Aplicación en plataforma Android para la visualización de datos en tiempo real del SCADA SAINUX

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Alexis Gongora Smith.

Tutor: Ing. Ridel Oscar García Mora.

Co-tutor: Ing. Yosvany Leyva Pizarroza.

Habana, Junio de 2016

"Año 58 de la Revolución"

“...este país vivirá de la inteligencia y de las producciones intelectuales”

Declaración de Autoría

Por este medio declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas (UCI) a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ___ días del mes de _____ del 2016.

_____

Autor: Alexis Gongora Smith

_____ _____

Tutor: Ing. Ridel Oscar García Mora Co-tutor: Ing. Yosvany Leyva Pizarroza

Datos de Contacto

Tutor: Ing. Ridel Oscar García Mora
Ciudadanía: cubana
Institución: Universidad de las Ciencias Informáticas (UCI)
Título: Ingeniero en Ciencias Informáticas
Categoría Docente:
E-mail:

Co-tutor: Ing. Yosvany Leyva Pizarroza
Ciudadanía: cubana
Institución: Universidad de las Ciencias Informáticas (UCI)
Título: Ingeniero en Ciencias Informáticas
Categoría Docente:
E-mail:

Agradecimientos

Dedicatoria

A ti madre, que me llevaste en tu vientre y me diste la vida. Tú que siempre estuvo a mi lado y me dio su apoyo in condicional cuando más lo necesité. Aunque no estas a mi lado, yo te sigo queriendo como la mujer más maravillosa del mundo.

Desde el cielo nunca me dejes de cuidar, te amo mamá.

Resumen

El constante desarrollo de las ciencias de la computación a partir del siglo pasado, ha traído como uno de sus beneficios la incorporación de los ordenadores a la supervisión y control de procesos industriales. En la última década las tecnologías de información y comunicación han cambiado el ritmo de vida de los seres humanos. El uso de internet ha posibilitado que las personas accedan a una fuente inagotable de información y les ha permitido conocerse y acercarse como nunca antes. En años recientes, el mundo adoptó dos dispositivos inteligentes, tales dispositivos son el *Smartphone* y la *Tablet*. El *hardware* que estos dispositivos ofrecen, abre la posibilidad de interactuar y controlar sistemas SCADA. Con la necesidad de aumentar la accesibilidad a la información generada por el Sistema SCADA SAINUX desarrollado en la UCI; esta investigación se traza como objetivo desarrollar una aplicación en Android para visualizar los datos en tiempo real obtenidos por el SCADA SAINUX en procesos industriales. Para el cumplimiento del mismo se aplicaron algunos métodos científicos como el Analítico-Sintético, Modelación, Observación, Experimento y Entrevista. Se utilizó la metodología AUP en su modalidad UCI, el lenguaje de modelación UML, le herramienta CASE Visual Paradigm, el lenguaje de programación Java, como framework de desarrollo el Android SDK, como entorno de desarrollo integrado Android Studio. Utilizando los métodos, tecnologías y herramientas se pudo obtener una aplicación en Android capaz de visualizar los datos en tiempo real obtenidos por el SCADA SAINUX dando cumplimiento a los objetivos propuestos por esta investigación.

Palabras claves: Android, HMI, SCADA, *supervisión y control*.

1 Índice de Contenidos

Índice

Índice

Introducción

El constante desarrollo de las ciencias de la computación a partir de la década de los años 50 del siglo pasado, ha traído como uno de sus beneficios la incorporación de los ordenadores a la supervisión y control de procesos industriales. Lo anterior se ha traducido no solo en un ahorro de recursos materiales y humanos en las industrias actuales, sino también en una mayor calidad de los productos fabricados y una significativa disminución de los accidentes industriales.

Particularmente en la última década las tecnologías de la información y las comunicaciones han cambiado drásticamente el ritmo de vida de los seres humanos. El uso generalizado de internet ha posibilitado que las personas accedan a una fuente inagotable de información y les ha permitido conocerse y acercarse como nunca antes. En años recientes, el mundo en general adoptó dos dispositivos inteligentes, que se han vuelto indispensables para algunos; tales dispositivos son el *Smartphone* y la *Tablet*. Estos dispositivos incluyen un gran número de aplicaciones orientadas para diferentes fines, aunque, el grueso de la población los ocupa como medios de entretenimiento y comunicación por redes sociales. Sin embargo, la creación de aplicaciones que apoyan la productividad en diferentes ramas ha crecido paulatinamente. El *hardware* que estos dispositivos ofrecen, abre la posibilidad de interactuar y controlar sistemas SCADA, acrónimo de *Supervisory* Control *And Data Acquisition* (Supervisión, Control y Adquisición de Datos) el cual es un *software* para ordenadores que permite controlar y supervisar procesos industriales a distancia. Facilita retroalimentación en tiempo real con los dispositivos de campo (sensores y actuadores) y controla el proceso automáticamente. Provee de toda la información que se genera en el proceso productivo (supervisión, control de la calidad con relativa facilidad, control de producción, almacenamiento de datos) y permite su gestión e intervención.

Con el objetivo de contribuir al incremento del nivel de automatización de las industrias cubanas, contando con una solución propia basada en *software* libre, el Centro de Informática Industrial perteneciente a la Universidad de las Ciencias Informáticas (UCI) desarrolla el SCADA SAINUX**.**

En la actualidad el SCADA SAINUX cuenta para la interacción del usuario con el proceso industrial, con dos módulos de interfaz hombre-máquina (HMI). El primero es una aplicación de escritorio que para su ejecución requiere de la instalación del sistema operativo Debian 7 y de dependencias de varios módulos del sistema. El segundo es un sistema que utilizando tecnologías *web* visualiza los datos del proceso supervisado por el SCADA SAINUX. La utilización de las dos soluciones mencionadas anteriormente

para supervisar y controlar procesos industriales desde dispositivos móviles o tabletas se dificulta debido a:

- ✓ En la interfaz gráfica de usuario (del inglés, *Grafic User Interface*, GUI) tanto del visualizador *web* como en el de escritorio, la visibilidad se afecta en pantallas con dimensiones menores a las 12 pulgadas presentes en los dispositivos móviles.
- ✓ Debido a problemas de dependencias y compatibilidad el sistema de visualización para computadores de escritorio no puede ser desplegado en la plataforma Android.

Teniendo en cuenta lo anteriormente planteado se define como **problema de la investigación**: ¿Cómo visualizar los datos en tiempo real de los procesos industriales obtenidos por el SCADA SAINUX en dispositivos con Sistema Operativo Android?
Definiendo como **objeto de estudio** la visualización de datos en tiempo real en dispositivos con Sistema Operativo Android.
Teniendo como **campo de acción** la visualización de datos en tiempo real obtenidos por el SCADA SAINUX utilizando dispositivos con Sistema Operativo Android.
Para dar solución a este problema se tiene como **objetivo general**: Desarrollar una aplicación en Android para visualizar los datos en tiempo real de los procesos industriales obtenidos por el SCADA SAINUX.

Como tareas específicos se plantea:

- ✓ Elaborar el marco teórico de la investigación a través del estudio del estado del arte que existe actualmente sobre el tema.
- ✓ Desarrollar un mecanismo para obtener datos a través de la API de la capa de servicios web del SCADA SAINUX.
- ✓ Desarrollar una aplicación en Android para la visualización de datos en tiempo real provistos por el SCADA SAINUX.
- ✓ Realizar el levantamiento de requisitos funcionales y no funcionales.
- ✓ Seleccionar la metodología, herramientas y tecnologías idóneas para el desarrollo de la solución propuesta.
- ✓ Diseñar las interfaces gráficas de usuario típicas de un módulo de visualización de sistemas SCADA para pantallas con dimensiones menores a 12’ pulgadas.
- ✓ Validar que la solución construida cumpla con los requisitos propuestos mediante el uso de técnicas para este fin.

Para cumplir con las tareas trazadas, se utilizarán los métodos científicos siguientes:

Métodos de investigación

El método científico de investigación es la forma de abordar la realidad, de estudiar la naturaleza, la sociedad y el pensamiento, con el propósito de descubrir su esencia y sus relaciones. (Hernández, 2011) Con el objetivo de dar solución a los objetivos y las tareas propuestas se seleccionan los siguientes métodos científicos:

Métodos Teóricos:

Analítico-Sintético: Se utiliza para analizar teorías y elementos bibliográficos relacionados con la visualización de datos en tiempo real de los sistemas SCADA**.**

Modelación: Se utilizará este método ya que permitirá crear un prototipo de la aplicación para la representación de todos los requerimientos que debe cumplir el sistema.

Métodos Empíricos:

Observación: Se emplea para estudiar las características y comportamientos de las soluciones similares, permitiendo obtener toda la información referente a la visualización de la información del SCADA.

Experimento

Mediante este método se comprobará que la aplicación funcione de manera eficiente y cumpla los objetivos para los cuales fue creada, mediante pruebas al sistema.

Entrevista

Este método facilita la captura de los requisitos; consiste en una conversación entre el investigador y el o los funcionarios para la obtención de los requisitos. En este caso fue de gran ayuda para la definición de los conceptos y requerimientos pertinentes para el análisis y diseño del proceso de control del SCADA/SAINUX.

Capítulo I. Fundamentación Teórica

Introducción

En el presente capítulo se abordarán diferentes temas relacionados con las aplicaciones en Android, pues en Cuba y especialmente en la UCI prevalece un creciente desarrollo de las mismas. Se expondrán los principales conceptos asociados al sistema SCADA y una síntesis del entorno de visualización del módulo HMI. Se analizará y fundamentará la selección de la metodología, tecnología, lenguaje de programación y herramienta CASE (del inglés, *Computer Aided Software Engineering*, Ingeniería de Software Asistida por Ordenador) seleccionada para el desarrollo de la aplicación en Android.

1.1 Sistemas de Supervisión, Control y Adquisición de Datos (SCADA).

1.1.1 Historia de los SCADA

Los sistemas SCADA utilizan la computadora y tecnologías de comunicación para automatizar el monitoreo y control de procesos industriales. Estos sistemas son partes integrales de la mayoría de los ambientes industriales complejos o muy geográficamente dispersos, ya que pueden recoger la información de una gran cantidad de fuentes muy rápidamente y la presentan a un operador en una forma amigable. Los sistemas SCADA mejoran la eficacia del proceso de monitoreo y control proporcionando la información oportuna para poder tomar decisiones operacionales apropiadas. (Montero, 2013)

Los primeros sistemas automatizados SCADA fueron altamente modificados con programas de aplicación específicos para atender a requisitos de algún proyecto particular. Ingenieros de varias industrias asistieron al diseño de estos sistemas, adquiriendo su percepción de los sistemas SCADA las características de su propia industria. Los proveedores de sistemas SCADA, deseando reutilizar su trabajo previo sobre nuevos proyectos, perpetuaron esta imagen de industria específica por su propia visión de los ambientes de control con los cuales tenían experiencia. Solamente cuando nuevos proyectos necesitaron funciones y aplicaciones adicionales, entonces los desarrolladores de estos sistemas tuvieron la oportunidad de desarrollar experiencias en otras industrias. (Montero, 2013)

En la actualidad, los proveedores de SCADA están diseñando sistemas que son pensados para resolver las necesidades de muchas industrias, con módulos de software industriales específicos disponibles para proporcionar las capacidades requeridas

comúnmente. No es inusual encontrar software SCADA comercialmente disponible adaptado para procesamiento de papel y celulosa, industrias de aceite y gas, hidroeléctricas, gerencia y provisión de agua, así como el control de fluidos. Los proveedores de SCADA aún tienen tendencia en favor de algunas industrias sobre otras, por esta razón, los compradores de estos sistemas a menudo dependen del proveedor para una comprensiva solución a su requisito y generalmente prefieren seleccionar un vendedor que pueda ofrecer una completa solución con un producto estándar que cumpla con las necesidades específicas del usuario final. En el caso de seleccionar a un vendedor con experiencia limitada en la industria del comprador, el mismo debe estar preparado para asistir al esfuerzo de ingeniería necesario para desarrollar el conocimiento adicional de la industria requerida por el vendedor para una exitosa puesta en ejecución del sistema. (Montero, 2013)

1.1.2 Sistemas SCADA

Los sistemas SCADA son aplicaciones de software para la adquisición, supervisión y control de datos mediante la comunicación con los dispositivos de campo, con la finalidad de controlar procesos automatizados desde la pantalla del ordenador. Una característica importante de estos sistemas es que pueden localizar los errores de forma rápida, minimizando así los períodos de paro en las instalaciones, esto repercute en la reducción de costes de mantenimiento. La instalación de un sistema SCADA necesita un *hardware* de señal de entrada y salida, sensores y actuadores, controladores, HMI, redes, comunicaciones, base de datos entre otros. Entre las funciones principales de un sistema SCADA se encuentran la adquisición de datos para recoger, procesar y almacenar la información recibida, supervisión para observar desde un monitor la evolución de las variables de control, control para modificar la evolución del proceso, actuando bien sobre los reguladores autónomos básicos (consignas, alarmas, menús) o directamente sobre el proceso mediante las salidas conectadas. Además, poseen funciones un poco más específicas como la transmisión de información con dispositivos de campo y otras PC (del inglés, *Personal Computer*), base de datos para la gestión de datos con bajos tiempos de acceso, presentación para la representación gráfica de los datos.

Algunas de sus principales funcionalidades son:

- ✓ Supervisión: Es la capacidad de observar o monitorear aquello que sucede en el proceso industrial, el equipo o la maquinaria, por ejemplo, conocer si un interruptor se encuentra encendido o apagado.

- ✓ Adquisición de datos: Es tener la capacidad de obtener información desde el sistema de control, por ejemplo, sobre valores de humedad o presión.

- ✓ Control: Es la posibilidad de ejecutar comandos; es decir poder enviar instrucciones hacia el sistema de control.

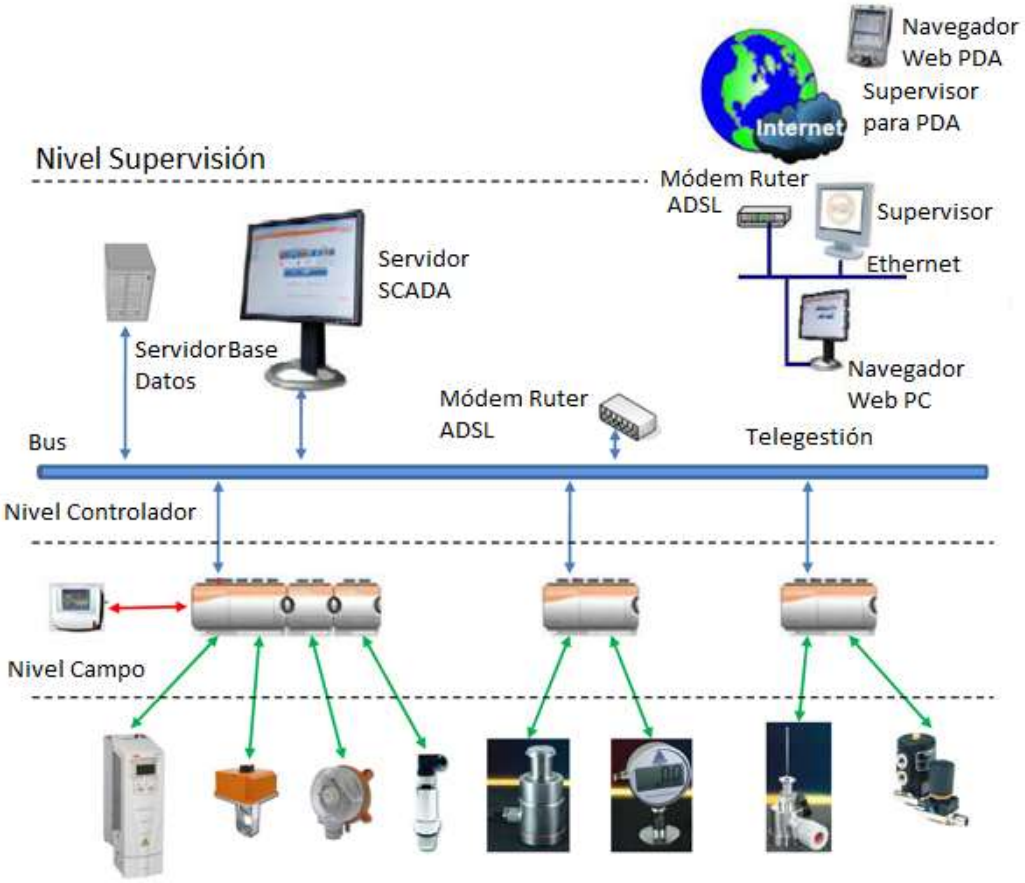


Ilustración 1 Esquema general de un sistema SCADA

1.1.3 SCADA SAINUX

La UCI desarrolla un proyecto de software con el objetivo de crear un producto SCADA, conocido como SCADA SAINUX.

