FARAUTE, Vol. 10, No. 1, 32-48, 2015
ISSN 1698-7418
Deposito Legal PP200402CA1617

Aplicación móvil para el manejo de solicitudes a una base de datos, usando SMS para conexiones sin Internet

Herrera, Mirella, Herrera, Gerardo , Jara, Augusto
Departamento de Computación Universidad de Carabobo, FACYT Valencia, Venezuela
***Autor de correspondencia**: mirella.herrera@gmail.com

Resumen
En la actualidad, resulta de vital interés en casi cualquier ámbito de la vida diaria (negocios, educación y entretenimiento) establecer un proceso de sincronización de información vía Internet a través de un teléfono móvil. Sin embargo, dicha conexión puede fallar por diversas razones interrumpiendo el proceso de sincronización. En orden de presentar una alternativa para disminuir los posibles efectos negativos en las organizaciones que dependen de la sincronización, se presenta un framework que provee un conjunto de componentes para que el programador, adapte sus aplicaciones para enviar las consultas mediante SMS y de esta manera sustituir la conexión a Internet. En el proceso de desarrollo del framework se siguió la metodología propuesta por Bosch (análisis del dominio, diseño arquitectónico y del framework, implementación, prueba del framework y documentación) y se realizaron pruebas instanciándolo en una aplicación del dominio de comercialización en el ramo ferretero, a fin de que los vendedores pudieran realizar operaciones CRUD de pedidos de venta en zonas sin conexión a Internet. Los resultados muestran la posibilidad de establecer conexión sin internet a una BD utilizando SMS desde un dispositivo móvil, se encontraron algunas dificultades que pudieran ser tratadas en futuras investigaciones y versiones del framework

Palabras Clave: Framework, Patrones de diseño, Internet, SMS-SQL, Android.

Mobile app to manage DB queries using SMS for no internet connections

Abstract
At present, it is now vitally important to establish a process of synchronizing information via the Internet through a mobile phone in almost any area of daily life (business, education and entertainment). However, the connection may fail for various reasons interrupting the synchronization process. In order to present an alternative to diminish the possible negative effects in the organizations, a framework oriented to programmers presents a set of components to adapt its applications or business model to send data base queries through SMS and substitute the internet connection. The framework development process, followed the methodology proposed by Bosch (domain analysis, architectural and framework design, implementation, framework test and documentation) and tests were carried out instantiating it in an application of the commercialization domain in the hardware industry. The sellers could perform CRUD operations and sales orders to the database without an Internet connection. The results show the possibility of establishing a non-internet connection to a database using SMS from a mobile device; also, we found some difficulties that could be addressed in future research and versions of the framework.

Keywords: Framework, Design Patterns, Internet; SMS; SQL; Android.

Introducción

En la actualidad, la telefonía móvil se ha convertido en una herramienta crítica para los negocios y para la vida diaria, en la mayoría de los países desarrollados. Es así como ha proliferado el uso de los teléfonos para realizar transacciones y operaciones en distintos ámbitos como la educación, negocios y entretenimiento, entre otros; convirtiéndose, en algunos casos, en el medio indispensable para establecer las conexiones necesarias y realizar dichas transacciones.

De la misma manera, son cada vez más las empresas que incorporan la utilización de aplicaciones móviles para llevar a cabo sus procesos de negocio y en consecuencia, sus empleados y clientes requieren de una adecuada y oportuna sincronización, a fin de establecer el acceso a los datos de la empresa, vía Internet.

Sin embargo, a nivel mundial existen ubicaciones geográficas donde los dispositivos móviles no cuentan con conexión a Internet, debido a que la señal no alcanza al teléfono con la suficiente frecuencia. Esta disminución en la frecuencia de la señal, ocasiona pérdida de información en tiempo real generando un sin número de incidencias negativas que, en muchos casos, pudieran conducir a la pérdida de oportunidades y fidelización del cliente, entre otras.

Es importante destacar que en Venezuela, una cantidad considerable de empresas, especialmente del ramo de la comercialización, utiliza la telefonía móvil como mecanismo esencial de sus operaciones de fuerza de ventas, presentando problemas para establecer la sincronización con sus bases de datos, desde lugares donde no hay acceso a la Internet.

Por lo tanto y en base a lo expresado, este trabajo plantea el desarrollo de una primera versión de un framework cuyo objetivo principal permite solventar las dificultades enunciadas, haciendo uso de la mensajería de texto o SMS como vía alterna de comunicación, al permitir una mayor oportunidad para la sostenibilidad de la comunicación.

Es así como el desarrollo de un framework flexible y adaptable a las necesidades de los programadores, que además pueda ser instanciado en el dominio particular de la empresa, conllevaría al logro de una solución rápida y de bajo costo para las organizaciones.

El presente artículo se organiza en cinco secciones, el marco teórico desarrolla una revisión de la literatura a partir de una selección de autores, términos y trabajos importantes para la contextualización del tema y la problemática. Asimismo, una sección para el abordaje metodológico del proceso de desarrollo del framework y una sección para la discusión de los resultados, en la que se despliegan los pasos seguidos en la metodología y la instanciación en el caso de estudio, a partir de una empresa comercializadora en el ramo ferretero. La última sección abarca los aspectos correspondientes a las conclusiones y trabajos a futuro.

Marco Teórico

Revisión de la Literatura

A continuación se presenta una serie de términos y acepciones importantes, que permiten contextualizar la investigación.

Framework o Marco de Trabajo: Un marco “es un conjunto de clases que encarna un diseño abstracto para soluciones a una familia de problemas”[1]. Otra definición es ün marco es un diseño reutilizable de todo o parte de un sistema que está representado por un conjunto de clases abstractas y la forma en que sus instancias interactúan”[2]. En términos generales, podría definirse framework como una aplicación que consta de un código desarrollado para todas las funciones básicas de un sistema, que se puede ajustar para el propósito de una aplicación concreta [3]. Puede considerarse también un conjunto de bloques de construcción prefabricados que los programadores pueden utilizar, extender o adaptar para soluciones computacionales [4]. Entonces, una de las razones principales para el desarrollo de frameworks, obedece al hecho de facilitar la reutilización del código.

El primer marco orientado a objetos, MVC para Smalltalk, se desarrolló durante los años 80s y desde entonces se ha producido una cantidad considerable de artículos y desarrollos con el fin de construir el marco perfecto. Sin embargo, aún no hay una metodología única ni para el desarrollo ni para su adecuada documentación, principalmente debido al amplio dominio de las aplicaciones existentes [5].

Patrones de Diseño de Frameworks: La mayoría de los autores coincide en el uso de patrones de diseño para el desarrollo de frameworks. Algunos de los más significativos son: Patrón de los tres ejemplos, Caja Blanca, Librería de Componentes, Puntos Calientes o Hot Spots, Caja Negra, Visual Builder, Herramientas de Lenguaje de Programación, entre otros. De acuerdo con [5] durante la fase de análisis del framework, el patrón de los tres ejemplos juega un papel de gran utilidad pues permite una definición del dominio, generando una abstracción para tres aplicaciones. El patrón de caja blanca sugiere el uso de la herencia como la forma más simple de hacer y cambiar código en el desarrollo de software orientado a objetos. El patrón de librería de componentes sugiere que se debe crear una biblioteca consistente de un conjunto de componentes, que se utilizan en la mayoría de las aplicaciones (por ejemplo, algunas clases de utilidad), simplificando el uso del marco en sentido amplio. El patrón de puntos calientes o Hot Spots muestra que el código que se cambiará para cada aplicación, debería estar ubicado en pocas clases. Esto simplificará el uso del framework, y los usuarios (programadores) sabrán dónde deben realizar cambios de código para la aplicación específica, generada usando el framework. Para esta investigación se utilizó el patrón de los tres ejemplos en la fase de análisis del framework y el patrón de puntos calientes o Hot Spots para la fase del desarrollo.

SMS Gateway: Una de las especificaciones de los mensajes de texto, en lo sucesivo, Short Message Service (SMS), es que pueden ser enviados de teléfono a teléfono, y también pueden ser enviados de computadora a teléfono. Los servicios SMS Gateway o de pasarela SMS proveen servicios computador-teléfono y teléfono-computador. Estos servicios hacen posible la comunicación desde o hacia un móvil, de actualizaciones de un sitio web por medio de SMS [6].

