otras características.

d. Sincronización push, Sincronización Mayor, Sincronización Strem

RhoSync permite configurar el tipo de sincronización que se desea usar para cada una de las tablas presentes en la aplicación móvil.

6. Evaluación de los lineamientos para el tratamiento de conflictos de datos

En la documentación de RhoSync, no existe una mención clara y específica sobre el tratamiento de conflictos de datos, por esta razón se realizaron algunas pruebas funcionales a RhoSync versión 2.1.17, con el ánimo de determinar qué métodos o mecanismos emplea para solucionar los conflictos de datos que se puedan presentar. A continuación se describen las pruebas funcionales y su resultado:

a. Arquitectura

A continuación se muestra, la arquitectura de referencia que se utilizará en las pruebas a RhoSync 2.1.17 y las herramientas utilizadas para la implementación de la arquitectura.

Figura 6. Arquitectura de referencia

b. Descripción de la Aplicación

Con el fin de realizar las pruebas a RhoSync, será desplegada en la arquitectura anteriormente expuesta, una aplicación móvil “Prueba1”; que permite a los clientes móviles manipular y sincronizar desde cada replica móvil una base de datos central alojada en el Back-End por medio del servidor de sincronización.

c. Escenario y Plan de pruebas

Las pruebas a las que será sometido el Framework por medio de la aplicación "Prueba1", consisten en generar diferentes tipos de conflictos desde dos clientes móviles, para determinar de qué manera el Framework responde, y que métodos utiliza para tratar los conflictos.

d. Proceso de Pruebas y resultados obtenidos

Las tablas resumen que se presentan a continuación, ilustran el comportamiento de Rhosync versión 2.1.17; en sus columnas, se registra el momento en que se ejecuta cada acción (Tiempo), la descripción de la acción, que sucede con los datos en el sitio maestro, y las réplicas de los dispositivos móviles respectivamente.

- *Conflictos de Unicidad:*

Se genera un conflicto de unicidad entre dos clientes móviles en una operación CREAR REGISTRO. En la siguiente tabla se presenta el resultado que permite determinar, el tratamiento que el RhoSync da a un conflicto de Unicidad.

Tabla 1: Prueba conflicto de unicidad

TIEMPO	ACCIÓN	SITIO MAESTRO	CLIENTE A	CLIENTE B
1	Cliente A crea un registro R1 con clave primaria PK=1		R1={PK=1…} (offline)	 (online)
2	Cliente B crea un 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)
4	Cliente B envía solicitud de sincronización de datos.	R1={PK=1…} R2 genera un error de inserción en la base de datos (Clave Primaria duplicada)	R1={PK=1…} (offline)	R2={PK=1…} (online)
5	El registro R2 no se inserta en la base de datos y es eliminado del cliente que envió la solicitud (B).	R1={PK=1…}	R1={PK=1…} (offline)	R1={PK=1…} R2 es eliminado (online)

De la tabla 1 se infiere que RhoSync no detecta el conflicto de unicidad, ya que la solicitud pasa directamente al sitio maestro ubicado en el Back-End, quien rechaza esta operación al violar la restricción de unicidad del esquema de datos. A su vez, el registro que causo el conflicto es eliminado del cliente móvil que envía la solicitud; en conclusión RhoSync no dispone de mecanismos para identificar, resolver, evitar o registrar conflictos de unicidad.

- *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 siguiente tabla se presenta el resultado que permite determinar el tratamiento que RhoSync da a un conflicto de Actualización.

Tabla 2: Prueba conflicto de actualización.

TIEMPO	ACCIÓN	SITIO MAESTRO	CLIENTE A	CLIENTE B
1	Convergencia de datos, en todos los sitios en el valor: R1= Carlos	R1= Carlos	R1= Carlos (offline)	R1= Carlos (offline)
2	Cliente A actualiza a R1= Juan	R1= Carlos	R1= Juan (offline)	R1= Carlos (offline)
3	Cliente B actualiza a R1 = Pedro	R1= Carlos	R1= Juan (offline)	R1 = Pedro (offline)
4	Cliente A sincroniza exitosamente los datos con el Sitio Maestro.	R1= Juan	R1= Juan (online)	R1 = Pedro (offline)
5	Cliente B sincroniza exitosamente los datos con el Sitio Maestro.	R1 = Pedro	R1= Juan (offline)	R1 = Pedro (online)
6	El Sitio Maestro almacena la versión del último registro que se sincroniza (Descarta el anterior)	R1 = Pedro	R1= Juan (offline)	R1 = Pedro (online)

De la tabla 2, se infiere que RhoSync no detecta el conflicto de actualización, ya que la solicitud de sincronización que genera el conflicto, pasa directamente al sitio maestro ubicado en el Back-End, quien inserta la última versión que es sincronizada y descarta la versión anterior. El cliente que realizó primero la sincronización, queda con una versión divergente de los datos ya que no se le informan los cambios.

En conclusión, RhoSync no dispone de mecanismos para identificar, resolver, evitar o registrar conflictos actualización.

- *Conflictos de Eliminación:*

Se genera un conflicto de eliminación entre dos clientes. En la siguiente tabla se presenta el resultado que permite determinar el tratamiento que RhoSync, da a un conflicto de Actualización.

