PRACTICA EMPRESARIAL: DESARROLLO DE APLICACIÓN MÓVIL PARA LA FACTURACIÓN EN SITIO

JULIO FERNANDO MANTILLA CARRILLO

UNIVERSIDAD INDUSTRIAL DE SANTANADER
FACULTAD DE INGENIERÍAS FISICOMECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2017

PRACTICA EMPRESARIAL: DESARROLLO DE APLICACIÓN MÓVIL PARA LA FACTURACIÓN EN SITIO

JULIO FERNANDO MANTILLA CARRILLO

Informe Práctica Empresarial

Tutor
Jorge Flechas Suta
Arquitecto del Sistema SAC
ACTSIS LTDA

Director
Jaime Octavio Albarracín Ferreira

UNIVERSIDAD INDUSTRIAL DE SANTANADER
FACULTAD DE INGENIERÍAS FISICOMECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2017

CONTENIDO

LISTA DE TABLAS

Pag

LISTA DE ILUSTRACIONES

Pag

LISTA DE ANEXOS

RESUMEN

TITULO: PRACTICA EMPRESARIAL: DESARROLLO DE APLICACIÓN MÓVIL PARA LA FACTURACIÓN EN SITIO.

AUTOR: JULIO FERNANDO MANTILLA CARRILLO**

PALABRAS CLAVE: PRÁCTICA EMPRESARIAL, FACTURACIÓN EN SITIO, APLICACIÓN MÓVIL

DESCRIPCIÓN:

ACTSIS Ltda. una empresa colombiana, con más de 20 años de experiencia, líder en la prestación de servicios y soluciones específicas de sistemas de información para todo tipo de empresas con especial énfasis en empresas de servicios públicos, decide por medio del convenio con la Universidad Industrial de Santander dar la oportunidad de realizar una práctica empresarial con el objetivo del desarrollo de aplicación móvil para la facturación en sitio. Para llevar a cabo este proyecto fue necesario combinar los conocimientos y experiencia de la empresa en materia de facturación con una investigación de tecnologías móviles.

Este proyecto se divide en tres frentes, y en cada una de ellos se aplicaron las fases de análisis, diseño, construcción y pruebas. En el primero de ellos se realizó la construcción del módulo liquidador, este es un módulo independiente el cual se encarga de realizar la liquidación de los conceptos relacionado en una orden de trabajo (consumo energía, alumbrado público, impuesto, subsidios, etc.). En el segundo frente se realizó la construcción del servicio web el cual permite la comunicación entre la terminal móvil donde está instalada la aplicación y el servidor de base de datos. Finalmente, en el último frente se hizo la construcción de la aplicación móvil que se encarga de administrar los procesos de lecturas, liquidación, facturación, impresión y descarga de las órdenes de trabajo que se le asignan a una terminal.

***Trabajo de grado**

****Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingeniería de Sistemas e Informática. Director: Jaime Octavio Albarracín Ferreira, Ingeniero de Sistemas.**

ABSTRACT

TITLE: BUSINESS PRACTICE: DEVELOPMENT OF MOBILE APPLICATION FOR BILLING ON SITE.

AUTHOR: JULIO FERNANDO MANTILLA CARRILLO**

KEYWORDS: BUSINESS PRACTICE, BILLING ON SITE, MOBILE APPLICATION

DESCRIPTION:

ACTSIS Ltda., A Colombian company, with more than 20 years of experience, leader in the provision of services and specific solutions of information systems for all types of companies with special emphasis on public utilities, decides by means of the agreement with the University Industrial Santander to give the opportunity to conduct a business practice with the goal of developing mobile application for billing on site. To carry out this project was necessary to combine the knowledge and experience of the company in the field of billing with a research of mobile technologies.

This project is divided in three fronts, and in each one of them the phases of analysis, design, construction and tests were applied. In the first one the construction of the liquidator module was carried out. This is an independent module which is in charge of carrying out the liquidation of the related concepts in a work order (energy consumption, public lighting, tax, subsidies, etc.). In the second front was constructed the web service which allows communication between the mobile terminal where the application are installed and the database server. Finally on the last front was the construction of the mobile application that is responsible for managing the processes of readings, billing, printing and download of the work orders assigned to a terminal.

***Bachelor Thesis**

****Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingeniería de Sistemas e Informática. Director: Jaime Octavio Albarracín Ferreira, Ingeniero de Sistemas.**

INTRODUCCIÓN

Hoy en día se es consciente de la necesidad de utilizar nuevas tecnologías que permitan optimizar los procesos en las compañías. Los dispositivos móviles, brindan la posibilidad de disponer de la información en cualquier tiempo y lugar, adquiriendo una mayor eficiencia en los procesos y una mejora en la productividad, permitiendo así extender las capacidades de negocio de una empresa. Estos dispositivos, son adecuados para el uso en diferentes actividades tanto dentro de la empresa como fuera de ella, en los sectores de la industria, comercio, salud, gobierno, transporte y logística.

Es por lo anterior que la empresa ACTSIS LTDA ha conformado un equipo de trabajo para realizar el análisis, diseño y construcción de un prototipo de una aplicación móvil para realizar la facturación en sitio del servicio público de energía eléctrica.

1. ACTSIS LTDA

ACTSIS Ltda. es una empresa colombiana, con más de 20 años de experiencia, líder en la prestación de servicios y soluciones específicas de sistemas de información para todo tipo de empresas. Durante su trayectoria ha contado con más de 20 casos de éxito, liderando el sector de empresas de servicios públicos:

- 4 Acueductos
- 17 Electrificadoras
- 1 Municipio

ACTSIS Ltda cuenta con un recurso humano calificado y comprometido con la calidad de los productos ofrecidos, respaldado por su Sistema de Gestión de Calidad, certificado por ICONTEC hace más de 7 años.

1.1 MISIÓN

ACTSIS LTDA es una empresa privada que ofrece soluciones específicas en Sistemas de Información para todo tipo de empresas, contando con un grupo humano experimentado, calificado y comprometido en brindar un excelente servicio, utilizando tecnología de última generación, para satisfacer las necesidades y expectativas del cliente.

1.2 VISIÓN

Para el año 2020 ACTSIS LTDA aumentará su cubrimiento nacional destacándose por la calidad de sus productos y servicios, creando valor de manera permanente, buscando incursionar en el mercado internacional.

2. OBJETIVOS

2.1 OBJETIVO GENERAL

Realizar el análisis, diseño y construcción de un prototipo de una aplicación móvil para realizar la facturación en sitio al momento de tomar lectura del medidor a usuarios del servicio público de energía eléctrica.

2.2 OBJETIVOS ESPECÍFICOS

1. Definir los criterios de selección, aplicación, y utilización de plataformas y herramientas de desarrollo que se usaran en la construcción del prototipo.
2. Especificar todos los requerimientos de los usuarios y los requisitos del sistema, los cuales habrán de ser implementados en el prototipo de la aplicación.
3. Realizar el análisis, diseño y construcción de un prototipo de aplicación móvil que cumpla con los siguientes requisitos:

a) Descargar los datos desde un servidor de base de datos a la terminal móvil antes de iniciar los recorridos a los usuarios.

b) Capturar y validar los datos en el terreno mismo y en el momento mismo de la lectura de cada contador domiciliario.

c) Liquidar, facturar y mostrar factura del consumo del período leído en cada contador.

d) Cargar de regreso los datos de la terminal móvil al servidor de base de datos después de terminar los recorridos a los usuarios.

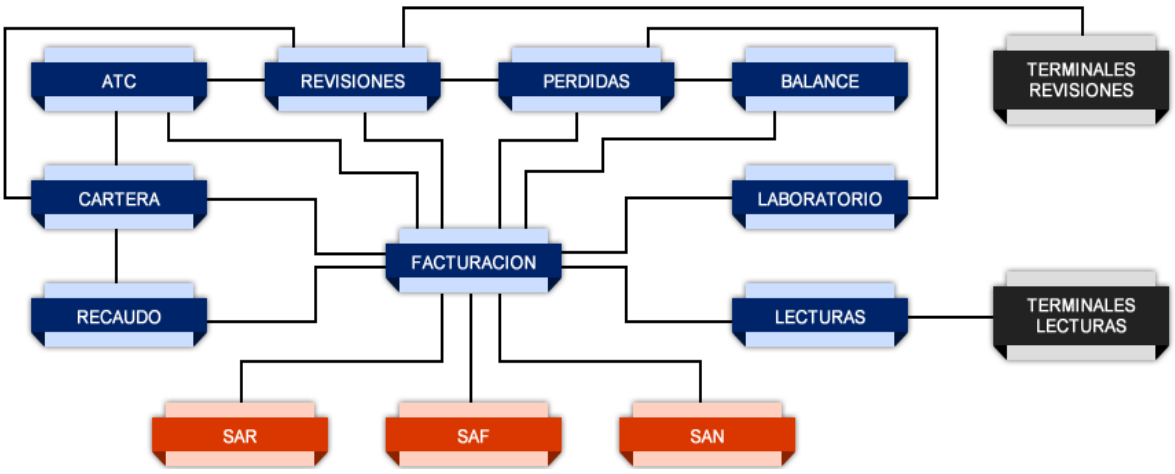
3. MARCO TEÓRICO – CONCEPTUAL

3.1 SAC

Sistema para la administración y operación de los procesos involucrados en la comercialización de servicios públicos (Energía, Agua, Alcantarillado, Aseo, Gas, entre otros). Permite gestionar en forma integral, sincronizada y controlada todas las acciones que se derivan de estos procesos y que se desarrollan en diferentes áreas de la organización.

El sistema incluye el cobro de consumos pre-pago y post-pago con tarifas por periodo, por franjas horarias, por horas y por horas-día. Para el servicio de energía eléctrica permite la liquidación de la penalización de la energía reactiva (factor de potencia). Adicionalmente posibilita el cobro de otros servicios o productos que factura la empresa ya sean propios o de terceros, registrando y suministrando la información en forma organizada y debidamente clasificada.

Ilustración 1, Módulos SAC



3.1.1 Facturación

- Esquema Tarifario
  - Parametrización de Conceptos, Valores, Tarifas, Conexos, etc.
- Agenda
- Mensajes para Recibos

- Parámetros de Crítica
- Procesos de Facturación
  - Generación de Libros
  - Crítica Automática
  - Actualización de Consumos
  - Liquidación
  - Impresión
  - Cierre
  - Los mismos procesos, pero de reversión.
- Verificación de Proceso de Facturación
  - Controles automáticos para cada una de las etapas del proceso de facturación
  - Estados de Clientes (Suscriptor y Facturación) y Medidores
- Ingreso masivo de Lecturas
- Ingreso masivo de Critica y Lecturas de Verificación
- Verificación de Lecturas y Crítica
- Ingreso individual de crítica

3.2 ANDROID

Es un sistema operativo basado en el núcleo Linux. Fue diseñado principalmente para dispositivos móviles con pantalla táctil, como teléfonos inteligentes, tablets o tabléfonos; y también para relojes inteligentes, televisores y automóviles. Inicialmente fue desarrollado por Android Inc., empresa que Google respaldó económicamente y más tarde, en 2005, la compró. Android fue presentado en 2007 junto la fundación del Open Handset Alliance para avanzar en los estándares abiertos de los dispositivos móviles. El primer móvil con el sistema operativo Android fue el HTC Dream y se vendió en octubre de 2008. Los dispositivos de Android venden más que las ventas combinadas de Windows Phone e IOS.

3.2.1 Android Studio. Es un entorno de desarrollo integrado para la plataforma Android. Fue anunciado el 16 de mayo de 2013 en la conferencia Google I/O, y reemplazó a Eclipse como el IDE oficial para el desarrollo de aplicaciones para Android. La primera versión estable fue publicada en diciembre de 2014. Está basado en el software IntelliJ IDEA de JetBrains, y es publicado de forma gratuita a través de la Licencia Apache 2.0. Está disponible para las plataformas Microsoft, Mac OS X y GNU/Linux.

3.3 ORACLE DATABASE.

Es un sistema de gestión de base de datos de tipo objeto-relacional desarrollado por Oracle Corporation. Se considera a Oracle Database como uno de los sistemas de bases de datos más completos, destacando: soporte de transacciones, estabilidad, escalabilidad, y soporte multiplataforma. Su dominio en el mercado de servidores empresariales había sido casi total hasta que recientemente tiene la competencia de Microsoft SQL Server y de la oferta de otros RDBMS con licencia libre como PostgreSQL, MySQL o Firebird. Las últimas versiones de Oracle han sido certificadas para poder trabajar bajo GNU/Linux.

3.4 SQLITE

SQLite es un sistema de gestión de bases de datos relacional compatible con ACID, contenida en una relativamente pequeña (~275 kiB) biblioteca escrita en C. SQLite es un proyecto de dominio público[1] creado por D. Richard Hipp. A diferencia de los sistemas de gestión de bases de datos cliente-servidor, el motor de SQLite no es un proceso independiente con el que el programa principal se comunica. En lugar de eso, la biblioteca SQLite se enlaza con el programa pasando a ser parte integral del mismo. El programa utiliza la funcionalidad de SQLite a través de llamadas simples a subrutinas y funciones. Esto reduce la latencia en el acceso a la base de datos, debido a que las llamadas a funciones son más eficientes que la comunicación entre procesos. El conjunto de la base de datos (definiciones, tablas, índices, y los propios datos), son guardados como un

sólo fichero estándar en la máquina host. Este diseño simple se logra bloqueando todo el fichero de base de datos al principio de cada transacción.
En su versión 3, SQLite permite bases de datos de hasta 2 Terabytes de tamaño, y también permite la inclusión de campos tipo BLOB.

3.4.1 Bulk insert. SQLite proporciona métodos de la clase SQLiteDatabase que se pueden utilizar para hacer todas las llamadas de inserción de un mismo lote en una sola transacción siguiendo el proceso de iniciar la transacción, realizar las operaciones de base de datos, confirmar la transacción y finalmente terminar la transacción. Para realizar este proceso se deben usar las siguientes instrucciones:

- **beginTransaction ():** Inicia una transacción en modo exclusivo.
- **compileStatement (String sql):** Compila una instrucción SQL en un objeto de declaración pre-compilado reutilizable. Más rápido cuando se utiliza una y otra vez.
- **setTransactionSuccessful ():** Confirma la transacción.
- **endTransaction ():** Finaliza la transacción.

3.5 TIPOS DE LIQUIDACIÓN DE ENERGIA ELECTRICA

3.5.1 Liquidaciones estándares. Las liquidaciones estándares contemplan la gran mayoría de los casos requeridos. El diseño propuesto está orientado a cumplir principalmente estos tipos de liquidaciones que cubren más del 95% de los clientes.

3.5.1.1 Consumo de activa. Toma el consumo calculado para la energía activa (el cual depende de una única lectura) y lo multiplica por el valor de la tarifa de activa (una única tarifa por cliente). Cuando existen múltiples tipos de energía activa (ej. alta, media, baja, en punta, fuera de punta) se debe manejar cada una de manera independientemente ya que cada una tiene un valor tarifario distinto.

3.5.1.2 CONSUMO DE REACTIVA

3.5.1.2.1 Basado en el totales. Toma la sumatoria de todos los consumos de reactiva y la divide por dos (2) y le resta la sumatoria de los consumos de activa. Si el resultante es mayor de cero (los valores negativos se vuelven 0) se multiplica por la tarifa de penalización de reactiva.

3.5.1.2.2 Basado en consumos independientes. Toma cada uno de los consumos de reactiva y lo divide por dos (2) y le resta el consumos equivalente de activa. Si el resultante es mayor de cero (los valores negativos se vuelven 0) se multiplica por la tarifa de penalización de reactiva. Esto se hace para cada uno de las parejas (activa-reactiva) de energía activa (ej. alta, media, baja, en punta, fuera de punta).

3.5.1.3 Subsidio. Toma el consumo calculado para la energía activa (el cual depende de una única lectura) y lo multiplica por el valor de la tarifa de activa multiplicada por un factor (valor negativo).

3.5.1.4 Contribución. Toma el consumo calculado para cada una de las energías activas y lo multiplica por el valor de la tarifa de activa correspondiente multiplicada por un factor (valor positivo). Se deberá crear un registro de contribución por cada una de las activas, pero en la factura solo se debe imprimir un registro (la sumatoria).

3.5.1.5 Valores fijos (cuotas de créditos, intereses, pagos diferidos). Cobrar un valor en pesos fijo.

3.5.1.6 Cobros de alumbrado público

3.5.1.6.1 Porcentaje del consumo activa. Toma el consumo calculado para la energía activa y lo multiplica por el valor de la tarifa deseada multiplicada por un factor.