Red de telefonía móvil: Actualmente las operadoras de telefonía móvil en Venezuela, se rigen por las dos plataformas de comunicación CDMA y la GSM. Los sistemas, GSM (Global System for Mobile Communications) o sistema global para comunicaciones móviles, y el CDMA (Code Division Multiple Access) traducido como división de código de acceso múltiple, hacen de la red de telefonía móvil un elemento indispensable para una buena comunicación. Los teléfonos GSM se identifican por llevar en su interior tarjetas SIM (módulo de identidad del suscriptor) y los CDMA solamente obtienen comunicación por configuración directa. El mercado nacional de acuerdo con la Comisión Nacional de Telecomunicaciones CONATEL en un informe de 2015, adjudica una suscripción del 23 % aproximadamente a los usuarios CDMA, quedando el resto para los usuarios del GSM. Este último fue seleccionado como base para el desarrollo del framework, por su uso generalizado a nivel nacional e internacional.

SMS (Short Message Service): Un mensaje SMS es aquel enviado desde o hacia un teléfono móvil, limitado a 140 bytes o 160 caracteres de 7 bits [7]. Algunas estadísticas de acuerdo con [8] indican que para el año 2009, 4 mil millones de usuarios GSM usaban el servicio, generando más de 50 mensajes cortos por usuario al mes. Esto lo presenta como un medio altamente utilizado y efectivo, para establecer una comunicación alternativa y llevar a cabo operaciones tipo CRUD (create, read, update, delete) en las organizaciones.

A. Aspectos Teóricos El estudio del proceso de desarrollo de un framework tiene sus inicios a mediados de los 90s, uno de los pioneros en el área es Booch, quien trabajó el proceso de desarrollo y el uso de patrones [9]. Desde entonces, han surgido una gran cantidad de trabajos, algunos de interés para la investigación, se presentan a continuación:

De acuerdo con Stanjoevic, Vlajic, Milov y Ognjanovic [5], Markiewicz y Lucena proponen el ciclo del desarrollo clásico de frameworks, comprendiendo las fases: análisis del dominio, diseño e implementación del framework [10]. En otro orden, Johnson considera este enfoque teórico ideal pero con poca correspondencia con la realidad, sugiriendo una opción iterativa en la cual, mientras un grupo desarrolla el framework otro grupo desarrolla la aplicación en un determinado dominio, utilizando el framework [11].

Por su parte, Xavier Amatriain [12] apoya el proceso de desarrollo iterativo propuesto por Johnson y añade la participación del usuario en las distintas etapas. Asimismo, indica que no siempre es necesario el modelo de análisis del dominio en etapas tempranas, puesto que podría realizarse durante el diseño del framework generando el meta-modelo del dominio más adecuado. La empresa Taligent aporta con sus prácticas para desarrollos pequeños, la conveniencia de

enfocar el framework en la solución de forma iterativa, con su correspondiente documentación y respaldo [4]. Bosch diferencia dos actividades principales en el desarrollo: 1) El core que comprende la parte del framework que no cambia con cada instanciación o uso, siendo conformado por los denominados frozen spots, es decir, interfaces y clases ya implementadas en el framework y que son usadas en cada instanciación. 2) El código generado por cada aplicación específica de acuerdo al modelo de dominio llamado internal add-ins. Las partes del código que lo hacen flexible son conocidos como hot spots, los cuales en muchos casos son clases abstractas o métodos que deben ser descartados [13]. En resumen, la aplicación definitiva generada utilizando el framework consiste del core, los add-ins o complementos internos y el código específico de la aplicación o hot spots. En este trabajo se desarrolló un framework basados en el hecho de que un número considerable de empresas del dominio cuya fuerza de ventas realiza operaciones tipo CRUD, desde sus celulares a las bases de datos. Asimismo, se utilizó el método propuesto por Bosch para el proceso de desarrollo en función de su característica iterativa y que además se refina con cada instanciación en los dominios de las aplicaciones, integrando en la revisión aquellos aspectos que facilitan al programador su uso. Finalmente, se trabajó utilizando la separación entre el core, adds-in y hot spots en la construcción de las clases, a fin de orientar con más detalle al programador en su uso.

B. Proceso de Desarrollo del Framework

Es claro que el desarrollo de un software difiere del desarrollo de un framework. Una de las principales diferencias es que el resultado del desarrollo de un software es una aplicación concreta para un dominio específico mientras que el resultado del desarrollo de un framework, es un marco que puede ser usado para desarrollar diversas aplicaciones [5]. Como referencia se siguieron las siguientes fases [14], (Figura 1):

1. Análisis del Dominio: En esta fase se llevaron a cabo las actividades de identificación de las características, descripción y modelación del dominio de las comunicaciones a través de dispositivos móviles. Asimismo, se realizó la modelación de los procesos de conexión a través de Internet, la descripción de los componentes del framework y finalmente la comunicación a través de mensajes SMS utilizando dispositivos móviles sin acceso a Internet, todo ello partiendo de la base del patrón de los tres ejemplos. El resultado fue un Modelo de Dominio contentivo de los elementos y conceptos, así como la relación entre ellos. Igualmente, se levantaron los requerimientos de una empresa comercializadora de productos, cuyas ventas están basadas en aplicaciones móviles que realizan operaciones CRUD sobre una base de datos vía Internet, los cuales son desplegados en la instanciación en el caso de estudio.
2. Diseño Arquitectónico: Durante esta fase, se utilizó el análisis del dominio como insumo para decidir cuál modelo de arquitectura de software se ajustó al desarrollo. El diseño de alto nivel resultante conformó las bases para el desarrollo del framework. Como artefacto se obtuvo el diagrama de estados y transiciones del framework.
3. Diseño del Framework: La fase sirvió para realizar el diseño de alto nivel del framework obtenido en el paso anterior y se diseñaron las clases adicionales necesarias. El resultado de esta actividad, incorporó el alcance de las funcionalidades del framework y las reglas de diseño basadas en la arquitectura a usar. Al finalizar, se obtuvo el conjunto de diagramas de clase correspondientes.

Resultados

En este apartado se presentan como resultado, los artefactos más importantes generados durante la ejecución de cada fase de la metodología. Asimismo, se muestran los resultados de la instanciación del framework en el caso de estudio.

A. Análisis del Dominio.

Siguiendo el patrón de los tres ejemplos para el análisis, se revisaron tres aplicaciones para determinar la arquitectura general del framework y los requisitos. Éstas fueron:

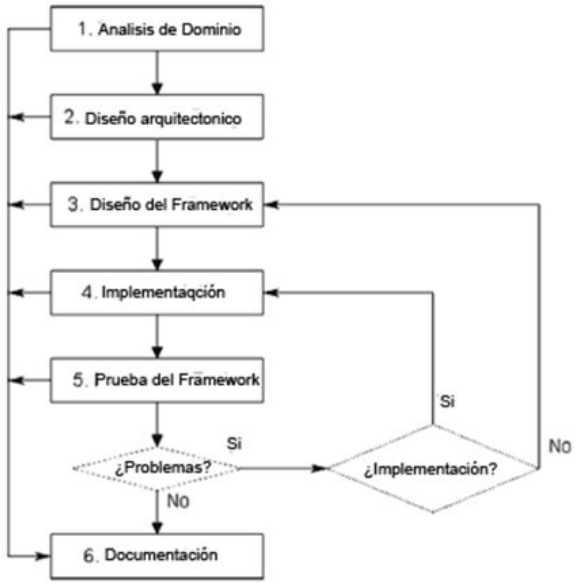

Figura 1. *Flujo de trabajo para el proceso de desarrollo del framework [14].*

1. El trabajo publicado por Sbaa y otros autores [15] describe el desarrollo de un sistema empotrado SMS-SQL para gestionar y consultar una base de datos, en el cual describe un sistema capaz de realizar consultas a bases de datos a través de comandos SMS usando un sistema basado en UNIX. Esta solución extiende las consultas de datos a las redes móviles de generaciones anteriores. Sin embargo, la solución es de difícil distribución y comercialización ya que precisa de la compra, configuración y programación de una computadora que sirve como SMS-SQL Gateway. A partir del trabajo de Sbaa se determina la factibilidad de establecer comunicación con una BD y también con un servidor por medio de SMS. La diferencia con esta investigación es que se desarrolla un framework flexible o adaptable para los programadores, partiendo de las necesidades de un conjunto de aplicaciones en un dominio en particular, el cual puede ser instanciado y adaptado para distintas aplicaciones. Adicionalmente, el modelo de dominio del trabajo de Sbaa en tres capas (teléfono, pasarela SMS-SQL y servidor de base de datos) es fundamental y base para el modelo de dominio planteado en la investigación.