Tabla 3: Prueba 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…} (offline)	R1={PK=12…} (offline)
2	Cliente A elimina el registro R1 con clave primaria PK=12	R1={PK=12…}	(offline)	R1={PK=12…} (offline)
3	Cliente B actualiza el registro R1 con clave primaria PK=12	R1={PK=12…}	(offline)	R1*={PK=12…} (offline)
4	Cliente A Sincroniza exitosamente sus datos con el Sitio Maestro.		(online)	R1*={PK=12…} (offline)
5	Cliente B envía solicitud de sincronización de datos.	Error, el Back-End no encuentra el registro (R1).	(offline)	R1*={PK=12…} (online)
6	Los datos enviados por el cliente B no se insertan en la base de datos y el registro (R1) es eliminado del cliente que envió la solicitud (B).		(offline)	(online)

De la tabla 3 se infiere que RhoSync no detecta el conflicto de eliminación, ya que la solicitud de sincronización que genera el conflicto pasa directamente al sitio maestro ubicado en el Back-End, quien no encuentra el registro que se ha modificado en el segundo cliente, y genera un error. Luego, el registro es borrado del dispositivo cliente que realiza la actualización (B).

En conclusión, RhoSync no dispone de mecanismos para identificar, resolver, evitar o registrar conflictos eliminación.

Anexo C. Manual de instalación de cRhosync.

La instalación de cRhoSync que se describe a continuación está dieseñada para equipos Windows y consta de los siguientes pasos:

Paso 1. Instalación de Rhosync 2.1.2 [1].

En la carpeta adjunta llamada *instaladores,* se encuenta el instalador de RhoSync 2.1.2 o se puede descargas desde la siguiente dirección https://github.com/rhomobile/rhosync/downloads. Ejecutar y seguir el asistente para su instalación; se recomienda seleccionar todas las opciones por defecto en este proceso.

Finalizada la instalación, se tendrá en el equipo la versión 2.1.2 de RhoSync así como la versión 2.1.1 de *redis*. Con el ánimo de no tener futuros problemas, se debe verificar que las rutas de acceso a redis y rhosync este agregadas en el path del sistema, figura 7.

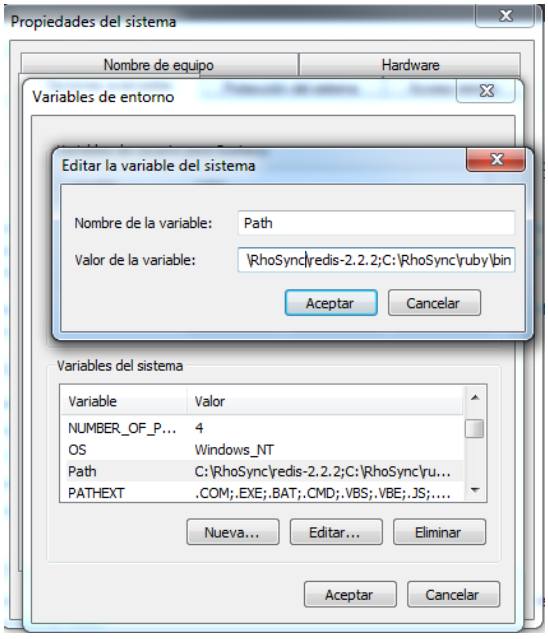

Figura 7. Rutas redis y rhosync en el path de Windows.

Paso 2. Actualizar Rhosync a la version 2.1.17 [2].

La actualización se hace con el ánimo de eliminar algunos bugs de la versión 2.1.2, y ya que Rhosync queda instalado en el equipo como una gema más de ruby, basta

[1] Para instalar RhoSync en Linux o Max, seguir las instrucciones descritas en: http://docs.rhomobile.com/rhosync/install

[2] Para este paso, es necesario tener una conexión a internet.

con ejecutar el siguiente comando desde un terminal de Windows: ***gem install rhosync.*** Este comando ejecutará la actualización no solo de Rhosync a la versión 2.1.17, sino también la versión de *redis* a la 2.2.2, para soportar los cambios ejecutados en esta.

Finalizado este paso, ya se puede desarrollar un proyecto rhosync con la versión 2.1.17. Sin embargo, no se tienen las funcionalidades de identificación, resolución y registro de conflictos de actualización, desarrolladas en el marco de este trabajo de grado.

Paso 3. Agregar los cambios hechos en la adaptación (cRhosync).

Antes de cambiar Rhosync a cRhosync, se debe instalar mysql 5 o mayor[3], para el funcionamiento de las operaciones de identificación y resolución de conflictos. Y a continuación se debe crear la base de datos llamada *rhosync,* con la ejecución del script adjunto en la carpeta instaladores, antes mencionada, este script tiene el nombre: *rhosync.sql* y se encargara de crear esquema de datos necesario.

Para esta tarea, se debe ingresar a una ventana de comandos de Windows, y en la carpeta bin de mysql[4], ejecutar el siguiente comando:

mysql -u usuario –p < /ruta al script/rhosync.sql

En el comando anterior, el usuario y la contraseña deben corresponder con la configuración en la instalación de mysql, se pedirá enseguida que se confirme la ejecución de la instrucción en la ventana de comandos.

Si se ha terminado exitosamente este proceso, se pasa a copiar la carpeta llamada *rhosync-2.1.17,* incluida con los instaladores[5] y remplazarla con la ubicada en la ruta *C:\RhoSync\ruby\lib\ruby\gems\1.8\gems\.* Con esta tarea terminada, tenemos cRhoscync listo para el desarrollo de un servicio de sincronización.

Paso 4. Configuración del entorno de desarrollo.

Descargar Eclipse IDE for Java EE Developers de la siguiente dirección: http://www.eclipse.org/downloads/. Descomprimir en una ruta corta, puede ser C:/RhoStudio, y copiar el archivo jar llamado "*com.rhomobile.rhostudio_2.0.0.201306110834*" que se encuentra en la carpeta de

[3] Se puede seguir el tutorial descrito aquí: http://dev.mysql.com/doc/refman/5.0/es/windows-installation.html

[4] Ruta de la carpeta bin de mysql por defecto es: C:\Program Files\MySQL\MySQL Server 5.#\bin

[5] Esta carpeta también se puede desacargar del repositorio github:

instaladores, a la carpeta *dropins* de eclipse; a continuación iniciar eclipse, ejecutando eclipse.exe que se encuentra dentro de la carpeta descomprimida.