3.5.1.6.1.1 Manejo de topes mínimos y máximos. Existe una variación al método anterior, incluyendo para cada rango un tope mínimo y tope máximo del valor a cobrar. Estos topes se deben aplicar después de calculado el valor.

3.5.1.6.2 Valor en pesos según el rango en que se encuentre el consumo de activa. Toma el consumo calculado para la energía activa y ubica dicho consumo en una matriz donde cada registro está definido por consumo desde y hasta; ubica el registro y retorna el valor definido para dicho registro. El valor retornado es el valor a liquidar.

3.5.1.6.3 Porcentaje del consumo de activa, cuyo porcentaje varía según el rango del consumo de activa. Toma el consumo calculado para la energía activa y ubica dicho consumo en una matriz donde cada registro está definido por consumo desde y hasta; ubica el registro y retorna el porcentaje para dicho registro. Multiplicada el consumo calculado para la energía activa por el valor de la tarifa de deseada multiplicada por el porcentaje recuperado según el rango.

3.5.1.6.3.1 Manejo de topes mínimos y máximos. Existe una variación al método anterior, incluyendo un tope mínimo y tope máximo del valor a cobrar. Estos topes se deben aplicar después de calculado el valor.

3.5.1.6.3.2 Manejo de topes mínimos y máximos según el rango de consumo. Existe una variación al método anterior, incluyendo para cada rango un tope mínimo y tope máximo del valor a cobrar. Estos topes se deben aplicar después de calculado el valor.

3.5.1.6.4 Porcentaje del valor neto de energía (energía – subsidio). Toma el valor calculado para la energía activa y le resta el subsidio aplicado a la misma, el resultado lo multiplica por un factor.

3.5.2 Liquidaciones especiales. Las liquidaciones especiales contemplan casos muy particulares requeridos para un número reducido de clientes, e incluso de empresas.

3.5.2.1 Áreas comunes

3.5.2.1.1 Áreas comunes con medidor independiente. Para centros comerciales, conjuntos y edificios. Se debe distribuir el consumo del área común entre todos los hijos, según el coeficiente.

3.5.2.1.1.1 El consumo del área común se cobra en concepto independiente. Para cada predio, se multiplica el consumo correspondiente (consumo del área común x el coeficiente del predio) por el valor de la tarifa correspondiente

3.5.2.1.1.2 El consumo del área común se cobra como parte del consumo de activa. Para cada predio, se suma el consumo correspondiente (consumo del área común x el coeficiente del predio) al consumo de activa y el total es usado para calcular el valor de la activa, subsidio, contribución y alumbrado.

3.5.2.1.2 Áreas comunes con medidor de frontera. Para algunos centros comerciales, conjuntos y edificios. Cuando el consumo del área común incluye los consumos de los hijos más el consumo propio del área común, se debe restar del consumo del área común la sumatoria de los consumos de los hijos (para evitar cobros dobles). El consumo resultante se utiliza de igual forma que en el punto anterior. (Nota: En algunos casos distribuyen la plata y no el consumo dentro los hijos).

3.5.2.2 Compensación por calidad del servicio. El sistema comercial debe calcular trimestralmente el valor máximo a compensar por calidad de servicio. Para el mes de aplicación, se debe multiplicar el consumo del mes por la tarifa de distribución, el valor a cobrar será el menor valor entre ésta multiplicación y el valor máximo previamente calculado.

3.6 HARDWARE PARA TRABAJO EN TERRENO

3.6.1 Terminales móviles. Hoy en día la palabra "Terminales móviles " sirve para describir un dispositivo móvil que opera con baterías y cubre la necesidad de capturar y registrar datos en lugares alejados de los Centros de cómputo. Las empresas que distribuyen productos para venta como refresqueras y repartidores de productos de consumo las utilizan olvidándose del lápiz el papel y los errores de captura. Se conocen como Terminales móviles que son operadas manualmente en cualquier lugar. Normalmente cuentan con lector de código de barras integrado, utilizan una batería recargable, gran capacidad de memoria de almacenamiento, 128MB, 256MB 512MB 1GB o más.

La mayoría de las terminales móviles utilizan los mismos componentes electrónicos que usted encuentra en un Pocket PC o PDA convencional, pero estas usan cubiertas duras, fuertes y están diseñadas para el trabajo a mano. Tienen diferentes formas algunas se ven como un PDA un poco más grande, otros tienen integrado una conveniente linterna propia de las anteriores generaciones de terminales de datos, algunos tienen empuñaduras de pistola para un fácil manejo, otros son comprimidos de pequeño tamaño. A la mayoría se le han integrado los escáneres de códigos de barras o láser y pueden utilizar cualquier cosa de tecnología mínima de CPU y muestra el estado de la técnica.

3.6.1.1 Características

Pantalla para exteriores

La mayoría de las terminales móviles serán utilizados al aire libre, y por lo tanto una pantalla que sea visible en exteriores es especialmente importante, a

diferencia de los portátiles comerciales donde la visibilidad exterior no se considera importante, PDAs y ordenadores de bolsillo han tenido pantallas visibles bajo la luz solar durante muchos años por lo que encontrar un dispositivo con una buena visibilidad es generalmente un problema menor. Sin embargo, la tecnología avanza, y estamos cada vez menos inclinados a aceptar las pantallas de bajo contraste o la reflexión excesiva y el deslumbramiento. La resolución también puede ser un problema. Durante años, los dispositivos Palm y Pocket PC utilizan pantallas de baja resolución (en su mayoría de 240 x 320 píxeles), pero a medida que las computadoras de mano se utilizan para aplicaciones más sofisticadas como los mapas o navegación, muchos dispositivos ahora tienen 480 x 640 pantallas a todo VGA y si las computadoras de mano industriales siguen los teléfonos inteligentes, que algunos sin duda, pronto veremos resoluciones que antes se creía impensable para estos dispositivos.

Diseño resistente

Computadoras de mano no son inmunes a las leyes de la física, pero gracias a su tamaño más pequeño, a su menos peso y la ausencia de unidades de disco duro, las hace capaces de soportar un castigo más extremo. Se han probado terminales que pueden sobrevivir a caídas de más de seis pies, y no sólo continuaron trabajando, incluso no se rayan o resultan pelados. Algunas están bien selladas para que puedan ser utilizadas en las lluvias torrenciales o incluso sumergidas en agua.

Ciclo de vida

Mientras que las electrónicas de consumo cambian muy rápidamente, ordenadores de mano industriales tienen un ciclo de vida mucho más larga. Muchos modelos están siendo utilizados de forma casi invariable año tras año. Sin embargo, el tiempo no se detiene y hay una serie de tecnologías que están encontrando su camino en las computadoras de mano y va a cambiar cómo se utilizan los dispositivos de mano. Algunas de estas tecnologías son GPS, radios inalámbricas de área amplia, RFID, lectores de imagen de alta calidad y escáneres y cámaras de alta resolución integrados. Ninguna de estas tecnologías es nueva,

pero sólo ahora están logrando auge y son lo suficientemente confiables para que puedan ser integrados en dispositivos de mano en lugar de ser periféricos a ellos. Esta integración, sin embargo, aumenta el costo y la complejidad.

Sistema operativo

Durante 2010 y 2011 surgieron los primeros dispositivos de mano industriales capaces de funcionar con Android, aunque el sistema operativo por defecto por lo general se mantuvo basado en Microsoft. A principios de 2012, la diferencia entre los dispositivos portátiles de consumo e industriales nunca había sido tan grande. Mientras que la gran mayoría de los dispositivos portátiles de consumo eran o iPhones o teléfonos inteligentes basados en Android, HTC, Samsung o LG, los dispositivos de mano industriales permanecieron basados en Windows CE o Windows Mobile. En 2013, se hizo cada vez más claro que Android se consideró una alternativa viable para terminales móviles e industriales. Varios fabricantes comenzaron a ofrecer versiones de Android de los dispositivos de mano seleccionados. En 2015, la mayoría de fabricantes de terminales móviles venden tanto los dispositivos Android y Windows. De estos últimos, la mayoría todavía utiliza Windows Embedded Handheld de 6.5, debido principalmente a la falta de una alternativa viable de Microsoft.

3.6.2 IMPRESORAS ZEBRA RW. Las impresoras Zebra RW420/RW220 de transferencia térmica directa están diseñadas para imprimir recibos y facturas. Son altamente resistentes y modulares, permite que los usuarios seleccionen entre varias opciones: inalámbricas, tarjetas lectoras, y accesorios, como soportes de anclaje a vehículos para simplificar la impresión en ruta. Su perfecta colocación y una pantalla angular permiten facilidad de uso durante el trabajo.Son impresoras portátiles fuertes y resistentes que cumplen con los estrictos requisitos de polvo y agua IP54, las RW420/RW220 pueden aguantar las más altas exigencias para la impresión en ruta: tickets, facturas, albaranes de entrega y más.

3.6.2.1 CPCL. El lenguaje CPCL permite al programador imprimir textos, gráficos, imágenes, códigos de barras y facilita la comunicación con las impresoras móviles Zebra. En este proyecto se usa especialmente la impresión de textos y códigos de barras.

3.7 SERVICIOS WEB

Es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de ordenadores como Internet. La interoperabilidad se consigue mediante la adopción de estándares abiertos. Las organizaciones OASIS y W3C son los comités responsables de la arquitectura y reglamentación de los servicios Web. Para mejorar la interoperabilidad entre distintas implementaciones de servicios Web se ha creado el organismo WS-I, encargado de desarrollar diversos perfiles para definir de manera más exhaustiva estos estándares. Es una máquina que atiende las peticiones de los clientes web y les envía los recursos solicitados.

3.7.1 REST. La Transferencia de Estado Representacional (Representational State Transfer) o REST es un estilo de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web. El término se originó en el año 2000, en una tesis doctoral sobre la web escrita por Roy Fielding, uno de los principales autores de la especificación del protocolo HTTP y ha pasado a ser ampliamente utilizado por la comunidad de desarrollo.

3.7.1.1 Descripción

- Un protocolo cliente/servidor sin estado: cada mensaje HTTP contiene toda la información necesaria para comprender la petición. Como resultado, ni el cliente ni el servidor necesitan recordar ningún estado de las comunicaciones entre

mensajes. Sin embargo, en la práctica, muchas aplicaciones basadas en HTTP utilizan cookies y otros mecanismos para mantener el estado de la sesión.

- Un conjunto de operaciones bien definidas que se aplican a todos los *recursos* de información: HTTP en sí define un conjunto pequeño de operaciones, las más importantes son POST, GET, PUT y DELETE. Con frecuencia estas operaciones se equiparan a las operaciones CRUD en bases de datos (ABMC en castellano: crear, leer, actualizar, borrar) que se requieren para la persistencia de datos, aunque POST no encaja exactamente en este esquema.

- Una sintaxis universal para identificar los recursos. En un sistema REST, cada recurso es direccionable únicamente a través de su URI.

- El uso de hipermedios, tanto para la información de la aplicación como para las transiciones de estado de la aplicación: la representación de este estado en un sistema REST son típicamente HTML o XML. Como resultado de esto, es posible navegar de un recurso REST a muchos otros, simplemente siguiendo enlaces sin requerir el uso de registros u otra infraestructura adicional.

3.7.2 SOAP. Es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. Este protocolo deriva de un protocolo creado por Dave Winer en 1998, llamado XML-RPC. SOAP fue creado por Microsoft, IBM y otros. Está actualmente bajo el auspicio de la W3C. Es uno de los protocolos utilizados en los servicios Web.

3.7.2.1 Características. Básicamente SOAP es un paradigma de mensajería de una dirección sin estado, que puede ser utilizado para formar protocolos más complejos y completos según las necesidades de las aplicaciones que lo implementan. Puede formar y construir la capa base de una "pila de protocolos de web service", ofreciendo un framework de mensajería básica en el cual los web services se pueden construir. Este protocolo está basado en XML y se conforma de tres partes:

- Sobre (envelope): el cual define qué hay en el mensaje y cómo procesarlo.
- Conjunto de reglas de codificación para expresar instancias de tipos de datos
- La Convención para representar llamadas a procedimientos y respuestas.

El protocolo SOAP tiene tres características principales:

- Extensibilidad (seguridad y WS-routing son extensiones aplicadas en el desarrollo).
- Neutralidad (SOAP puede ser utilizado sobre cualquier protocolo de transporte).
- Independencia (SOAP permite cualquier modelo de programación).

3.7.3 Procedimiento almacenado. Un procedimiento almacenado es un programa (o procedimiento) almacenado físicamente en una base de datos. Su implementación varía de un gestor de bases de datos a otro. La ventaja de un procedimiento almacenado es que al ser ejecutado, en respuesta a una petición de usuario, es ejecutado directamente en el motor de bases de datos, el cual usualmente corre en un servidor separado. Como tal, posee acceso directo a los datos que necesita manipular y sólo necesita enviar sus resultados de regreso al usuario, deshaciéndose de la sobrecarga resultante de comunicar grandes cantidades de datos salientes y entrantes.

3.7.4 Extensible markup language. Es un meta-lenguaje que permite definir lenguajes de marcas desarrollado por el World Wide Web Consortium utilizado para almacenar datos en forma legible. Proviene del lenguaje SGML y permite definir la gramática de lenguajes específicos para estructurar documentos grandes. A diferencia de otros lenguajes, XML da soporte a bases de datos, siendo útil cuando varias aplicaciones deben comunicarse entre sí o integrar información. XML no ha nacido únicamente para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable. XML es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.

3.7.5 Google maps api. Permite agregar mapas basados en datos de Google Maps a la aplicación. La API administra en forma automática el acceso a servidores, descargas de datos, visualización de mapas y respuesta a gestos de mapas de Google Maps. El API posee funciones para agregar marcadores, polígonos y superposiciones a un mapa básico, e instrucciones que permiten cambiar la vista del usuario de modo que se muestre un área del mapa en particular. Estos objetos proporcionan información adicional de ubicaciones en el mapa y permiten la interacción del usuario con este. La API te permite agregar los siguientes gráficos a un mapa:

- iconos anclados en posiciones específicas del mapa (marcadores);
- conjuntos de segmentos de líneas (polilíneas);
- segmentos cerrados (polígonos);
- gráficos de mapa de bits anclados en posiciones específicas del mapa (marcadores);
- conjuntos de imágenes que se muestran sobre los mosaicos de mapas básicos (superposiciones de mosaicos).

4. METODOLOGÍA

4.1 ETAPAS DEL PROCESO DE DESARROLLO DE SOFTWARE

ACTSIS LTDA siendo una empresa certificada en CMMI nivel tres en cada una de las fases de desarrollo utiliza las prácticas propuestas por este modelo para optimizar cada etapa de desarrollo del software y garantizar un resultado esperado.

Las etapas que se tendrán en cuenta para este proyecto serán las siguientes:

- ANALISIS
- ESPECIFICACIÓN TANTO DE REQUISITOS DEL SISTEMA COMO DE REQUERIMIENTOS DEL USUARIO
- DISEÑO
- CONSTRUCCIÓN
- PRUEBAS

ANÁLISIS

En esta etapa se hizo una investigación para escoger las herramientas que se usaran para llevar a cabo el proyecto. También se estudió el entorno en que se usará el prototipo de la aplicación de facturación en sitio. Además, se estudió la herramienta que actualmente se usa para la facturación por lectura. Finalmente se levantarán los requisitos que deberá cumplir el prototipo de la aplicación.

DISEÑO

Se diseñaron las estructuras de datos del prototipo de la aplicación y del programa liquidador que efectuara los cálculos necesarios para hacer la facturación, también se diseñaron los procesos que se deberán seguir durante la ejecución del programa. Finalmente se diseñó la interfaz que usaran los usuarios finales para

realizar la labor de descarga de datos tanto de las órdenes de trabajo como del proceso de liquidación, toma de lecturas y liquidación de las órdenes de trabajo.

CONSTRUCCIÓN

Se escogieron los entornos de programación Android Studio para la aplicación móvil y Visual Studio 2015 para el servicio web con los lenguajes de programación de Java y C# .NET versión 4.5 respectivamente. Se hicieron las construcciones pertinentes conforme al diseño hecho previamente.

PRUEBAS

Se harán las distintas pruebas necesarias para encontrar los posibles errores.

4.2 REQUISITOS DEL SISTEMA Y REQUERIMIENTOS DEL USUARIO

4.2.1 REQUISITOS DEL SISTEMA

El prototipo de la aplicación:

1. Deberá funcionar en terminales móviles con una capacidad de memoria RAM de 4 GB o superior.
2. Deberá funcionar en terminales móviles con un procesador de 1.6 GHz o superior.
3. Deberá funcionar en terminales móviles con un almacenamiento interno de 16 GB o superior.
4. Deberá funcionar en terminales móviles con un sistema operativo Android 4.0 o superior.