SAINUX implementa funcionalidades que le permiten adquirir información del proceso que se desea supervisar, procesa la información que se recibe y brinda los resultados a los operarios del sistema.

La información se recolecta de los dispositivos de campo. Estos son utilizados para obtener datos importantes del proceso bajo supervisión. Pueden ser sensores de temperatura, de presión, controladores de válvulas, interruptores, entre otros.

Los valores que se recolectan de los dispositivos de campo se representan en el sistema como puntos. Pueden ser analógicos o digitales. Los puntos analógicos toman valores

en un rango de un dominio. Pueden representar la temperatura de una caldera, el nivel de un tanque, la presión en un área determinada. Los puntos digitales toman valores específicos de un dominio. Con frecuencia representan valores binarios (verdadero o falso). Pueden representar el estado de un motor (encendido o apagado), de una válvula (abierta o cerrada), o de un interruptor (encendido o apagado).
Los puntos calculados se obtienen de aplicar funciones de cálculos configuradas a uno o varios puntos, ya sean recolectados o calculados.
Las situaciones excepcionales que pueden ocurrir en el proceso bajo supervisión, se representan en el sistema como alarmas. Ejemplo de situaciones excepcionales pueden ser: la falla de comunicación con un dispositivo de campo, el desbordamiento de un tanque, las altas temperaturas en una caldera, fallas en interruptores y motores, entre otras.
Para operar sobre las alarmas y los puntos, son utilizados los comandos. Estos representan órdenes enviadas hacia el módulo de procesamiento, por parte de un operario o de uno de los restantes módulos del SCADA. Las operaciones sobre los puntos, hacen referencia a la lectura y escritura de sus valores. Las operaciones a realizar sobre las alarmas son de reconocer (el operario del sistema reconoce la existencia de la alarma), inhibir (la alarma deja de ser procesada por el sistema), desinhibir (la alarma puede ser procesada por el sistema) y resetear (los miembros de la alarma toman sus valores por defecto). Se utiliza el comando de sincronización para solicitar al módulo de procesamiento que publique el sumario de alarmas (contenedor con las alarmas activas). Al realizar una operación mediante un comando, se genera otro de respuesta para informar sobre el resultado de la acción realizada

El SAINUX es un sistema distribuido en módulos que trabajan de manera conjunta posibilitando el funcionamiento del sistema como un todo. Estos módulos se encuentran interconectados a través de un software para la distribución de los servicios en la red, conocido como software de comunicación entre aplicaciones. La distribución de los módulos existentes en el SCADA permite obtener configuraciones escalables en dependencia de los requisitos que presente cada aplicación. Es un sistema en tiempo real y presenta una arquitectura distribuida. Está dividido en varios subsistemas entre los que se encuentran:

- ✓ **Comunicación:** Es la capa de software, que se encarga de la comunicación entre los diferentes módulos que forman parte del sistema. Este módulo tiene como finalidad proporcionar la capa de comunicación de alto nivel, tanto

sincrónica, como asincrónica, para la comunicación de todos los módulos que conforman el sistema SCADA.

- ✓ **Adquisición:** Es el encargado de la adquisición, recepción, procesamiento y distribución de los datos provenientes del campo.

- ✓ **Configuración:** El servicio de configuración está formado por un grupo de componentes cada uno con una tarea específica y una base de datos que contendrá las configuraciones de cada uno de los módulos que conformen el proyecto activo. Es el encargado de almacenar, persistir y suministrar la información base para el funcionamiento de los demás módulos del SCADA.

- ✓ **Almacenamiento de datos históricos:** Es el encargado de almacenar la información del sistema para que posteriormente pueda ser empleada, por ejemplo, en generación de reportes, tendencias o en gestión de producción. La base de datos histórica (BDH) contendrá la información persistente de los datos recolectados de los dispositivos.

- ✓ **Seguridad:** Proporciona las funcionalidades necesarias para garantizar el trabajo autorizado a usuarios y módulos, además brinda las herramientas para la protección contra ataques maliciosos o involuntarios al sistema por parte de personas o recursos, tales como fallas de energía, problemas de red o servidores.

- ✓ **Módulo HMI**

  El módulo de HMI en el SCADA se encarga de representar, en un ordenador, los procesos que ocurren en el campo, muestra los componentes implicados, los sensores, las estaciones remotas, y el sistema de comunicación dándole al operador total control. Este módulo es el que permite al operador estar en contacto directo con el sistema, realizar la supervisión y el control del proceso en general. Está compuesto por dos partes fundamentales:

  - ➢ **Entorno de configuración (EC) o Editor:** Permite configurar varios procesos o partes de ellos, aquí se definen y gestionan las variables, los manejadores, los comandos, las alarmas y variadas opciones adicionales. Este entorno funciona como una aplicación de diseño tradicional, con la peculiaridad que los sinópticos se confeccionan a partir

de objetos y primitivas básicas predefinidas, que se pueden agrupar, combinar, transformar, importar y exportar entre otras. (Penin, 2011)

- **Entorno de visualización (EV) o Visualizador**: Se encarga de visualizar las animaciones y los objetos definidos en el editor, muestra lo que está ocurriendo en el campo en tiempo real, es el que envía los comandos a las estaciones remotas, quién recibe los valores de las variables, interactúa con la mayoría de los operadores pues se emplea para supervisar el proceso de manera directa. (Penin, 2011)

1.2 Conceptos Esenciales

1.2.1 Variable

Derivada del término en latín *variabilis*, variable es una palabra que representa a aquello que varía o que está sujeto a algún tipo de cambio. Se trata de algo que se caracteriza por ser inestable, inconstante y mudable. En otras palabras, una variable es un símbolo que permite identificar a un elemento no especificado dentro de un determinado grupo. Este conjunto suele ser definido como el conjunto universal de la variable (universo de la variable, en otras ocasiones), y cada pieza incluida en él constituye un valor de la variable. (Variable, 2016)

1.2.2 Alarma

Se entiende por alarma la señal o aviso que advierte sobre la proximidad de un peligro. El aviso de alarma informa a la comunidad en general o a una entidad específica que deben seguir ciertas instrucciones de emergencia dado que se ha presentado una amenaza. (Alarma, 2016)

1.2.3 Comando

En informática, un comando (*command*, orden, mandato) es una orden que se le da a un programa de computadora que actúa como intérprete del mismo, para así realizar una tarea específica. Generalmente un comando se le da a una interfaz de línea de comandos, como un *Shell* (terminal). (ALEGSA, 2016)

1.2.4 Puntos

El flujo principal de información en los sistemas SCADA lo constituyen las variables (puntos), también conocidas como *tags* o señales de campo. Estas variables pueden

representar innumerables indicadores como son: presión, temperatura, flujo, potencia, peso, intensidad de corriente, voltaje, potencial hidrógeno, densidad, carga, resistencia o capacitancia entre otros. Las variables son adquiridas mediante instrumentación o utilizando sensores conectados a autómatas o equipos de control, luego de convertidas a señales eléctricas, estas variables pasan a ser estructuras que contienen datos, los cuales pueden ser de tipos simples (entero, flotante, cadenas, u otros) o de tipos complejos. Además es posible obtener puntos calculados, como resultado de la realización de operaciones matemáticas sobre un conjunto de variables. La información de estas variables permite conocer el estado del sistema y su historia. (ZAPATA, 2011)

1.2.5 Programación multihilos

Debido a que la frecuencia de reloj del procesador es limitada, resulta necesario dividir el procesamiento en varias tareas que se ejecuten de forma simultánea, con el objetivo de aumentar su rendimiento y alcanzar mayor eficiencia. Una de las formas de realizar procesamiento simultáneo es mediante la utilización de hilos. Los hilos o procesos ligeros constituyen una abstracción de un programa en ejecución. Son una representación más ligera de los procesos tradicionales, los cuales están formados por un hilo principal, totalmente independiente, con su propia zona de memoria y contador de programa. El mejor aprovechamiento del procesador no se realiza con un único hilo o proceso. La creación dentro de un proceso de múltiples hilos, posibilita resolver distintas tareas de forma simultánea y aprovecha el trabajo de forma cooperada entre los distintos hilos.

La utilización de hilos tiene su ventaja sobre los procesos. Se tarda mucho menos tiempo en crear un hilo nuevo dentro de un proceso existente que en crear un nuevo proceso. Igualmente se necesita menos tiempo para terminar un hilo y para cambiar el estado entre dos hilos de un mismo proceso.

Otra de las formas en que los hilos aportan eficiencia es en la comunicación entre programas en ejecución. En la mayoría de los sistemas operativos, la comunicación entre procesos independientes necesita del núcleo para ofrecer protección y proporcionar los mecanismos necesarios para la comunicación. Mientras que los hilos de un mismo proceso al compartir el mismo espacio de memoria no necesitan de la intervención del núcleo para establecer la comunicación. (Stallings, 1997)

Al igual que los procesos, un hilo puede estar en diferentes estados:

- ✓ **Listo:** se encuentra en espera de su turno de ejecución.

- ✓ **Ejecución:** posee el uso del procesador y se encuentra activo.
- ✓ **Bloqueado:** en espera que otro hilo elimine el bloqueo.
- ✓ **Terminado:** cuando ha finalizado su trabajo.

1.2.6 Sumario

Representa en forma tabulada la información referente a las mediciones o datos que el sistema genera u obtiene ya sea desde los dispositivos de campo o de las bases de datos. Dicha mediciones o datos pueden agruparse o clasificarse en alarmas, eventos, estados del sistema, puntos, reportes, despliegues, estadísticas.

1.2.7 Despliegue

Es un resumen esquematizado con la ventaja de permitir visualizar la estructura y organización de los elementos que componen el proceso, así como el comportamiento de los mismos, los cuales pueden estar ubicados en un área geográficamente extensa para que el operador monitoree constantemente cada de uno de los componentes que conforman el proceso.

1.3 Análisis de los sistemas existentes

1.3.1 Visualizador de HMI Qt

Software de Interfaz Hombre-Máquina para la supervisión y control empleando el SCADA SAINUX. Está diseñado para ejecutarse en plataformas Linux y es desarrollado por el Centro de Informática Industrial (CEDIN) de la UCI empleando el *framework* Qt. Este sistema brinda un conjunto de funcionalidades para la supervisión y control de procesos industriales entre las que destacan: visualización de información en tiempo real en sumarios de puntos, alarmas, subcanales y dispositivos. Permite la ejecución de acciones operacionales y de control sobre el sistema a través de interfaces de detalles de puntos y dispositivos. Este sistema es capaz de mostrar despliegues o sinópticos y gráficos de tendencias. Requiere para su ejecución de requisitos mínimos un procesador Core i3 y 1 Gb de RAM.

1.3.2 Servidor HMI

Software de HMI para brindar información y atender las solicitudes que se realizan desde clientes HMI, es desarrollado por el CEDIN de la UCI. Este sistema brinda un conjunto de funcionalidades para la supervisión y control de procesos industriales entre las que destacan: el envío a los clientes de la información referente a los sumarios de puntos, alarmas, subcanales y dispositivos. Permite atender las solicitudes de ejecución de acciones operacionales y de control sobre el sistema.

1.3.3 Capa de Servicios Web

Es una aplicación desarrollada en la línea de interfaz hombre-máquina del Centro de Informática Industrial, utilizando el lenguaje de programación PHP. Las funcionalidades de la misma están disponibles a través del protocolo Http mediante el servidor web Apache. El intercambio de información entre los clientes y el servidor se realiza utilizando la notación JSON.

1.3.4 InTouch

Es un SCADA de la rama *Wonderware* de la división de administración de Operaciones de la compañía inglesa *Invensys*. Permite configurar alarmas y establecerles hasta 999 niveles de prioridad y hasta 8 niveles de jerarquía entre grupos de alarma con posibilidad de hasta 16 subgrupos para cada uno de ellos. Posibilita visualizar todas o un extracto de ellas de forma histórica y grabar en disco o imprimir en diferentes formatos personalizables. Las funciones de alarmas distribuidas incluyen reconocimiento global o selectivo, desplazamiento por la lista y visualización de alarmas procedentes de diferentes servidores en un único panel. (MEASURESOFT, 2009)

1.3.5 Simatic WinCC apk

El *SIMATIC WinCC Sm@rtClient* apk (aplicación), en combinación con el *SIMATIC WinCC Sm@rtServer*, permite la operación remota desde el móvil y la observación de *SIMATIC* HMI a través de *Ethernet* / *WLAN* (red inalámbrica). La aplicación está disponible para los paneles *SIMATIC* HMI *Comfort* y *WinCC Runtime Advanced*.

La estación toma el papel de un *Sm@rtServer*, mientras que el *Smartphone* / *Tablet* asume el papel de la *Sm@rtClient*, la funcionalidad *Sm@rtServer* puede activarse fácilmente con un solo clic del ratón en las opciones de configuración del dispositivo. La pantalla se puede visualizar en el *Sm@rtClient* por medio de una pantalla *Sm@rtClient* en el modo "Solo", sino una operación coordinada en toda regla se puede configurar también. "Operación coordinada" implica que sólo un usuario a la vez tiene el derecho de explotación, es decir, ya sea la estación del operador con la función *Sm@rtServer* o

el *Sm@rtClient* App. Además, la aplicación es compatible con la configuración de fácil conexión mediante la detección automática del panel de operador.

No sólo la aplicación *Sm@rtClient* muestra una pantalla seleccionada, sino que, además, muestra la disposición completa del dispositivo de la estación, incluyendo los botones funcionales de hardware en el dispositivo. Como tal, el funcionamiento del dispositivo se puede realizar como si el usuario estaba directamente en frente del dispositivo, sólo que los botones no se presionan en realidad, sino más bien se activan con una prensa en la pantalla de un teléfono inteligente o tableta. (SIEMENS, 2016)

1.3.6 Visualizador HMI Web

Software de Interfaz Hombre-Máquina para la supervisión y control empleando el SCADA SAINUX. Está diseñado para ejecutarse en un navegador *web* sin necesidad de software adicional sobre cualquier plataforma y es desarrollado por el Centro de Informática Industrial de la Universidad de las Ciencias Informáticas. Este sistema brinda un conjunto de funcionalidades para la supervisión y control de procesos industriales entre las que destacan: visualización de información en tiempo real en sumarios de puntos, alarmas, subcanales y dispositivos. Permite la ejecución de acciones operacionales y de control sobre el sistema a través de interfaces de detalles de puntos y dispositivos. Este sistema es capaz de mostrar despliegues o sinópticos y gráficos de tendencias. Requiere para su ejecución de requisitos mínimos navegador *web Google Chrome* versión 35.0 o superior, Core i3 y 512 Mb de RAM.

Después de analizar los sistemas existentes en la actualidad, para la creación de la aplicación se utilizó diversas funcionalidades presentes en los mismo las cuales fueron: como seria la comunicación entre el cliente y servidor, la representación gráfica de las funcionalidades que iba a tener la aplicación, se seleccionó como mostraríamos los sumarios, como estrían compuestas las peticiones que realizaría la aplicación y cuál sería la estrategia marcaria que íbamos a utilizar.

1.4 Lenguajes de programación para Android

Un lenguaje de programación es una técnica estándar de comunicación que permite expresar las instrucciones que han de ser ejecutadas en una computadora (Lévénez, 2007).

En la actualidad existen varios lenguajes de programación que son utilizados para desarrollar aplicaciones en Android. A continuación se analizaran las características del lenguaje de programación utilizado.

1.4.1 Java

Este lenguaje de programación fue desarrollado por James Gosling y su equipo en Sun *Microsystems*, entre 1990 y 1994, pensado originalmente como un reemplazo de C++, orientado a dispositivos embebidos, y a la televisión interactiva, posteriormente para rescatarlo del fracaso y fue reorientado hacia su aplicación en la *Web*. (Python, 2013)
El lenguaje Java tiene cinco características principales entre las que se encuentran: orientado a objetos, multiplataforma, soporte integrado para redes de computadoras, diseñado para ejecutar código de fuentes remotas de modo seguro, y fácil de usar.
Este lenguaje de programación ha sido probado, mejorado y ampliado. Gracias a su versatilidad, eficiencia y portabilidad, se ha convertido en un recurso inestimable.