2. En el mismo orden, Ozeki NG - SMS Gateway, es una herramienta que sirve de pasarela SMS-SQL y permite enviar y recibir mensajes SMS, usando un servidor de base de datos con la ayuda de consultas SQL [16]. Este trabajo plantea una solución para la gestión de mensajes de texto vía SMS, aportando bases para lograr la pasarela SMS-SQL que es uno de los componentes claves en el desarrollo del framework.

3. Por su parte, Wei Cui, LiaHuang, LiJing, en su trabajo, "The Research of PHP Development Framework Based on MVC Pattern" [17], plantearon la necesidad del uso de frameworks con estructura modular ya que facilita su uso, aumenta el rendimiento de la aplicación, mejora la mantenibilidad y aumenta la taza de reutilización de código, incrementando la eficacia del desarrollo de la aplicación. La modularidad es uno de los aspectos clave en el proceso de desarrollo del framework en cuestión. Es importante considerar el uso de patrones de diseño en el desarrollo de un framework, es por ello que Chanchai Supaartagorn, en su artículo, plantea la necesidad del uso del patrón MVC, por las razones siguientes [18]:

    1. Loosely-couples: Muchos tipos de componentes pueden interactuar de manera flexible.

    2. Desarrollo en Paralelo: Al tener una meta clara es posible dividir el sistema en componentes para que personas distintas puedan hacer el desarrollo en paralelo.

    3. Expansibilidad: El controlador puede ser expandido simplemente añadiéndole un módulo nuevo, sin necesidad de modificar ninguna de las otras capas.

A partir del estudio de estas tres aplicaciones se configuraron en alto nivel, los componentes del framework y los aspectos arquitectónicos del mismo.

Descripción de los Componentes del Framework

En este sentido y tomando como base el trabajo de Sbaa [15], se definieron los componentes requeridos para implantar una aplicación bajo el sistema operativo Android, el cual actúa como servidor a las peticiones recibidas vía SMS. Esta parte del frame-

work contempló el funcionamiento de la pasarela SMS-SQL, pero adaptado al manejo de peticiones HTTP a fin de realizar consultas al servidor.

A continuación, los componentes y sus funcionalidades:

SMSJSONParser: Se encarga de traducir el mensaje de texto en una petición HTTP la cual retorna su resultado en formato JSON para luego ser enviada a la aplicación cliente.

IncomingSMS: Este es el componente que escucha por mensajes de texto entrantes. Cuando un mensaje realiza una petición al servidor, éste la dirige al SMSJSONParser para que lo maneje.

SmsManager: Este componente es el encargado de enviar la respuesta en formato JSON al usuario vía SMS, para que la aplicación cliente la traduzca. Asimismo, se desarrollaron las clases que pueden ser usadas por los programadores para que sus aplicaciones hagan consultas mediante textos SMS, cuando no logren establecer conexión a Internet.

Las clases principales son:

JSONParser: Esta clase se encarga de realizar la petición HTTP y retornar el resultado en formato JSON; de no haber conexión a Internet, se le pasa la petición al SMSManager. ConnectionDetector: Esta clase es creada por el JSONParser al momento de hacer la petición HTTP para saber si hay conexión a Internet.

SMSManager: Envía las peticiones HTTP por textos SMS al servidor.

IncomingSMS: Esta clase se encarga de recibir los mensajes que envía el servidor en formato JSON, como respuesta a las consultas enviadas y como si se recibieran de +una petición HTTP hecha vía Internet.

DatabaseHelper: Esta clase ofrece las operaciones básicas para sincronizar la información obtenida del servidor y así contar con registro de la información de manera local en el teléfono. Esta clase es completamente modificable para que se pueda adaptar al modelo de negocio del usuario.

Cabe señalar, que para solventar la dificultad de Android para realizar consultas SQL directamente en bases de datos externas, se desarrolló una API REST que proporciona respuesta a las solicitudes HTTP y retorna los resultados en formato JSON. De esta manera, el programador puede realizar las operaciones a la base de datos que más se adapten a sus necesidades.

Tomando las consideraciones propuestas anteriormente, se presenta en la Figura 2, el Diagrama de modelado del Dominio, el cual establece los requisitos del framework.

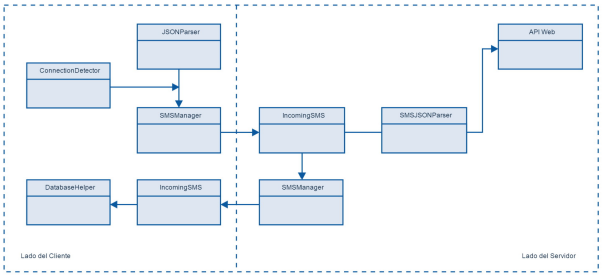

Figura 2. *Diagrama de modelado del Dominio. Fuente: Autor.*

B. Diseño Arquitectónico

La Figura 3 presenta la estructura física del framework, cuyos componentes se describen a continuación:

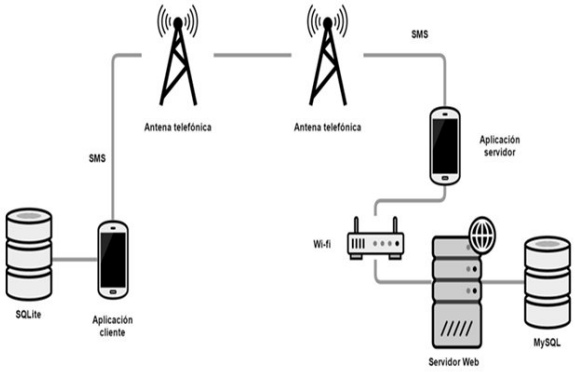

Figura 3. . *Estructura Física del framework. Fuente: Autor.*

Conexiones de red: Son manejadas a través de antenas telefónicas, las cuales permiten que exista una conexión GSM para el envío de mensajes de texto por SMS. Por otro lado, el wireless router (en la Figura 3, Wi-fi), permite la conexión del teléfono con el servidor en red local.

Aplicación cliente: La aplicación del lado del cliente se adapta al modelo del negocio para el cual el programador desarrolla sus aplicaciones. Es importante

señalar, que las aplicaciones podrán hacer uso de clases intermediarias, para establecer las conexiones HTTP y para realizar sincronizaciones y consultas a la base de datos. Asimismo, las conexiones que dependen de Internet para poder realizar las peticiones al servidor, utilizan también estas clases intermediarias para tener la opción de establecer las conexiones vía SMS, cuando la conexión a Internet falle.

SQLite: Es el gestor de base de datos que se instala de forma nativa en sistemas Android, se utilizó para crear una base de datos con la información necesaria para la sincronización, de esta manera puede ser accedida sin conexión a otras redes.

Aplicación servidor: Es uno de los componentes principales del framework y se encarga de recibir las peticiones HTTP en textos SMS, para luego realizar la consulta en la red local, tal y como lo haría la aplicación cliente si tuviese Internet. Al recibir la respuesta del servidor Web en formato JSON, la aplicación servidor le envía el JSON a la aplicación cliente. Si el JSON tiene más de 160 caracteres, este se divide en tantos SMS como sea necesario. Además, la aplicación cuenta con un sistema de usuarios, para restringir el acceso a aquellos no autorizados, estableciendo un esquema básico de seguridad para autorizar las consultas a la base de datos, utilizando mensajes de texto.