Ya se tiene configurado Eclipse Juno, para trabajar en el desarrollo de proyectos cRhosync; para esta tarea, referirse al Anexo D.

Anexo D. Desarrollo de un servidor cRhosync con RhoStudio.

1. Creación de un proyecto cRhosync a través de un asistente.

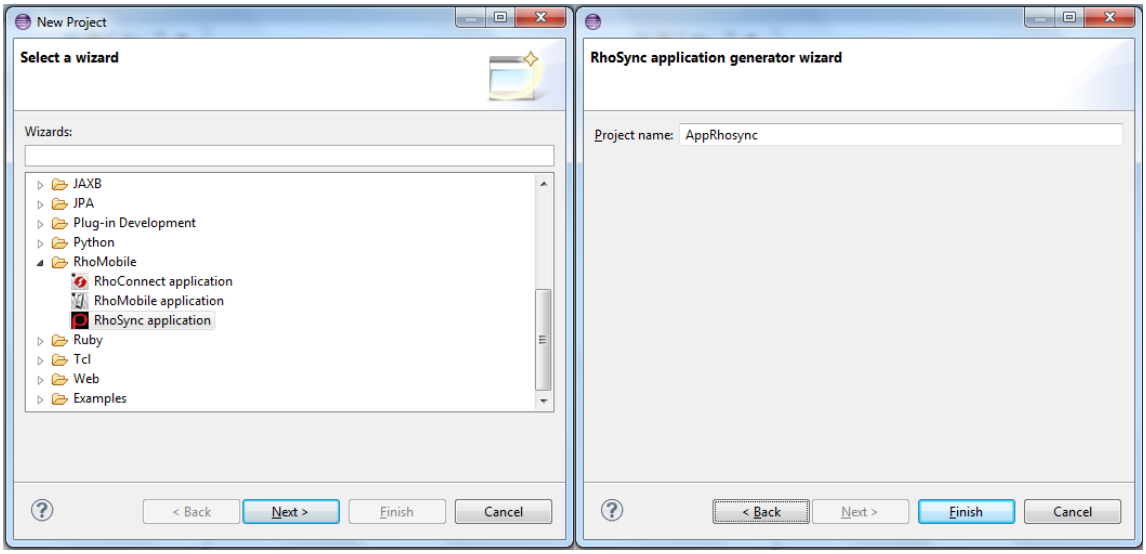

Figura 8. Asistente de creación de un proyecto RhoSync.

El asistente para la creación de un proyecto cRhosync que se muestra en la figura anterior, consta de la ventana de selección del tipo de proyecto y a continuación se debe indicar el nombre necesario para finalizar la creación.

Luego de finalizar la creación de un proyecto, se puede observar el esquema de la aplicación como se muestra en la figura 9. Esta es la descripción general de los componentes:

- Carpeta *settings*: contiene los datos de licencia y de configuración de la aplicación.
- Carpeta *spec*: contiene los archivos que carga el ambiente de funcionamiento de la aplicación.
- Carpeta *sources*: contiene todos los adaptadores fuente que crea el desarrollador.
- *config.ru*: encargado de cargar la aplicación.
- *aplicattion.rb*: es la clase principal de la aplicación, que se encarga de cargar todas las funcionalidades de Framework. Se puede implementar aquí un inicio de sesión general.
- *Rakefile*: contiene las tareas que se pueden ejecutar desde el terminal.

2. Creación de adaptadores fuente a través de un asistente.

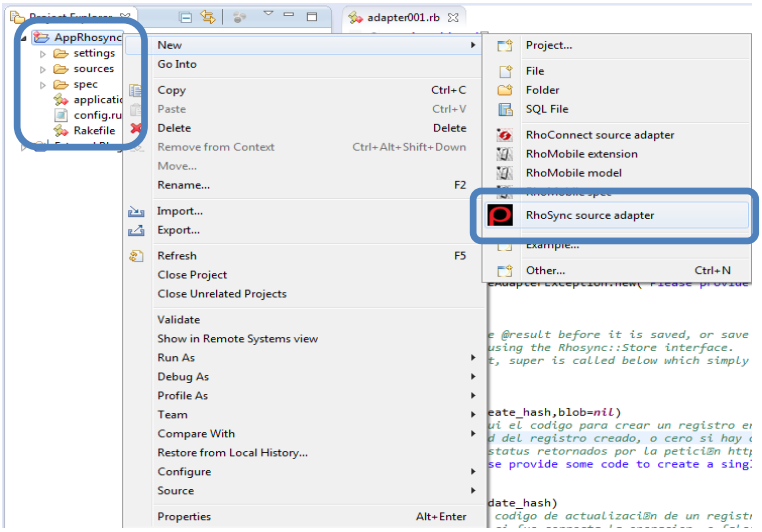

Figura 9. Menú de creación de un adaptador fuente y estructura de una aplicación Rhosync.