"Un lenguaje simple. Orientado al objeto, distribuido, interpretado, sólido, seguro, de arquitectura neutral, portable, de alto desempeño, de multihilos y dinámico". (Java, 2015)

Se escoge el lenguaje Java dado que presenta las siguientes ventajas a la hora del desarrollo de Aplicaciones:

Simplicidad: Java proporciona un interfaz fácil para los usuarios y los desarrolladores. Es considerado como el lenguaje más simple en comparación a otros lenguajes de programación. En Java se ha eliminado el uso de punteros y también remplazado la complejidad de las múltiples herencias presentes en C++. (Java, 2015)

Portabilidad e Independiente de la Plataforma Comportamiento: Java es independiente de la plataforma. Proporciona la facilidad para "Escribir una Vez y Ejecutar en cualquier Lugar". Las aplicaciones desarrolladas por el uso de este lenguaje se pueden ejecutar en cualquier plataforma de hardware y software, son compatibles con todos los navegadores. (Java, 2015)

Asignación: Java tiene la característica de asignación de Pila del sistema. Ayuda a que los datos se almacenan y se pueden restaurar fácilmente. La pila de gestión es un proceso de arreglo de objetos en LIFO (Last In First Out). Este sistema de gestión hace que sea fácil de almacenar y restaurar cualquier objeto. A diferencia de otros lenguajes donde el programador tiene que asignar datos y recoger la basura, Java tiene la facilidad de recolección de basura automática y la asignación de memoria. (Java, 2015)

Distribuido: La plataforma en la que dos o más equipos pueden trabajar juntos en una red. Java tiene una gran capacidad de red. La creación de redes en Java es más fácil que escribir un programa de redes, se siente como el envío y recepción de archivos. (Java, 2015)

Además de las ventajas mencionadas anteriormente, el lenguaje de programación Java fue elegido por los creadores de Android para la creación del Kit de desarrollo de aplicaciones por defecto de este sistema operativo. Es debido a lo expresado anteriormente que las interfaces de programación de aplicaciones (API) más extensas y con soporte oficial para la plataforma Android se encuentran disponibles para el lenguaje de programación Java.

1.5 Notación para la transferencia de datos

1.5.1 JSON

JSON (*JavaScript Object Notation*) es un formato de intercambio de datos ligero. Es fácil para los seres humanos a leer y escribir. Es fácil para las máquinas para analizar y generar. Se basa en un subconjunto del lenguaje de programación JavaScript, estándar ECMA-262 3ª Edición - Diciembre de 1999. JSON es un formato de texto que es completamente independiente del lenguaje, pero utiliza las convenciones que son familiares para los programadores del C-familia de lenguajes, incluyendo C, C ++, C #, Java, *JavaScript*, *Perl*, *Python*, y muchos otros. Estas propiedades hacen JSON un lenguaje ideal de intercambio de datos. (JSON, 2016)

JSON se basa en dos estructuras:

- ✓ Una colección de pares nombre / valor. En varios idiomas, esto se realiza como un objeto, registro, estructura, diccionario, tabla hash, lista con clave, o una matriz asociativa.
- ✓ Una lista ordenada de valores. En la mayoría de los idiomas, esto se realiza como una matriz, vector, lista o secuencia.

Estas son estructuras de datos universales. Prácticamente todos los lenguajes de programación modernos los apoyan de una forma u otra. Tiene sentido que un formato de datos que es intercambiable con los lenguajes de programación también se basará en estas estructuras. (JSON, 2016)

La estructura de un JSON está compuesta de la siguiente forma:
Un objeto es un conjunto desordenado de pares nombre / valor. Un objeto comienza con {(llave izquierda) y termina con} (llave derecha). Cada nombre es seguido por: (dos puntos) y el nombre / valor pares están separados por, (coma). (JSON, 2016)

1.6 Entorno de Desarrollo Integrado (IDE)

Los Entornos de Desarrollo Integrado (IDE) son programas informáticos compuestos por herramientas de programación como; un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI) que permite desarrollar un *software* utilizando un lenguaje específico. El IDE a emplear se describe a continuación.

1.6.1 **Android Studio** (Android, 2015)

Android Studio es el nuevo entorno de desarrollo creado por Google para facilitar la tarea de programación dentro de su sistema operativo. Hasta este momento toda la documentación oficial gira en torno a Eclipse como IDE recomendado por Google. Fue en Mayo de 2013 durante el Google I/O, donde Google dio a conocer este nuevo IDE para Android, y creemos que a partir de este momento se centrarán en migrar y adaptar toda la documentación oficial para que gire alrededor de Android Studio.
Está basado en el IDE *Intellij* IDEA *Community Edition*, con lo cual goza de todas las características de este entorno de desarrollo más algunas otras características propias que ha incorporado Google.

Características más importantes

- ✓ **Sistema de construcción y Ayudas para Codificación:** Nuevo sistema de construcción y empaquetación de un proyecto Android. Añade compatibilidad con Maven, utiliza un lenguaje específico de dominio basado en *Groovy* y permite añadir nuestros servidores de integración continua. Permite edición más fluida, refactorización más potente y mejorado análisis de código.
- ✓ **Previsualización y Generación de recursos:** Permite previsualización de recursos. Al situarse sobre un drawable con el cursor, se puede visualizar de forma rápida el recurso *drawable* en varios tipos de densidades y resoluciones. Añade un selector de color que aparece tras indicar un color en nuestro código y pulsar sobre él.
- ✓ **Ayudas para el Diseño y Detección de Errores:** Incluye diálogos para facilitar la generación de recursos para distintas configuraciones. Cuenta con un mejorado depurador de código que facilita y ayuda en la labor de corrección de errores de forma automática. Indica fallos y mejoras que se puede incluir en el código. El sistema de análisis es personalizado permitiendo que trabaje sobre áreas específicas de Android. Por ejemplo, trabajar sobre un método en concreto al que se pasa como parámetro un S*tring*. El sistema comprueba y visualiza una

serie de opciones para pasarle como parámetro y así evitar posible fallos y ralentizar el trabajo permitiendo mejorar la productividad.

- ✓ **Acceso a Servicios de Google y Refactorización potente:** Posee un sistema de refactorización mejorado que permite por ejemplo que se pueda cambiar el nombre de un recurso de tipo imagen y este modifica automáticamente el nombre del identificador. Además Google ha facilitado el uso de sus recursos en los proyectos. Android Studio facilita el acceso a herramientas desde el propio entorno, integrando por ejemplo los servicios GCM y la inclusión de un nuevo plugins (ADT Translation Manager Plugins) para la traducción de las aplicaciones e integrado con el servicio de traducción de *Google Play Developer Console*.

Además de las características anteriores este IDE aportó las siguientes ventajas en el cumplimiento del objetivo del trabajo:

- ✓ Está basado en IntelliJ IDEA, uno de los IDE para java de primer nivel. Más serio, más versátil, más potente, más actual, y más parecido a un proyecto en java.
- ✓ Utiliza Gradle, nos posibilita:

    - ➢ Facilita reusar código y recursos.
    - ➢ Facilita configurar, extender y personalizar el proceso de construcción de la aplicación.
    - ➢ Facilita la distribución del código y por tanto trabajar en equipos.
    - ➢ Gestiona las dependencias de una forma cómoda y potente.
    - ➢ Permite compilar desde línea de comandos.
    - ➢ Hace fácil crear distintas versiones de la aplicación, dando la posibilidad de hacer una distribución multi-apk, para distintos dispositivos o una versión gratis y otra de pago, o una versión de prueba que carga distintos recursos, apuntando a *webservices* distintos y usa estadísticas distintas.

1.7 Metodologías de desarrollo de software

La Real Academia Española define la Metodología como un conjunto de métodos que se siguen en una investigación científica o en una exposición doctrinal.

Se plantea que una metodología es el conjunto ordenado de pasos a seguir para cumplir un objetivo. Dicho objetivo, en la ingeniería de software, es el desarrollo de *software* de alta calidad que cumpla con las necesidades del cliente dentro de un plan y un presupuesto predecible. (Duenser, 2012)

Una metodología representa el camino para desarrollar software de una manera sistemática. Las metodologías persiguen tres necesidades principales:

- ✓ Mejores aplicaciones, conducentes a una mejor calidad.
- ✓ Un proceso de desarrollo controlado.
- ✓ Un proceso normalizado en una organización, no dependiente del personal.

Las características se la metodología a seguir durante el desarrollo de la aplicación se analizan a continuación.

1.7.1 **Metodología AUP**

Existen metodologías ágiles y robustas. Las metodologías ágiles son apropiadas para guiar proyectos de poco volumen que requieran una rápida implementación, un ejemplo es *Extreme Programming* (XP). Las metodologías robustas pueden ser empleadas para guiar el proceso de desarrollo de proyectos grandes o pequeños, aunque son más apropiadas para proyectos grandes que por su importancia requieren una fuerte planificación y generación de documentación, ejemplo RUP (*Rational Unified Process*). Este proyecto tiene gran importancia ya que se enfoca en la gestión semi-automatizada de los artefactos que se generan durante el proceso de Prueba, por lo que se requiere de una metodología capaz de guiar el proceso con precisión, que contribuya a obtener un *software* fiable, sin errores. (G.d, 2014)

Proceso Unificado Ágil (AUP, por sus siglas en inglés) es la metodología que se ajusta a la necesidad del proyecto porque combina características de la metodología ágil XP con los artefactos de RUP. (G.d, 2014) Dentro de las características particulares de AUP, tenemos que es una versión simplificada de la metodología RUP.

AUP-UCI versión 1.2

Actualmente el desarrollo de software dentro de la actividad productiva de la UCI se caracteriza por el uso de diferentes metodologías de desarrollo entre robustas y ágiles, a pesar de la variedad de metodologías usadas, se ha comprobado que muy pocos proyectos la aplican en su totalidad. Por lo que para lograr erradicar los problemas detectados, se propone crear un cronograma tipo y un método de estimación para cada una de las metodologías que hoy se usan en los proyectos o converger a una única metodología que cubra las particularidades de cada uno. Por lo tanto se decide escoger una metodología para ser adaptada a lo que ya la Universidad ha estado proponiendo como ciclo de vida de los proyectos, sin alejarse de lo que hasta el momento se ha trabajado e introducir la menor cantidad de cambios posibles. (Sánchez, versión 1.2)

Al no existir una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigiéndose así que el proceso sea configurable. Se decide hacer una variación de la metodología AUP, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI. (Sánchez, versión 1.2)

Una metodología de desarrollo de software tiene entre sus objetivos aumentar la calidad del software que se produce, de ahí la importancia de aplicar buenas prácticas, para ello nos apoyaremos en el Modelo CMMI-DEV v1.3. El cual constituye una guía para aplicar las mejores prácticas en una entidad desarrolladora. Estas prácticas se centran en el desarrollo de productos y servicios de calidad. (Sánchez, versión 1.2)

Fases de AUP-UCI (Rodríguez, 2010)

- ✓ **Inicio:** Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo; y decidir si se ejecuta o no el proyecto.
- ✓ **Ejecución:** En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elabora la arquitectura y el diseño, se implementa y se libera el producto.
- ✓ **Cierre:** En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

Esta versión ágil de la metodología RUP se basa en los principios siguientes:

- ✓ **Simplicidad**: Refiere a que todo se describe de forma concisa usando poca documentación, no muchas de ellas. (Rodríguez, 2010)
- ✓ **Agilidad**: Refiere al ajuste de los valores y principios de la Alianza Ágil. Es por ello que es necesario centrarse en actividades de alto valor: La atención se centra en las actividades que en realidad lo requieren, no en todo el proyecto. (Rodríguez, 2010)
- ✓ **Herramienta de la independencia**: Refiere a que se puede usar cualquier conjunto de herramientas que desea con el AUP. Sugiere utilizar las herramientas idóneas para el trabajo, que normalmente son herramientas simples o incluso herramientas de código abierto. (Rodríguez, 2010)
- ✓ **Adaptabilidad:** Refiere a que la metodología AUP es un producto de fácil uso, independiente de la herramienta usada. (Rodríguez, 2010)

Descripción de las disciplinas

AUP propone 7 disciplinas (Modelo, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de proyecto y Entorno), se decide para el ciclo de vida de los proyectos de la UCI tener 7 disciplinas también pero a un nivel más atómico que el definido en AUP. Los flujos de trabajo: Modelado de negocio, Requisitos y Análisis y diseño en AUP están unidos en la disciplina Modelo, en la variante para la UCI se consideran a cada uno de ellos disciplinas. Se mantienen la disciplina Implementación, en el caso de Prueba se desagrega en 3 disciplinas: Prueba Internas, de Liberación y Aceptación. Las restantes 3 disciplinas de AUP asociadas a la parte de gestión para la variante UCI se cubren con las áreas de procesos que define CMMI-DEV v1.3 para el nivel 2, seria CM (Gestión de configuración), PP (Planeación de proyecto) y PMC (Monitoreo y control de proyecto). Para una mayor comprensión se muestra la siguiente tabla algunas de las disciplinas. (Sánchez, versión 1.2)

Disciplinas AUP	**Disciplinas Validación AUP- UCI**	**Objetivos Disciplinas(Variación AUP-UCI)**
Modelo	Modelado de negocio	El Modelado del Negocio es la disciplina destinada a comprender los procesos de negocio de una organización. Se comprende cómo funciona el negocio que se desea informatizar para tener garantías de que el software desarrollado va a cumplir su propósito. Para modelar el negocio se proponen las siguientes variantes : **1-**Casos de Uso del Negocio (CUN). **2-**Descripción de Proceso de Negocio (DPN). **3-**Modelo Conceptual (MC). A partir de las variantes anteriores se condicionan cuatro escenarios para modelar el sistema en la disciplina Requisitos.

	Requisitos	El esfuerzo principal en la disciplina Requisitos es desarrollar un modelo del sistema que se va a construir. Esta disciplina comprende la administración y gestión de los requisitos funcionales y no funcionales del producto. Existen tres formas de encapsular los requisitos [Casos de Uso del Sistema (CUS), Historias de usuario (HU) y Descripción de requisitos por proceso (DRP)], agrupados en cuatro escenarios condicionados por el Modelado de negocio.
	Análisis y diseño	En esta disciplina, si se considera necesario, los requisitos pueden ser refinados y estructurados para conseguir una comprensión más precisa de estos, y una descripción que sea fácil de mantener y ayude a la estructuración del sistema (incluyendo su arquitectura). Además, en esta disciplina se modela el sistema y su forma (incluida su arquitectura) para que soporte todos los requisitos, incluyendo los requisitos no funcionales. Los modelos desarrollados son más formales y específicos que el análisis.
Implementación	Implementación	En la implementación, a partir de los resultados del Análisis y Diseño se construye el sistema.

Pruebas	Pruebas internas	En esta disciplina se verifica el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas. Se deben desarrollar artefactos de prueba como: diseños de casos de prueba, listas de chequeo y de ser posibles componentes de prueba ejecutables para automatizar las pruebas.
	Pruebas de liberación	Pruebas diseñadas y ejecutadas por una entidad certificadora de la calidad externa, a todos los entregables de los proyectos antes de ser entregados al cliente para su aceptación.
	Pruebas de Aceptación	Es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido.

Tabla 1. 1 Algunas disciplinas utilizadas en AUP-UCI

A continuación se esboza brevemente como será aplicada la metodología AUP-UCI para conducir el proceso de desarrollo de la aplicación:

Para la creación de la aplicación se utilizará en la disciplina de modelado de negocio el caso modelo conceptual donde: "*Un Modelo Conceptual o Modelo de Dominio captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan las "cosas" que existen o los eventos que suceden en el entorno en el que trabaja el sistema*". (JACOBSON, 2000), en la disciplina de requisitos se utilizará la historia de usuario, la cual es la técnica donde se especifican los requisitos del software. Las mismas no son más que tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es dinámico y flexible, en

cualquier momento historias de usuario pueden romperse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. (Fontanarossa, 2006)

Se utilizará el diagrama de componentes el cual es uno de los diagramas del Lenguaje Unificado de Modelado. Representa cómo un sistema de software es dividido en componentes y muestra las dependencias entre estos. Los componentes físicos incluyen archivos, cabeceras, bibliotecas compartidas, módulos, ejecutables, o paquetes. Este tipo de diagrama prevalece en el campo de la arquitectura de software pero pueden ser usados para modelar y documentar cualquier arquitectura de sistema. (Sparxsystems, 2016) También se empleará para modelar la vista estática de un sistema. Muestra la organización y las dependencias entre un conjunto de componentes. No es necesario que un diagrama incluya todos los componentes del sistema, normalmente se realizan por partes. Cada diagrama describe un apartado del sistema. Con el Modelo de Despliegue, el cual es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo, se logra contar con una entrada fundamental en las actividades de diseño e implementación debido a que la distribución del sistema tiene una influencia principal en su diseño. (González, 2013)

1.8 Herramienta CASE para UML

Las Herramientas CASE son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero. Estas herramientas pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores, entre otras. (Rodríguez, 2010)

Principales objetivos de las herramientas CASE:

- ✓ Mejorar la productividad en el desarrollo y mantenimiento del *software*.
- ✓ Aumentar la calidad del *software*.
- ✓ Mejorar el tiempo, coste de desarrollo y mantenimiento de los sistemas informáticos.
- ✓ Mejorar la planificación de un proyecto.
- ✓ Aumentar la biblioteca de conocimiento informático de una empresa ayudando a la búsqueda de soluciones para los requisitos.