API Web: Es la encargada de hacer posible las consultas a la base de datos. Para que las aplicaciones puedan realizar consultas, debe haber una API que responda a éstas por medio de peticiones HTTP. Estas peticiones sirven para ejecutar los procesos que presentan los resultados de las consultas a la base de datos en formato JSON, para que posteriormente la aplicación cliente maneje la información con mayor facilidad. El servidor debe estar configurado de forma tal que, el programador pueda consultar su base de datos con facilidad.

Gestor de Base de Datos MySQL: Es el gestor de base de datos utilizado en el desarrollo del proyecto por diversas razones, entre ellas, es un software libre de código abierto, es de uso común en una amplia gama de modelos de negocio por lo que se espera, contribuya con la factibilidad de que el framework sea utilizado por un mayor número de usuarios. Además de su uso generalizado a nivel de mercado, esta herramienta es utilizada por un grupo considerable de desarrolladores, por lo cual existe suficiente documentación y grupos de colaboración para el soporte. A pesar de que MySQL no cumple con la conformidad del estándar SQL ANSI ISO/IEC 9075, es su velocidad a la hora de realizar las operaciones y el bajo consumo de recursos CPU y memoria, lo que le hace uno de los gestores que ofrecen mayor rendimiento. En el caso específico de la investigación, la empresa caso de estudio utiliza este gestor. Sin embargo, es importante señalar que el uso de MySQL no limita la posibilidad que tiene el programador de adaptar sus API WEB, para que realicen consultas en los gestores de base de datos de su preferencia.

Adicionalmente, la Figura 4 plantea el Diagrama de Estado del framework, con el fin de mostrar con detalle, los componentes y funcionamiento del framework

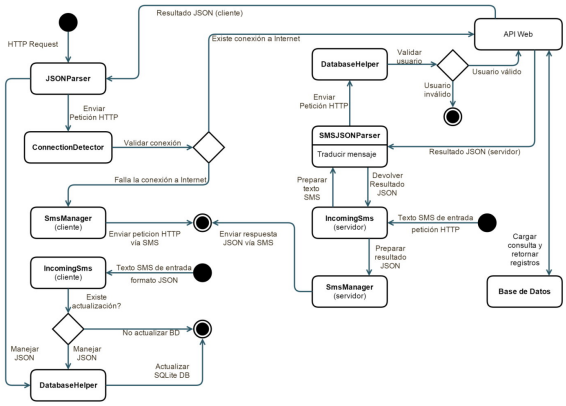

Figura 4. *Diagrama de Estado del framework. Fuente: Autor.*

Finalmente, la arquitectura planteada presenta una alternativa proclive a que la aplicación establezca una conexión segura. Asimismo, el diseño arquitectónico del framework está conformado por componentes intercambiables y modificables que le permiten al programador hacerse de su uso, para solventar los problemas de comunicación de sus aplicaciones.

C. Diseño del Framework

A continuación se definen las clases en Java para Android, como repertorio a ser usado por el programador al desarrollar su propia aplicación. Las clases son clasificadas de acuerdo con Bosch en core, add-ins y hot spots; para facilitar al programador su uso y adaptación al modelo del negocio . *Clases que conforman la Aplicación Servidor*

SMSJSONParser: Esta clase se encarga de realizar

la petición HTTP recibida por SMS. Del lado del servidor la conexión sehace en red local, por lo que es una conexión más segura (Figura 5).

Figura 5. *Diagrama de clase SmsJSONParser. Fuente: Autor.*

Métodos

- Construc: constructor de la clase SmsJSONParser
- makeHttpRequest: recibe el mensaje de texto con la petición HTTP y la traduce para realizar la conexión, luego retorna el resultado en un JSONObject.

Las siguientes clases extienden a menudo la clase Activity, propia de Android para agregar y ajustar a las necesidades de esta aplicación.

Configuracion: Esta clase hereda de la clase Activity, tiene como propósito llevar la configuración de las direcciones a las cuales se conectan las aplicaciones para que la aplicación pueda hacer la conexión en red local, con el servidor Web (Figura 6).

Figura 6. *Diagrama de clase Configuracion. Fuente: Autor.*

Atributos

- inputDomain: Nombre del dominio al cual accede la aplicación cliente en un ambiente sin problemas de conexión a Internet.
- inputLocalServerIP: Dirección local del servidor Web. IncomingSMS: Esta clase hereda de BroadcastReceiver, es decir, la clase es un receiver que va a estar escuchando a partir de mensajes de texto entrantes. Esta clase tiene la funcionalidad de recibir las peticiones HTTP enviadas por SMS desde el cliente. (Figura 7)

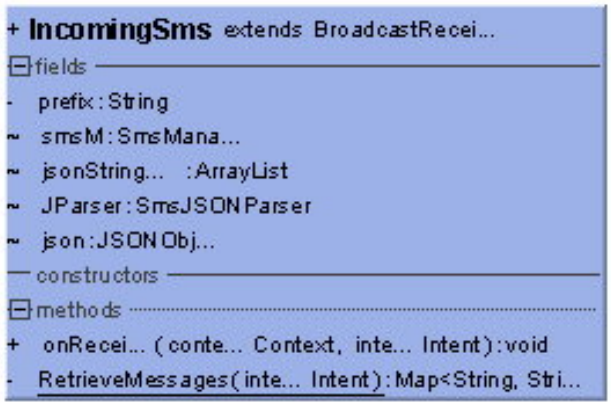

Figura 7. *Diagrama de clase IncomingSMS. Fuente: Autor.*

Atributos

- prefix: Es el prefijo con el cual la aplicación cliente conoce que el servidor envía una respuesta en formato JSON.

Métodos

- RetrieveMessages: Medio por el cual la clase recibe los mensajes de entrada para ser procesados.

DatabaseHelper: Esta clase maneja las consultas a las tablas de usuarios y de configuraciones. Los usuarios que estén en la base de datos son los autorizados a realizar las peticiones HTTP vía SMS (Figura 8)

Atributos

- Context: El contexto desde el cual se instancia la clase.

Métodos

- Construc: Construtor de la clase DatabaseHelper.
- createUser: Crea un usuario y retorna su ID.

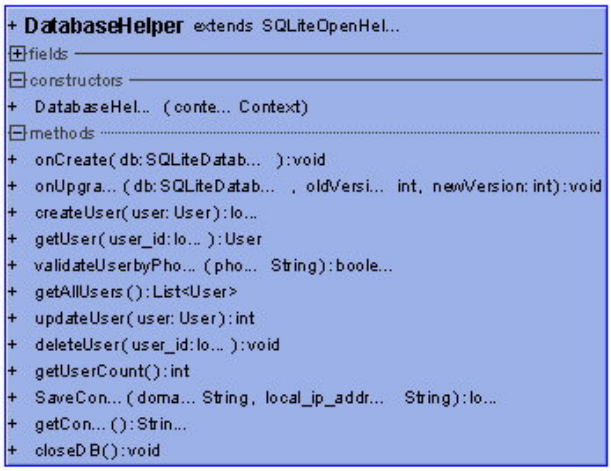

Figura 8. *Diagrama de clase DatabaseHelper. Fuente: Autor.*

- getUser: Retorna el usuario dado su ID correspondiente.
- validateUserByPhone: Comprueba que exista un usuario con un número de teléfono dado.
- getAllUsers: Retorna todos los usuarios en una lista.
- updateUser: Actualiza al usuario enviado por parámetro.
- deleteUser: elimina el usuario dado su ID correspondiente.
- SaveConfig: guarda la configuración de las direcciones para conectar al servidor web.
- getConfig: Obtiene la configuración guardada en la base de datos.
- closeDB: cierra la conexión con la base de datos.

User: Esta clase representa a los usuarios del sistema que tienen permitido enviar peticiones HTTP por medio de SMS. (Figura 9)

Atributos

- name: Nombre del usuario
- phone_number: Número de teléfono del usuario

Métodos

- Construc: Cuenta con tres constructores para inicializar el usuario con los valores en nulo o con valores predefinidos.
- setId: Modifica el ID del usuario.
- setName: Modifica el nombre del usuario.
- setPhone_number: Modifica el número de teléfono del usuario.
- getID: Devuelve el valor del ID del usuario.
- getName: Devuelve el valor del nombre del usuario.
- getPhone_number: Devuelve el número de teléfono del usuario.