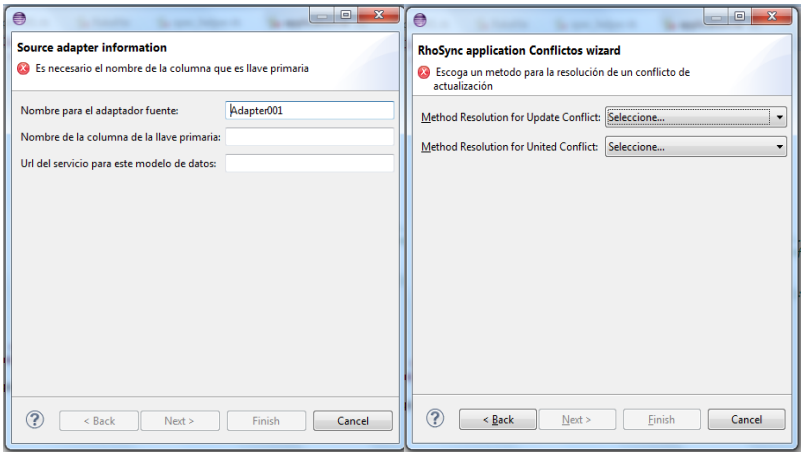

Figura 10. Asiste de creación de un adaptador fuente.

El asistente para la creación de un adaptador fuente Rhosync (Figura 10), permite indicar el nombre del adaptador y la ruta especifica de acceso al servicio, para la tabla que manejará este adaptador. En el siguiente paso, se indica la configuración del manejo de conflictos, escogiendo los métodos que se desea usar para la resolución.

Al finalizar la creación de un adaptador fuente cRhosync, se crea la carpeta *sources* (si esta no existe aún) y el adaptador fuente, acompañado de un arhivo de configuración en formato json para la resolución de conflictos; como se muestra en la figura 11.

3. Lógica de acceso al servicio del servidor central.

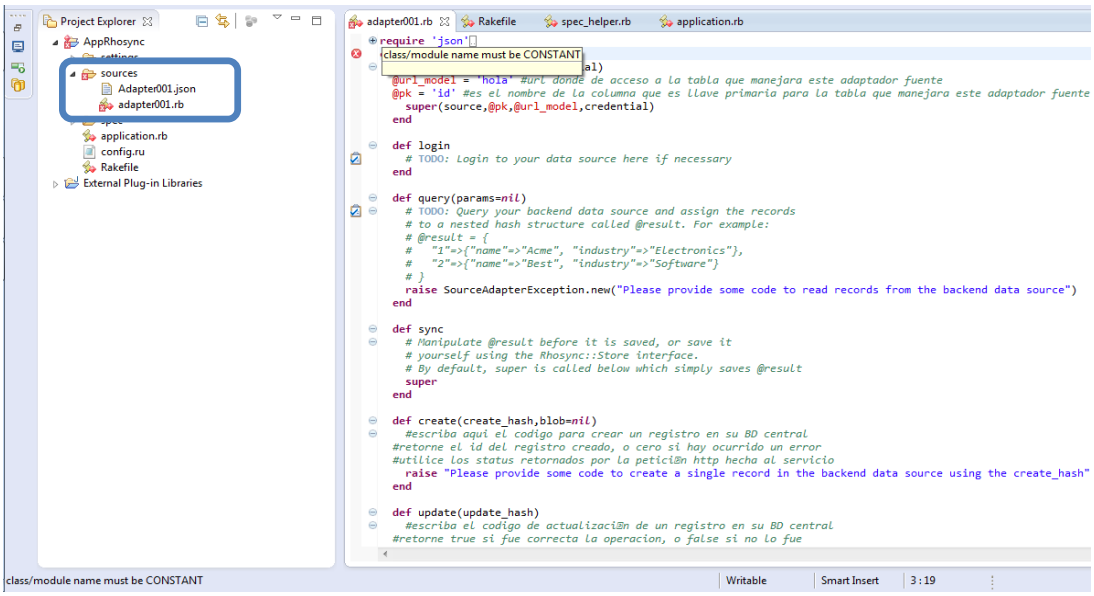

Figura 11. Editor y depurador Ruby.

Una las características importantes de un entorno de desarrollo, es la depuración de los códigos en tiempo de escritura. El RhoStudio original incluye un editor y depurador Ruby que se ha integrado para los proyectos cRhosync y que puede ser de gran utilidad en la implementación de la lógica de los adaptadores fuente, principalmente.

Para el acceso a un servicio Rest desde los adaptadores fuente se utilizar la gema ResClient de ruby, esta gema, permite realizar las acciones get, post, put y delete del protocolo http.

Los códigos de ejemplo que se muestran a continuación, corresponden al piloto implementado en este trabajo de grado (servicio Rest Back-end, desarrollado en JAVA), por lo que es una guía de ejemplo para el uso de la gema *RestClient;* sin embargo, para servicios Rest desarrollados en otras plataformas, podrían necesitarse cambios, pero representa una guía general del uso de los métodos que posee el adaptador fuente.

3.1. Consulta (método *query* en el adaptador).

Para una consulta a un servicio Rest, se puede utilizar el método *get* de la gema ResClient. En la figura 12, se muestra un ejemplo de una implementación, para el método *query* de un adaptador fuente.

La variable *@url_model,* contiene la ruta principal de acceso al servicio Rest, y es común para todos los ejemplos que se sugieren en estas sección al igual que la variable *@pk,* que contiene el nombre de la columna que es la llave primaria. Ambas variables, son inicializadas con la creación del adaptador fuente (Figura 10).

```
def query(params=nil)
  respuesta = RestClient.get("#{@url_model}",:accept => 'application/json',:content_type => 'application/json')
  parsed = JSON.parse(respuesta.body)
  @result={}
  parsed.each do |item|
    supervision = {}
    id = item["#{@pk}"]
    item.each do |key,value|
      supervicion["#{key}"] = value
    end if item
    @result["#{id}"] = supervision
  end if parsed
  return @result
end
```

Figura 12. Método *query* del adaptador fuente, para el acceso a los registros del back-end.