Las características de la herramienta CASE a utilizar se exponen a continuación.

1.8.1 Visual Paradigm

Visual Paradigm es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de *software*: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. También proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML.

Emplea una rápida respuesta utilizando moderadamente los tiempos del procesador, lo que le permite manejar grandes y complicadas estructuras de un proyecto en una forma eficiente, que requiera de una configuración de escritorio.

Esta herramienta case ofrece estabilidad de ejecución en diferentes sistemas operativos y la facilidad de abrir y trabajar con un modelo UML empleando el mismo programa sin importar el sistema operativo y sin afectar el trabajo hecho; además esta herramienta guarda todo el modelo en un solo fichero donde basta con copiarse solo ese fichero. Visual Paradigm es una herramienta que trabaja en ordenadores “poco potentes”.

1.9 Framework de desarrollo

Kit de desarrollo de software (SDK)

Un kit de desarrollo de software o SDK (siglas en inglés de software development kit) es generalmente un conjunto de herramientas de desarrollo de software que le permite al programador o desarrollador crear aplicaciones con el fin de lograr un mejor uso de las técnicas a desarrollar en este sistema. (ORACLE, 2015)

El SDK de Android, incluye un conjunto de herramientas de desarrollo. Comprende un depurador de código, biblioteca, un simulador de teléfono basado en QEMU, documentación, ejemplos de código y tutoriales. Las plataformas de desarrollo soportadas incluyen GNU/Linux, Mac OS X 10.5.8 o posterior, y Windows XP o posterior. La plataforma integral de desarrollo (IDE, *Integrated* Development *Environment*) soportada oficialmente es Android Studio junto con el complemento ADT (Android Development Tools *plugin*). Además, los programadores pueden usar un editor de texto para escribir ficheros Java y XML y utilizar comandos en un terminal (se necesitan los paquetes JDK, Java Development Kit y Apache Ant) para crear y depurar aplicaciones, así como controlar dispositivos Android que estén conectados (es decir, reiniciarlos, instalar aplicaciones en remoto, etc.).

Las Actualizaciones del SDK están coordinadas con el desarrollo general de Android. El SDK soporta también versiones antiguas de Android, por si los programadores necesitan instalar aplicaciones en dispositivos ya obsoletos o más antiguos. Las herramientas de desarrollo son componentes descargables, de modo que una vez instalada la última

versión, pueden instalarse versiones anteriores y hacer pruebas de compatibilidad. (Conran, 2012)

1.10 Conclusiones parciales del capítulo

Con el estudio del marco teórico que encierra esta investigación se enunciaron los conceptos principales asociados al problema a resolver y se sentaron las bases necesarias para la comprensión del objetivo de estudio. Además se investigaron y dieron uso las tecnologías a utilizar en el desarrollo de la propuesta de solución las cuales fueron: como lenguaje de programación para establecer la lógica entre las clases Java, como entorno de desarrollo integrado Android Studio, como metodología AUP-UCI, como lenguaje de modelación UML, como herramienta CASE Visual Paradigm y como notación para la comunicación con el servidor JSON. Además después de una análisis de los sistemas existentes pudimos seleccionar aspectos de los mismo en la aplicación a desarrollarlos cuales son: estructura de la aplicación, los medio de comunicación, la capacidad en cuanto a la representación gráfica de la aplicación y la forma de conformar las peticiones que iban a ser enviadas a la capa de servicios web. A partir de estos puntos se comenzará el desarrollo de la propuesta de solución.

Capítulo II. Análisis y Diseño de la Aplicación

Introducción

En este capítulo se abordan las características que el sistema debe tener. El desarrollo de la aplicación se centra en el Proceso Unificado Ágil (AUP-UCI) para la modelación de los artefactos utilizando Visual Paradigm. Según la metodología, durante el ciclo de vida del proceso de desarrollo del software, se llevan a cabo un conjunto de disciplinas entre las que se encuentran: Modelación del negocio, Requerimientos, Análisis y Diseño, con el objetivo de detallar los procesos que describen el dominio y así poder entender el universo en que se emplaza el sistema y contribuir a la comprensión de los requisitos del sistema. Es por ello que se presenta el Modelo de Dominio de la solución que se propone, así como la descripción de los requisitos funcionales y no funcionales identificados. Además, para la interacción con el cliente se utilizan las historias de usuario. Se detalla tanto la arquitectura utilizada para la creación de la aplicación como los patrones de diseños utilizados en la misma, incluyendo diagrama de clases, brindando una entrada fundamental al desarrollo del mismo.

2.1 Análisis de la Solución

La presente investigación está orientada al desarrollo de una aplicación Android, capaz de consumir la API (*application interface*) de la Interfaz de Servicios Web para visualizar los datos en tiempo real provenientes del SCADA SAINUX en un teléfono móvil o tableta. En la aplicación se deberán visualizar los datos en tiempo real de varias entidades del sistema SCADA tales como: puntos, alarmas, subcanales y dispositivos. La representación de los datos deberá estar acorde con las normas definidas en el proyecto para cada tipo de entidad, teniendo en cuenta además el tamaño reducido de las pantallas de los dispositivos a los cuales está destinada la aplicación.

2.2 Modelo de dominio

Un modelo del dominio es una representación visual de las clases conceptuales u objetos del mundo real en un dominio de interés, por lo que se le mostrará al usuario los principales conceptos que intervienen en el dominio del sistema y ayudará a identificar varias clases que se utilizarán dentro del mismo.

A continuación se presenta el modelo de dominio creado para el desarrollo de la aplicación:

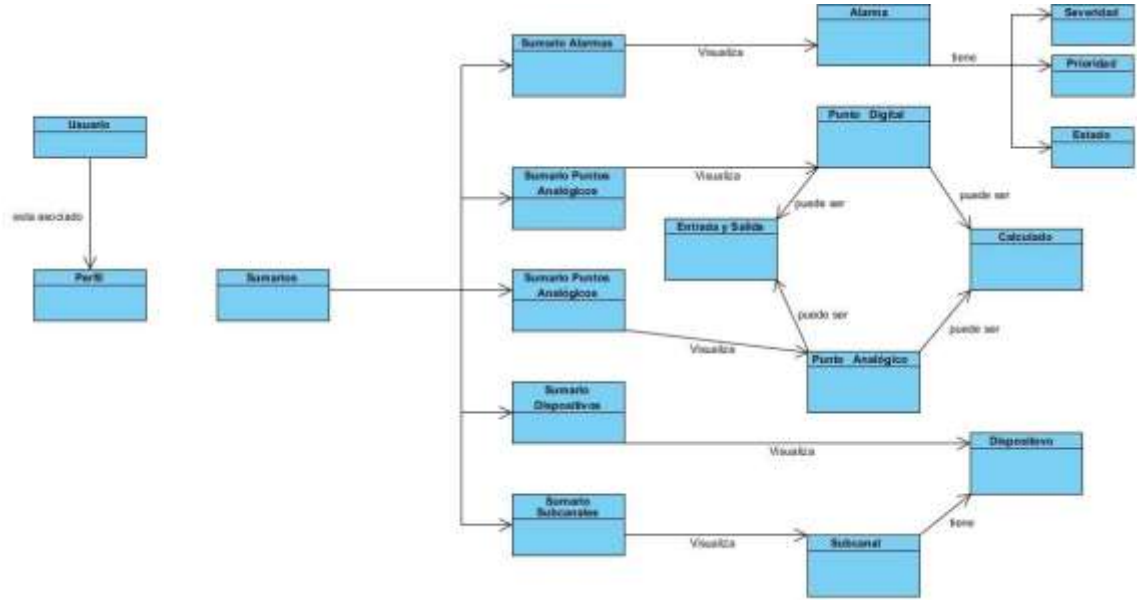


Figura 2.1. Modelo de Dominio.

Seguidamente se detallan cada uno de los elementos que lo componen:

Usuario: El usuario para la interacción con la aplicación cuenta con un nombre de usuario, una contraseña y uno o varios perfiles.

Perfil: Es la asignación que se tiene un usuario dependiendo a sus niveles de privilegios en el sistema.

Sumario Alarmas: Visualiza el listado de las alarmas existentes en el sistema.

Alarma: Las alarmas están organizadas teniendo en cuenta su severidad, prioridad y estado.

Sumario Puntos Analógicos: Visualiza el listado de los puntos analógicos existentes en el sistema.

Punto Analógico: Cada punto analógico se puede comportar como de entrada y salida o calculado.

Sumario Puntos Digitales: Visualiza el listado de los puntos digitales existentes en el sistema.

Punto Digital: Cada punto digital se puede comportar como de entrada y salida o calculado.

Sumario Dispositivos: Visualiza el listado de los dispositivos fisicos que se encuentran desplegados en el campo.

Sumario Subcanales: Visualiza el listado de subcanales existente en el sistema.

Subcanal: Es el protocolo de comunicación de los dispositivos fisicos que se encuentran desplegados en el campo.

2.3 Requerimientos del sistema

Para el desarrollo de la aplicación es necesario descubrir qué es lo que debe o no hacer esta, en otras palabras es preciso efectuar una captura de los requisitos que debe cumplir. “Los requisitos no son más que las condiciones o capacidades que tienen que ser alcanzadas por un sistema para satisfacer las necesidades del cliente. Los mismos se clasifican en Requisitos funcionales y Requisitos no funcionales”. (JACOBSON, 2000)

2.3.1 Requisitos funcionales del sistema

Un requisito funcional define una función del sistema de software o sus componentes. Una función es descrita como un conjunto de entradas, comportamientos y salidas. Los requisitos funcionales pueden ser: cálculos, detalles técnicos, manipulación de datos y otras funcionalidades específicas que se supone que un sistema debe cumplir. Los mismos establecen los requerimientos de comportamiento del sistema para cada requerimiento funcional que se muestran en la historia de usuario. (Pressman, 2002)

Los requerimientos que debe cumplir la aplicación a desarrollar se relacionan a continuación**.**

RF1: Iniciar la aplicación.
RF2: Autenticar usuario.
RF3: Mostrar perfiles de usuario.
RF4: Cerrar sesión.
RF5: Visualizar sumario de alarmas.
RF6: Visualizar sumario de puntos analógicos.
RF7: Visualizar sumario de dispositivos.
RF8: Visualizar sumario de subcanales.
RF9: Realizar paginado de sumarios.
RF10: Notificación sonora de la existencia de alarmas.
RF11: Visualizar cada vista como máximo una sola vez.

2.3.2 Requerimientos no funcionales del sistema

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. (Pressman, 2002)

Usabilidad

RNF1: El sistema debe poder ser usado por cualquier persona que tenga conocimientos básicos de computación.
RNF2: Se debe lograr un producto altamente configurable y extensible, de manera que sea posible incorporar a éste nuevas funcionalidades, sin que se afecte el mismo.

Confiabilidad
RNF3: El sistema debe ser capaz de manejar los errores y recuperarse.

Interfaz de Usuario
RNF4: Forma de interacción de la interfaz: Basada en *activities*.
RNF5: El sistema debe contar con la política marcaria de interfaz única de la institución.
RNF6: El texto del sistema debe ser en idioma español y tener una apariencia uniforme.

Software
RNF7: Implementación con tecnologías libres.
RNF8: Sistema Operativo Android versión 4.2.2.

Hardware
RNF9: Como requisitos mínimos de hardware es necesario contar con memoria RAM 512MB y un Procesador Dual-Core 1GHz.

2.4 Historias de Usuarios

La historia de usuario (HU) es lo suficientemente comprensible y delimitada para que los desarrolladores puedan implementarla en unas semanas. (Fontanarossa, 2006) Los desarrolladores dialogarán de forma directa con los clientes cuando llegue la hora de la implementación, para así obtener los detalles necesarios.
A continuación se presenta las historias de usuario consideradas más importantes, las demás se encuentran en el Anexo 7.1:

Historia de usuario	
Numero: 2	**Nombre:** Mostrar Perfiles de usuario
Usuario: Usuario	
Prioridad de negocio: Alta	**Riesgo de desarrollo:** Alta
Puntos estimados:1	**Iteración asignada:**1
Programador responsable: Alexis Gongora Smith	
Descripción: El sistema debe poseer una interfaz donde el usuario introduce su nombre de usuario y contraseña, para solicitar los perfiles asociados a este.	

Observaciones:
Interfaz: 

Tabla 2.1 Historia de usuario #2

Historia de usuario	
Numero: 5	**Nombre:** Visualizar sumario de alarmas
Usuario: Usuario	
Prioridad de negocio: Alta	**Riesgo de desarrollo:** Alta
Puntos estimados:1	**Iteración asignada:**3
Programador responsable: Alexis Gongora Smith	
Descripción: El sistema debe permitir al usuario visualizar el estado de las alarmas, ordenas por criticidad del sistema.	
Observaciones:	

Interfaz:

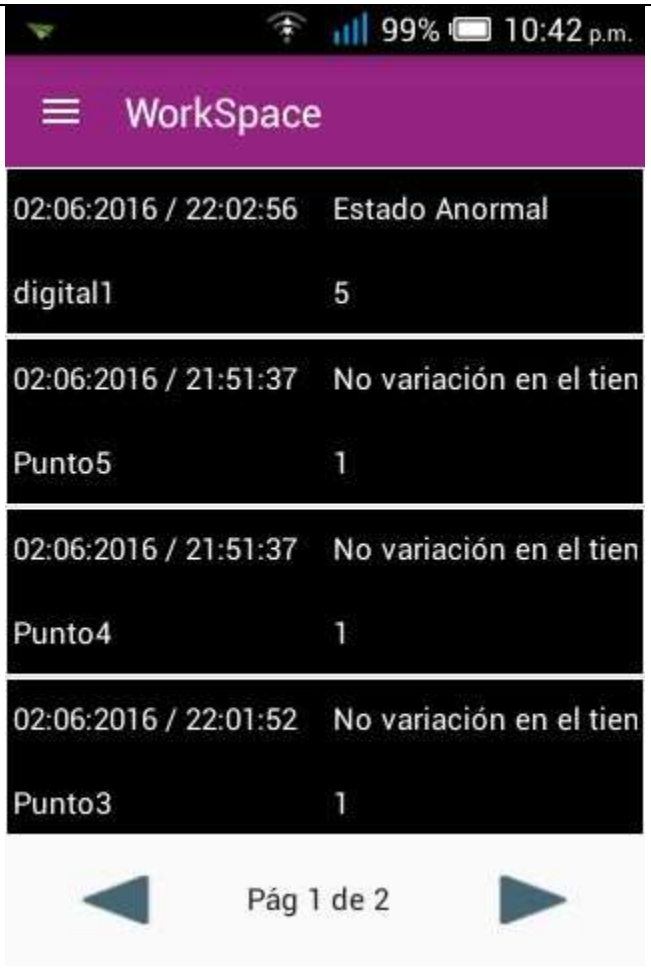


Tabla 2.2 Historia de usuario #5

La descripción de las demás historias de usuario se encuentra en el anexo.