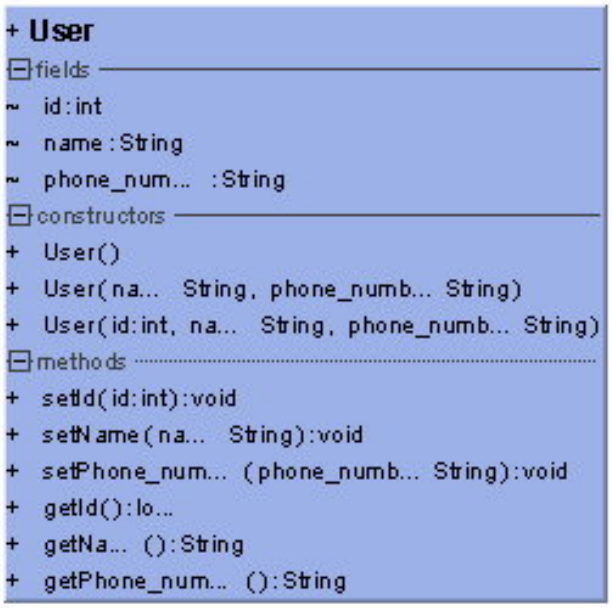

Figura 9. *Diagrama de clase User. Fuente: Autor*

NuevoUsuario: Hereda de la clase Activity, esta clase se encarga de recibir los parámetros para crear un usuario nuevo por pantalla e ingresarlo a la base de datos local. (Figura 10).

Atributos

- inputNombre: Nombre del nuevo usuario.
- inputNumeroTelefono: Número de teléfono del usuario a crear.

EdUsuario: Hereda de la clase Activity, esta clase se encarga de recibir los parámetros para editar un usuario por pantalla y actualizarlo en la base de datos local. (Figura 11)

Atributos

Figura 10. *Diagrama de clase NuevoUsuario. Fuente: Autor*

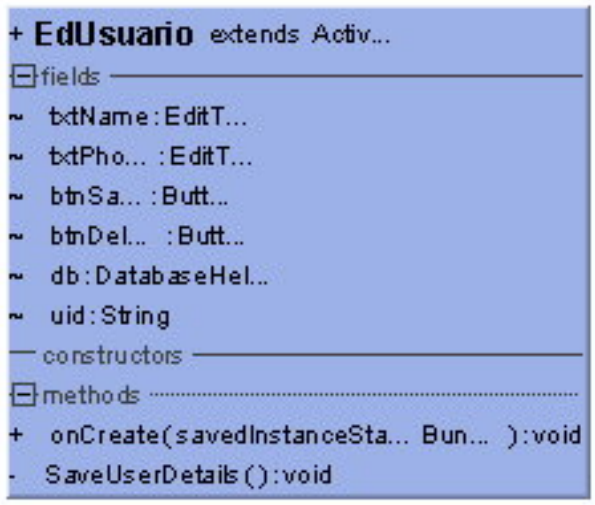

Figura 11. *Diagrama de clase EdUsuario. Fuente: Autor*

- inputNombre: Nuevo nombre del usuario.
- inputNumeroTelefono: Nuevo número de teléfono del usuario.

Métodos

- SaveUserDetails: Guarda los cambios del usuario a modificar.

TodosLosUsuarios: Esta clase hereda de ListActivity y se encarga de mostrar por pantalla una lista de todos los usuarios registrados en la base de datos local. (Figura 12)

ConnectionDetector: Es la clase encargada de verificar la conexión a Internet, permitiendo que todo el proceso pueda transcurrir sin que la aplicación cliente se detenga por problemas de conexión a Internet. (Figura 13)

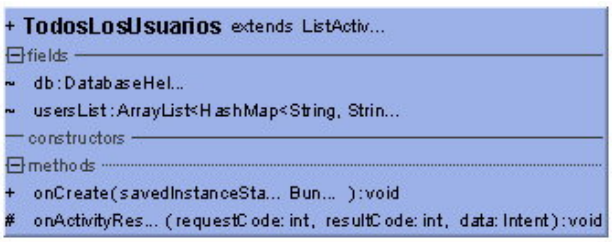

Figura 12. *Diagrama de clase TodosLosUsuarios. Fuente: Autor*

Figura 13. *Diagrama de clase ConnectionDetector (Fuente Autor)*

Atributos

- context: Contexto en el cual fue creado el objeto que instancia la clase, este contexto es necesario para poder verificar el estado de la conexión

Métodos

- Construc: Pasa como parámetro el context para poder construir el objeto.
- isConnectingToInternet: Función que activa la clase, por este medio se sabrá si el teléfono está conectado o no a Internet.

DatabaseHelper: Esta clase provee al programador de operaciones de base de datos con funciones predefinidas, fáciles de adaptar a su modelo de negocio. En este caso se desarrollaron procedimientos y funciones relacionadas al funcionamiento de un inventario. De esta manera, el programador tendrá un repertorio de operaciones básicas de inserción, modificación, eliminación y consulta, que normalmente se realizarían en un inventario sencillo (Figura 14).

La clase hereda de SQLiteOpenHelper la cual sirve de ayuda para gestionar la creación de la base de datos y las correspondientes versiones.

Atributos

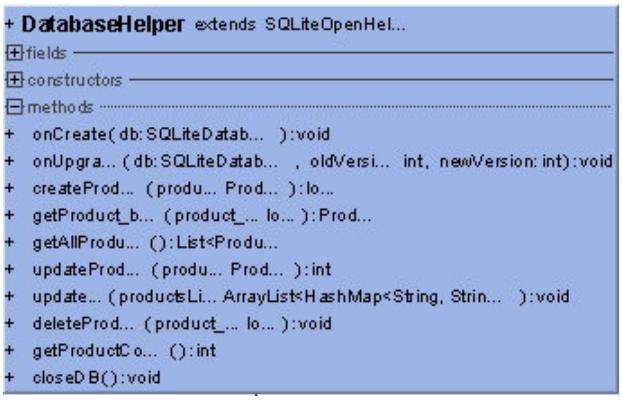

Figura 14. *Diagrama de clase DatabaseHelper. Fuente: Autor*

- context: Contexto en el cual fue creado el objeto que instancia la clase.
-

Métodos

- Construct: Recibe el context para la creación de la base de datos.
- createProduct: Crea un producto y retorna la posición en la cual se registró.
- getProductById: Retorna el producto consultando por su ID.
- getAllProducts: Devuelve todos los productos en una lista.
- UpdateProduct: Actualiza el producto pasado por parámetro.
- updateAllProducts: Actualiza todos los productos a partir de una lista de productos.
- deleteProduct: Elimina el producto que corresponda con el ID pasado como parámetro.
- closeDB: Cierra la conexión con la base de datos.

IncomingSMS: Hereda de BroadcastReceiver, es decir, la clase es un receiver que escucha por mensajes de texto entrantes. Es utilizada para recibir los resultados en JSON enviados por el servidor. (Figura 15)

Atributos

- ServerNumber: Atributo a ser modificado con el número de teléfono que tenga asignada la aplicación servidor, para poder aceptar y procesar los resultados que lleguen vía SMS.
- prefix: Prefijo con el que llegan las respuestas del servidor, permite diferenciar un resultado en JSON de un mensaje normal

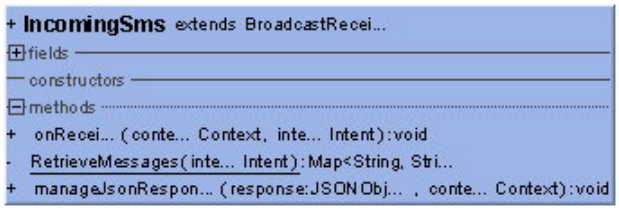

Figura 15. *Diagrama de clase IncomingSMS. Fuente: Autor*

Métodos

- retrieveMessages: Recupera los mensajes a ser procesados.
- manageJsonResponse: Lee el JSON y gestiona los cambios correspondientes para efectuar la sincronización.