El servicio web:

5. Que permitirá la carga y descarga de datos entre la terminal móvil y el servidor de bases de datos, deberá funcionar para la versión 4.5 de .NET o superior.

4.2.2 REQUERIMIENTOS DEL USUARIO

El prototipo de la aplicación deberá:

1. Ser capaz de cargar datos de forma inalámbrica desde un servidor de base de datos hacia la terminal móvil.
2. Administrar el seguimiento de las órdenes de una ruta de lectura.
3. Permitir capturar y validar los datos de terreno de la lectura, la observación de anomalía o de no lectura de acuerdo a las condiciones del proceso en sitio.
4. Posibilitar la búsqueda de los datos de una orden.
5. Liquidar los valores a cobrar de acuerdo a los datos capturados en terreno y la información cargada para una determinada orden.
6. Consultar los valores liquidados.
7. Ser capaz de descargar los datos desde la terminal móvil al servidor de base de datos después de terminar los recorridos a los usuarios (lectores).

5. ACTIVIDADES REALIZADAS

La ejecución del proyecto se llevó a cabo en diferentes frentes y en cada una de ellos se aplicaron las fases de análisis, diseño, construcción y pruebas. Los frentes en los cuales se dividió el proyecto fueron:

- Modulo Liquidador
- Comunicación vía servicio web.
- Construcción de la aplicación móvil.

5.1 MODULO LIQUIDADOR

ANÁLISIS

- Estudió del sistema SAC desarrollado por ACTSIS del cual se extiende la modalidad de facturación en sitio y que es el responsable de generar los datos necesarios para ser usados en el módulo liquidador.
- Estudio y definición del modelo del módulo liquidador teniendo en cuenta los datos que son generados por el SAC.
- Estudio de la base de datos SQLite usada por el entorno Android Studio y las diferencias de tipos de datos que tiene con la base de datos Oracle que usa el sistema SAC.
- Estudio de los distintos tipos de liquidación que se deben manejar en la facturación en sitio.

DISEÑO

- Diseño del modelo datos del módulo liquidador definiendo las tablas, datos y llaves primarias y foráneas que en ellas estarán incluidas.
- Diseño de las funciones que conforman el motor principal para la interpretación de la información contenida en las tablas de la base de datos.

- Diseño de las funciones que se usarán para para el proceso de cálculo de la liquidación teniendo en cuenta los distintos tipos de liquidación y haciendo uso de los datos que se recibirán.

CONSTRUCCIÓN

- Construcción de las clases y funciones que se encargara de la creación y conexión de la base de datos y de las tablas en SQLite según el modelo definido en la etapa de diseño.
- Construcción de las funciones del motor principal de interpretación de información contenida en las tablas de la base de datos.
- Construcción de las funciones que efectuaran el proceso de cálculo de la liquidación teniendo en cuenta los distintos tipos de liquidación.

PRUEBAS

- Aplicación de una prueba manual sin usar el sistema SAC ingresando manualmente todos los datos necesarios para efectuar el proceso de liquidación.
- Aplicación de una prueba utilizando datos que fueron generados por el SAC.

5.2 COMUNICACIÓN VÍA SERVICIO WEB

ANÁLISIS

- Investigación de las alternativas de arquitecturas para la construcción de servicios web enfocados a la transferencia de grandes volúmenes de datos.
- Pruebas para la elección de la arquitectura que se usara en la construcción de servicio web.
- Elección de la opción de la arquitectura REST para la construcción del servicio web.
- Estudio de la librería KSOAP2 que permite la comunicación entre el servicio web y la aplicación móvil.

- Estudio de la librería propia de ACTSIS corenet.dll que facilita la comunicación entre el servicio web y la base de datos Oracle.
- Definición de los procesos almacenados que se encargaran de retornar los datos solicitados por el servicio web.
- Definición del modelo de datos necesario para efectuar el proceso de carga y descarga.
- Definición del esquema de seguridad que se usara para el acceso de la base de datos.
- Definición de los procesos de carga de datos hacia la terminal móvil y de descarga de datos hacia el servidor de base de datos.
- Investigación de los métodos disponibles en Android y SQLite para la inserción eficiente de grandes volúmenes de datos.
- Elección del método Bulk Insert Transaction para realizar las inserciones masivas de datos en la terminal móvil.

DISEÑO

- Diseño del proceso de carga de datos en la terminal móvil.
- Diseño de las clases y funciones que permiten:
  - Hacer las peticiones al servicio web desde la terminal usando la librería KSOAP.
  - Recibir e interpretar los datos enviados por el servicio web.
  - Insertar en la base de datos de la terminal los datos recibidos.
- Diseño de las funciones del servicio web que llaman a los procedimientos almacenados, reciben los datos, interpretan los datos y finalmente los envía a la terminal móvil.
- Diseño de los procedimientos almacenados que reciben parámetros del servicio web y retornan los datos solicitados al servicio web.

CONSTRUCCIÓN

- Construcción de las clases y funciones que permiten:
  - Hacer las peticiones al servicio web desde la terminal usando la librería KSOAP.
  - Recibir e interpretar los datos enviados por el servicio web.
  - Insertar en la base de datos de la terminal los datos recibidos.
- Construcción de las funciones del servicio web que llaman a los procedimientos almacenados, reciben los datos, interpretan los datos y finalmente los envía a la terminal móvil.
- Construcción de los procedimientos almacenados que reciben parámetros del servicio web y retornan los datos solicitados al servicio web.

PRUEBAS

- Aplicación de una prueba en la que se hace un cargue de datos masivo desde la base de datos hacia la terminal.
- Aplicación de una prueba en la que se hizo un descargue de datos masivo desde la terminal hacia la base de datos.

5.3 CONSTRUCCIÓN DE LA APLICACIÓN MÓVIL

ANÁLISIS

- Definición de los perfiles de usuarios que usaran la aplicación, se definen dos perfiles de usuarios los cuales son administrador y operador (lector).
- Definición de las actividades, permisos y opciones que tiene cada perfil de usuario.
- Definición del proceso de login en la aplicación y logout.
- Definición de los permisos (cargue, descargue y modificación de datos).
- Definición de las distintas pantallas que conforman la aplicación.
- Definición de los procesos de cargue, gestión, lectura, liquidación y descargue de las ordenes de trabajo.
- Definición de los botones de navegación entre las pantallas.

DISEÑO

- Definición de los estándares de fondos, fuentes y colores de fuentes usados en la aplicación.
- Diseño de las pantallas según el perfil del usuario tal como se definió en el análisis.
- Diseño de los botones y menús de la aplicación.
- Diseño del proceso de login y logout de la aplicación.
- Diseño de los procesos de cargue, gestión, lectura, liquidación y descargue de órdenes de trabajo.

CONSTRUCCIÓN

- Construcción de las pantallas según el perfil del usuario tal como se definió en el análisis.
- Construcción de los menús de la aplicación.
- Construcción de las clases y funciones de login y logout de la aplicación.
- Construcción de las clases y de funciones que se usan en los procesos de cargue, gestión, lectura, liquidación y descargue de órdenes de trabajo.

PRUEBAS

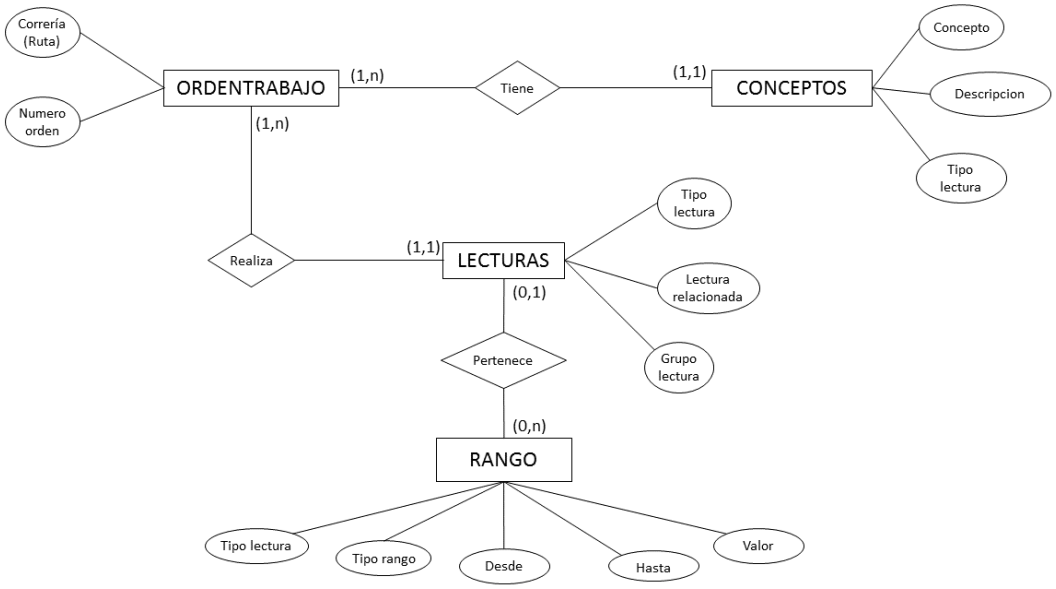
- Se realiza una prueba completa de la aplicación en donde se logre hacer la carga de las órdenes de trabajo, la lectura de las mismas, la liquidación y finalmente la descarga hacia el servidor de base de datos.

6. MODULO LIQUIDADOR

Se decide construir de manera separada un módulo que se encargue de interpretar los datos que son generados por el sistema SAC de ACTSIS (pre liquidación), que junto con las lecturas tomadas de un medidor de una orden de trabajo especifica generen la liquidación de conceptos de dicha orden de trabajo de manera inmediata.

6.1 MODELO DE DATOS

Ilustración 2, Modelo de datos liquidador



6.1.1 Tablas

Tabla 1, Tabla de datos FAC_GRUPOS_LECTURA

TIPO DE OBJETO	Tabla (Maestra - Parámetros)			
NOMBRE	FAC_GRUPOS_LECTURA			
OBJETIVO	Tabla de maestra de grupos de tipos de lecturas			
DESCRIPCIÓN	Almacena los grupos de lecturas que corresponden a múltiples tipos de lectura. Un tipo de lectura podría estar en más de un grupo.			
COLUMNA	**TC**	**O**	**TIPO DATO**	**DESCRIPCION**
codprograma	PK	S	TEXT	Código del programa de SIRIUS
idgrupolectura	PK	S	TEXT	Identificador del grupo de lectura
idtipolectura	PK	S	TEXT	Identificador del tipo de lectura
idtipolecturarelac	O	N	TEXT	Identificador del tipo de lectura relacionado (la activa equivalente para una reactiva)

Tabla 2, Tabla de datos FAC_LABORCONCEPTOS

TIPO DE OBJETO	Tabla (Datos. Entrada y Salida: I/O)			
NOMBRE	FAC_LABORCONCEPTOS			
OBJETIVO	Tabla de conceptos a facturar (y conceptos facturados)			
DESCRIPCIÓN	Almacena los conceptos por cada OT, detallando las variables requeridas para realizar la liquidación. Esta tabla debe ser enviada por el sistema comercial para cada una de las OTs. La terminal después de calcular los valores los debe retornar al sistema comercial.			
COLUMNA	**TC**	**O**	**TIPO DATO**	**DESCRIPCIÓN**
codprograma	PK	S	TEXT	Código del programa de SIRIUS
correria	PK	S	TEXT	Número de la correría
numorden	PK	S	TEXT	Numero único de la Orden de Trabajo.
idconcepto	PK	S	INTEGER	Identificador del concepto
orden	PK	S	INTEGER	Orden de liquidación del concepto (para crear llave primaría única y para futuros usos si se requiere un orden estricto de liquidación, es decir unos conceptos dependan de otros.
saldoanterior	O	S	TEXT	Saldo anterior del concepto
idgrupolectura	O	N	TEXT	Identificador del grupo de lectura. Solo para conceptos que dependen de un consumo (ej. Activa, Activa Alta/Media/Baja, Reactiva, Alumbrado, Subsidio, Contribución)
cantbase	O	N	TEXT	Base de la Cantidad
canttiporango	O	N	TEXT	Multitabla de Rangos para Cantidad
cantcodrango	O	N	TEXT	Código para ubicar el registro en la

				Multitabla de Rangos para Cantidad
cantrutina	O	N	TEXT	Función para ajustar Cantidad (después de base y rango).
cantidad	I/O	N	TEXT	Cantidad a facturar. Si es nulo debe calcularlo.
factbase	O	N	TEXT	Base del factor
facttiporango	O	N	TEXT	Multitabla de Rangos para Factor
factcodrango	O	N	TEXT	Código para ubicar el registro en la Multitabla de Rangos para Factor
factrutina	O	N	TEXT	Función para ajustar factor (después de base y rango).
factor	I/O	N	TEXT	Factor para facturar. Si es nulo debe calcularlo.
tarifabase	O	N	TEXT	Base del Tarifa
tarifatiporango	O	N	TEXT	Multitabla de Rangos para Tarifa
tarifacodrango	O	N	TEXT	Código para ubicar el registro en la Multitabla de Rangos para Tarifa
tarifarutina	O	N	TEXT	Función para ajustar Tarifa (después de base y rango).
tarifa	I/O	N	TEXT	Tarifa para facturar. Si es nulo debe calcularlo.
valorbase	O	N	TEXT	Base para el Valor
valortiporango	O	N	TEXT	Multitabla de Rangos para Valor
valorcodrango	O	N	TEXT	Código para ubicar el registro en la Multitabla de Rangos para Valor
valorrutina	O	N	TEXT	Función para ajustar Valor_Total (después de base y rango).
valor	I/O	N	TEXT	Valor a facturar del concepto. Si es nulo debe calcularlo.

Tabla 3, Tabla de datos FAC_CONCEPTOS

TIPO DE OBJETO		Tabla (Maestra - Parámetros)		
NOMBRE		FAC_CONCEPTOS		
OBJETIVO		Tabla maestra de conceptos		
DESCRIPCIÓN		Almacena los posibles conceptos y su parametrización		
COLUMNA	**TC**	**O**	**TIPO DATO**	**DESCRIPCION**
codprograma	PK	S	TEXT	Código del programa
idconcepto	PK	S	TEXT	Identificador del concepto
descripcion	PK	S	TEXT	Descripción del concepto
unidad	I	N	TEXT	Unidad que maneja el concepto (ej: KWh, M3, KVA)

Tabla 4, Tabla de datos FAC_RANGOS

TIPO DE OBJETO	Tabla (Maestra - Parámetros)			
NOMBRE	FAC_RANGOS			
OBJETIVO	Parametrización de valores para múltiples tipos de rangos			
DESCRIPCIÓN	Contiene la parametrización de múltiples tipos de rangos, donde cada tipo de rango está regido por un codrango y dentro de él se definen los rangos del desde y hasta para recuperar el campo Valor. Si Valor es nulo se debe asumir el valor de entrada.			
COLUMNA	**TC**	**O**	**TIPO DATO**	**DESCRIPCIÓN**
codprograma	PK	S	TEXT	Código del programa.
tiporango	PK	S	TEXT	Identificador del tipo de rango de valores

codrango	PK	S	TEXT	Código del rango para un tipo de rango de valores
desde	PK	S	INTEGER	Valor inicial del rango (valor base > desde)
hasta	PK	S	INTEGER	Valor final del rango (valor base < hasta)
valor	O	S	INTEGER	Valor del rango (si es nulo se debe asumir el valor de entrada)
valorresta	O	N	INTEGER	Valor a restar del rango. Normalmente es nulo o 0. Este valor se debe restar del Valor para obtener el valor real del rango.

6.2 CLASES

6.2.1 BD_FENS. Clase java extendida de SQLiteOpenHelper una clase de ayuda para gestionar la creación de bases de datos y gestión de versiones. En esta se encuentran las funciones que necesita el modulo liquidador para consultar y actualizar la base de datos de la terminal móvil durante el proceso de liquidación.

6.2.1.1 Variables

- **private static final String DB_NAME**

String que contiene el nombre de la base de datos de la terminal móvil.

- **private static final int SCHEME_VERSION**

Variable int que se usa para el control de versiones de la base de datos.

- **private SQLiteDatabase db**

Expone métodos para gestionar una base de datos SQLite.

6.2.1.2 Funciones BD_FENS

- **public BD_FENS(Context contex)**

Constructor de la clase, se usa para inicializar variable db.

- **public void onCreate(SQLiteDatabase db)**

Función propia de la clase SQLiteOpenHelper se ejecuta cuando se crea la base de datos no es usada en el módulo.

- **public void Update(String query)**

Ejecuta un comando SQL contenido en un String se usa para actualizar registros en la base de datos.

- **public Cursor cursor_consulta(String query)**

Ejecuta un comando de consulta SQL contenido en un String, retorna desde un registro hasta una tabla completa.