2.5 Patrones de Arquitectura

La arquitectura de la aplicación está basada en los bloques de construcción principales que no son más que los componentes que se utilizan en el desarrollo de una aplicación Android. Estos son elementos conceptuales que deben ser puestos juntos para crear algo más grande. Los bloques de construcción principales son los siguientes:

Activities: Una *activity* es usualmente una simple vista que el usuario ve en el dispositivo. Una aplicación generalmente cuenta con múltiples activities y el usuario puede navegar entre estas. (Learning, 2013)

Intents: Los *intents* son mensajes que se envían entre los bloques de construcción principales. Ellos provocan que una *activity* se muestre, que un servicio se inicie o se detenga, o son simples mensajes. Los *intents* son asíncronos lo cual significa que el código que los envía no tiene que esperar a que se ejecute el código que lo recibe. (Learning, 2013)

Services: Los servicios se ejecutan de forma paralela y no tienen componentes de interfaz de usuario, pueden ejecutar las mismas acciones que las *activities*. (Learning, 2013)

Content Providers: Los proveedores de contenido son interfaces para compartir datos entre aplicaciones como por ejemplo las listas de contactos, preferencias, entre otros. (Learning, 2013)

Broadcast Receivers: Los *broadcast receivers* son la implementación de Android del mecanismo publicación suscripción o más precisamente el patrón observador. El recibidor no es más que un código que se activa cuando un evento ocurre en lo que él está suscrito. (Learning, 2013)

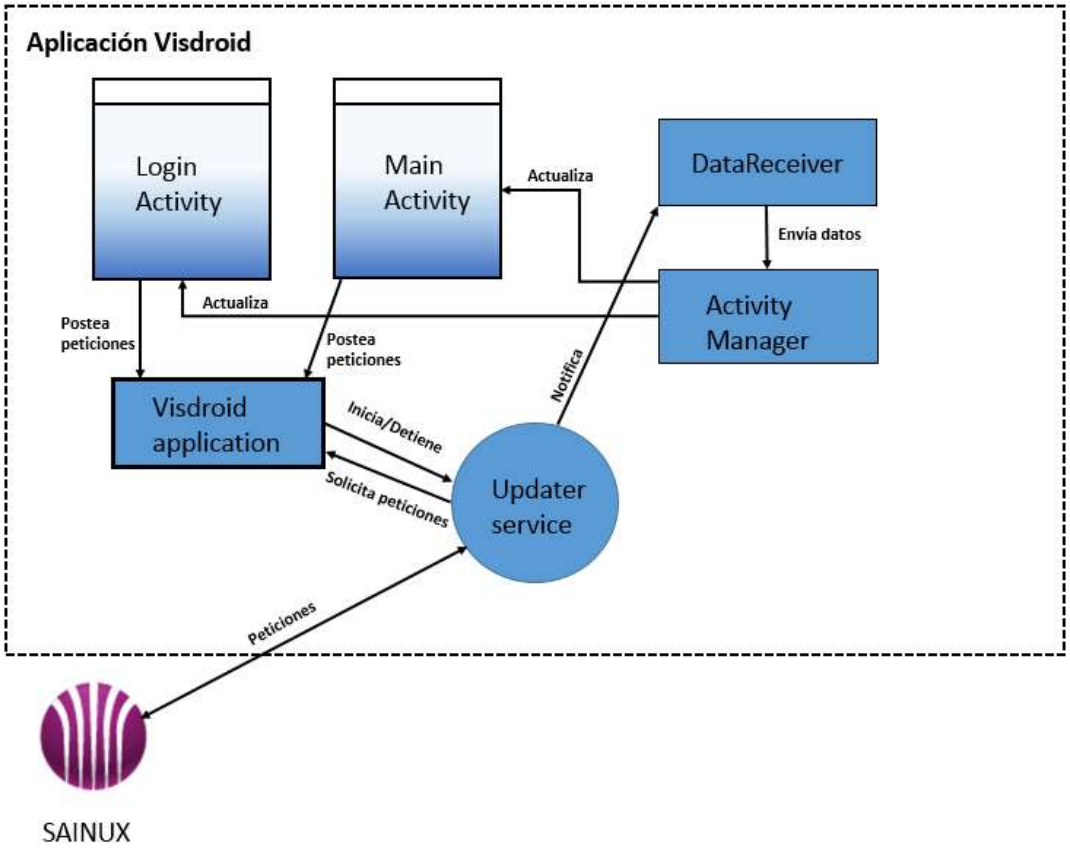


Figura 2.2. Vista de alto nivel de la arquitectura de la aplicación Visdroid.

La arquitectura de la aplicación se detalla a continuación:

La aplicación para la interacción con el usuario cuenta con dos *activity* o vistas. Tanto en la vista *Login Activity* como en la *Main Activity* es donde se realizan la peticiones, las mismas se ponen en cola en la clase principal de la aplicación esperando que *UpdateService* (el servicio se despierta cada un segundo, es decir solicita las peticiones que están en la clase principal cada un segundo) solicite las peticiones, las cuales son enviadas al servidor para ser procesadas; devuelta la respuesta al servicio, este notifica a la clase *DataReceiver* que es la encargada de recibir los datos que provienen de

la capa de servicios web y los envía a la clase *ActivityManager* la cual es la encargada de administrar las *activity* y la misma actualiza la vista donde se haya realizado la petición.

2.6 Diagrama de Clases de Diseño

Un diagrama de clases de diseño (DCD) representa las especificaciones de las clases e interfaces software en una aplicación. Entre la información general se encuentra:

- ✓ Clases, asociaciones y atributos.
- ✓ Interfaces, con sus operaciones y constantes.
- ✓ Métodos.
- ✓ Información acerca del tipo de los atributos.
- ✓ Navegabilidad.
- ✓ Dependencias.

Las clases de diseño de los DCD muestran las definiciones de las clases software en lugar de los conceptos del mundo real. (Patrones, 2003)

En la Figura 2.4, se presenta el Diagrama de Clase del Diseño correspondientes la aplicación.

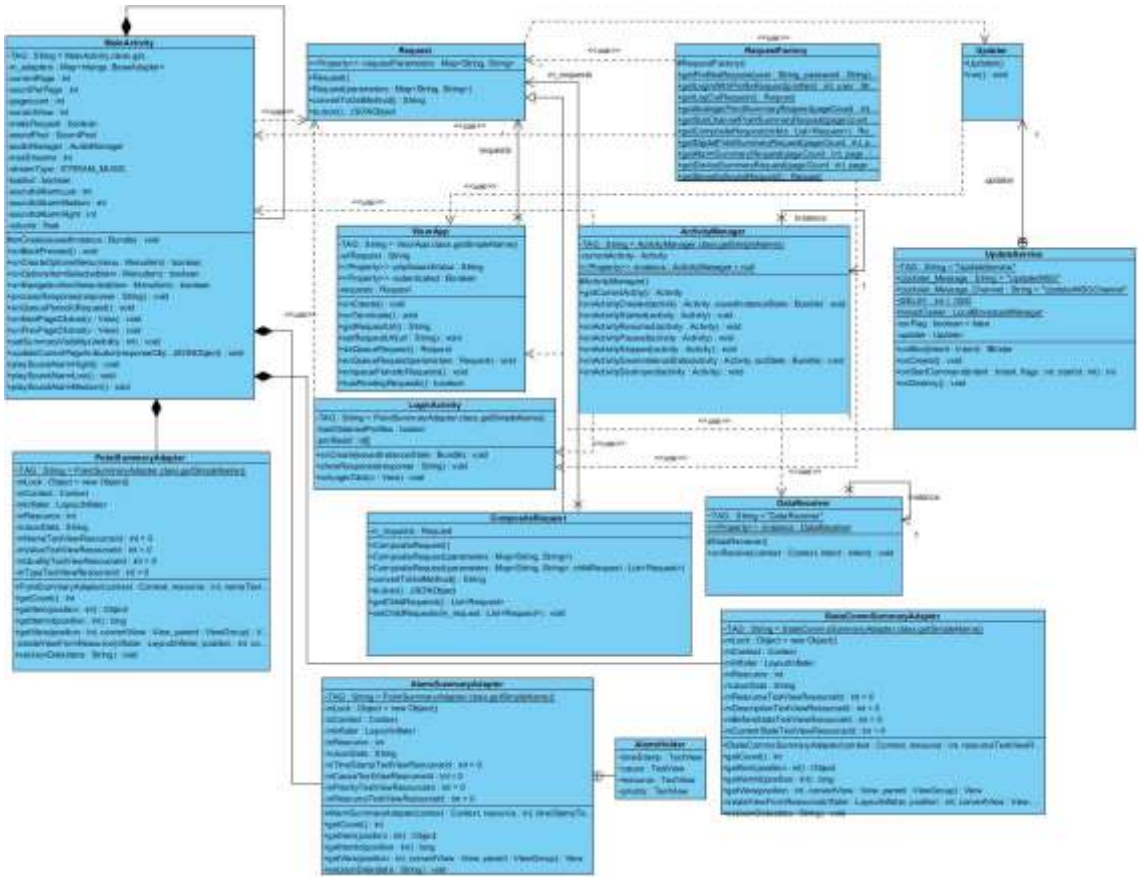


Figura 2.3. Diagrama de Clase de Diseño.

LoginActivity: Es la vista encargada de la autentificación.

MainActivity: Es la vista principal de la aplicación.

VisorApp: Es el controlador principal de la aplicación.

ActivityManager: Es el manejador de *activity* el cual siempre conoce a la *activity* que está activa en la aplicación.

Request: Representa una petición y sus parámetros a la capa de servicios web.

RequestFactory: Es la clase encargada de construir las peticiones.

DataReceiver: Es la clase encargada de recibir los datos que provienen de la capa de servicios web.

UpdateService: Es el servicio encargado de ejecutar las peticiones en la capa de servicios web.

PointSummaryAdapter: Es la clase encargada de adaptar los datos a la vista de sumario de puntos analógicos y digitales.

AlarmSummaryAdapter: Es la clase encargada de adaptar los datos a la vista de sumario de alarmas.

StateCommSummaryAdapter: Es la clase encargada de adaptar los datos a la vista de sumario de dispositivos y subcanales.

Updater: Es el hilo de ejecución se activa en el *updateservice* para el envío de peticiones a la capa de servicios web.

CompositeRequest: Esta clase permite el envio de múltiples peticiones al servidor dentro de una sola, evitando así la sobrecarga de la conexión http con el servidor.

AlarmHolder: Es una clase utilitaria que se emplea para guardar referencia de los campos de las vistas para aumentar el rendimiento del *listview* de Android.

2.7 Patrones de Diseño

Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software. En otras palabras, brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares. Se deben tener presente en un patrón los elementos siguientes: su nombre, el problema (cuando aplicar un patrón), la solución (descripción abstracta del problema) y las consecuencias (costos y beneficios). (MADRID, 2012)

Los patrones de diseño se agrupan en dos tipos principales los GOF (*Gang of Four*) y los GRASP (*General Responsibility Assignment* Software *Patterns*). Los patrones GOF son un conjunto de posibles soluciones a problemas que suelen ser comunes cuando se desarrollan software. Dentro de los patrones GOF existen tres tipos: los creacionales (que abordan problemas de creación de objetos), los estructurales (que abordan problemas sobre relaciones entre entidades) y los de comportamiento (que abordan problemas de comunicación entre entidades). (JACOBSON, 2000)

Los patrones GRASP son usados para asignar responsabilidades a los objetos o clases en el diseño a partir de los diagramas de interacción realizados en el Modelo de Análisis. (Patrones, 2003)

Los patrones GOF y GRASP utilizados en el desarrollo de la solución propuesta se describen a continuación:

2.7.1 Patrones GOF

Observador/Observer

Patrón de tipo comportamiento, a nivel de objetos; que tiene como propósito un efecto lateral muy frecuente en aquellos sistemas que se fragmentan en un conjunto de clases que cooperan, es la necesidad de mantener la consistencia entre los distintos objetos interrelacionados. Para no recurrir las soluciones fuertemente acopladas (que reducen la posibilidad de reutilización), este patrón define una dependencia “uno-a muchos” entre objetos, para que, cuando uno de ellos cambie su estado, todos los que dependan de él sean avisados y puedan actualizarse convenientemente. (MADRID, 2012)

Un ejemplo del uso de este patrón se puede observar a continuación:

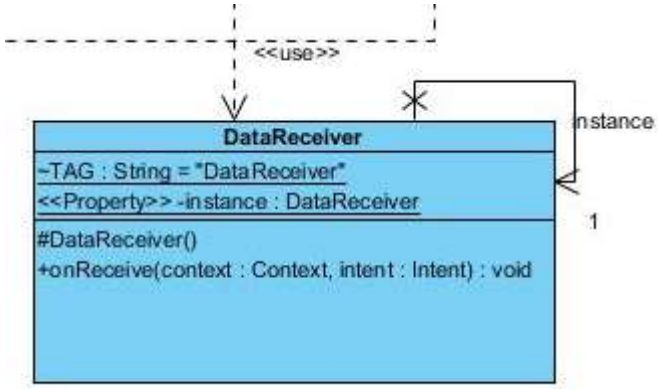


Figura 2. 4 Ejemplo del patrón Observador.

Instancia Única: Asegura que una clase tenga solamente una única instancia y provee un punto de acceso global a ella.

Un ejemplo del uso de este patrón se puede observar a continuación:

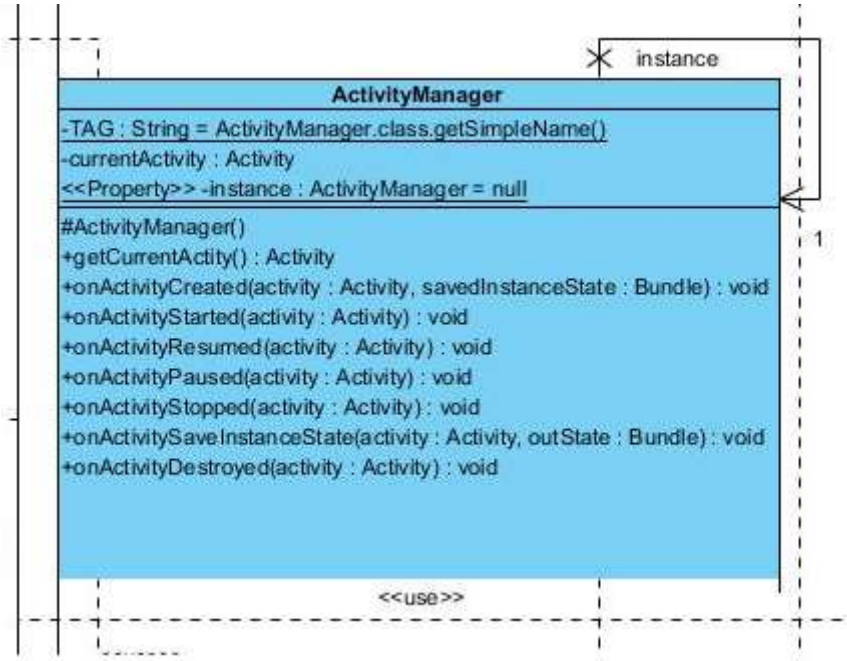


Figura 2. 5 Ejemplo del patrón Instancia Única.

Fábrica Abstracta: Se utiliza para proporcionar una interfaz común para crear una serie de objetos (productos) bajo una u otra arquitectura o framework, sin que los clientes de dichos productos tengan conocimiento de la arquitectura elegida para implementar cada familia de productos.

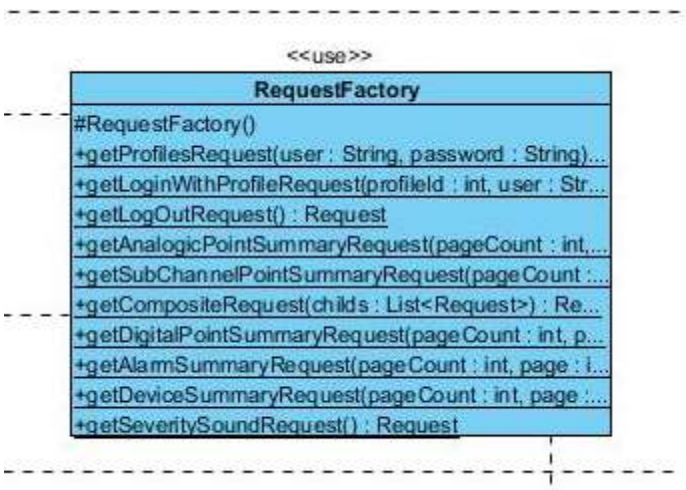


Figura 2. 6 Ejemplo del patrón Fabrica Abstracta.

2.7.2 Patrones GRASP

Experto

Patrón utilizado para asignar las responsabilidades al experto en la información, es decir a la clase que cuenta con la información necesaria para cumplir la responsabilidad. (Patrones, 2003) Se evidencia en la clase *LoginActivity* la cual es la encargada de realizar todas las operaciones de autenticación en el sistema.

Bajo acoplamiento

Proporciona que entre las clases exista la menor cantidad de dependencias posibles. De tal forma que en caso de producirse una modificación en alguna de ellas, tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases. Este patrón es respetado durante la implementación de la aplicación, pues se ha incluido la menor cantidad de dependencias entre las clases. (Patrones, 2003)

Alta cohesión

Consiste en que cada elemento de un diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable. Una clase con baja cohesión hace muchas cosas no relacionadas o hace demasiado trabajo. En la aplicación que se desarrolla, existe colaboración entre las clases para realizar tareas con cierta dificultad. Por ejemplo, si para realizar una funcionalidad se necesita llevar a

cabo varias labores, estas son asignadas a las clases responsables de realizarlas. (Patrones, 2003)

Se evidencia el uso del patrón **Bajo Acoplamiento** y **Alta Cohesión**, debido a que las clases fueron diseñadas de tal forma que existiera la menor relación posible entre ellas, a la vez logrando que la información almacenada en cada clase fuera coherente. Con este diseño se logra que de existir una modificación tenga la mínima repercusión posible en el resto de las clases, se garantiza la reutilización de código y disminuye la dependencia entre clases.