SONParser: Se encargara de realizar las peticiones HTTP y de no ser posible establecer la conexión, envía la consulta vía SMS. (Figura 16)

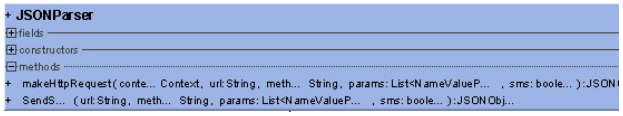

Figura 16. *Diagrama de clase JSONParser. Fuente: Autor*

Atributos

- ServerNumber: Número de teléfono asignado en el servidor, para poder enviar la consulta por mensaje de texto SMS.

Métodos

- Construc: Constructor para la clase JSONParser.
- makeHttpRequest: Realiza la petición HTTP vía Internet, si el parámetro sms es verdadero entonces envía la petición HTTP vía SMS, si la conexión a Internet falla.
- SendSMS: Envía el mensaje con la petición HTTP y los parámetros al servidor.

Product: Es la clase de ejemplo del framework, representa una recomendación para trabajar orientado a objeto, de manera que las actualizaciones a la base de datos se realicen en forma rápida. Dependiendo del modelo de negocio, esta clase puede desaparecer o adaptarse a sus necesidades al igual que las clases asociadas. (Figura 17)

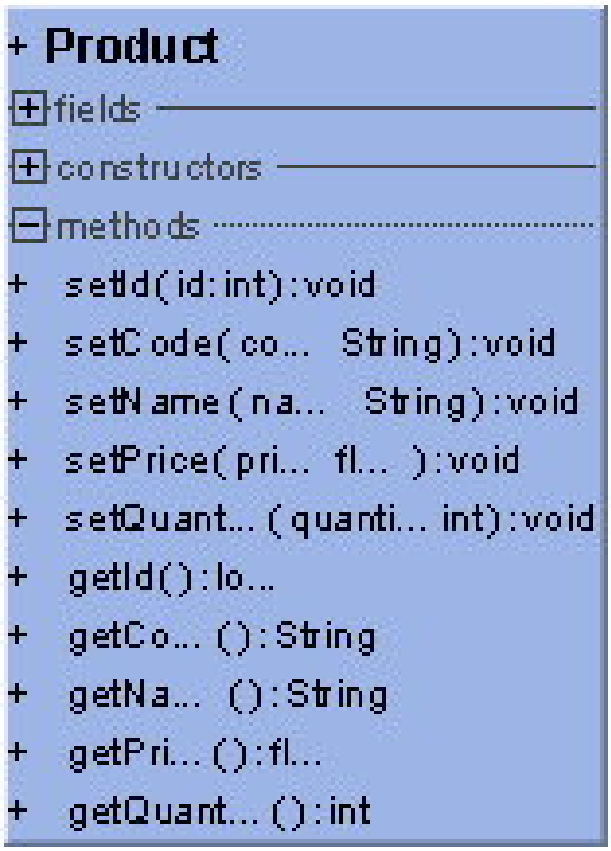

Figura 17. *Diagrama de clase Product. Fuente: Autor*

Atributos

- Id: Id del producto.
- Code: Código del producto.
- Name: Nombre del producto.
- Price: Precio del producto.
- Quantity: Cantidad de ese producto.

Métodos

- Construc: Cuenta con dos constructores para inicializar el producto con los valores en nulo o con valores predefinidos por el usuario.
- setId: Método para modificar el ID del producto.
- setCode: Método para modificar el código del producto.
- setName: Método para modificar el nombre del producto.
- setPrice: Método para modificar el precio del producto.
- setQuantity: Método para modificar la cantidad del producto.
- getId: Retorna el ID del producto.
- getCode: Retorna el código del producto.
- getName: Retorna el nombre del producto.
- getPrice: Retorna el precio del producto.
- getQuantity: Retorna la cantidad del producto.

NuevoProducto: Esta clase al igual que la clase Product dependen del modelo de negocio del programador y puede ser descartada o modificada para el mejor uso del framework. La clase hereda de la clase Activity para poder interactuar con el usuario mediante EditTexts y a su vez está compuesta por la clase CreateNewProduct que hereda de la clase AsyncTask, esta última realizara la conexión con el servidor en segundo plano de la aplicación.(Figura 18)

Figura 18. *Diagrama de clase NuevoProducto. Fuente: Autor*

Atributos

- txtCode: Lee el valor escrito por el usuario en pantalla para el código de un producto.
- txtName: Lee el valor escrito por el usuario en pantalla para el nombre del producto.
- txtPrice: Lee el valor escrito por el usuario en pantalla para el precio del producto.
- txtQuantity: Lee el valor escrito por el usuario en pantalla para el nombre del producto.

- url: Representa la dirección hacia la cual se realiza la petición HTTP para crear el producto.

EditarProducto: Esta clase al igual que la clase Product dependen del modelo de negocio y puede ser descartada o modificada para el mejor uso del framework. La clase hereda de la clase Activity para poder interactuar con el usuario mediante EditTexts y a su vez está compuesta por tres clases que heredan de la clase AsyncTask, estas últimas tres clases realizan la conexión con el servidor en segundo plano de la aplicación, para realizar sus operaciones. (Figura 19)

Figura 19. *Diagrama de clase EditarProducto. Fuente: Autor*

Atributos

- txtCode: Lee el valor escrito por el usuario en pantalla para el código de un producto.
- txtName: Lee el valor escrito por el usuario en pantalla para el nombre del producto.
- txtPrice: Lee el valor escrito por el usuario en pantalla para el precio del producto.
- txtQuantity: Lee el valor escrito por el usuario en pantalla para el nombre del producto.
- url: Representa la dirección hacia la cual se realiza la petición HTTP para cada operación.

TodosLosProductos: La clase hereda de la clase ListActivity para poder desplegar una lista de productos al usuario, a su vez está compuesta por la clase CargarProductos que hereda de la clase AsyncTask, esta última realiza la conexión con el servidor en segundo plano de la aplicación. (Figura 20)

Atributos

- url: Representa la dirección hacia la cual se realiza la petición HTTP para cada operación.

Las Figuras 21 y 22 a continuación, resumen las clases y sus relaciones tanto para la aplicación del lado del cliente como del lado del servidor.

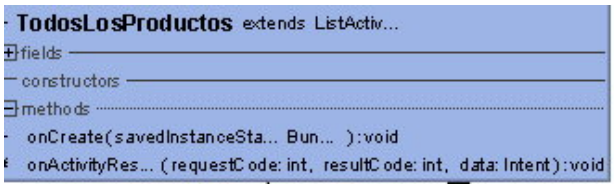

Figura 20. *Diagrama de clase TodosLosProductos. Fuente: Autor*

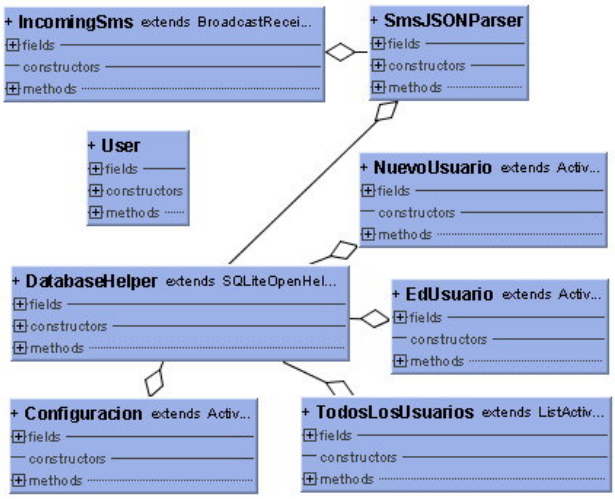

Figura 21. *Diagrama de clase de la aplicación servidor. Fuente: Autor*

Para la aplicación servidor, las clases se estructuran como:

- Core: SmsJASONParser, IncomingSms, User.
- Adds-in: DataBaseHelper.
- Hotspot: TodoslosUsarios, EdUsuario, NuevoUsuario.