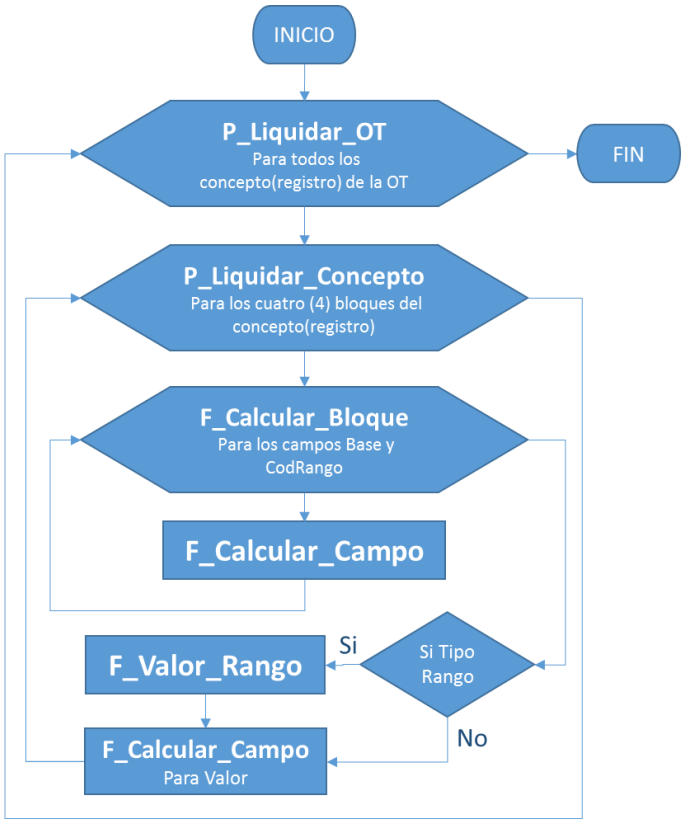
- **public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)**

Función propia de la clase SQLiteOpenHelper se usa para hacer las versiones de base de datos, no se usa en el módulo.

6.2.2 FAC_LABORCONCEPTOS. Esta clase contiene variables que guardan las columnas de la tabla FAC_LABORCONCEPTOS, se usa para crear objetos que contengan todos los datos de un registro de esta tabla, de esta manera es más fácil de manipular su valor durante la ejecución del programa y facilita la actualización en la base de datos tras la ejecución del liquidador.

6.2.3 LIQUIDADOR. Esta clase se encarga de hacer todo el proceso de liquidación, incluyendo el cálculo de los valores teniendo en cuenta los datos encontrados en la tabla FAC_LABORCONCEPTOS. El siguiente diagrama muestra la interacción de las funciones principales:

Ilustración 3, Interacción de las funciones



6.2.3.1 Variables

- **private final Context ctx**

Permite llamar al padre para realizar operaciones a nivel de la aplicación, como lanzar Activities, difundir mensajes por el sistema, recibir Intents, etc.

- **private BD_FENS help**

Con este objeto se pueden acceder a las funciones de la clase base de datos y que permite la actualización y consulta de datos.

- **private FAC_LABORCONCEPTOS r_ot**

En este objeto se guardan el valor de todas las columnas de un registro de la tabla FAC_LABORCONCEPTOS facilitando su acceso y modificación durante la ejecución del procedimiento de liquidación.

6.2.3.2 Funciones liquidador

- **F_Calcular_Campo**

Realiza el proceso de calcular el valor definitivo de un campo. El valor inicial de un campo podrá ser un número o una función (F_<nombre de función>). En el campo solo estará la función sin parámetros, cómo son funciones estándares previamente definidas el sistema sabrá que parámetros debe usar para cada caso.

- **F_Calcular_Bloque**

Realiza el proceso de calcular los valores de un bloque o grupo de datos para obtener el valor para el mismo. Cada uno de los campos de un bloque puede ser un valor o una función.

- **F_Valor_Rango**

Retornar el valor correspondiente para un Rango, código de rango y valor base según la base de datos**.**

- **F_Liquidar_Concepto**

Realiza el proceso de liquidar un concepto (registro) de una orden de trabajo.

- **P_Liquidar_OT**

Realiza el proceso de liquidar todos los conceptos de una orden de trabajo.

- **F_Consumo**

Retornar la sumatoria de los consumos de una OT para un grupo de lecturas.

- **F_Consumo_AC**

Retornar la sumatoria de los consumos de la OT Padre para un grupo de lecturas específico.

- **F_Consumo_Reactiva**

Retornar el consumo a facturar de reactiva.

- **F_Cantidad**

Retornar el valor existente en el mismo registro para el campo Cantidad.

- **F_Factor**

Retornar el valor existente en el mismo registro para el campo factor.

- **F_Tarifa**

Retornar el valor existente en el mismo registro para el campo tarifa.

- **F_Valor_Compensacion**

Retornar el valor a compensar por la calidad del servicio.

- **F_Valor_Total**

Retornar el valor total a cobrar para un concepto.

- **F_Suma_Conceptos**

Retornar el valor correspondiente a la suma del valor de los conceptos ingresados en el campo base.

- **F_Redondeo**

Ajusta y aproxima el valor de los conceptos liquidados según la manera que se sea especificada (decimas, unidades, decenas, centenas) por el cliente.

7. COMUNICACIÓN VÍA SERVICIO WEB

Se hizo necesario el uso de un servicio web ya que este facilita la transmisión de datos entre dos motores de datos diferentes como lo son el SQLite de la terminal móvil y el Oracle Database del servidor de base de datos.

7.1 ELECCIÓN DE ARQUITECTURAS Y FORMA DE ENVIAR DATOS

Teniendo en cuenta la cantidad de datos que maneja la aplicación antes de elegir la arquitectura de software REST esta fue comparada con la plataforma WFC para ver cuál de ellas permitía una transferencia de datos más eficiente especialmente en tiempo y memoria puesto que esta última es limitada en las terminales móviles. Como producto de esto se hicieron tres pruebas para ver cuál era más óptima.

7.1.1 DESCRIPCIÓN DE LAS PRUEBAS

En las tres pruebas se intentará trasmitir e insertar la tabla FAC_IMPRESION de 10 columnas y 85050 registros que tiene la siguiente estructura:

codprograma VARCHAR2(2) not null
correria VARCHAR2(10) not null
numorden VARCHAR2(20) not null
idimpresion VARCHAR2(10) not null
posxy VARCHAR2(20)
fuente VARCHAR2(30)
justificacion VARCHAR2(2)
funcion VARCHAR2(5)
parametros VARCHAR2(500)
valor VARCHAR2(500)

Pruebas

1. En la primera prueba se cargará la tabla desde la base de datos hacia la terminal móvil uniendo todas las columnas de cada registro en una sola columna separada por "|".

2. En la segunda prueba se cargará la tabla desde la base de datos hacia la terminal móvil sin ninguna modificación.

3. En la tercera prueba se descargará la tabla desde la terminal hacia la base de datos.

7.1.2 Resultados

Primera Prueba

Tabla 5, Resultados primera prueba

	Webservice	Peso Max XML [Kb]	Registros	Tiempo Inserción [s]	Tiempo Transmisión [s]	Tiempo Total [s]
Celular	WCF	37111	74682	15	75	90
	REST	33661	82829	17	69	86
Emulador	WCF	37111	76829	6	28	34
	REST	33661	85050	10	28	38

Segunda Prueba

Tabla 6, Resultados segunda prueba

	Webservice	Peso Max XML [Kb]	Registros	Tiempo Inserción [s]	Tiempo Transmisión [s]	Tiempo Total [s]
Celular	WCF	10122	85050	29	12	41
	REST	9181	85050	29	12	41
Emulador	WCF	10122	85050	12	5	17
	REST	9181	85050	12	6	18

Tercera Prueba

Tabla 7, Resultados tercera prueba

	Webservice	Peso Max XML [Kb]	Registros	Tiempo Inserción [s]	Tiempo Transmisión [s]	Tiempo Total [s]
Celular	WCF	(---)	700	0	2	2
	REST	(---)	45000	1	47	48
Emulador	WCF	(---)	700	0	0	0
	REST	(---)	45000	0	20	20

7.1.3 Conclusiones

- Se decide que se enviaran todos los datos como una sola columna separando el valor de cada columna con el carácter "|". Ya que de esta manera se reduce a un tercio el tamaño del archivo XML generado.
- La plataforma WCF tiene más limitaciones respecto a la cantidad de datos que se pueden enviar, esto es evidente en la prueba de descargue de datos.
- Se decide que para la construcción del servicio web se usara la arquitectura REST ya que, aunque en la trasmisión de datos usando el celular, el tiempo de transmisión fue más alto que en WCF este último tiene más limitaciones respecto a la cantidad de datos que es posible enviar, además de que el archivo XML que genera esta plataforma es más pesado y por esto no se adecua al trabajo con la terminal móvil donde el consumo de memoria es un punto crítico.

7.2 TERMINAL MÓVIL

Para consumir el servicio web se necesita en primer lugar que la aplicación en cuestión tenga permiso de la terminal móvil para acceder a internet, en el caso de esta adicionalmente se necesitó de la librería KSOAP2.

Ilustración 4, Petición de permisos aplicación móvil

AndroidManifest.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:tools="http://schemas.android.com/tools"
4      package="actsis.fens.com">
5
6      <uses-permission android:name="android.permission.INTERNET" />
7      <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

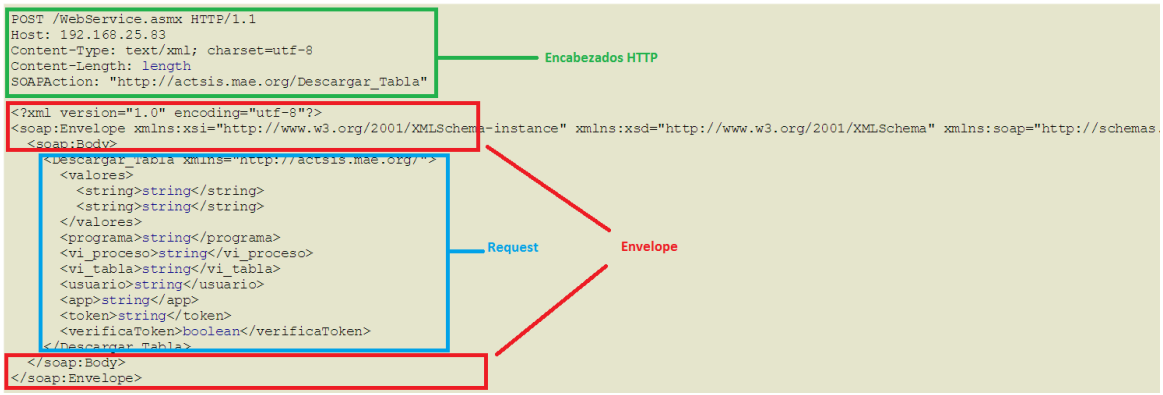
7.2.1 KSOAP2-ANDROID. Debido a que Android no incluye “de serie” ningún tipo de soporte para el acceso a servicios web de tipo SOAP, se utilizó una librería externa para hacer más fácil esta tarea. Entre la oferta actual, la opción más popular y más utilizada es la librería ksoap2-android. Esta librería es un fork, especialmente adaptado para Android, de la antigua librería kSOAP2. Este framework nos permitirá de forma relativamente fácil y cómoda utilizar servicios web que utilicen el estándar SOAP.

7.2.1.1 Estructura de las peticiones. Cada petición que se hace tiene una determinada estructura que se deberá cumplir para que las peticiones sean aceptadas por el servicio web y que este dé una respuesta. La estructura de los mensajes varía de acuerdo a la función que se esté solicitando ejecutar en el servicio. A continuación se muestra un ejemplo de la estructura de la petición de la función prDescargarTabla.

Ilustración 5, Estructura petición SOAP

SOAP 1.1

A continuación se muestra un ejemplo de solicitud y respuesta para SOAP 1.1. Es necesario reemplazar los marcadores de posición que aparecen con valores reales.



7.2.1.2 Uso de ksoap2 en android. Para usar la librería en Android como primer paso se definen cuatro constantes que se usan en varias ocasiones en el código de la petición:

- **NAMESPACE:** Espacio de nombres utilizado en nuestro servicio web.
- **URL:** Dirección URL para realizar la conexión con el servicio web.
- **METHOD_NAME:** Nombre del método web concreto que vamos a ejecutar.
- **SOAP_ACTION:** Equivalente al anterior, pero en la notación definida por SOAP.

Ilustración 6, Constantes petición SOAP

Los siguientes pasos del proceso son crear la petición SOAP a enviar al servicio web, enviarla al servidor y recibir la respuesta.

Para la creación de la petición al método Descargar Tabla se crea un objeto SoapObject agregándole el namespace y el nombre del método web, a este objeto se le asocian los parámetros de entrada del método web por medio del método addProperty ().

Ilustración 7, Adición de parámetros a una petición SOAP en Android

```
135     SoapObject request = new SoapObject(NAMESPACE, METHOD_NAME);
136     SoapObject objeto = new SoapObject(NAMESPACE,"parametrosPr");
137
138     for (int i = 0; i < parametroValues.length; i++)
139     {
140         objeto.addProperty("string", parametroValues[i]);
141     }
142
143     request.addProperty("programa", programa);
144     request.addProperty("usuario", usuario);
145     request.addSoapObject(objeto);
146     request.addProperty("token", Token);
147     request.addProperty("verificaToken", verificaToken);
```

El siguiente paso es crear el contenedor SOAP (envelope) y asociarle la petición. Para esto se crea un nuevo objeto SoapSerializationEnvelope indicando la versión de SOAP que vamos a usar (en el proyecto se usa la versión 10 por ser más estable que la última versión). En este caso en particular se indica que se trata de un servicio web .NET activando la propiedad dotNet. Finalmente se asocia la petición antes creada al contenedor usando el método setOutputSoapObject().

Ilustración 8, Creación del sobre y asociación de la petición

```
149     SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(SoapEnvelope.VER10);
150     envelope.dotNet = true;
151     envelope.setOutputSoapObject(request);
```

Para hacer el envió de la petición crearemos el objeto que se encargará de realizar la comunicación HTTP con el servidor, de tipo HttpTransportSE, al que se le adjunta la URL de conexión a nuestro servicio web. Para finalizar se hace la llamada al servicio web mediante el método call(). Tras hacer la llamada se obtiene la respuesta mediante el método getResponse(). Dependiendo del tipo de resultado que se espera se debe convertir esta respuesta a un tipo u otro. En este caso, el resultado que se espera es un valor simple (un String) por esto la respuesta se convierte a un objeto SoapPrimitive, que después se puede convertir a una variable String.

Ilustración 9, Llamada al servicio web

```
386         HttpTransportSE transporte = new HttpTransportSE(URL);
387         try {
388             transporte.call(SOAP_ACTION, envelope);
389
390             SoapPrimitive resultado = (SoapPrimitive)envelope.getResponse();
391
392             respuesta = resultado + "";
393
394         } catch (Exception e) {
395
396         }
```

7.2.2 Tareas en segundo plano. En la aplicación se usan varias tareas en segundo plano para realizar las distintas llamadas al servicio web y tratar los datos que se reciben o se envían a este. La razón que se usen tareas en segundo plano es que Android a partir de la versión 3 restringe que se ejecuten conexiones http en el hilo principal de ejecución por que estas pueden llevar mucho tiempo y por ende causan un bloqueo de la interfaz. Las tareas usadas en la aplicación fueron las siguientes:

7.2.2.1 P_User_Login. Esta tarea se encarga de validar el ID de usuario y contraseña ingresadas en las aplicación, si estos son válidos se obtiene un Token de seguridad, para esto realiza una petición al método "ejecutarProcedimientoRtaStr" haciendo uso del procedimiento almacenado "P_Valida_Usuario" de la base de datos. Como respuesta recibe un String que contiene el Token de seguridad o el informe de error si los datos ID y contraseña no corresponden.

7.2.2.2 P_User_Registro. Esta tarea se encarga de registrar un nuevo usuario de la aplicación, para esto realiza una petición al método "ejecutarProcedimientoRtaStr" del servicio web haciendo uso del procedimiento almacenado "P_Registra_Usuario" de la base de datos. Como respuesta recibe un String que contiene un mensaje confirmando el registro del nuevo usuario o el mensaje de error.

7.2.2.3 P_Proceso_Rutas. Esta tarea se encarga de obtener la lista de rutas asignadas a la terminal móvil, para esto realiza una petición al método "ejecutarProcedimientoRtaArrayString" haciendo uso del procedimiento almacenado "P_Pro_Rutas_Bd" de la base de datos. Como respuesta recibe una lista de String en donde están las rutas asignadas y la descripción de cada ruta si está disponible.