2.8 Diagrama de Paquete

Los paquetes constituyen un medio para organizar el modelo de diseño en piezas manejables. (JACOBSON, 2000). A continuación se presenta el diagrama de paquete de la solución en el cual se organizan en estructura los componentes que la conforman.

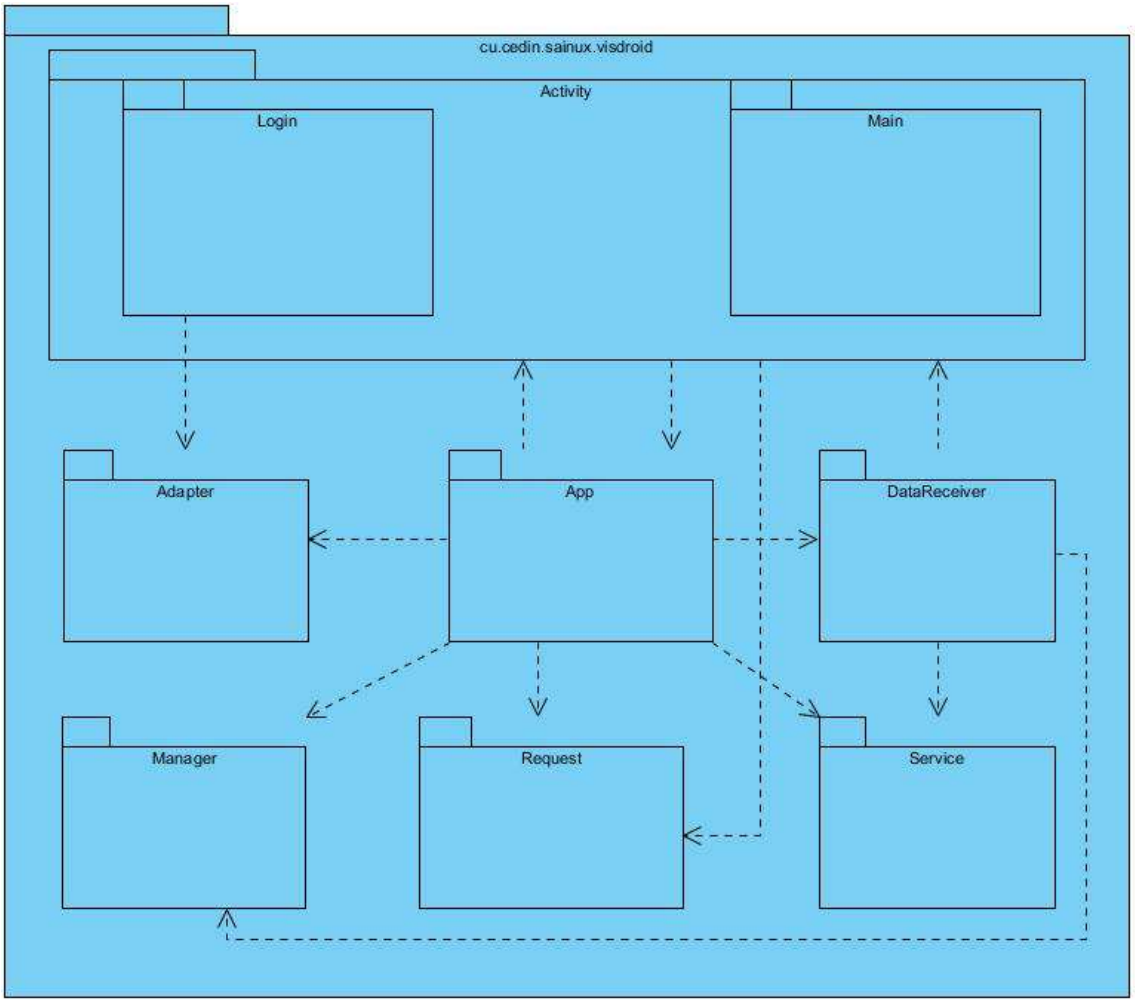


Figura 2.7. Diagrama de Paquete.

Seguidamente se detallan cada uno de los elementos que componen el diagrama:

cu.cedin.sainux.visdroid: Es el paquete principal de la aplicación en el cual engloban el conjunto de paquetes y componentes de las vistas.

Activity: Es el paquete que engloba los paquetes *Login* y *Main.*

***Login*:** Es el paquete donde se encuentra la *activity* encargada de gestionar los procesos de autentificación en la aplicación.

Main: Es el paquete donde se encuentra la *activity* principal de la aplicación, la cual es la encargada de gestionar todo los procesos referentes al sistema después que un usuario se autentica.

Adapter: Es el paquete donde se encuentran las clases encargadas de adaptar los datos procedentes de la capa de servicios web del SCADA SAINUX en formato JSON.

App: Es el paquete donde se encuentra la clase principal de la aplicación.

DataReceiver: Es el paquete donde se encuentra las clases encargadas de recibir los datos procedentes de la capa de servicios web que son proporcionados por el componente servicio y entregarlos a las vistas del sistema.

Manager: Es el paquete donde se encuentra las clases encargadas de la administración de las vistas del Sistema.

Request: Es el paquete donde se encuentran las clases encargadas de la creación y conversión al formato adecuado de las peticiones que se realizan a la capa de servicios *web* del SCADA SAINUX.

Service: Este el paquete donde se encuentra las clases encargadas de la ejecución de las peticiones que realiza la aplicación a la capa de servicios *web* del SCADA SAINUX.

2.9 Conclusiones del Capitulo

En este capítulo se inicia el desarrollo de la propuesta de solución que se desea implementar. Se utilizaron los diferentes artefactos que establece la metodología de desarrollo de software empleada. Además se realizó un levantamiento los requisitos funcionales y no funcionales que debe tener la aplicación para el cumplimiento del objetivo general, por lo que se crearon las historias de usuario referentes a los mismos. Finalmente se diseñó varios diagramas de clases del diseño que constituyen la entrada fundamental para las actividades de implementación y prueba.

3 Capítulo III. Implementación y prueba

Introducción

En este capítulo se desarrollan los flujos de trabajo implementación y prueba. En la implementación se empezó con el resultado del diseño y se crea el sistema en términos de componentes, es decir, ficheros de código fuente, ficheros de código binario, ejecutables y similares.

El flujo de trabajo de análisis y diseño se propone crear un plano del modelo de implementación, por lo que sus últimas actividades están vinculadas a la creación del Modelo de Despliegue. El flujo de trabajo de implementación describe cómo los elementos del Modelo del Diseño se implementan en términos de componentes y cómo estos se organizan de acuerdo a los nodos específicos en el Modelo de Despliegue.

3.1 Modelo de implementación

La disciplina de implementación se empieza con el resultado del diseño y se implementa el sistema en términos de componentes, es decir, ficheros de código fuente, scripts, ficheros de código binario, ejecutables y similares. (Jacobson, 2005)

3.1.1 Diagrama de Componentes

Uno de los usos principales del diagrama de componentes es que puede ser utilizado para ver que componentes pueden compartirse entre sistemas o entre diferentes partes de un sistema.

A continuación le presentamos el diagrama de componentes propuesto para nuestra aplicación donde se hará uso de un glosario de términos para identificar todos los componentes que se utilizarán en mismo:

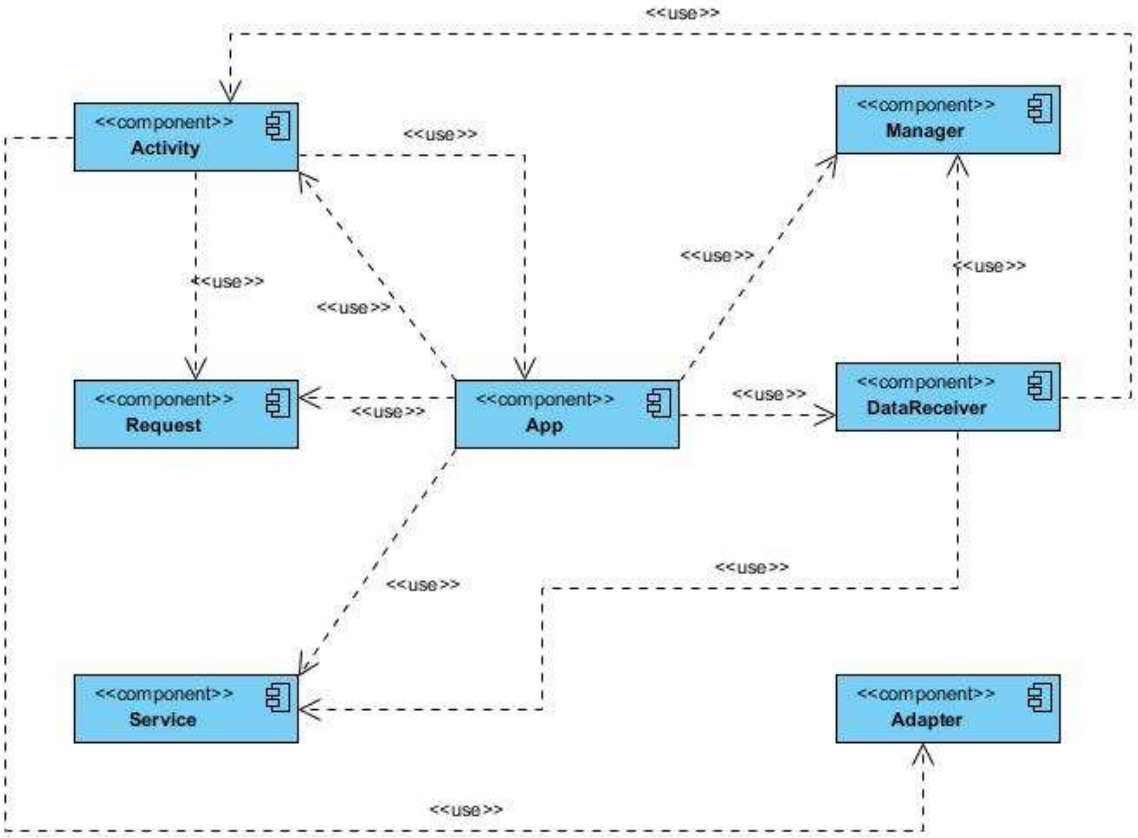


Figura 3. 1Diagrama de Componentes

Seguidamente se detallan cada uno de los elementos que lo componen:

Activity: El componente *Activity* se encarga de la definición de las vistas de la aplicación.

App: Es el componente principal de la aplicación.

Manager: Es el componente encargado de la administración de las vistas del Sistema, se utiliza para conocer cuál es la vista que esta active en la aplicación en todo momento.

Request: Es el componente encargado de la creación y conversión al formato adecuado de las peticiones que se realizan a la *interface* de servicios *web* del SCADA SAINUX.

Service: Este componente es el encargado de la ejecución de las peticiones que realiza la aplicación a la *interface* de servicios *web* del SCADA SAINUX mediante el protocolo http.

Adapter: Es el componente encargado de adaptar los datos procedentes de la interface de servicios web del SCADA SAINUX en formato JSON para su visualización utilizando las vistas que provee el Android SDK.

DataReceiver: Es el componente encargado de recibir los datos procedentes de la interfaz de servicios web que son proporcionados por el componente servicio y entregarlos a las vistas del sistema.

3.2 Modelo de Despliegue

Los Diagramas de Despliegue muestran la disposición física de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación. Un nodo es un recurso de ejecución tal como un computador, un dispositivo o memoria.
En la Figura 3.2 se representa el Modelo de Despliegue del sistema.

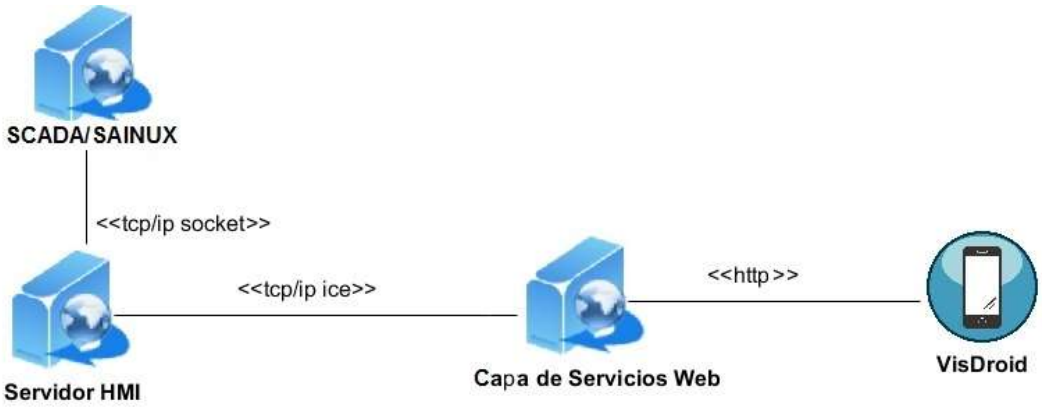


Figura 3.2 Diagrama de despliegue

Seguidamente se detallan cada uno de los elementos que lo componen:

VisDroid: Aplicación que para su ejecución necesita de un *Smartphone* o *Tablet* con Sistema Operativo Android.

Capa de Servicios Web: Software encargado del intercambio de información entre los clientes y el servidor se realiza utilizando la notación JSON.

Servidor HMI: Software de HMI para brindar información y atender las solicitudes que se realizan desde clientes HMI.

SCADA/SAINUX: Software para ordenadores que permite controlar y supervisar procesos industriales a distancia.

Ejemplo de cómo se observar el despliegue de nuestra aplicación:

La aplicación se encuentra instalada en un Smartphone o *Tablet* con sistema operativo Android versión 4.2.2 el cual se conecta a una red wi-fi donde también a ella pertenece

un servidor Scada con sus dependencias las cuales son Servidor HMI y capa de servicios web. Al ejecutar la aplicación mediante la red utilizando el protocolo http se comunica con la capa de servicios web (envía peticiones) esta se comunica a través del protocolo tcp/ip y la biblioteca ice con el servidor HMI y este con el SCADA/SAINUX a través del mismo protocolo utilizando socket. SAINUX conforma la respuesta y utiliza el mismo canal para enviarla.

3.3 Estándares de codificación

Los estándares de codificación establecen un conjunto de reglas que los desarrolladores deben seguir para escribir el código fuente de un software. Esto tiene como objetivo que dicho código sea entendible por cualquier desarrollador del grupo de trabajo, no solo por el que lo creó; garantizando un mantenimiento del sistema más rápido y eficiente, ya sea creando nuevas funcionalidades o modificando las ya existentes.

Para el desarrollo del sistema propuesto se utilizan varios estándares de codificación, tales como:

- ✓ Los atributos de las clases deben comenzar con la letra m seguido de guión bajo y a continuación el nombre del atributo, si existen atributos compuestos la segunda palabra debe comenzar con mayúscula.

  Ejemplo: m_isColorFlange

- ✓ Las funciones no deben exceder de las 200 líneas de código.
- ✓ Las secciones *public*, *protected* y *private* serán declaradas en este orden.
- ✓ El código será escrito en inglés.
- ✓ Las variables y funciones comienzan con letra minúscula, cada palabra consecutiva en el nombre comienza con letra mayúscula.
- ✓ Los valores de los numerativos deben ser con letras mayúsculas.

3.4 Validación y prueba

Las pruebas y la validación son procesos de revisión que verifica que el sistema de software producido cumple con las especificaciones y que logra su cometido. Es normalmente una parte del proceso de pruebas de software de un proyecto, que también utiliza técnicas tales como evaluaciones, inspecciones y tutoriales. La validación es el proceso de comprobar que lo que se ha especificado es lo que el usuario realmente quería.