En la implementación del método query del adaptador fuente, es importante considerar que al final del método, se debe retornar una variable con los datos consultados al servidor central. Esta variable puede ser un *array* o un *hash.*

3.2. Creación (método *créate* en el adaptador fuente)

Para crear un registro a través de un servicio Rest, en el adaptador fuente, se puede utilizar el método *post* de la gema ResClient. En la figura 13, se muestra un ejemplo de una implementación, para el método *create* de un adaptador fuente.

```
def create(create_hash,blob=nil)
  ids = create_hash["#{@pk}"]
  create_hash.delete("#{@pk}")
  create_hash["#{@pk}"] = ids.to_i
  create_hash = create_hash.to_json
  puts 'json a crear'
  puts create_hash
  res = RestClient.post(@url_model,create_hash,:accept => 'application/json',:content_type => 'application/json')
  case res.code
    when 204#forma en que responde este servicio en particular para una operacion exitosa
      return ids
    else
      return 0
  end
end
```

Figura 13. Método *create* del adaptador fuente, para la creación de un registro en el back-end.

El método *create* del adaptador fuente, debe retornar el valor de llave primaría del registro, si este se crea exitosamente; de lo contrario se debe retornar un cero. Esto con el ánimo de verificar, si la operación se realizó exitosamente en el back-end.

3.3. Actualización (método *update* en el adaptador fuente)

Para actualizar un registro a través de un servicio Rest, en el adaptador fuente; se puede utilizar el método *put* de la gema ResClient. En la figura 14, se muestra un ejemplo de una implementación, para el método *create* de un adaptador fuente.

```
def update(update_hash)
  puts 'actualizar'
  ids = update_hash["#{@pk}"]
  update_hash.delete("#{@pk}")
  update_hash["#{@pk}"] = ids.to_i
  update_hash = update_hash.to_json
  puts update_hash
  respuesta = RestClient.put("#{@url_model}",update_hash,:accept => 'application/json',:content_type => 'application/json')
  case respuesta.code
    when 204#forma en que responde este servicio en particular para una operacion exitosa
        return true
    else
      return false
  end
end
```

Figura 14. Método *update* del adaptador fuente, para actualizar un registro en el back-end.

En el método *update,* se debe retornar un valor lógico que indique si la actualización fue exitosa o no en el back-end. De manera similar, esta bandera debe estar relacionada con el estado de la respuesta a la petición *put.*

3.4. Eliminación (método *delete* en el adaptador fuente)

Para eliminar un registro a través de un servicio Rest, en el adaptador fuente; se puede utilizar el método *delete* de la gema ResClient. En la figura 15, se muestra un ejemplo de una implementación, para el método *delete* de un adaptador fuente.

```
def delete(delete_hash)
  puts 'delete'
  ids = delete_hash["#{@pk}"]
  respuesta = RestClient.delete("#{@url_model}/"+ids)
  case respuesta.code
    when 204#forma en que responde este servicio en particular para una operacion exitosa
      return true
    else
      return false
  end
end
```

Figura 15. Método *delete* del adaptador fuente, para eliminar un registro en el back-end.

En la implementación de este método, también es necesario devolver un valor lógico, que indique si la petición de eliminación se procesó correctamente. De igual forma, la manera de hacerlo se enfoca en el código de estado, que retorna la respuesta de la petición *delete*.

a. Despliegue de Aplicaciones cRhosync.

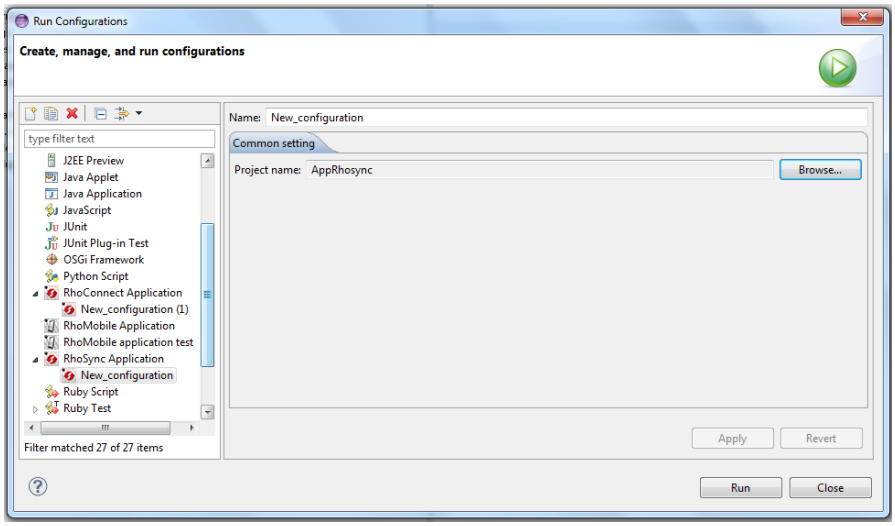

Figura 16. Configuración de ejecución de cRhosync.

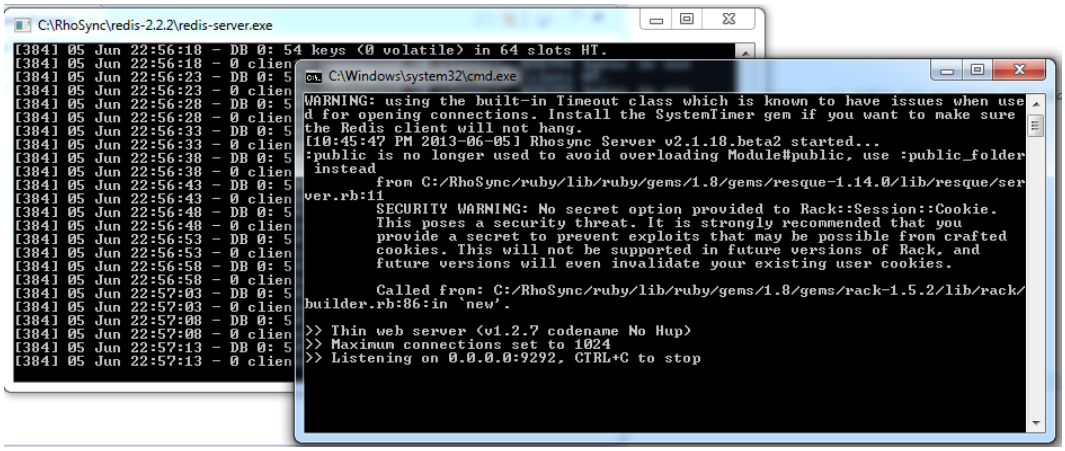

Figura 17. Ejecución de una aplicación cRhosync.