7.2.2.4 P_Proceso_Tablas. Esta tarea se encarga de obtener la lista de tablas asignadas para el cargue de datos para esto realiza una petición al método "ejecutarProcedimientoRtaArrayString" haciendo uso del procedimiento almacenado "P_Pro_Tablas_BD" de la base de datos. Como respuesta recibe una lista de String en donde están el nombre de las tablas, el tipo de tabla y la fecha de actualización. Las tablas que se retornan pueden ser filtradas por tipo según los parámetros que se envíen al procedimiento almacenado.

7.2.2.5 P_Proceso_Columnas. Esta tarea se encarga de obtener la lista de columnas de una determinada tabla para esto realiza una petición al método "ejecutarProcedimientoRtaArrayString" haciendo uso del procedimiento almacenado "P_Pro_Tablas_Col_Bd" de la base de datos. Como respuesta recibe una lista de String en donde están los nombres de las columnas de la tabla y el tipo de dato que estas tiene en la base de datos Oracle.

7.2.2.6 P_Proceso_Carga_Tablas. Esta tarea tiene dos funciones principales las primera es obtener los datos de una tabla haciendo una petición al método “ejecutarProcedimientoRtaArrayString” haciendo uso del procedimiento almacenado “P_Pro_Carga_Tabla_Bd” de la base de datos recibiendo una lista de String con todos los datos solicitados. La segunda tarea consiste en insertar los datos recibidos en la base de datos de la terminal móvil. Esta tarea sirve para cargar datos tanto de tablas detalle como para tablas maestras.

7.2.2.7 P_Proceso_Descarga_Tablas. Esta tarea se encarga de enviar los datos desde la terminal móvil al servicio web para estos hace uso de la tarea P_Proceso_Tablas para obtener las tablas que se deben enviar al servicio web, después se usa la tarea P_Proceso_Columnas para saber que columnas de cada tabla es necesario enviar. Luego de obtener los datos que se van enviar se hace una petición al método “prDescargarTabla” haciendo uso del procedimiento almacenado “P_Pro_Descarga_Tabla_Bd” de la base de datos como respuesta recibe un String que contiene el número de datos que fueron descargados en la base datos.

7.2.2.8 P_Actualizar_Maestros. La función de esta tarea en la de actualizar la tabla de maestros de la terminal. Esta tarea hace uso de la tarea P_Proceso_Tablas (especificando que el tipo de tabla sea “Maestros”) para obtener la lista de las tablas a actualizar para luego usar la tarea P_Proceso_Carga_Tablas para obtener los datos de las tablas y actualizarlos en la base de datos.

7.2.2.9 P_Cargar_Configuracion. Esta tarea se encarga de cargar la configuración de la terminal haciendo una petición al método "ejecutarProcedimientoRtaArrayString" haciendo uso del procedimiento almacenado "P_Config_Terminal" de la base de datos. Por el momento como respuesta se recibe una lista de String que contiene el número que se asigna a la terminal y la URL de conexión nueva en el caso que se quiera cambiar de servidor (servicio web).

7.2.3 Inserción de datos. Como la cantidad de datos que se maneja en esta aplicación puede llegar a ser bastante grande se hizo necesario encontrar un método que permitiera la inserción masiva de datos de forma eficiente. Como respuesta a este problema se consideró el uso del método Bulk Insert para hacer las inserciones de datos. Para poder usar este método se usa la información de la base datos de la terminal móvil para construir la instrucción SQL de forma dinámica, por esto es necesario que las tablas que se trasmitirán tengan una estructura similar en la base de datos local y en el servidor de base de datos.

7.3 Servicio web. El servicio web se desarrolló en lenguaje C# usando el entorno Visual Studio Community 2015 usando la arquitectura REST para aprovechar las ventajas que proporciona que este use un protocolo sin estado facilitando la transferencia de grandes volúmenes de datos.

7.3.1 Funciones. En el servicio web hay dos tipos de funciones las que son WebMethod que permiten que se les realicen peticiones a través de SOAP y las funciones internas que solo son accesibles dentro del servicio web.

7.3.1.1 WebMethod

7.3.1.1.1 ejecutarProcedimientoRtaStr. Esta función es genérica y ejecuta cualquier procedimiento almacenado en la base de datos que reciba como parámetros variables tipo VARCHAR2 y retorne un VARCHAR2 como respuesta. Como parámetros recibe un String que contenga el nombre del procedimiento almacenado y un array de String con los parámetros que necesite el procedimiento. Como respuesta retorna un String que contiene el VARCHAR2 devuelto por el procedimiento.

7.3.1.1.2 ejecutarProcedimientoRtaArrayString. Esta función es genérica y ejecuta cualquier procedimiento almacenado en la base de datos que reciba como parámetros variables tipo VARCHAR2 y retorne un REFCURSOR como respuesta. Como parámetros recibe un String que contenga el nombre del procedimiento almacenado y un array de String con los parámetros que necesite el procedimiento. Como respuesta retorna una lista String que contiene el REFCURSOR devuelto por el procedimiento.

7.3.1.1.3 prDescargarTabla. Esta no es una función genérica y se usa para ejecutar el procedimiento almacenado “P_Pro_Descarga_Tabla_Bd”. Como parámetros recibe cinco String que contienen el programa, proceso, nombre de la tabla, ID de usuario, nombre de la aplicación y el Token. Como respuesta retorna un String con la cantidad de registros que fueron insertados en la base de datos.

7.3.1.2 FUNCIONES INTERNAS

7.3.1.2.1 prVerificarTokenString Esta no es una función genérica y se usa para ejecutar el procedimiento almacenado “P_Valida_Token”. Como parámetros recibe el nombre de la aplicación, ID de usuario y el Token de seguridad. Como resultado retorna un String que contiene un valor que indica si el Token fue validado con éxito o si este fallo.

7.3.1.2.2 abrirConexion. Esta función se encarga de establecer una conexión con la base de datos usando una variable String de conexión que se encuentra encriptado en el archivo Web.config del servicio web, para realizar este proceso se usa la función EstablecerConexiones de la librería corenet de ACTSIS. Como respuesta retorna un objeto de conexión.

7.3.1.2.3 encode64Base. Esta función se encarga de codificar y retornar el String que recibe.

7.3.1.2.3.1 decode64Base. Esta función se encarga de decodificar y retornar el String que recibe.

7.4 SERVIDOR BASE DE DATOS

Para el desarrollo del proyecto se usa un servidor de base de datos Oracle en el cual se encuentran todas las tablas y procedimientos almacenados mediante los cuales se pueden adquirir los datos que se usaran en la aplicación.

Los procedimientos almacenados que se encargan de enviar los datos al servicio web y posteriormente a la aplicación se encuentran en dos paquetes principales los cuales son PK_FENS_SERVER y PK_GENAPP.

7.4.1 PK_GENAPP. En este paquete se encuentran los procedimientos almacenados para el manejo de seguridad en el registro y autenticación de usuarios. Actualmente este paquete es usado por otras aplicaciones y se espera que se pueda usar en futuros proyectos.

7.4.1.1 P_Valida_Usuario. Este procedimiento se encarga de verificar la existencia de un usuario en la base de datos. Como parámetros recibe tres VARCHAR2 los cuales contienen el nombre de la aplicación, ID de usuarios y clave. Como respuesta retorna un VARCHAR2 que en caso de que el usuario y la contraseña coincidan contiene un Token de seguridad de ocho caracteres aleatorios. Si el usuario no existe o no coinciden usuario y contraseña se retorna un mensaje de error.

7.4.1.2 P_Valida_Token. Este procedimiento se encarga de verificar el estado del Token de seguridad que posee actualmente el usuario. Como parámetros recibe tres VARCHAR2 los cuales contienen el nombre de la aplicación, ID de usuarios y Token de seguridad. Como respuesta retorna un VARCHAR2 que contiene el estado actual del Token los cuales son valido, inválido y excedido.

7.4.1.3 P_Registra_Usuario. Con este procedimiento se puede registrar un nuevo usuario en la base de datos. Como parámetros recibe cinco VARCHAR2 los cuales con el nombre de la aplicación, ID de usuario, nombre de usuario, correo electrónico y clave. Como respuesta retorna un VARCHAR2 que contiene una confirmación de la creación del nuevo usuario.

7.4.2 PK_FENS_SERVER. En este paquete se encuentran todos los procedimientos almacenados que hacen posible el cargue automático de datos en la terminal móvil. Este paquete fue creado para el desarrollo del proyecto y actualmente solo este lo usa, pero no se descarta la opción de que sea usado en más proyectos en el futuro.

7.4.2.1 P_Config_Terminal. Este procedimiento almacenado la verifica la existencia de una terminal en la base de datos. Como para parámetros recibe un VARCHAR2 que contiene el número de serie de la terminal y como respuesta retorna un REFCURSOR que contiene el ID de la que se le asignó a la terminal móvil y la URL del servicio web, esta última se usa en caso de que se quiera cambiar de servidor del servicio web.

7.4.2.2 P_Pro_Rutas_Bd. Este procedimiento se encarga de consultar y retornar la lista de rutas de trabajo asignadas a una terminal móvil especifica. Como parámetros recibe tres VARCHAR2 los cuales contienen el nombre de aplicación, el proceso y el ID de la terminal. Como respuesta retorna un REFCURSOR con las listas de rutas asignadas a la terminal.

7.4.2.3 P_Pro_Tablas_Bd. Este procedimiento se encarga de consultar y retornar la lista de tablas que deben ser cargadas o descargadas por la terminal. Como parámetros recibe tres VARCHAR2 los cuales contienen el nombre de aplicación, el proceso y el tipo de tabla. Como respuesta retorna un REFCURSOR con la lista de tablas según el proceso y el tipo que se soliciten, cada fila de la lista contiene el nombre de la tabla aplicación, tabla base de datos, el tipo de tabla, la fecha de actualización y el orden todos separados por el carácter “|”.

7.4.2.4 P_Pro_Tablas_Col_Bd. Este proceso almacenado se encarga de seleccionar las columnas de una tabla asociadas a un proceso de carga o descarga de datos. Como parámetros recibe tres VARCHAR2 los cuales contienen el nombre de aplicación, el proceso y el nombre de la tabla. Como respuesta retorna un REFCURSOR con una lista en la que cada fila contiene el nombre de la columna, el tipo de dato y si esta es una llave primaria todos esto separados por el carácter "|".

7.4.2.5 P_Pro_Carga_Tabla_Bd. Este procedimiento selecciona y retorna los datos de una tabla en la base de datos estos datos son filtrados por fecha de actualización y también pueden ser o no filtrados por ruta. Siempre se retorna datos por encima de la fecha que recibe manteniendo así los datos actualizados. Como parámetros de entrada recibe cinco VARCHAR2 los cuales son nombre de la aplicación, proceso, nombre de la tabla, ruta (solo tablas tipo D) y fecha de actualización. Como respuesta retorna un REFCURSOR con todos los registros resultantes. Para cada registros se unen el valor de todas las columnas de dicho registros y se separan por el carácter "|".

7.4.2.6 P_Pro_Descarga_Tabla_Bd. Este procedimiento se encarga de insertar en la base de datos los nuevos registros obtenidos tras hacer la liquidación de órdenes de trabajo en la terminal móvil. Como parámetros de entrada recibe tres VARCHAR2 que contienen el nombre de aplicación, el proceso que se está ejecutando y el nombre de la tabla de donde provienen los datos, también recibe un tabla de VARCHAR2 en donde están los datos que se van a insertar. Este procedimiento no retorna un resultado, pero en caso de ocurrir un error este lo reporta.

7.5 SEGURIDAD

Se usan dos manejos diferentes para garantizar la seguridad de la información en la aplicación.

7.5.1 Registro de terminales. Consiste en registrar en la base de datos del servidor el número de serie de las terminales móviles en donde se instala la aplicación. Después que se instala la aplicación esta no contiene ningún dato ni usuarios, ordenes de trabajo o información de conexión, por esta razón la aplicación le solicita al usuario de forma automáticamente la dirección de conexión del servicio web. Una vez ingresada la dirección se ejecuta tarea P_Cargar_Configuracion en la cual se envía el número de serie de la terminal a la base de datos y si esta se encuentra registrada se le retorna el ID de la terminal y una dirección URL de conexión la cual se almacena y se usa en futuras tareas. De esta manera se garantiza que solo las terminales registradas en la base de datos puedan usar la aplicación.

7.5.2 Token de seguridad. Este proceso se inicia en el momento en que el usuario ingresa su ID y contraseña en la aplicación e inicia sesión, al hacer esto se inicia la tarea P_User_Login y se obtiene un Token de seguridad si el ID y contraseña son correctos. El Token se guarda en la base de datos local de la terminal móvil y es enviado en todas peticiones futuras al servicio web. El Token tiene un tiempo de vida y en el momento que este sea excedido se le informa a la terminal y esta debe hacer una revalidación automática, en caso de que no sea posible hacer la revalidación o esta resulte incorrecta se hace un cierre de la aplicación y se debe ingresar de nuevo la ID y contraseña del usuario.

7.6 CARGA DE DATOS Y DESCARGUE DE DATOS

7.6.1 CARGA DE DATOS

7.6.1.1 Carga de datos general. En este proceso se inicia ejecutando las tareas P_Proceso_Rutas y P_Proceso_Tablas especificando como proceso “CARGA DATOS” obteniendo las listas de rutas y tablas que se deben cargar, después de esto para cada una de las rutas se cargan todas las tablas de la lista ejecutando la tarea P_Proceso_Carga_Tablas. Este proceso es válido para tablas tipo detalle y tablas tipo maestro.

7.6.1.2 Carga de maestros. Este proceso se inicia ejecutando el proceso P_Proceso_Tablas especificando el tipo de tabla como maestros y como proceso “CARGA DATOS” obteniendo la lista de tablas maestras que se deben cargar, después de esto se ejecuta para cada una de las tablas la tarea P_Proceso_Carga_Tablas. Este proceso es válido solo para las tablas tipo maestros.

7.6.2 DESCARGA DE DATOS. Este proceso se inicia ejecutando el proceso P_Proceso_Tablas especificando como proceso “DESCARGA DATOS” obteniendo la lista de tablas que se deben descargar, después de estos para cada una de las tablas en la lista se ejecuta la tarea P_Proceso_Columnas para obtener que columnas de la tabla se deben retornar. Tras estos se ejecuta la tarea P_Proceso_Descarga_Tablas en el cual se hace la recuperación de datos y se envían al servidor de base de datos. Este proceso se hace filtrando por rutas y el estado de órdenes de trabajo permitiendo solo descargar de una ruta a la vez los datos de las ordenes de trabajo que estén en estado leído o facturado. Además, se hace la descarga de anexos a las órdenes de trabajo.

8. APLICACIÓN MÓVIL

Como resultado del proyecto se crea FENSAPP, una aplicación móvil para el sistema operativo Android 4.0. La aplicación fue desarrollada en el entorno Android Studio y esta se encarga de integrar y administrar los procesos de cargue de datos hacia la terminal móvil, toma de lecturas de medidores, liquidación de órdenes de trabajo, generación e impresión de facturas y descargue de datos hacia el servidor de base datos. La aplicación fue pensada para trabajar sin estar conectada a internet es por esto que es posible trabajar con o sin conexión una vez que fue cargada en la terminal la información necesaria, tal como las ordenes de trabajo y la información de usuario que la está usando.

8.1 PANTALLAS

En la programación Android se puede decir que cada pantalla de la aplicación en una “activity” que está conformada a su vez por una parte lógica y una parte gráfica. La parte lógica es una clase java que hereda de la clase activity que tiene Android definida con sus respectivos métodos asignados. Para la parte grafica existe un archivo XML que tiene todos los elementos que estamos viendo de una pantalla declarados con etiquetas parecidas a las del HTML, es decir, que el diseño de una aplicación en Android se hace similar a una página web.

8.1.1 LOGIN

8.1.1.1 Elementos

- **EditText txtUsuario:**

Este elemento permite el ingreso de ID del usuario.

- **EditText txtPassword:**

Este elemento permite el ingreso de la contraseña del usuario.

- **Button btnIniciarSesion:**

Este elemento inicia el proceso de identificación del usuario.

- **TextView tvTerminal:**

Este elemento muestra el numero ID asignado a la terminal.

- **ProgressBar progress:**

Este elemento se muestra mientras se ejecuta una tarea en segundo plano.

8.1.1.2 Funcionamiento. Una vez se inicia la aplicación esta busca en su base de datos si tiene la información de conexión, si no la tiene esta le envía un mensaje solicitándosela al usuario. Si la información es correcta se mostrara en el TextView tvTerminal el número de la terminal que le fue asignado.