3.4.1 Prueba de aceptación

Las pruebas de aceptación verifican que el sistema que recibe funciona y lo hace de acuerdo con las especificaciones. Las pruebas de aceptación se realizan de acuerdo con protocolos específicos del producto en nuestra fábrica o en terreno. Al participar en un proceso de prueba de aceptación, puede obtener un conocimiento más profundo acerca de cómo funciona el equipo. (JACOBSON, 2000)

El cliente al cual validó las pruebas de aceptación fue la línea de desarrollo del HMI

A continuación se relaciona un caso de prueba aplicada a cada historia de usuario, los demás casos se encuentran en el anexo 2:

Caso de Prueba Aceptación	
Número: 2	**Historia de usuario:** 2
Nombre: Mostrar Perfiles de usuario.	
Descripción: El sistema debe poseer una interfaz donde el usuario introduce su nombre de usuario y contraseña, para solicitar los perfiles asociados a este.	
Condiciones de Ejecución: El usuario de haberse autenticado rellenando los campos de nombre de usuario, contraseña y dirección del servidor.	
Pasos de Ejecución:	
Resultado esperado: Después de haberse autenticado llenando los campos de nombre de usuario, contraseña y dirección del servidor surge una lista con los perfiles permitidos para el usuario	

Tabla 3.1 Prueba de aceptación #2

Caso de Prueba Aceptación	
Número: 6	**Historia de usuario:** 6
Nombre: Visualizar sumario de puntos analógicos.	
Descripción: El sistema debe permitir al usuario visualizar el estado de los puntos analógicos.	
Condiciones de Ejecución: El usuario debe de estar autenticado.	
Pasos de Ejecución: Al desplazar el menú que se encuentra oculta en el lado izquierdo de la pantalla seleccionamos la opción Sumario Ptos Analógicos, donde se representa un listado de los puntos analógicos existentes donde cada un segundo el sistema envía la petición al servidor actualizándose la vista.	

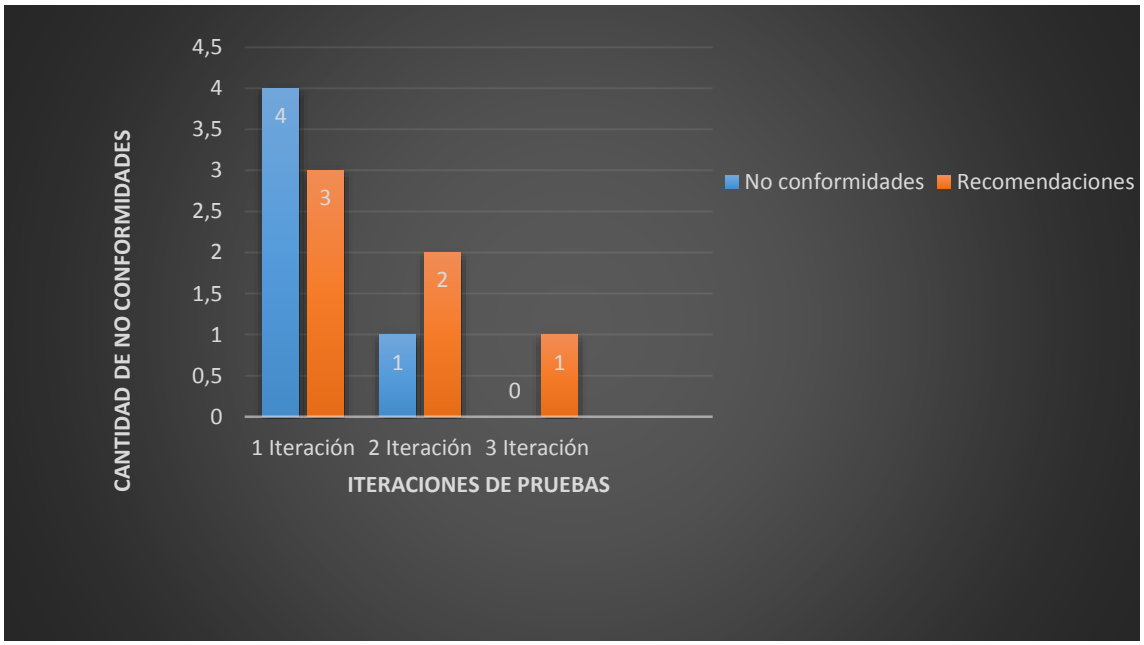
Resultado esperado: El sistema debe permitir al usuario visualizar los puntos analógicos existentes.

Tabla 3.2 Prueba de aceptación #6

Al aplicarles las pruebas de aceptación arrojaron los siguientes resultados:

Sistema	HU	Iteración	No conformidad	Cerradas	No procede	Recomendaciones
Visdroid	10	1ra	4	4	0	3
		2da	1	1	0	2
		3ra	0	0	0	1

A partir de las historias de usuario se diseñaron 11 casos de pruebas con el objetivo de obtener la aceptación del cliente de la solución implementada, al ejecutar las pruebas, las mismas arrojan como resultado que para una 1ra iteración se detectaron 4 no conformidades donde se resuelven las cuatro, para una 2da iteración surgen 2 no conformidades las cuales son resueltas en la misma iteración y en una 3ra iteración no se detectaron no conformidades.



3.4.2 Pruebas de Rendimiento

Pruebas de carga, pruebas de estrés y pruebas de capacidad son diferentes tipos de pruebas de rendimiento. Efectivamente, se trata de valoración de cualquier experiencia

de usuario al realizar guiones contra la aplicación de destino. Esto permite evaluar capacidad de la máquina y la infraestructura. Las pruebas de rendimiento, de carga, de estrés y de capacidad se basan en requerimientos de negocio. Es un enfoque profesional para evaluar el rendimiento de una aplicación. Se requiere simulación de guiones de carga reales que se ejecuten contra la aplicación o sitio web de destino. (AgileLoad, 2016)

Para la realización de la prueba de rendimiento se hizo un análisis del uso del procesador, consumo de memoria y batería. Las características de hardware y software del dispositivo móvil donde se ejecutará el sistema y los resultados que arrojaron los diferentes factores a evaluar al iniciar la aplicación y después de ejecutarse 24 horas consecutivas se exponen a continuación:

Prestaciones de Hardware:

- ✓ Memoria RAM: 512 Mb
- ✓ Procesador: Dual-Core 1GHz

Sistema Operativo:

- ✓ Android versión 4.2.2.

Funcionalidad a probar:

- ✓ Visualización de alarmas.

Sistema Operativo	Consumo de Memoria (Mb)	Uso del Procesador (%)	Uso de la batería (%)
Android versión 4.2.2.	**5**	**1**	**100**

Tabla 3.3 Consumo de Memoria, Batería y uso del Procesador al iniciar la aplicación

Sistema Operativo	Consumo de Memoria (Mb)	Uso del Procesador (%)	Uso de la batería (%)
Android versión 4.2.2.	**20**	**6**	**81**

.Tabla 3. 4 Consumo de Memoria, Batería y uso del Procesador después de 24 h de ejecución

De acuerdo con los resultados obtenidos, las pruebas se realizaron satisfactoriamente alcanzando valores positivos en concordancia con el tiempo de actualización que debe poseer el visualizador para dispositivos móviles con Sistema Operativo Android en la representación de los procesos pertenecientes al SCADA SAINUX. Llegando a la

conclusión de que utilizando el mismo hardware la aplicación estaría en óptimo y continuo rendimiento durante aproximadamente 5 días.

3.4.3 Resultado de las pruebas

Al culminar las pruebas de aceptación y rendimiento las cuales estaban destinadas al control del nivel de aceptación del cliente y el comportamiento de la aplicación en los aspectos del consumo de la memoria RAM, procesador y batería se pudo tener en cuenta que la aplicación trabaja bajo los parámetros adecuados ya que fue comparada con otra aplicación similar, llevándonos a la conclusión que es un producto el cual puede ser utilizado para la supervisión y control de los procesos industriales.

3.5 Conclusiones Parciales

Durante el desarrollo del capítulo se estableció el modelo de implementación el cual está conformando por el diagrama de componentes y modelo de despliegue de la aplicación. Además se abordó los estándares de codificación utilizados en la implementación. Finalmente se realizaron las pruebas de aceptación y rendimiento en las cuales comprobaron el correcto funcionamiento del sistema, dando cumplimiento de los requisitos definidos en el Capítulo 2.

4 Conclusiones

Partiendo de la investigación realizada en el presente trabajo, donde se evidencia la necesidad de que el sistema SCADA SAINUX cuente con una aplicación para la visualización de datos en tiempo real en dispositivos con pantallas de dimensiones menores a las 12 pulgadas se arriban a las siguientes conclusiones:

- ✓ Se desarrolló una aplicación de visualización de datos en tiempo real en dispositivos móviles utilizando las tecnologías, herramientas y metodología anteriormente expuestas, contando con una documentación sobre las funcionalidades que brinda la aplicación.
- ✓ El uso de un estilo de programación orientado a objeto permitió que los subsistemas desarrollados como parte del presente trabajo puedan ser reutilizados en la construcción de aplicaciones con características similares.
- ✓ Se validó la aplicación mediante pruebas de aceptación y rendimiento demostrando la robustez de la solución y la conformidad del cliente con la misma, además se corrigieron algunas deficiencias presentes.
- ✓ La investigación desarrollada permitió realizar un estudio sobre la plataforma de desarrollo, lográndose adquirir una idea general de todas las potencialidades de dicha plataforma y las ventajas que tiene a la hora de supervisar y controlar los procesos industriales.

5 Recomendaciones

Sería significativo destacar que durante el proceso de investigación se detectó una recomendación, las cual se pretenden incluir en posteriores versiones de la herramienta y es: actualizar la aplicación a medida que vayan surgiendo nuevas versiones de Android debido a que funcionalidades pueden quedar obsoletas.

6 Bibliografía

AgileLoad. [En linea] [Citado el 24 de mayo de 2016] http://es.agileload.com/performance-testing/application-performance-testing/what-is-performance-testing.

Alarma, Qué es, Significado y Concepto - Definición de alarma En: Sitio en Internet definicion.de., [Citado el 16 de enero de 2016] Disponible en: http://definicion.de/alarma/#ixzz3wzttJa4B.

ALEGSA, Sitio Web. ALEGSA.com.ar En: Sitio en Internet alegsa.com.ar, [Citado el 20 de enero de 2016] Disponible en: http://www.alegsa.com.ar/Dic/comando.php#sthash.oxYKn3SF.dpuf.

Android, Sitio Web. Academia Android - IDE para Android: IntelliJ IDEA, Android Studio y AIDE, 2015, En: Sitio en Internet academiaandroid.com Disponible en: http://academiaandroid.com/ide-android-intellij-android-studio-aide.

Conran, Aaron. Sencha. [En línea] 3 de mayo de 2012. [Citado el: 28 de abril de 2016]. Disponible en: http://www.sencha.com.

Duenser, Andreas. Mathematics and Computing. [En línea] 23 de Julio de 2012. http://mcs.open.ac.uk/pervasive/pdfs/duenserEdutainment07.pdf.

Fontanarossa, Diego, Garderes, Javier. Gestión de Software. Extreme Programming. 2006.

G.d. Metodologías de desarrollo., 2014, Obtenido de http://www.um.es/docencia/barzana/IAGP/lagp2.html.

González, Software bajo tecnología web para el ingreso, control y respaldo de cabezales de registros de pozos petroleros [2013] / autor. Rossany González Johanna Macías.- Venezuela. Universidad Alonso de Ojeda. Facultad de Ingeniería. Escuela de Computación. http://www.scribd.com.

Hernández, Rolando Alfredo León, Sayda Coello González. El proceso de investigación científica. S.l.: Editorial Universitaria, 2011.C. ISBN978-959-16-1307-3.

IntegraXor. 2013. IntegraXor. [Online] Ecava Sdn Bhd, 2013. Cited: Febrero 29, 2016.] http://www.integraxor.com.

JACOBSON, I. B., GRADY RUMBAUGH, JAMES. El Proceso Unificado de Desarrollo de Software .Editado por: Rumbaugh, I. J. G. B. J. 2000. 4, 27, 28, 112, 107, 127, 168, 177, 208, 210, 283 y 284 p. The Addison-Wesley Object Technology Series.

JACOBSON, Jacobson-Booch-Rumbaugh. 2005. El lenguaje unificado de modelado, manual de referencia.

Java, Por qué los desarrolladores de softwareeligen Java., 2015 En: Sitio en Internet de Java.Disponible en: http://java.sun.com.

JSON, Introducing JSON, 2016, http://www.json.org.

Learning, Libro: Learning Android, 2013, editorial: O' Relly, autor: Marko Gargenta.

Patrones, UML y Patrones. 2ª Edición. Craig Larman. Prentice Hall. 2003.

Letelier, Patricio, Penades, María del Carmen. Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP).2012.

Lévénez, É. Computer Languages History. En: Página personal en Internet de Éric Lévénez. 2007. Disponible en: http://www.levenez.com/lang.

MADRID, U. P. D. Patrones-del-Gang-of-Four.2012. 27, 41, 53 p.

MEASURESOFT, 2009. Disponible en: http://www.measuresoft.com/products/scadapro-server.

Montero, Dagoberto, Barrante, David B, 2013, Quiróz, Jorge M.

Introducción a los Sistemas.

ORACLE, Sitio Web. ORACLE, 2015 En: Sitio en Internet oracle.com. Disponible en: http://www.oracle.com/technetwork/java/archive-139210.html.

Patiño, Yuri VictorMunayev. 2012. XStormClient, visualizador web para el SCADA Guardián del Alba.

Pressman, Roger (2002). Ingeniería del Software, un enfoque práctico, Mc-Graw Hill

PROGEA, I. A. S. Movicon X, 2011. [09/1/2016]. Disponible en: www.movicon-services.de.

Python. Documentacion de Python. En: Sitio en Internet acerca de Python. 2013 Disponible en: http://pyspanishdoc.sourceforge.net.

Rodríguez, Elizabeth De la Cruz Rodríguez, Daliamny Guzmán Hernández. Propuesta de herramienta CASE para los proyectos del Centro de Desarrollo de Informática Industrial. Ciudad de la Habana: Universidad de las Ciencias Informáticas, 2010.

Sanchez, Tamara Rodríguez Sánchez, Metodología de desarrollo para la Actividad productiva de la UCI, versión 1.2.

SANGUINETTI, C. Análisis y Diseño de Sistema, 2010.

SIEMENS, 2016. Disponible en: http://w3.siemens.com/mcms/human- machine-interface/en/visualization-software/scada/Pages/system-overview.aspx.

Sparxsystems. [En línea] [Citado el: 5 de abril de 2016.] http://www.sparxsystems.com.ar/resources/tutorial/uml2_deploymentdiagram.

Stallings, W. (1997). Sistemas Operativos (segunda edición ed.). Madrid: PRENTICE HALL.

Variable, Sitio Web. Que es, Significado y Concepto - Definición de variable En: Sitio en Internet definicion.de., [Citado el 16 de enero de 2016] Disponible en: http://definicion.de/variable/#ixzz3wzo0HBeE.

ZAPATA, CARLOS MARIO ZAPATA, C. D. J. C. Comparación de las características de algunas herramientas de software para pruebas de carga. In Revista Avances en Sistemas e Informática.2011, vol. 8.

7 Anexos

7.1 Anexo 1 Historia de usuario

8 **Historia de usuario**	
Numero: 1	**Nombre:** Iniciar Aplicación
Usuario: Usuario	
Prioridad de negocio: Alta	**Riesgo de desarrollo:** Alta
Puntos estimados:1	**Iteración asignada:**1
Programador responsable: Alexis Gongora Smith	
Descripción: Inicia la *activity autenticar, en la cual se muestran los campos usuario, contraseña, dirección del servidor y la acción entrar.*	
Observaciones:	
Interfaz : 	

Tabla 2.3. Historia de usuario # 1

Historia de usuario

Numero: 3	**Nombre:** Autenticar usuario.
Usuario: Usuario	
Prioridad de negocio: Alta	**Riesgo de desarrollo:** Alta
Puntos estimados:1	**Iteración asignada:**1
Programador responsable: Alexis Gongora Smith	
Descripción: El sistema debe poseer una interfaz de autenticación donde el usuario selecciona el perfil para abrir una sesión de trabajo en el sistema.	
Observaciones:	
Interfaz: 	

Tabla 2.4 Historia de usuario #3

Historia de usuario	
Numero: 4	**Nombre:** Cerrar sesión
Usuario: Usuario	
Prioridad de negocio: Baja	**Riesgo de desarrollo:** Baja
Puntos estimados:0.5	**Iteración asignada:**2
Programador responsable: Alexis Gongora Smith	
Descripción: El sistema debe permite al usuario cerrar su sesión de trabajo en el mismo.	