Las figuras anteriores, muestran el proceso de despliegue de una aplicación cRhosync en RhoStudio. En la Figura 16, se indica la configuración de ejecución, donde se selecciona el tipo y el proyecto a desplegar. Este asistente se puede acceder desde el menú *run configurations…,* después de dar click derecho en el proyecto y seleccionar el menú desplegable *run as*.

Las ventanas de comandos que se muestran en la figura 17, representa el servicio en ejecución de *Redis* y de la aplicación Rhosync respectivamente. *Redis* es un servidor de estructuras de datos clave-valor, que se incluye en las gemas de ruby y que es utilizado por RhoSync para almacenar todos los datos de funcionamiento.

Anexo E. Piloto: Análisis y diseño del Software.

Este anexo aborda de forma general el análisis y diseño del software, descrito con los diagramas de casos de uso, de clases para las aplicaciones cliente y back-end, y el diagrama de despliegue del piloto.

a. Diagrama de casos de uso del piloto.

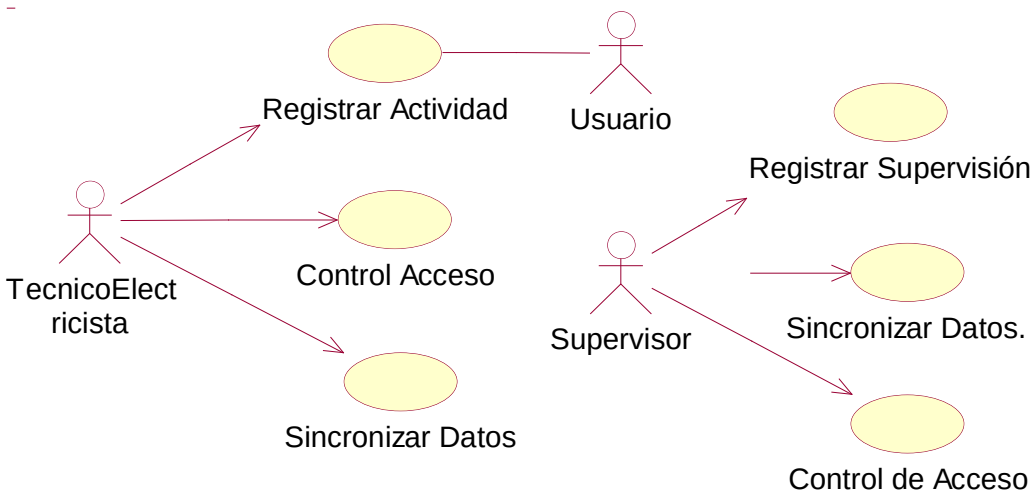

Figura 18. Modelo de Casos de Uso del Sistema.

b. Descripción detallada de los casos de uso esenciales

Nombre: Control de Acceso,
Iniciador: Técnico Electricista, Supervisor
Propósito: La finalidad de este caso de uso es gestionar el acceso al sistema de cada uno de los actores relacionados.
Precondiciones: El actor debe tener establecida una conexión a internet en su dispositivo y haber iniciado la aplicación.
Escenario:

Actor (es)	Sistema
1. El actor digita sus datos de usuario, ingresa su contraseña y envía una solicitud de ingreso al sistema.	2. Toma los datos que ingresó el usuario, valida su ingreso al sistema, posteriormente le presenta una interfaz inicial según el rol.

Poscondiciones:

- Se guarda la sesión de usuario para necesaria para las tareas a ejecutar.

Flujo alternativo o Asíncrono:

Actor (es)	Sistema
1. El actor digita sus datos de usuario, ingresa su contraseña y envía una solicitud de ingreso al sistema.	2. Toma los datos de inicio de sesión de usuario, pero comprueba que son incorrectos. A continuación le pide que lo vuelva a intentar, por medio de un mensaje de información.

Excepciones:

Actor (es)	Sistema
1. El actor digita sus datos de usuario, ingresa su contraseña y envía una solicitud de ingreso al sistema.	2. No se puede establecer comunicación entre el servidor de sincronización y la base de datos central para ejecutar la operación, se presenta una excepción y se notifica al usuario.

Interface de usuario relacionadas:

Caso de uso número: 2.
Nombre: Registrar Supervisión
Actor: Supervisor
Propósito: El usuario registra un nueva actividad, captura los datos correspondientes y los almacena en la base de datos del dispositivo.
Precondiciones: El Supervisor debe haberse identificado en el sistema con su nombre de usuario y su contraseña respectiva.
Escenario:

Actor (es)	Sistema
1. El Supervisor elige la opción registrar nueva actividad del menú principal.	2. La aplicación despliega una interfaz con el formulario que permite el ingreso de los datos correspondientes.
3. El Supervisor ingresa los datos en el formulario desplegado, luego de ingresar todos los campos	4. Se despliega una pantalla de confirmación donde se resume los campos principales del registro a crear y se presenta el número

requeridos, confirma el registro de la actividad.	consecutivo con el que se registrara el reporte.
5. El Supervisor confirma el registro de la actividad.	6. La aplicación almacena los datos en la base de datos local del dispositivo y despliega un mensaje de confirmación.
7. El Técnico Electricista recibe el mensaje de confirmación, que le informa que la actividad ha sido creada con éxito.	