Si el usuario usa el botón btnIniciarSesion, se verifica que los campos estén txtUsuario y txtPassword no estén vacíos y en primer lugar busca dentro de la base de datos de la terminal si el usuario existe y la contraseña coincide, en caso de no encontrar el usuario, ejecuta la tarea en segundo plano P_User_Login para verificar si el usuario existe en el servidor de bases de datos. Una vez confirme la validez del usuario avanzara a la siguiente pantalla.

8.1.1.3 Consideraciones

- Solo las terminales registradas en el servidor de base de datos pueden conectarse a este.
- La única manera de ingresar en la aplicación es que el usuario exista en la base de datos de la terminal o en el servidor de la base de datos.
- En caso de que el usuario exista solo en el servidor de base de datos se debe ingresar una URL de conexión válida para poder acceder a la aplicación.

8.1.2 Menú principal

8.1.2.1 Elementos

- **Button btnGestionar:**

Inicia la pantalla gestionar rutas.

- **Button btnAdministrar:**

Inicia la pantalla de administración.

- **Menú de la aplicación:**

Muestra el menú principal de la aplicación en donde se da la opción de salir de la aplicación. Este elemento aparece en todas las pantallas con excepción de la pantalla Login.

8.1.2.2 Funcionamiento

Dependiendo cual opción elija el usuario este será dirigido a pantalla diferente.

8.1.2.3 Consideraciones

- En caso de ingresar en el menú de la aplicación y elegir la opción cerrar sesión se devolverá al usuario a la pantalla Login y se deberá volver a identificar si quiere ingresar de nuevo a la aplicación.

8.1.3 Administración

8.1.3.1 Elementos

- **Button RegistrarUsuario:**

Inicia la pantalla registrar usuario.

- **Button impresora:**

Inicia la pantalla configurar impresora.

- **Button configuración:**

Inicia el proceso de configuración de la terminal.

- **Button maestros:**

Inicia el proceso de actualización de maestros.

- **ProgressBar progress:**

Este elemento se muestra mientras se ejecuta una tarea en segundo plano.

8.1.3.2 Funcionamiento. En esta pantalla existen dos tipos de opciones, los que inician otras pantallas y los que ejecutan directamente tareas en segundo plano, los botones maestros y configuración ejecutan las tareas P_Actualizar_Maestros y P_Cargar_Configuracion respectivamente.

8.1.3.3 Consideraciones

- Para usar las opciones maestros y configuración se requiere tener una conexión con el servidor de base de datos.
- Algunas de las opciones solo pueden ser realizadas por usuarios con perfil de administrador.

8.1.4 Registrar usuario

8.1.4.1 Elementos

- **EditText txtNombre:**

Campo donde se ingresa el nombre del nuevo usuario.

- **EditText txtCorreo:**

Campo donde se ingresa el correo del nuevo usuario.

- **EditText txtUsuario:**

Campo donde se ingresa el ID usuario del nuevo usuario.

- **EditText txtClave:**

Campo donde se ingresa la clave del nuevo usuario.

- **EditText txtConfirmarClave:**

Campo donde se ingresa la confirmación de la clave del nuevo usuario.

- **Button btnRegistrar:**

Inicia el proceso de registro de un nuevo usuario.

8.1.4.2 Funcionamiento. Cuando el usuario usa el botón btnRegistrar se verifica si los campos están debidamente diligenciados (sin campos vacíos y que las contraseñas coincidan) y se inicia la tarea en segundo plano P_User_Registro.

8.1.4.3 Consideraciones

- El registro de usuarios solo puede ser realizado por usuarios con perfil de administrador.
- El registro de usuarios se realiza solamente en el servidor de base de datos.
- Para hacer el registro se requiere conexión con el servidor de base de datos.

8.1.5 Impresora

8.1.5.1 Elementos

- **ListView ListaImpresora:**

Muestra la lista de dispositivos bluetooth que fueron encontrados por el dispositivo móvil.

- **ProgressBar progress:**

Este elemento se muestra mientras se realiza la búsqueda de dispositivos bluetooth cercanos a la terminal móvil.

8.1.5.2 Funcionamiento. Cuando se inicia la pantalla se ejecuta de manera automática una búsqueda de los dispositivos que tengan bluetooth y que estén visibles en los alrededores del dispositivo móvil. Una vez que es finalizada la búsqueda se despliega una lista con el nombre y dirección Mac de los dispositivos que fueron encontrados. De esta lista el usuario puede elegir uno de los dispositivos (se espera que se elija una impresora bluetooth portátil) y este será guardado en la base de datos local y se usara posteriormente en labores de impresión de facturas, tras ser guardada la información de la impresora se retorna automáticamente a la pantalla de Menú Principal.

8.1.5.3 Consideraciones

- Para efectuar la búsqueda de dispositivos bluetooth se debe incluir el permiso de acceso y control del servicio de bluetooth del dispositivo móvil en el archivo de manifiesto de Android de la aplicación.
- Se debe configurar la impresora para que esta sea visible para otros dispositivos bluetooth.
- El tiempo de búsqueda de dispositivos bluetooth puede variar de un dispositivo móvil a otro.

8.1.6 Gestionar rutas

8.1.6.1 Elementos

- **ListView ListasRutas:**

Muestra la lista de las rutas de trabajo que están cargadas y disponibles en el dispositivo.

- **Button cargarRutas:**

Se encarga de iniciar la pantalla de cargue de datos.

- **Button eliminar:**

Se muestra para cada uno de los elementos de la ListView ListasRutas.

- **TextView ruta**

Muestra la descripción de la ruta si está disponible.

8.1.6.2 Funcionamiento. Cuando se inicia la pantalla se ejecuta de manera automática una consulta en la base de datos local a la tabla SCM_RUTAS para saber que rutas están disponibles en la terminal móvil. Después se muestra en la ListView ListaRutas todas las rutas que están disponibles para trabajar en la terminal móvil.Adicionalmente, en la lista, si se pulsa el botón eliminar, se da la opción de borrar rutas que estén cargadas en la terminal.En caso de que se pulse sobre el nombre de la ruta se iniciara la pantalla Información Ruta.

8.1.6.3 Consideraciones

- La opción de eliminar rutas esta solo disponible para usuarios con el perfil de administrador.
- Si existen órdenes de trabajo leídas o facturadas en la terminal, la aplicación pedirá al usuario que confirme la acción de borrar dos veces.
- En caso de no haber rutas cargadas en la terminal, la aplicación no muestra ninguna lista y muestra un mensaje en la pantalla.

8.1.7 Cargue de datos

8.1.7.1 Elementos

- **ListView ListasRutas**

Muestra la lista de las rutas de trabajo que están asignadas a la terminal móvil y que pueden ser cargadas.

- **Button Cargar**

Se encarga de iniciar el proceso de carga de datos desde el servidor de base de datos hacia la terminal móvil.

- **ProgressBar progress**

Este elemento se muestra mientras se ejecuta una tarea en segundo plano.

- **CheckBox checkRuta**

Elemento de la lista que permite seleccionar las rutas que se desean cargar.

- **TextView ruta**

Muestra la descripción de la ruta si está disponible.

8.1.7.2 Funcionamiento. Una vez iniciada la pantalla se ejecuta la tarea en segundo plano P_Proceso_Rutas para obtener las rutas asignadas a la terminal, una vez obtenida la información esta se despliega en la lista ListasRutas. El usuario puede elegir de la lista de rutas disponibles, cuales desea cargar haciendo uso del CheckBox e iniciar el proceso de carga usando el Button Cargar.Una vez se termina el proceso de carga se muestran mensajes informando que tablas fueron cargadas exitosamente o si ocurrió algún error durante el proceso.

8.1.7.3 Consideraciones

- Si una ruta fue cargada con anterioridad ya no se mostrará de nuevo en la lista de rutas asignadas a la terminal.
- El cargue de datos a la terminal puede ser realizado por cualquier perfil de usuario.
- Se debe tener una conexión a internet estable para un cargue de datos exitoso.

8.1.8 Información ruta

8.1.8.1 Elementos

- **Button Actualizar**

Se encarga de iniciar el proceso de actualización de datos de una ruta.

- **Button Iniciar**

Se encarga de iniciar la pantalla lista órdenes de trabajo.

- **Button GPS**

Se encargar de iniciar la pantalla Mapas.

- **Button descargarRuta**

Se encarga de iniciar el proceso de descarga de datos desde la terminal móvil hacia el servidor de base de datos.

- **ProgressBar progress**

Este elemento se muestra mientras se ejecuta una tarea en segundo plano.

- **TextView descripcion**

Muestra la descripción de la ruta si está disponible.

- **TextView ordenesTotales**

Muestra el número total de órdenes de trabajo.

- **TextView ordenesPendientes**

Muestra el número de órdenes de trabajo que no han sido leídas, facturas o descargadas.

- **TextView ordenesLeidas**

Muestra el número de órdenes de trabajo que fueron leídas por el usuario.

- **TextView ordenesFacturadas**

Muestra el número de órdenes que de trabajo que han sido facturadas.

- **TextView ordenesDescargadas**

Muestra el número de órdenes de trabajo que han sido descargadas.

8.1.8.2 Funcionamiento. En el momento en que se inicia la pantalla se hace una consulta a la base de datos de la terminal para buscar la información correspondiente a la ruta que fue elegida en la pantalla Gestionar Rutas, se consulta la descripción de la ruta, número total de órdenes de trabajo, ordenes que faltan por trabajarse, numero de ordenes leídas, numero de ordenes facturadas y numero de órdenes que fueron descargadas en el servidor de base de datos, esta información se muestra en los TextView descripcion, ordenesTotales, ordenesPendientes, ordenesLeidas, ordenesFacturadas y ordenesDescargadas respectivamente.

8.1.8.3 Consideraciones

- Cuando el usuario usa el botón Actualizar se ejecuta el mismo proceso de cargue de datos, pero se especifica que solo se busquen datos de las tablas detalle que correspondan a la ruta que se está mostrando actualmente, en caso de que ningún dato de dicha ruta se haya modificado no se retornan nuevos datos en la terminal.
- El descargue de datos hacia la terminal se hace por órdenes de trabajo.
- Solo se puede descargar órdenes de trabajo que estén leídas o facturadas.
- Si una orden de trabajo es descargada esta queda bloqueada y no se podrá modificar o facturar.

8.1.9 Lista ordenes de trabajo

8.1.9.1 Elementos

- **Button continuarRuta**

Se encarga de iniciar el proceso de actualización de datos de una ruta.

- **ListView ListaOT**

Lista que contiene las ordenes de trabajo de una ruta

8.1.9.2 Funcionamiento. En el momento en que se inicia la pantalla se realiza una consulta a la base de datos para obtener las ordenes de trabajo programadas en la ruta actual, con esta información se construye una lista con todas las ordenes de trabajo. El usuario puede acceder a cada orden de trabajo pulsando sobre esta o puede usar el botón continuarRuta para ingresar a la primera orden de trabajo que se encuentre en estado pendiente.Para que el usuario pueda saber el estado de las órdenes de trabajo de forma más rápida la lista muestra un icono diferente para las rutas según sea su estado.

8.1.10 Información ordene de trabajo

8.1.10.1 Elementos

- **TextView posicion**

Muestra un contador que indica en qué posición esta la orden de trabajo actual con respecto al total de órdenes de trabajo.

- **TextView direccionOT**

Muestra la dirección de la orden de trabajo

- **TextView nombreOT**

Muestra el nombre del cliente o razón social del predio de la orden de trabajo.

- **TextView indicacionOT**

Muestra, si está disponible, un punto de referencia para la ubicación de la orden de trabajo.

- **TextView marca**

Muestra el identificador de la marca del medidor.

- **TextView serie**

Muestra la serie del medidor.

- **TextView modelo**

Muestra el identificador del modelo del medidor.

- **TextView ubicacionMedidor**

Muestra la localización del medidor.

- **TextView activa**

Indica que se está mostrando la información del medidor de lectura activa.

- **TextView reactiva**

Indica que se está mostrando la información del medidor de lectura reactiva si este existe.

- **TextView observacion**

Muestra una observación para la orden de trabajo puede ser respecto al consumo o a una causa de no lectura. Inicia el dialog con una lista de observaciones definidas que pueden ser elegidas.

- **TextView observacionAdicional**

Muestra una observación para la orden de trabajo que no está relacionada con el consumo o una causa de no lectura. Inicia el dialog con una lista de observaciones definidas que pueden ser elegidas

- **TextView lector**

Espacio en donde se muestra la lectura ingresada para la orden de trabajo. También se encarga de iniciar la pantalla lector

- **Button observacionNueva**

Inicia el dialog donde se ingresa una observación de texto.

- **Button liquidarOT**

Inicia el proceso de liquidación de la orden de trabajo. Muestra el dialog de los resultados tras la liquidación.

- **Button back**

Carga la orden de trabajo anterior.

- **Button next**

Carga la siguiente orden de trabajo.

- **ListView ListaConceptos**

Muestra la lista de conceptos que fueron liquidados.

- **Button imprimir**

Inicia el proceso de impresión de la factura.

- **Button pdf**

Inicia el proceso de creación de la factura en formato PDF.

- **Button jpg**

Inicia el proceso de creación de la factura en formato JPG.

- **Button cerrar**

Cierra el dialog que muestra los resultados tras la liquidar la orden de trabajo.

- **EditText observacionTexto**

Permite ingresar una observación de texto.

- **Button guardarObservacion**

Guarda en la base de datos la observación ingresada en observacionTexto.

- **Button cancelar**

Cierra el dialog para ingresar la obsevacionTexto.

- **ListView ListaObservaciones**

Muestra una lista con las observaciones que se pueden ingresar a la orden de trabajo. Las observaciones pueden ser de consumo, de causas de no lectura o observaciones adicionales.

- **Button cerrar**

Cierra el dialog con la lista de observaciones.

8.1.10.2 Funcionamiento. En el momento en que se inicia la pantalla se realiza una consulta a la base de datos para obtener la información de la orden de trabajo que se está tramitando actualmente, se carga en los TextView información sobre la dirección de la orden de trabajo, nombre del usuario, información sobre el medidor e información acerca de las lecturas que tiene el medidor. Además, el TextView lector ajusta el número de decimales y enteros según la información que proporcione la tabla SCM_ELEMENTOSLECTURAS. Después de pulsar sobre el TextView Lector en ingresar la lectura pueden ocurrir varios escenarios:

1. La lectura ingresada es válida, quiere decir que el consumo que genera esta lectura está dentro del rango permitido para la orden de trabajo. Con la lectura valida el usuario puede liquidarla directamente o descargarla.
2. La lectura ingresada no es válida, quiere decir que el consumo generado por la lectura no se encuentra dentro del rango que se menciona en el numeral anterior, en este caso si el usuario desea liquidar la orden, primero debe ingresar una observación de consumo y tomar una evidencia fotográfica, tras hacer esto se podrá liquidar la orden de trabajo.
3. No es posible ingresar una lectura, en este caso si el usuario desea liquidar la orden de trabajo debe ingresar una observación de causa no lectura y tomar una evidencia fotográfica, tras hacer esto se podrá liquidar la orden de trabajo.

Una vez liquidada la orden de trabajo se muestra en la pantalla un dialog con los conceptos que se liquidaron para la orden de trabajo justo con las opciones de imprimir, generar PDF o generar un JPG.

8.1.10.3 Consideraciones

- Si la orden de trabajo tiene lectura activa y lectura reactiva se muestran los TextView activa y reactiva y pulsando sobre ellos se visualiza la información correspondiente a cada una de las lecturas.
- Las observaciones pueden ser al consumo o por causa no lectura, si se detecta una lectura ingresada se muestran las observaciones al consumo, si no se detecta ninguna lectura se muestran las observaciones de causa no lectura.
- Si la lectura es válida se muestra en color verde y no se valida se muestra en color rojo.
- El rango valido para las lecturas de las órdenes de trabajo es obtenido de la tabla SCM_ELEMENTOSLECTURAS.
- Es posible liquidar una orden de trabajo ingresado una lectura invalida o sin haberla ingresado, esto es posible usando el código de solconsumo de las tablas de observaciones el cual se relaciona con un valor solconsumo de la tabla SCM_ELEMENTOSLECTURAS el cual asigna un valor de consumo predefinido con el cual se realiza la liquidación de la orden de trabajo.
- Cada vez que se salga de la pantalla se guardaran todos los datos cambiados en la base de datos. Cuando se ingrese de nuevo a la orden de trabajo se muestran los datos actualizados.

8.1.11 Lector

8.1.11.1 Elementos

- **Button btGuardar**

Verifica y guarda la lectura ingresada.

- **Button btCancelar**

Cierra la pantalla lector y regresa a la pantalla información orden de trabajo.

- **Button btReset**

Limpia el TextView Lector.