Para la aplicación cliente, las clases se estructuran como:

- Core: JSONParser, IncomingSMS, ConnectionDetector, PantallaPrincipal, Product
- Adds-in: DataBaseHelper,
- Hotspot: NuevoProducto, TodoslosProductos, EditarProducto

D. Implementación

La fase de implementación del framework consistió en la generación del código fuente en sí mismo, el cual está disponible en su primera versión en el repositorio Git Hub, en el sitio: *https://github.com/augustotajar/SMSync-*

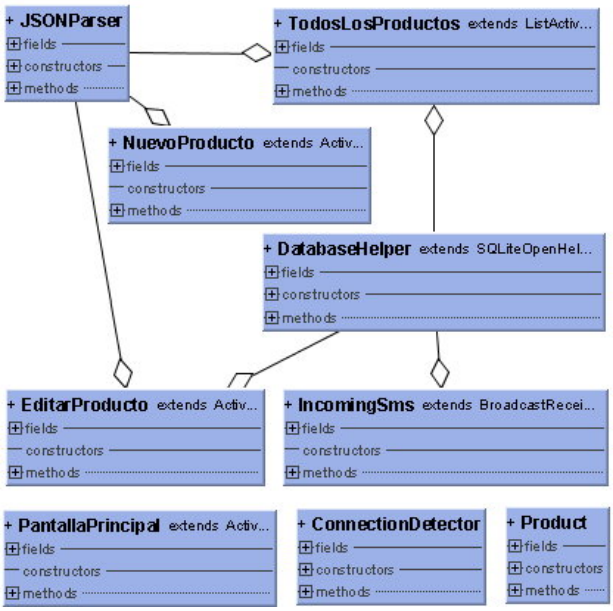

Figura 22. *Diagrama de clase de aplicación cliente. Fuente: Autor*

Framework.git. Es necesario mencionar en este punto, que el proceso de desarrollo de un framework es una tarea poco fácil puesto que no es posible eliminar el nivel de dependencia del dominio particular de las aplicaciones. Sin embargo, los desarrollos colaborativos apuntan a que con su uso se generen nuevas y mejores versiones, no solo del producto sino de su documentación.

E. Prueba del Framework

Para llevar a cabo las pruebas de funcionamiento del framework, se creó un microsistema de inventario con el cual se realizaron operaciones del tipo CRUD (create, read, update, delete). En este sentido, la Tabla 1 a continuación, describe los dispositivos hardware y el software utilizados.

Tabla 1. Dispositivos hardware y software utilizados en las pruebas. Fuente: Autor

Equipo	Marca/ Modelo	Sistema Operativo	Memoria RAM	Procesador
Teléfono móvil	Motorola Moto G2	Android 5.0.2 Lollipop	1 GB 1 GB	Snapdragon 400 quad-core a 1.2GHz
Teléfono móvil	LG G2 Mini	Android 4.4.2 Kitkat	1 GB 1 GB	Snapdragon 400 quad-core a 1.2GHz
Computador (Laptop)	ASUS S56C Series	Windows 8.1	6 GB 6 GB	Intel Core i5 1.7GHz

Es importante señalar que en el proceso de desarrollo del framework fueron tomados en cuenta aspectos de seguridad, al restringir el acceso a usuarios sin permisos, así como uno de los tópicos más importantes cuando se trabaja con dispositivos móviles, como lo es el criterio del espacio que ocupan las aplicaciones en dichos dispositivos. La Tabla 2 a continuación presenta el desempeño de las aplicaciones cliente y servidor, observándose en líneas generales, que el framework ocupa un espacio reducido en memoria RAM.

Tabla 2. Rendimiento en memoria de las aplicaciones cliente y servidor en los dispositivos móviles. Fuente: Autor

Almacenamiento	Motorola Moto G2	LG G2 Mini
Almacenamiento interno total	8 GB	8 GB
Almacenamiento disponible para la instalación	474 MB	2,62 GB
Espacio ocupado por la aplicación de pedidos de venta (Caso de Estudio)	4,83 MB (1,01 % del almacenamiento disponible, 0.06 % del total)	3,25 MB (0,12 % del almacenamiento disponible, 0.04 % del total)
Espacio ocupado por la aplicación cliente por defecto del framework	4,57 MB (0,96 % del almacenamiento disponible, 0.057 % total)	3,09 MB (0,11 % del almacenamiento disponible, 0.039 % del total)
Espacio ocupado por la aplicación servidor del framework	4,52 MB (0,95 % del almacenamiento disponible, 0.056 % total)	3,07 MB (0,11 % del almacenamiento disponible, 0.038 % del total)

Otro aspecto importante, lo constituye el hecho de que el framework está programado para escribir información en el log del sistema, de forma tal que, a medida que se ejecutan los procesos, pueden llevarse a cabo tareas de seguimiento y control de las aplicaciones. De esta manera, se puede realizar un análisis exhaustivo de la utilización de este nuevo medio y establecer un balance en la efectividad de la respuesta a las solicitudes del usuario.

Instanciación del Framework en un caso de estudio.

La instanciación en un caso de estudio se realizó a partir de una aplicación de pedidos de venta, la cual permite a los vendedores de una empresa comercializadora y mayorista de ferretería, la facilidad de realizar un pedido de venta desde una aplicación móvil.

A efectos de este artículo, solo se mostrará el caso de uso de la aplicación de pedidos de venta (Figura 23) y las pantallas de los dispositivos móviles comprobando la realización de operaciones de consulta de estatus de pedidos.

En la Figura 24 parte izquierda, se observa la estructura de archivos del framework con las clases

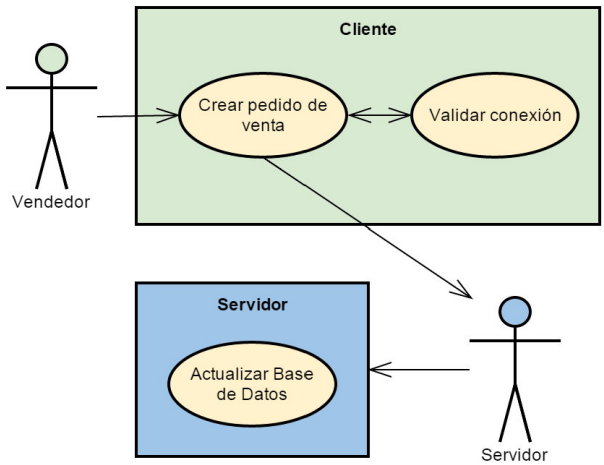

Figura 23. *Caso de uso de la aplicación Pedidos de ventas. Fuente: Autor*

predeterminadas, y en la parte derecha se puede ver que se han adaptado las clases al caso de estudio, para realizar operaciones de pedidos de venta. Estas clases en conjunto con las clases principales o core del framework, generan las nuevas tareas independientes del ejemplo original.

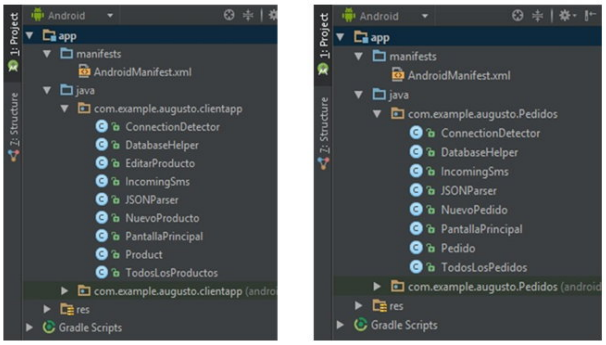

Figura 24. *Comparación de clases de la aplicación del lado del cliente, con la aplicación de Pedidos de venta. Fuente: Autor*

En las operaciones se utilizó la tabla de productos generada para el manejador de base de datos MySQL y creada para las pruebas (Figura 25).