Pos-condiciones:

- Se registran los datos correspondientes a la actividad en la base de datos del dispositivo móvil.

Flujo alternativo o Asíncrono:

Actor (es)	Sistema
5. El Supervisor cancela la confirmación del registro de la actividad.	6. Se muestra nuevamente el formulario de registro de la actividad para que el Supervisor cambie los datos que desee.
7. El Supervisor cambia los campos deseados.	8. Se muestra una pantalla de confirmación donde se resume los nuevos datos.
9. El Supervisor confirma el registro de la actividad con los nuevos datos.	10. La aplicación almacena los datos en la base de datos local del dispositivo y despliega un mensaje de confirmación.
11. El Supervisor recibe el mensaje de confirmación, que le informa que la actividad ha sido creado con éxito.	

Excepciones:

Actor (es)	Sistema
11. El Técnico Electricista recibe el mensaje de error, donde se le informa que los datos que desean almacenar en la base de datos no pueden ser procesados y/o almacenados por el	

sistema, y le indica la razón (error en los datos, memoria insuficiente).	

Interface de usuario relacionadas:

Caso de uso numero: 3.
Nombre: Sincronizar Datos
Actor: Técnico Electricista, Supervisor
Propósito: Tiene como finalidad sincronizar los datos recogidos en campo durante la jornada laboral o el periodo de desconexión, con los datos almacenados en el servidor de bases de datos central.
Precondiciones: El actor debe haberse identificado en el sistema con su nombre de usuario y su contraseña respectiva.
Escenario:

Actor (es)	Sistema
1. Comienza cuando el actor a elige la opción sincronizar.	2. El sistema establece la conexión con el servidor de sincronización.
3. El actor confirma la operación de sincronización.	4. El sistema comprueba los campos de datos que se van a actualizar y ejecuta la sincronización de datos.
5. El actor recibe el mensaje de confirmación, donde se le informa que la operación de sincronización resulto exitosa.	6. El sistema termina la conexión entre el dispositivo móvil y el servidor central.

Pos-condiciones:

- Se actualiza la información entre la base de datos central y el dispositivo móvil con los nuevos datos.

Flujo alternativo o Asíncrono:

Actor (es)	Sistema
3. El actor cancela la confirmación de la sincronización de datos.	4. Se despliega en pantalla de la aplicación el menú principal.

Excepciones:

Actor (es)	Sistema
5. El actor recibe el mensaje de confirmación, donde se le informa que los datos que desea sincronizar con la base de datos central no pueden ser procesados y/o almacenados por el sistema, y le indica la razón (error en los datos, conflicto de replicación, etc).	

Actor (es)	Sistema
	2. No se puede establecer conexión entre el dispositivo móvil y el sistema central de información, por errores de la Red de transmisión de datos o por fallas en la disponibilidad de alguno de los componentes del sistema.

Interface de usuario relacionadas:

a. Diagrama de clases de la aplicación cliente

En la figura 19, se puede observar el diagrama de clases de la aplicación cliente donde se describen las clases que incluye el Framework para el manejo de la base de datos local, las solicitudes de sincronización de datos, el control y la navegación por la aplicación.