- **Button btRetro**

Elimina el último número de lectura ingresado.

- **Button numeros**

Ingresan los valores de la lectura.

- **TextView tipolectura**

Muestra el tipo de lectura que se está ingresando.

- **TextView lector**

Muestra la lectura ingresada.

8.1.11.2 Funcionamiento. En el momento que se inicia esta pantalla se revisa que tipo de lectura se está ingresa y se ajustan los enteros y decimales del TextView lector de acuerdo con la información de la tabla SCM_ELEMENTOSLECTURAS.Una vez que se ingrese toda la lectura y se pulse guardar se evaluara si la lectura es válida en caso de que se válida la lectura se guardara y cerrara la pantalla lector y regresara a la pantalla información orden de trabajo, en caso de que no sea una lectura valida se mostrara un mensaje de error.

8.1.11.3 Consideraciones

- El botón guardar no se activará si el usuario no ingresa todos los dígitos que indica el TextView lector.
- Se tiene tres intentos para ingresar una lectura correcta, si no se ingresa la lectura correcta se guardará la última lectura ingresada y no se permitirá ingresar lecturas nuevas a la orden de trabajo.

8.1.12 Mapas

8.1.12.1 Funcionamiento. Esta pantalla tiene dos inicios diferentes por en cuando se inicia verifica desde que pantalla se inició y realiza dos acciones diferentes. Si se inicia desde la pantalla información ruta entonces se busca en la tabla SCM_ORDENES de trabajo y se obtiene la información del GPS entonces se ubica en el objeto GoogleMap marcadores con la ubicación de la orden de trabajo cada adicionalmente la dirección de la orden de trabajo. También se muestra la ubicación del usuario en el mapa. Si se inicia la pantalla desde la pantalla información orden de trabajo se realiza el mismo proceso, pero el marcador de la orden de trabajo en al que se encontraba la pantalla anteriormente se resaltará en un color diferente se hará un enfoque sobre este.

8.1.12.2 CONSIDERACIONES

- Se debe tener actualizado el Google Play Service en la terminal móvil para que funcione esta pantalla correctamente.
- Se debe agregar la API de Google Maps para que funciones correctamente esta pantalla.
- Se necesita tener el permiso de acceder al GPS de la terminal móvil.

8.1.13 Tomar foto

8.1.13.1 Elementos

- **RadioButton conexionButton**

Permite elegir el tipo de foto que se tomara.

- **Button predioButton**

Permite elegir el tipo de foto que se tomara.

- **Button medidorButton**

Permite elegir el tipo de foto que se tomara.

- **EditText descripcionFoto**

Permite ingresar una descripción a la evidencia fotográfica tomada.

- **Button aceptar**

Guarda la foto.

- **Button cancelar**

Cierra la pantalla Tomar Foto y regresa a la panta Información orden de trabajo.

8.1.13.2 Funcionamiento. Cuando se inicia la pantalla se muestra un dialog en el cual se consulta que tipo de foto se va tomar y se da la opción de ingresar una descripción de la evidencia fotográfica que se toma, después si se una el botón acepta se cierra el dialog y se abre la cámara del dispositivo. Una vez que es tomada la foto se guarda la información de esta en la tabla SCM_ANEXOS se cierra la pantalla y se regresa a la pantalla Información orden de trabajo.

8.1.13.3 CONSIDERACIONES

- La descripción tiene un mensaje por defectos según la situación que se presente (lectura no valida o que no se tenga una lectura).
- Es posible que la orden de trabajo exija tomar una foto sin que sea necesario que se presente alguna anomalía en la toma de la lectura.}
- La aplicación debe tener el permiso de acceso a la cámara para poder tomar la fotografía.

8.2 LIBRERÍAS

Para el desarrollo de la aplicación fue necesario el uso de librerías de terceros que facilitan la ejecución de algunas de las tareas de la aplicación como es el caso de

Google Maps el cual proporciona mapas para visualizar mejor la ubicación de las órdenes de trabajo.

8.2.1 KSOAP2. Facilita la conexión y trasmisión de datos con el servicio web.

8.2.2 ZSDK ANDROID API. Facilita la conexión e interacción con las impresoras Zebra. Esta librería fue creada por la empresa Zebra Technologies con el objetivo de facilitar el desarrollo de software que se usae en conjunto con sus productos.

8.2.3 iTextG. Es la versión de IText para Android la cual permite crear y manipular archivos PDF, RTF, y HTML en Java. Esta librería es bastante potente ya que soporta firmas basadas en PKI de PDF, cifrado de 40-bit y 128-bit, corrección de colores, PDF/X, gestión de colores por perfiles ICC entre otras características.

8.2.4 PDFVIEW BY JOAN ZAPATA. Es una librería que proporciona un visor PDF rápido para Android, con animaciones, movimientos, y zoom. Se basa en VuDroid para decodificar el archivo PDF. En el proyecto se usa para decodificar el PDF creado y construir un archivo JPG a partir de este.

8.2.5 Google maps API. Permite agregar mapas basados en datos de Google Maps a la aplicación. La API administra en forma automática el acceso a servidores, descargas de datos, visualización de mapas y respuesta a gestos de mapas de Google Maps.

9. CONCLUSIONES

- La facturación de energía eléctrica en Colombia se ve afectada por distintos factores que como resultado producen varios tipos de liquidación de los cuales uno o varios pueden estar presentes en una única orden de trabajo.
- Se deben identificar de manera precisa los cuellos de botella que se encuentren en un proceso para aplicar las correcciones correctas que lleven a la optimización del proceso en lugar de empeorarlo. (Transmisión de datos WinPhone).
- Los tiempos en el cronograma de trabajo deben ser lo mas apegados a la realidad posible para que sea posible su cumplimiento.
- Los proyectos se deben hacer de la manera mas modular posible para facilitar los cambios futuros.
- Es importante hablar con expertos que tengan experiencia para aprender de sus errores y evitarlos.

10. RECOMENDACIONES

- Se deben tener todas claras todas las funcionalidades y condiciones antes de iniciar la construcción de un proyecto.
- Tratar de tomar un enfoque mas practico en las clases que son impartidas en la universidad.

BIBLIOGRAFÍA

- ACTUALIZACIONES DE SISTEMAS LTDA. [sito web]. Bucaramanga: ACTSIS LTDA. [Consulta: 20 enero 2017]. Disponible en: http://www.actsis.com/.
- Android. Wikipedia. [en línea]. [Consultado 20 septiembre 2017]. Disponible en: https://es.wikipedia.org/wiki/Android.
- Android SQLite bulk insert and update example. MySampleCode. [en línea]. [Consultado 20 noviembre 2017]. Disponible en: http://www.mysamplecode.com/2011/10/android-sqlite-bulk-insert-update.html.
- DIMAS, José. ¿Qué es una Activity? DesarrolloWeb.com [en línea]. [Consultado 7 septiembre 2017]. Disponible en: http://www.desarrolloweb.com/articulos/android-que-es-una-activity-o-actividad.html.
- Extensible Markup Language. Wikipedia. [en línea]. [Consultado 20 septiembre 2017]. Disponible en: https://es.wikipedia.org/wiki/Extensible_Markup_Language.
- Impresores móviles Guía del usuario. Zebra Technologies [en línea]. [Consultado 30 noviembre 2017]. Disponible en: https://www.zebra.com/content/dam/zebra/manuals/es-la/printers/imzseries-ug-es.pdf.
- Introducción a la Google Maps Android API. Google Developers. [Consultado 30 noviembre 2017]. Disponible en: [Consultado 30 octubre 2017]. Disponible en: https://developers.google.com/maps/documentation/android-api/intro.
- INVARATO, Ramón. Context de Android. Jarroba.com. [en línea]. [Consultado 30 noviembre 2017]. Disponible en: http://jarroba.com/context-de-android/.
- ITEXT. iTextG [librería Android para la creación de documentos PDF]. Última versión: 5.5.10 Disponible para la descarga en https://github.com/itext/itextpdf/releases.

- Mobile Printing Systems CPCL Programming Manual. Zebra Technologies [en línea]. [Consultado 12 noviembre 2017]. Disponible en: https://www.zebra.com/content/dam/zebra/manuals/en-us/printer/cpcl-pm-en.pdf.
- PRODUCTOS. ¿Qué es SAC? ACTSIS LTDA. [en línea]. [Consultado 7 agosto 2017]. Disponible en: http://www.actsis.com/?men_id=7.
- PRODUCTOS. Facturación. ACTSIS LTDA. [en línea]. [Consultado 7 agosto 2017]. Disponible en: http://www.actsis.com/?men_id=24.
- Representational State Transfer. Wikipedia. [en línea]. [Consultado 20 septiembre 2017]. Disponible en: https://es.wikipedia.org/wiki/Representational_State_Transfer.
- SQLite. Wikipedia. [en línea]. [Consultado 10 septiembre 2017]. Disponible en: https://es.wikipedia.org/wiki/SQLite.
- Servicio web. Wikipedia. [en línea]. [Consultado 20 septiembre 2017]. Disponible en: https://es.wikipedia.org/wiki/Servicio_web.
- Procedimiento almacenado. Wikipedia. [en línea]. [Consultado 20 septiembre 2017]. Disponible en: https://es.wikipedia.org/wiki/Procedimiento_almacenado.
- SARMIENTO, Iván Darío. FD-10_ FENS_FACTURACION_SITIO. Documento interno de ACTSIS LTDA, 2015. 29 p.
- Simple Object Access Protocol. Wikipedia. [en línea]. [Consultado 20 septiembre 2017]. Disponible en: https://es.wikipedia.org/wiki/Simple_Object_Access_Protocol.
- SGOLIVER. Acceso a Servicios Web SOAP en Android. SGOLIVER.NET. [en línea]. [Consultado 15 noviembre 2017]. Disponible en: http://www.sgoliver.net/blog/acceso-a-servicios-web-soap-en-android-22/.
- SQLiteOpenHelper. Android Developers. [en línea]. [Consultado 20 noviembre 2017]. Disponible en: https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html.

- SQLiteDatabase. Android Developers. [en línea]. [Consultado 20 noviembre 2017]. Disponible en: https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html.
- KSOAP2 [librería Android para la conexión con servicios web que usan el estándar SOAP]. Última versión: 3.4.0. Disponible para la descarga en https://sourceforge.net/projects/ksoap2/.
- ZEBRA TECHNOLOGIES. Link-OS Multiplatform SDK [librería Android para la conexión con impresoras Zebra]. Última versión: 2.11.2800. Disponible para la descarga en https://www.zebra.com/us/en/products/software/barcode-printers/link-os/link-os-sdk.html.
- ZAPATA, Joan. Android PDFView [librería Android para la visualización de documentos PDF]. Última versión: 1.0.4 Disponible para la descarga en https://github.com/JoanZapata/android-pdfview.