En las pruebas se tomó en cuenta la existencia de los productos en el inventario para determinar si el pedido cambia de estatus “En proceso” o “Rechazado”, en la Figura 26 se tienen los pedidos y su estatus cuando son consultados vía SMS sin Internet.

	id	code	name	price	quantity	created_at
Editar Copiar Borrar	5	ESM001	Esmeril	555555.00	500	2015-03-24 12:39:43
Editar Copiar Borrar	49	ALI001	Alicate	999999.00	500	2015-04-04 14:51:00
Editar Copiar Borrar	51	TUB003	Tubería 1/2	999999.00	500	2015-04-04 14:51:00
Editar Copiar Borrar	52	TOB001	Tobo	999999.00	500	2015-04-04 14:51:00
Editar Copiar Borrar	53	LLA001	Llave Inglesa	999999.00	500	2015-04-04 14:51:00
Editar Copiar Borrar	54	MAN001	Manguera	999999.00	500	2015-04-07 09:57:59
Editar Copiar Borrar	69	TAL001	Taladro	99999.00	500	2015-05-08 23:36:30
Editar Copiar Borrar	70	SIE001	Sierra	99999.00	500	2015-05-08 23:46:36

Figura 25. *Base de Datos MySQL de productos de ferretería.*

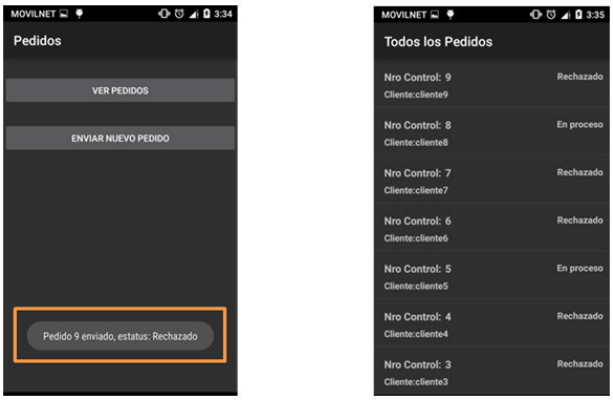

Figura 26. *Consulta de estatus de pedidos, sin conexión a Internet. Fuente: Autor.*

Conclusiones Trabajos futuros

El objetivo de este artículo consistió en mostrar los aspectos más relevantes en la construcción de un framework, orientado a un programador que necesite desarrollar aplicaciones con procesos de sincronización de información, usando herramientas que le permitan establecer conexiones por medios diferentes a la Internet desde dispositivos móviles. Esta problemática en la actualidad parece común en muchos países de Latinoamérica y específicamente en Venezuela, y a pesar de ello, es inminente el desarrollo de aplicaciones móviles como modelo de negocios emergente. El empleo de una estrategia sistemática basada en la metodología propuesta por Bosch, que incluye las fases: análisis del dominio, diseño arquitectónico y del framework, implementación, prueba del framework y documentación, permitieron lograr un producto en su primera versión cuyas pruebas cumplieron con el objetivo trazado. Específicamente, para las fases de análisis y diseño los autores utilizaron el patrón de tres ejemplos, a fin de obtener una abstracción de los componentes del framework, su interrelación y arquitectura. Asimismo, se desplegaron diagramas de modelado del dominio y diagrama de estado, los cuales ilustran con detalle los componentes y estructura del

framework. En estas fases se realizaron abstracciones con el fin de obtener las clases que se mantendrán sin cambios en cada aplicación generada y las que serán diferentes dependiendo del modelo de dominio específico. Esto permitirá al programador una visión de facilidad de uso, extensibilidad y adaptabilidad al modelo de negocio, con relativamente poco esfuerzo y facilidad de comprensión del mismo. Durante la instanciación del framework en la fase de pruebas, como era de esperarse, surgieron algunas interrogantes y dificultades que se sugieren como punto de partida para futuros trabajos:

- El uso excesivo de mensajes de texto tiene un costo asociado de acuerdo al servicio de telefonía al que esté suscrito el usuario, por lo tanto, para el tratamiento de consultas que generen una cantidad de mensajes considerable, deben estudiarse métodos alternativos.
- En caso de usar procesos automáticos de sincronización de información a partir del móvil, se recomienda que los tiempos entre sincronizaciones sean altos, de esta manera si el cliente permanece mucho tiempo sin conexión a Internet, se evitaría el consumo excesivo de mensajes de texto.
- Los aspectos de seguridad del framework están circunscritos a la autorización de usuarios. Por lo tanto, los aspectos correspondientes al cifrado de los mensajes no fueron tratados en este trabajo. Este pudiera ser un aspecto a reforzar en futuras versiones del framework, en caso de que se quisiera trabajar con datos bancarios o información estratégica de la empresa.
- Al ser un desarrollo utilizando los criterios de software libre, se recomienda profundizar en el estudio del código fuente, a fin de generar el mantenimiento correspondiente de la solución del software y alternativamente incluir nuevos elementos que permitan reducir el tiempo de desarrollo, ofreciendo la automatización de algunos procesos.

Bibliografía

[1] Johnson, R. E. and Foote, J. (1988). Designing Reusable Classes. Journal of Object Oriented Programming, 1(2):22-35.

[2] M. Mattsson, J. Bosch, Characterizing Stability involving Frameworks, In Proceedings of the 29th International Conference on Technology of Object-Oriented Languages and Systems, TOOLS EUROPE '99, Nancy, France, pp. 118-130, June.

[3] Dr Siniša Vlajić: Projektovanje programa (skripta), FON, Beograd 2004.

[4] Taligent (1994): Building Object-Oriented Frameworks, A Taligent White Paper. https://lhcb-comp.web.cern.ch/lhcbcomp/Components/postscript/buildingoo.pdf.

[5] V. Stanojevic S. Vlajic M. Milic and M. Ognjanovic. Guidelines for Framework Development Process. In Software Engineering Conference in Russia (CEE-SECR) 7th Central and Eastern European pp. 1-9 Nov 2011.

[6] Karch, M. Android for Work: Productivity for Professionals. Apress. 2010.

[7] Retford, B., and Schwartz, J. How to build an SMS Service. O'Reilly Media Inc., California, USA. 2007.

[8] Hillebrand F, Trosby F, Holley K. Short Message Service: The Creation of Personal Global Text Messaging. Chichester: Wiley. John Wiley & Sons. (2010).

[9] Booch, Grady. Designing an Application Framework. Dr. Dobb's Journal 19, Nro. 2. 1994.

[10] Marcus Markiewicz, Carlos J.P. Lucena: Object Oriented Framework Development, "Crossroads" Volume 7, Issue 4 (June 2001), ISSN: 1528-4972.

[11] Don Roberts, Ralph Johnson: Evolving Frameworks, A Pattern Language for Developing Object-Oriented Frameworks, http://stww.cs.illinois.edu/users/droberts/evolve.html.

[12] Xavier Amatriain: CLAM: A Framework for Audio and Music Application Development. IEEE Software 24(1) pp.82-85 (2007).

[13] M. Mattsson, J. Bosch, M. Fayad: Framework

Integration: Problems, Causes, Solutions, Communications of the ACM, Volume 42, Issue 10 (October 1999), pp. 80-87, 1999, ISSN: 0001-0782.

[14] Jan Bosch, Peter Molin, Michael Mattson, PerOlaf Bengtsson, and Mohamed E. Fayad (1997). Framework problems and experiences. In Mohamed E. Fayad, Douglas C. Schmidt, and Ralph E Johnson, editors, Building Application Frameworks: Object-Oriented Foundations of Framework Design, chapter 3, pp 55–82. Wiley, 1999.

[15] Sbaa, A. El Bejjet, R. & Medromi, H. An SMS-SQL based On-board system to manage and query a database. International Journal of Advanced Computer Science and Applications (IJACSA) Vol. 3, No.6 (2012:141).

[16] Ozeki Informatics Ltd. Ozeki NG - SMS Gateway. 2014. Recuperado el 14 de Octubrede 2014, de http://www.ozekisms.com/index.php?owpn=159.

[17] Wei Cui, Lin Huang, LiJing Liang, Jing Li (2009) "The Research of PHP Development Framework Based on MVC Pattern", Fourth International Conference on Computer Sciences and Convergence Information Technology (ICCIT), pp.947-949.

[18] C. Supaartagorn, "PHP Framework for database management based on MVC pattern", Department of Mathematics Statistics and Computer, Ubon Ratchathani University, Thailand, 2011.