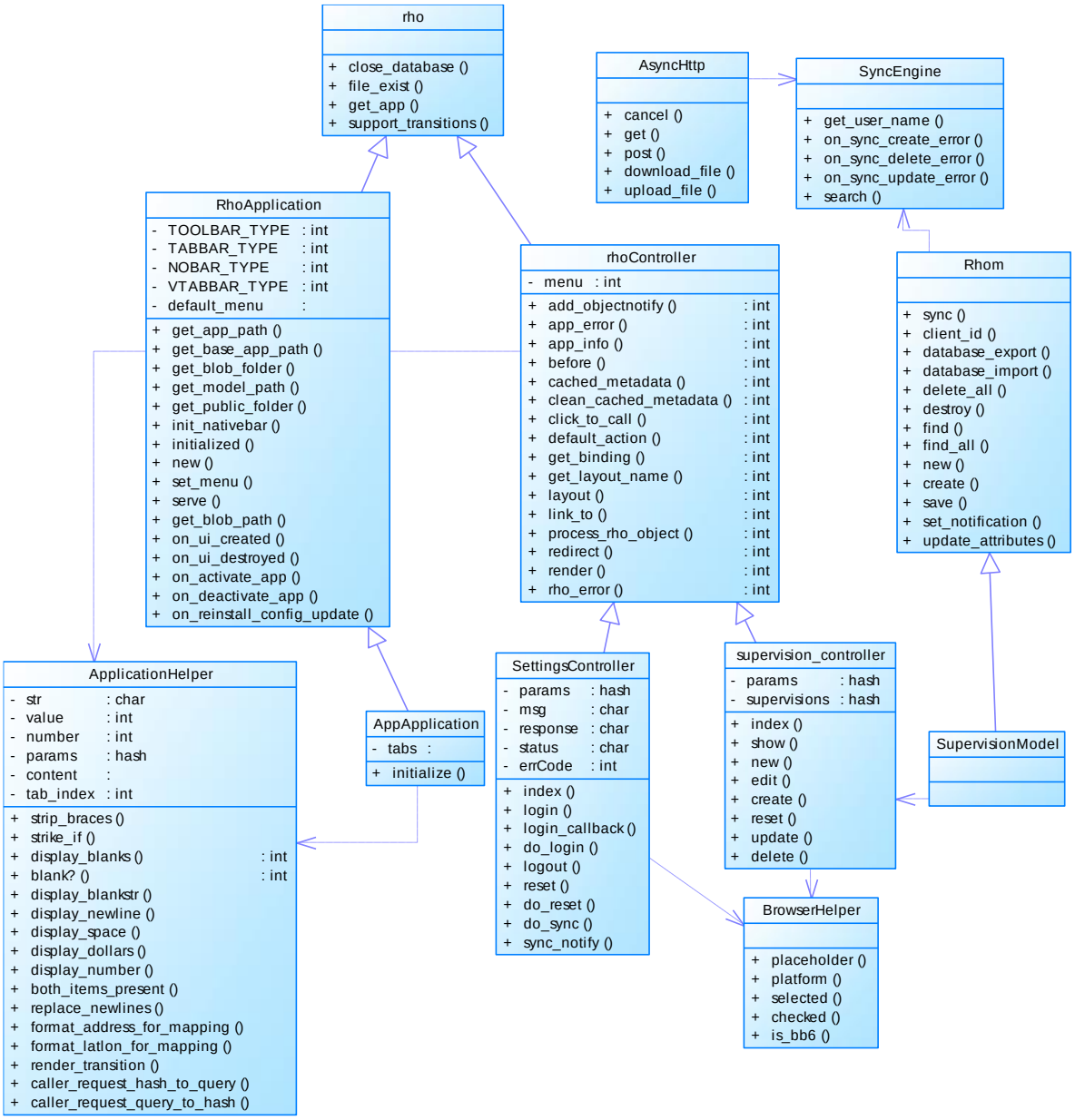

Figura 19. Diagrama de clases de la aplicación móvil.

b. Diagrama de clases de la aplicación empresarial (Back-End)

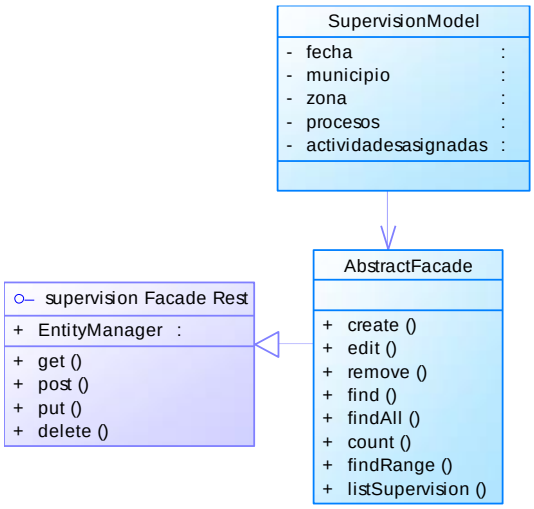

Figura 20. Diagrama de clases de la aplicación Web.

c. Diagrama de despliegue del piloto

El diagrama de despliegue del piloto muestra los tres nodos principales de un sistema de sincronización desarrollado con cRhosync, que corresponden a la aplicación cliente Rhodes, el servidor de sincronización y la aplicación empresarial.

También se pueden observar las bases de datos en cada nodo, la central ubicada en el servidor empresarial, los datos clave-valor redis que se guardan en el cache del servidor de sincronización, y finalmente la réplica que se tiene en cada cliente móvil, en el servidor de sincronización se puede observar una base de datos MySql, que corresponde a la usada para el manejo de conflictos.

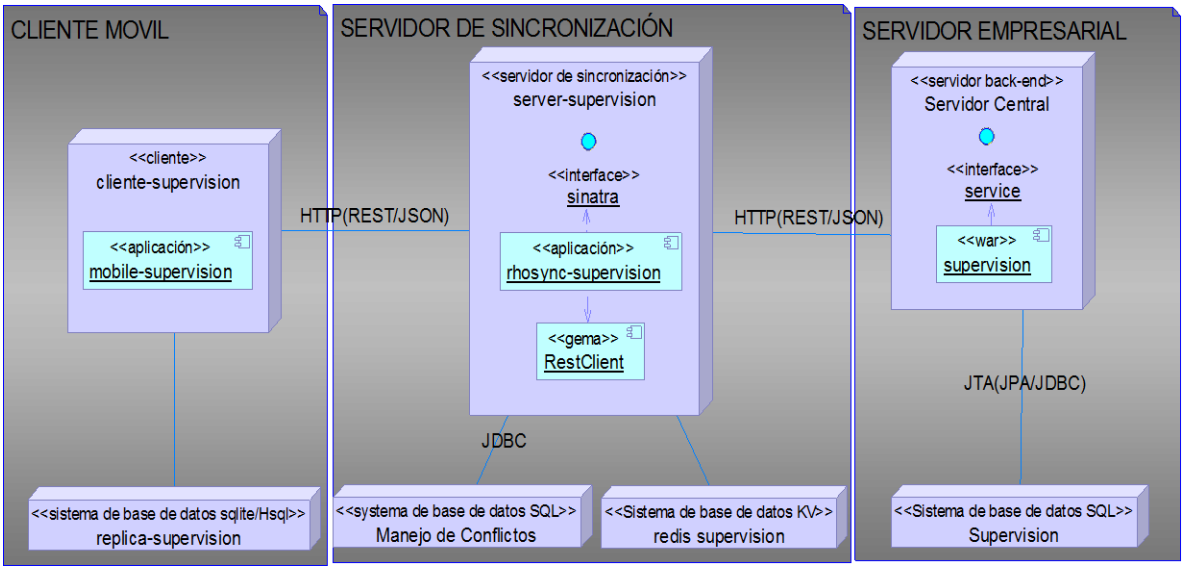

Figura 21. Diagrama de despliegue del piloto.