ANEXOS

Anexo A. CÓDIGO FUENTE SERVICIO WEB

```
using Oracle.ManagedDataAccess.Client;
using System;
using System.Web.Services;
using System.Data;
using System.Collections.Generic;
using corenet.Conexiones;
using TipoEnum = corenet.Aplicacion.TiposEnum;
using System.Configuration;
using corenet.Acciones;
using corenet.Componentes;
using System.IO;
using corenet.Aplicacion;

[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]

public class WebService : System.Web.Services.WebService
{
    public WebService()
    {
    }

    #region Declaraciones
    public const string K_PK_GENAPP  = "PK_GENAPP.";
    public const string K_PK_SACAPP  = "PK_SACAPP.";
    public const string K_PK_FENSAPP = "PK_FENS_SERVER.";
    #endregion
```

```
#region Funciones Públicas

[WebMethod]
public string ejecutarProcedimientoRtaStr(string  programa,
                                          string app,
                                          string usuario,
                                          string[] parametrosPr,
                                          string token = "",
                                          bool verificaToken = true,
                                          bool encriptaIn = false,
                                          bool encriptaOut = false,
                                          string esquema = "GEN")
{
    if (encriptaIn)
    {
        parametrosPr = encodeDecodeArray(parametrosPr, true);
    }

    string nomPK_Esquema;
    switch (esquema)
    {
        case "SAC":
            nomPK_Esquema = K_PK_SACAPP;
            break;
        case "FENS":
            nomPK_Esquema = K_PK_FENSAPP;
            break;
        case "GEN":
            nomPK_Esquema = K_PK_GENAPP;
```

```
            break;
        default:
            nomPK_Esquema = "";
            break;
    }

    string retorno = "";
    string validaToken = "OK";

    if (verificaToken == true)
    {
        validaToken = prVerificarTokenString(app, usuario, token);
    }

    if (validaToken == "OK")
    {
        object conexion = new object();
        try
        {
            ConexionBaseDatos procedimientoBD = new ConexionBaseDatos();
            OracleParameterCollection returnToken;
            OracleParameter[] parametros;

            parametros = new OracleParameter[parametrosPr.Length + 1];

            for (int i = 0; i < parametrosPr.Length; i++)
            {
                string[] splitPar = parametrosPr[i].Split('|');
              parametros[i] = new OracleParameter(splitPar[0],
OracleDbType.Varchar2, splitPar[1], System.Data.ParameterDirection.Input);
```

```
            }

            parametros[parametrosPr.Length] = new
OracleParameter("vo_respuesta", OracleDbType.Varchar2, 200, null,
System.Data.ParameterDirection.Output);


            conexion = abrirConexion();

            returnToken = procedimientoBD.ejecutarProcedimiento(conexion,
                                                nomPK_Esquema + programa,
                                                parametros);

            EstablecerConexiones.cerrarConexion(conexion);
            retorno = returnToken[parametrosPr.Length].Value.ToString();
        }
        catch (Exception ex)
        {
            EstablecerConexiones.cerrarConexion(conexion);
            retorno = "Error: " + ex.Message;
        }
    }
    else
    {
        retorno = validaToken;
    }

    if (encriptaOut)
    {
        return encode64Base(retorno);
```

```
    }

    return retorno;
}

[WebMethod]
public DataTable ejecutarProcedimientoRtaDt(string  programa,
                                    string app,
                                    string usuario,
                                    string[] parametrosPr,
                                    string token = "",
                                    bool verificaToken = true,
                                    bool encriptaIn = false,
                                    bool encriptaOut = false,
                                    string esquema = "GEN")
{
    if (encriptaIn)
    {
        parametrosPr = encodeDecodeArray(parametrosPr, true);
    }

    string nomPK_Esquema;
    switch (esquema)
    {
        case "SAC":
            nomPK_Esquema = K_PK_SACAPP;
            break;
        case "FENS":
            nomPK_Esquema = K_PK_FENSAPP;
            break;
```

```
    case "GEN":
        nomPK_Esquema = K_PK_GENAPP;
        break;
    default:
        nomPK_Esquema = "";
        break;
}

DataTable dtRetorno = new DataTable();
object conexion = new object();
string validaToken = "OK";

if (verificaToken == true)
{
    validaToken = prVerificarTokenString(app, usuario, token);
}

if (validaToken == "OK")
{

    try
    {
        ConexionBaseDatos procedimientoBD = new ConexionBaseDatos();
        DataTable returnToken;
        conexion = abrirConexion();

        returnToken  = procedimientoBD.ejecutarProcedimientoRtaDt(conexion
                                        , nomPK_Esquema + programa
                                        , parametrosPr);
        EstablecerConexiones.cerrarConexion(conexion);
```

```
            dtRetorno = returnToken;

        }
        catch (Exception ex)
        {
          String retorno = "Error: " + ManejoErrores.Validar_Exception(conexion,
ex);
            DataSet ds = new DataSet();
            DataTable dtErr = new DataTable();
            dtErr.Columns.Add("codError");
            dtErr.Columns.Add("mensajeError");
            dtErr.Rows.Add("1", retorno);
            EstablecerConexiones.cerrarConexion(conexion);
            ds.Tables.Add(dtErr);
            dtRetorno = ds.Tables[0];

        }
    }
    else
    {
        DataSet ds = new DataSet();
        DataTable dtErr = new DataTable();
        dtErr.Columns.Add("codError");
        dtErr.Columns.Add("ErrorToken");
        dtErr.Rows.Add("1", "ErrorToken|" + validaToken);
        EstablecerConexiones.cerrarConexion(conexion);
        ds.Tables.Add(dtErr);
        dtRetorno = ds.Tables[0];
    }
```

```
    if (encriptaOut)
    {
        dtRetorno = encodeDecodeDT(dtRetorno, false);
    }
    return dtRetorno;
}

[WebMethod]
public string prDescargarTabla(string programa,
                               string app,
                               string usuario,
                               string[] parametrosPr,
                               string[] datos,
                               string token = "",
                               bool verificaToken = true,
                               bool encriptaIn = true,
                               bool encriptaOut = true,
                               string esquema = "GEN")
{
    if (encriptaIn)
    {
        parametrosPr = encodeDecodeArray(parametrosPr, true);
    }

    string nomPK_Esquema;
    switch (esquema)
    {
        case "SAC":
            nomPK_Esquema = K_PK_SACAPP;
```

```
            break;
        case "FENS":
            nomPK_Esquema = K_PK_FENSAPP;
            break;
        case "GEN":
            nomPK_Esquema = K_PK_GENAPP;
            break;
        default:
            nomPK_Esquema = "";
            break;
    }

    string validaToken = "OK";

    if (verificaToken == true)
    {
        validaToken = prVerificarTokenString(app, usuario, token);
    }

    if (validaToken == "OK")
    {
        object conexion = new object();
        try
        {
          ProcedimientoBaseDatos procedimientoBD = new
ProcedimientoBaseDatos();
            ConexionBaseDatos procedimiento = new ConexionBaseDatos();
            ValidacionesGlobales array = new ValidacionesGlobales();
```

```
        OracleParameter[] parametros = new
OracleParameter[parametrosPr.Length + 1];

        for (int i = 0; i < parametrosPr.Length; i++)
        {
            string[] splitPar = parametrosPr[i].Split('|');
          parametros[i] = new OracleParameter(splitPar[0],
OracleDbType.Varchar2, splitPar[1], System.Data.ParameterDirection.Input);
        }

        parametros[parametrosPr.Length] =
array.convertirOracleParameter(datos, "vi_tab_linea");

        conexion = abrirConexion();

        procedimiento.ejecutarProcedimiento(conexion,
                                    nomPK_Esquema + programa,
                                    parametros);


        procedimientoBD.ejecutarProcedimientoAlmacenado(conexion,
                                            TipoEnum.TipoTransaccion.C_ommit,
                                            "",
                                            "P_Commit");

        EstablecerConexiones.cerrarConexion(conexion);
        if (encriptaOut)
        {
          return encode64Base("Operacion exitosa " + datos.Length + "
registros insertados");
```

```
            }
            else
            {
                return "Operacion exitosa " + datos.Length + " registros insertados";
            }
        }
        catch (Exception e)
        {
            EstablecerConexiones.cerrarConexion(conexion);
            if (encriptaOut)
            {
                return encode64Base("Error: " + e.ToString());
            }
            else
            {
                return "Error: " + e.ToString();
            }
        }
    }
    else
    {
        if (encriptaOut)
        {
            return encode64Base(validaToken);
        }
        else
        {
            return validaToken;
        }
    }
```

```
}

[WebMethod]
public string[] ejecutarProcedimientoRtaArrayString(string  programa,
                                        string app,
                                        string usuario,
                                        string[] parametrosPr,
                                        string token = "",
                                        bool verificaToken = true,
                                        bool encriptaIn = true,
                                        bool encriptaOut = true,
                                        string esquema = "GEN")
{
   DataTable dtr = new DataTable();
   List<string> tablas = new List<string>();

   dtr = ejecutarProcedimientoRtaDt(programa,
                              app,
                              usuario,
                              parametrosPr,
                              token,
                              verificaToken,
                              encriptaIn,
                              encriptaOut,
                              esquema);

   if (dtr.Columns[0].ColumnName  == "codError")
   {
      if(dtr.Columns[1].ColumnName  == "ErrorToken")
      {
```

```
            foreach (DataRow row in dtr.Rows)
            {
                tablas.Add(Convert.ToString(row["ErrorToken"]));
            }
            return tablas.ToArray();
        }
        else
        {
            return null;
        }
    }
    else
    {
        foreach (DataRow row in dtr.Rows)
        {
            tablas.Add(Convert.ToString(row["A"]));
        }

        return tablas.ToArray();
    }
}

[WebMethod]
public string guardarArchivoLocal(string arcBinaryArray, string nomArchivo, string extArchivo)
{
    string strdocPath;
    strdocPath = ConfigurationManager.AppSettings["va_ruta_imangen"] + nomArchivo + extArchivo;
    //strdocPath = "C:\\PruImagen\\" + nomArchivo + extArchivo;
```

```
      FileStream objfilestream = new FileStream(strdocPath, FileMode.Create,
FileAccess.ReadWrite);
       try
       {
          byte[] arcByte = decode64BaseByte(arcBinaryArray);
          objfilestream.Write(arcByte, 0, arcByte.Length);
          objfilestream.Close();

          return strdocPath;
       }
       catch (Exception ex)
       {
          objfilestream.Close();
          return "";
       }

   }

   #endregion

   #region Funciones Privadas

   private object abrirConexion()
   {
      object conexion = null;
     string                                connString                                =
ConfigurationManager.ConnectionStrings["connString"].ConnectionString;
       conexion = EstablecerConexiones.abrirConexion(connString);
       return conexion;
   }
```

```
    private string encode64Base(string texto)
    {
        //Encode
        var plainTextBytes  = System.Text.Encoding.UTF8.GetBytes(texto);
        string encode = System.Convert.ToBase64String(plainTextBytes);

        return encode;
    }

    private string decode64Base(string texto)
    {
        //Decode
        var base64EncodedBytes = System.Convert.FromBase64String(texto);
       String                                    decode                                    =
System.Text.Encoding.UTF8.GetString(base64EncodedBytes);
        return decode;
    }

    private string prVerificarTokenString(string  app, string usuario, string token)
    {
        string retorno = "OK";
        object conexion = new object();
        try
        {
            OracleParameterCollection returnToken;
            ConexionBaseDatos procedimientoBD = new ConexionBaseDatos();
            OracleParameter[] parametros;

            parametros = new OracleParameter[4];
```

```
        parametros[0] = new OracleParameter("vi_app", OracleDbType.Varchar2,
app, System.Data.ParameterDirection.Input);
        parametros[1] = new OracleParameter("vi_usuario",
OracleDbType.Varchar2, usuario, System.Data.ParameterDirection.Input);
        parametros[2] = new OracleParameter("vi_token", OracleDbType.Varchar2,
token, System.Data.ParameterDirection.Input);
        parametros[3] = new OracleParameter("vo_respuesta",
OracleDbType.Varchar2, 200, null, System.Data.ParameterDirection.Output);

          conexion = abrirConexion();

          returnToken = procedimientoBD.ejecutarProcedimiento(conexion,
                                              "PK_GENAPP.P_Valida_Token",
                                              parametros);
          EstablecerConexiones.cerrarConexion(conexion);
          retorno = returnToken[3].Value.ToString();

      }
      catch (Exception ex)
      {
          //retorno = "Ocurrio un error al validar el token";
          retorno = "ESV"; //Error al ejecutar el servicio
          EstablecerConexiones.cerrarConexion(conexion);
      }
      return retorno;
  }

  private string[] encodeDecodeArray(string[] datos, bool encripta)
  {
      if (encripta)
```

```
    {
        for (int i = 0; i < datos.Length; i++)
        {
            datos[i] = decode64Base(datos[i]);
        }
    }
    else
    {
        for (int i = 0; i < datos.Length; i++)
        {
            datos[i] = encode64Base(datos[i]);
        }
    }
    return datos;
}

private byte[] decode64BaseByte(string texto)
{
    //Decode
    var base64EncodedBytes = System.Convert.FromBase64String(texto);
    return base64EncodedBytes;
}

private DataTable encodeDecodeDT(DataTable datos, bool encripta)
{
    if (encripta)
    {
        foreach (DataRow Row in datos.Rows)
        {
            foreach (DataColumn Col in datos.Columns)
```

```
            {
                Row[Col] = decode64Base(Row[Col].ToString());
            }
        }
    }
    else
    {
        foreach (DataRow Row in datos.Rows)
        {
            foreach (DataColumn Col in datos.Columns)
            {
                Row[Col] = encode64Base(Row[Col].ToString());
            }
        }
    }
    return datos;
}

#endregion
}
```

Anexo B. FRAGMENTO CÓDIGO FUENTE APLICACIÓN MÓVIL

Adapter_Row_Gestion_Ruta

```
package actsis.fens.com.Adaptadores;
import android.app.AlertDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.database.Cursor;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;
import com.app.actsis.bd_app.BD_APP;
import java.util.ArrayList;
import actsis.fens.com.Aplicacion.Constantes;
import actsis.fens.com.Aplicacion.VAR;
import actsis.fens.com.Layout.Gestionar_Rutas;
import actsis.fens.com.Layout.Info_Ruta;
import actsis.fens.com.Objetos.Row_Gestion_Ruta;
import actsis.fens.com.R;

public class Adapter_Row_Gestion_Ruta extends
ArrayAdapter<Row_Gestion_Ruta> implements View.OnClickListener
{
    private LayoutInflater layoutInflater;
    private Context ctx;
    private BD_APP help;
    private Gestionar_Rutas activity;

    public Adapter_Row_Gestion_Ruta(Context context,
ArrayList<Row_Gestion_Ruta> objects, Gestionar_Rutas activity) {
        super(context, 0, objects);
        this.ctx = context;
        this.help = new BD_APP(Constantes.DB_NAME,context);
        this.activity = activity;
```

```
        layoutInflater = LayoutInflater.from(context);
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        // holder pattern
        Holder holder;
        if (convertView == null)
        {
            holder = new Holder();

            convertView = layoutInflater.inflate(R.layout.row_gestion_ruta,parent,
false);
            holder.setTextViewRuta((TextView)  convertView.findViewById(R.id.ruta));
            holder.setButtonDelete((Button)convertView.findViewById(R.id.eliminar));
            convertView.setTag(holder);
        }
        else
        {
            holder = (Holder) convertView.getTag();
        }

        final Row_Gestion_Ruta  row = getItem(position);
        holder.getTextViewRuta().setText(row.getDescripcion());
        holder.getTextViewRuta().setTag(position);
        holder.getButtonDelete().setTag(position);
        holder.getButtonDelete().setOnClickListener(this);
        holder.getTextViewRuta().setOnClickListener(this);

        return convertView;
    }

    @Override
    public void onClick(View view) {
        final int position = (Integer)view.getTag();
        switch (view.getId()){
            case R.id.ruta:
                Intent  intent = new Intent(ctx,  Info_Ruta.class);
                VAR.Ruta = getItem(position).getRuta();
```

```
            ctx.startActivity(intent);
            break;
        case R.id.eliminar:
            String perfil = help.B_K("APP_USUARIOS","perfil","usuario",
VAR.Usuario);
            String query = "Select * FROM scm_ordenes_trabajo WHERE estado
!= 'P'";
            String mensaje = "";
            Cursor c = help.Cursor(query);
            if(c.getCount() > 0){
                mensaje = "Ya se han tomado lecturas para esta rura. ¿Está
seguro que desea borrarla?";
            }else {
                mensaje = "¿Está seguro que desea borra esta ruta?";
            }
            if(c.getCount() == 0 || (perfil != null && perfil.equals("ADM"))){
                final String mensajeF = mensaje;
                new AlertDialog.Builder(ctx)
                        .setIcon(android.R.drawable.ic_dialog_info)
                        .setTitle("Borrar Ruta")
                        .setMessage("¿Está seguro que desea borrar esta ruta?")
                        .setNegativeButton(android.R.string.cancel, null)
                        .setPositiveButton(android.R.string.ok, new
DialogInterface.OnClickListener() {
                            @Override
                            public void onClick(DialogInterface dialog, int which){
                                new AlertDialog.Builder(ctx)
                                        .setIcon(android.R.drawable.ic_dialog_info)
                                        .setTitle("Borrar Ruta")
                                        .setMessage(mensajeF)
                                        .setNegativeButton(android.R.string.cancel, null)
                                        .setPositiveButton(android.R.string.ok, new
DialogInterface.OnClickListener() {//un listener que al pulsar, la actividad actual y
envie a otra
                                            @Override
                                            public void onClick(DialogInterface dialog, int
which){

                                                String[] where = {"ruta"};
```

```
                                        String[] wvalues  = {getItem(position).getRuta()};


help.DeleteTableNK("scm_rutas",where,wvalues);

help.DeleteTableNK("scm_elementos_lecturas",where,wvalues);

help.DeleteTableNK("scm_ordenes_trabajo",where,wvalues);

help.DeleteTableNK("fac_laborconceptos",where,wvalues);

help.DeleteTableNK("fac_impresion",where,wvalues);

help.DeleteTableNK("fac_rangos",where,wvalues);

help.DeleteTableNK("scm_anexos",where,wvalues);

                                                activity.Lista_Gestion_Rutas();
                                            }
                                        })
                                        .show();
                                }
                            })
                            .show();
                    }else {
                        Toast.makeText(ctx, "No tiene los permisos necesarios" ,
Toast.LENGTH_SHORT).show();
                    }

                    break;
                }
            }

            private static class Holder {
                TextView textViewRuta;
                Button buttonDelete;

                public Button getButtonDelete() {
                    return buttonDelete;
```

```
    }

    public void setButtonDelete(Button buttonDelete) {
        this.buttonDelete = buttonDelete;
    }

    public TextView getTextViewRuta()
    {
        return textViewRuta;
    }

    public void setTextViewRuta(TextView textViewRuta)
    {
        this.textViewRuta = textViewRuta;
    }
  }
}
```

Adapter_Row_OT

```
package actsis.fens.com.Adaptadores;
import android.content.Context;
import android.content.Intent;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ImageView;
import android.widget.RelativeLayout;
import android.widget.TextView;
import java.util.ArrayList;
import actsis.fens.com.Aplicacion.VAR;
import actsis.fens.com.Layout.Info_OT;
import actsis.fens.com.Objetos.Row_OT;
import actsis.fens.com.R;

public class Adapter_Row_OT extends ArrayAdapter<Row_OT> implements
View.OnClickListener
{
    private LayoutInflater layoutInflater;
    private ArrayList<Row_OT> OTS;
```

```
    private Context ctx;
    private int ot_actual;

    public Adapter_Row_OT(Context context, ArrayList<Row_OT> objects,int
actual)
    {
        super(context, 0, objects);
        this.OTS = objects;
        this.ctx = context;
        this.ot_actual = actual;
        layoutInflater = LayoutInflater.from(context);
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent)
    {
        // holder pattern
        Holder holder = null;
        if (convertView == null)
        {
            holder = new Holder();

            convertView = layoutInflater.inflate(R.layout.row_ot,parent, false);
            holder.setTextViewOT((TextView) convertView.findViewById(R.id.OT));

holder.setTextViewDirecion((TextView)convertView.findViewById(R.id.DireccionO
T));
            holder.setIcono((ImageView)convertView.findViewById(R.id.icono_ot));

holder.setRootRow_OT((RelativeLayout)convertView.findViewById(R.id.rootRow_
ot));
            convertView.setTag(holder);
        }
        else
        {
            holder = (Holder) convertView.getTag();
        }

        final Row_OT row = getItem(position);
```

```
        holder.getTextViewOT().setText(row.getOT());
        holder.getTextViewDirecion().setText(row.getDireccion());
        holder.getRootRow_OT().setTag(position);

        int image;
        switch (row.getEstado()){
            case "F":
                image = R.drawable.ic_facturado;
                break;
            case "T":
                image = R.drawable.ic_leido;
                break;
            case "DF":
                image = R.drawable.ic_descargada;
                break;
            case "DT":
                image = R.drawable.ic_descargada;
                break;
            default:
                image = R.drawable.ic_pendiente;
                break;
        }
        holder.getIcono().setImageResource(image);

        if(ot_actual == position){
            holder.getRootRow_OT().setBackgroundResource(R.color.separador);
        }else {
            holder.getRootRow_OT().setBackgroundResource(0);
        }
        holder.getRootRow_OT().setOnClickListener(this);

        return convertView;
    }

    @Override
    public void onClick(View view) {
        int position = (Integer)view.getTag();
        Intent intent = new Intent(ctx, Info_OT.class);
        VAR.Position = position;
```

```
    VAR.OTS = OTS;
    ctx.startActivity(intent);
}

static class Holder
{
    TextView textViewOT;
    TextView textViewDirecion;
    ImageView icono;
    RelativeLayout  rootRow_OT;

    public ImageView getIcono() {
        return icono;
    }

    public void setIcono(ImageView  icono) {
        this.icono = icono;
    }

    public RelativeLayout  getRootRow_OT() {
        return rootRow_OT;
    }

    public void setRootRow_OT(RelativeLayout  rootRow_OT) {
        this.rootRow_OT = rootRow_OT;
    }

    public TextView getTextViewDirecion() {
        return textViewDirecion;
    }

    public void setTextViewDirecion(TextView  textViewDirecion) {
        this.textViewDirecion  = textViewDirecion;
    }

    public TextView getTextViewOT()
    {
        return textViewOT;
    }
```

```
        public void setTextViewOT(TextView  textViewOT)
        {
           this.textViewOT = textViewOT;
        }
    }
}
```