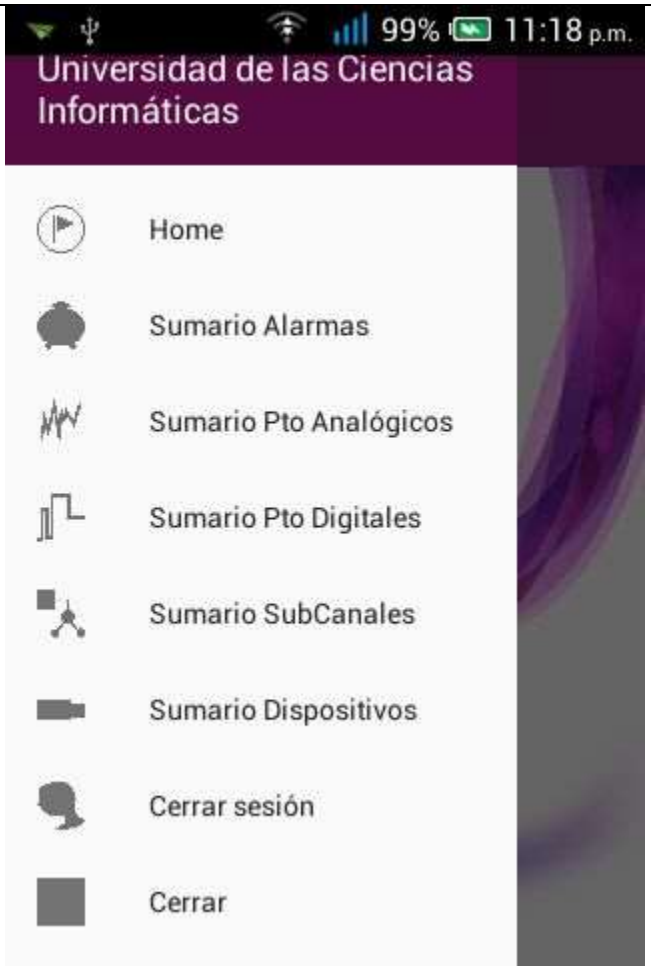
Observaciones:
Interfaz: 

Tabla 2.5 Historia de usuario #4

Historia de usuario	
Numero: 6	**Nombre:** Visualizar sumario de puntos analógicos.
Usuario: Usuario	
Prioridad de negocio: Alta	**Riesgo de desarrollo:** Alta
Puntos estimados:1	**Iteración asignada:**3
Programador responsable: Alexis Gongora Smith	
Descripción: El sistema debe permitir al usuario visualizar el estado de los puntos analógicos.	
Observaciones:	

Interfaz:

Tabla 2.6 Historia de usuario #6

Historia de usuario	
Numero: 7	**Nombre:** Visualizar sumario de puntos digitales.
Usuario: Usuario	
Prioridad de negocio: Alta	**Riesgo de desarrollo:** Alta
Puntos estimados:1	**Iteración asignada:**3
Programador responsable: Alexis Gongora Smith	
Descripción: El sistema debe permitir al usuario visualizar el estado de los puntos digitales.	
Observaciones:	

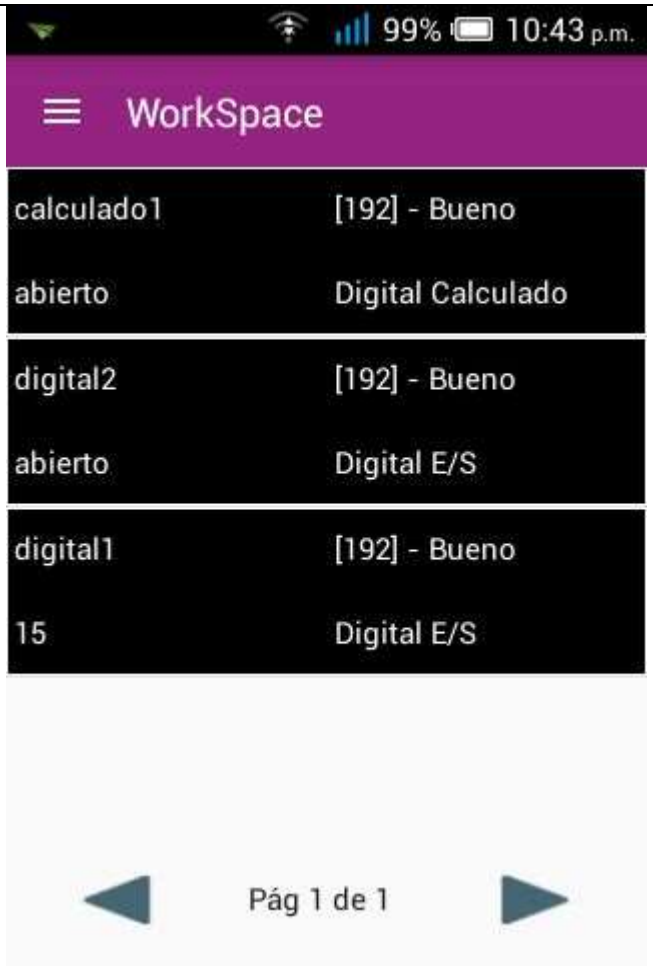
Interfaz:	

Tabla 2.7 Historia de usuario #7

Historia de usuario	
Numero: 8	**Nombre:** Visualizar sumario de dispositivos.
Usuario: Usuario	
Prioridad de negocio: Alta	**Riesgo de desarrollo:** Alta
Puntos estimados:1	**Iteración asignada:**4
Programador responsable: Alexis Gongora Smith	
Descripción: El sistema debe permitir al usuario visualizar el estado de los dispositivos.	
Observaciones:	

Interfaz:

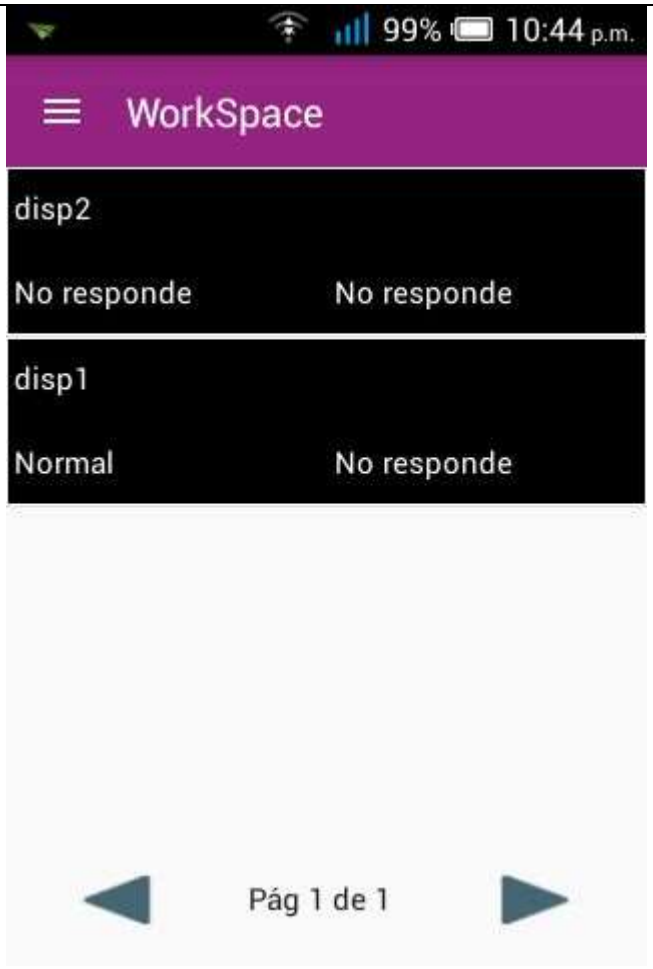


Tabla 2.8 Historia de usuario #8

Historia de usuario	
Numero: 9	**Nombre:** Notificación sonora de la existencia de alarmas.
Usuario: Usuario	
Prioridad de negocio: Baja	**Riesgo de desarrollo:** Baja
Puntos estimados:0.5	**Iteración asignada:**7
Programador responsable: Alexis Gongora Smith	
Descripción: El sistema debe notificar de forma sonora al usuario la existencia de una alarma activa no reconocida en el SCADA SAINUX.	
Observaciones:	
Interfaz:	

Tabla 2.9 Historia de usuario #9

Historia de usuario	
Numero: 10	**Nombre:** Visualizar cada vista de sumario como máximo una sola vez.
Usuario: Usuario	
Prioridad de negocio: Baja	**Riesgo de desarrollo:** Baja
Puntos estimados:0.5	**Iteración asignada:**7
Programador responsable: Alexis Gongora Smith	
Descripción: El sistema debe permitir al usuario visualizar cada vista de sumario como máximo una sola vez.	
Observaciones:	
Interfaz:	

Tabla 2.10 Historia de usuario #10+6

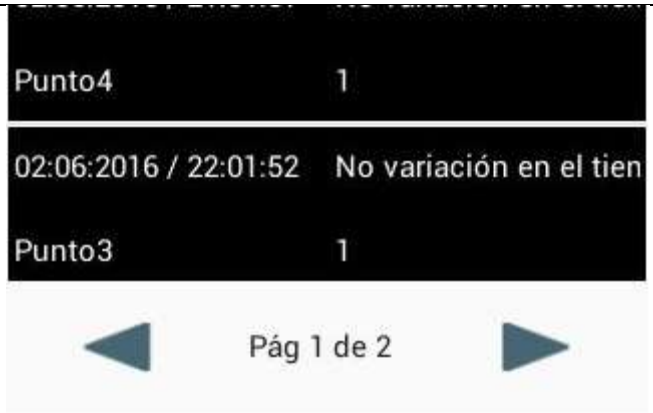
Historia de usuario	
Numero: 11	**Nombre:** Realizar paginado de los sumarios.
Usuario: Usuario	
Prioridad de negocio: Baja	**Riesgo de desarrollo:** Baja
Puntos estimados:0.5	**Iteración asignada:**7
Programador responsable: Alexis Gongora Smith	
Descripción: El sistema debe permitir el paginado de los sumarios, de esta manera se dividen contenidos de gran longitud en varias páginas más cortas.	
Observaciones:	
	
Interfaz:	

Tabla 2.11Historia de usuario #11

Anexos

8.1 Anexo 2 Pruebas de Aceptación

9 Caso de Prueba Aceptación	
Número: 1	**Historia de usuario:** 1
Nombre: Iniciar Aplicación.	
Descripción: Inicia la *activity autenticar, en la cual se muestran los campos usuario, contraseña y la acción entrar.*	
Condiciones de Ejecución: Sistema Operativo Android en su versión 4.2.2.	
Pasos de Ejecución: El usuario ejecuta la aplicación.	
Resultado esperado: Se ejecuta la aplicación como se esperada, mostrando la vista de autenticarse con los campos de nombre de usuario, contraseña, dirección de servidor y el botón de entrar.	

Tabla 3.5 Prueba de aceptación #1

Caso de Prueba Aceptación	
Número: 3	**Historia de usuario:** 3
Nombre: Autenticar usuario.	
Descripción: El sistema debe poseer una interfaz de autenticación donde el usuario selecciona el perfil para abrir una sesión de trabajo en el sistema.	
Condiciones de Ejecución: Debe existir creado un usuario con una contraseña que pertenezcan a un servidor, además de tener ese usuario un perfil asignado**.**	
Pasos de Ejecución: El usuario introduce correctamente el nombre de usuario, la contraseña y la dirección del servidor. El sistema realiza la petición con el servidor. El servidor envía la respuesta de la petición al sistema. El sistema muestra el listado de los perfiles asociados el usuario. El usuario selecciona el perfil y entra al sistema.	
Resultado esperado: Se autentica el usuario como se esperada, mostrando la vista principal de la aplicación.	

Tabla 3.6 Prueba de aceptación #3

Caso de Prueba Aceptación	
Número: 4	**Historia de usuario:** 4

Nombre: Cerrar sesión.
Descripción: El sistema debe permite al usuario cerrar su sesión de trabajo en el mismo.
Condiciones de Ejecución: El usuario debe de estar autenticado.
Pasos de Ejecución: El usuario cierra la sesión. El sistema envía la petición al servidor. El sistema cierra la sesión
Resultado esperado: Se cierra la sesión.

Tabla 3.7 Prueba de aceptación #4

Caso de Prueba Aceptación	
Número: 5	**Historia de usuario:** 5
Nombre: Visualizar sumario de alarmas.	
Descripción: El sistema debe permitir al usuario visualizar el estado de las alarmas, ordenas por criticidad del sistema.	
Condiciones de Ejecución: El usuario debe de estar autenticado.	
Pasos de Ejecución: Al desplazar el menú que se encuentra oculta en el lado izquierdo de la pantalla seleccionamos la opción Sumario Alarmas, donde se representa un listado de las alarmas que se encuentran activas donde cada un segundo el sistema envía la petición al servidor actualizándose la vista.	
Resultado esperado: Se muestran el listado de alarmas que se encuentran activas en el sistema.	

Tabla 3.8 Prueba de aceptación #5

Caso de Prueba Aceptación	
Número: 7	**Historia de usuario:** 7
Nombre: Visualizar sumario de puntos digitales.	
Descripción: El sistema debe permitir al usuario visualizar el estado de los puntos digitales.	
Condiciones de Ejecución: El usuario debe de estar autenticado.	

Pasos de Ejecución: Al desplazar el menú que se encuentra oculta en el lado izquierdo de la pantalla seleccionamos la opción Sumario Ptos Digitales, donde se representa un listado de los puntos digitales existentes donde cada un segundo el sistema envía la petición al servidor actualizándose la vista.
Resultado esperado: El sistema debe permitir al usuario visualizar el listado de los puntos digitales existentes.

Tabla 3.9 Prueba de aceptación #7

Caso de Prueba Aceptación	
Número: 8	**Historia de usuario:** 8
Nombre: Visualizar sumario de dispositivos.	
Descripción: El sistema debe permitir al usuario visualizar el estado de los dispositivos.	
Condiciones de Ejecución: El usuario debe de estar autenticado.	
Pasos de Ejecución: Al desplazar el menú que se encuentra oculta en el lado izquierdo de la pantalla seleccionamos la opción Sumario Dispositivos, donde se representa un listado de los dispositivos existentes en el campo donde cada un segundo el sistema envía la petición al servidor actualizándose la vista	
Resultado esperado: El sistema debe permitir al usuario visualizar el listado de dispositivos existentes en el campo.	

Tabla 3.10 Prueba de aceptación #8

Caso de Prueba Aceptación	
Número: 9	**Historia de usuario:** 9
Nombre: Notificación sonora de la existencia de alarmas.	
Descripción: El sistema debe notificar de forma sonora al usuario la existencia de una alarma activa no reconocida en el SCADA SAINUX.	
Condiciones de Ejecución: El usuario debe de esta autenticado y debe haber una alarma activa no reconocida**.**	
Pasos de Ejecución:	

Resultado esperado: El sistema emite sonido.

Tabla 3.11 Prueba de aceptación #9

Caso de Prueba Aceptación	
Número: 10	**Historia de usuario:** 10
Nombre: Visualizar cada vista de sumario como máximo una sola vez.	
Descripción: El sistema debe permitir al usuario visualizar cada vista de sumario como máximo una sola vez.	
Condiciones de Ejecución: El usuario debe d estar autenticado	
Pasos de Ejecución:	
Resultado esperado: El sistema solo puede visualizar un sumario de cada tipo.	

Tabla 3.12 Prueba de aceptación #10

Caso de Prueba Aceptación	
Número: 11	**Historia de usuario:** 11
Nombre: Realizar paginado de sumarios.	
Descripción: El sistema debe permitir al usuario cuando este visualizando cada vista de sumario poder cambiar entre las diferentes páginas que puedan tener las mismas.	
Condiciones de Ejecución: El usuario debe d estar autenticado y visualizando un sumario	
Pasos de Ejecución: El usuario abre un sumario y presiona el botan de adelante o atrás para cambiar a la página siguiente o anterior.	
Resultado esperado: El sistema realiza el paginado y visualiza las entidades referentes a esa página.	

Tabla 3.13 Prueba de aceptación #10

9.1 Anexo 3 Entrevista

Para establecer una serie de conceptos necesarios para la fundamentación teórica de la investigación, de los cuales no se contaba con una definición en fuentes bibliográficas confiables, se realizó una entrevista al especialista Luis Andrés Valido Fajardo para conocer una definición de los mismos desde el punto de vista de los desarrolladores del proyecto SCADA.

Entrevista al especialista Luis Andrés Valido Fajardo:

- ✓ Entre los conceptos que se manejan en el visualizador del SCADA SAINUX se encuentra el de despliegue, ¿me podría dar una definición del mismo desde su punto de vista?
- ✓ Una de las vistas que más se utiliza en la interfaz hombre-máquina para representarle la información en tiempo real al operador es la de sumario, ¿me podría decir la definición de este término para los desarrolladores del proyecto?
- ✓ En el proyecto Sainux recientemente se ha desarrollado un servidor para aligerar y atender las solicitudes de clientes interfaz hombre-máquina, ¿me podría decir algunas de sus funcionalidades?