Universidad de Matanzas
Facultad de Ciencias Técnicas

Visualizador de la Interfaz Hombre-Máquina del Sistema ROFLEXIN/LC para Dispositivos Móviles

Tesis Presentada como Requisito Parcial para la Obtención del Título de Máster en Ingeniería Asistida por Computadora

Autor: Ing. Luis Andrés Valido Fajardo

Tutor: Dr.C. Ramón Quiza Sardiñas

Matanzas, 2019

Declaración de Autoría y Nota Legal

Yo, Ing. Luis Andrés Valido Fajardo , declaro que soy el único autor de la siguiente tesis, titulada Visualizador de la Interfaz Hombre-Máquina del Sistema ROFLEXIN/LC para Dispositivos Móviles y, en virtud de tal, cedo el derecho de copia de la misma a la Universidad de Matanzas, bajo la licencia Creative Commons de tipo Reconocimiento No Comercial Sin Obra Derivada, con lo cual se permite su copia y distribución por cualquier medio siempre que mantenga el reconocimiento de sus autores, no haga uso comercial de la obra y no realice ninguna modificación de ella.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____

Ing. Luis Andrés Valido Fajardo

Resumen

El Centro de Estudios de Fabricación Avanzada y Sostenible (CEFAS) de la Universidad de Matanzas desarrolla el proyecto ROFLEXIN/LC: Sistema robusto, flexible e inteligente, de bajo costo, para monitoreo de sistemas y procesos mecánicos. Este proyecto esta dirigido fundamentalmente a pequeñas y medianas empresas, donde el uso de las alternativas disponibles en el mercado mundial no es financieramente factible. La presente investigación surge a partir de la necesidad del personal que utiliza el ROFLEXIN/LC, de visualizar la información recopilada por dicho sistema en dispositivos móviles y tablets. El objetivo fundamental es desarrollar una aplicación para móviles capaz de visualizar en tiempo real la información del sistema ROFLEXIN/LC. Para el desarrollo de la solución propuesta se realiza un análisis de las principales herramientas, tecnologías y metodologías que se utilizan en la construcción de un software. El proceso estuvo guiado por el uso de las siguientes herramientas y tecnologías: *Visual Paradigm* como herramienta CASE, UML como lenguaje de modelado, JSON como formato de intercambio de datos y Java como lenguaje de programación. Desarrollado con el IDE Eclipse con la extensión ADT. Para validar que los resultados alcanzados fueron los esperados se realizó un conjunto de pruebas. Se logró corregir todas las no conformidades detectadas lo que significa que la solución propuesta está lista para ser desplegada. Finalmente se obtuvo un Visualizador para dispositivos móviles y tablets con sistema operativo Android capaz de representar en tiempo real la información proveniente del sistema ROFLEXIN/LC.

Palabras claves: Android, Interfaz Hombre-Máquina, Sistema de monitoreo.

Abstract

The Center for Advanced and Sustainable Manufacturing Studies (CEFAS) of the University of Matanzas develops the ROFLEXIN/LC (Robust, flexible , intelligent and low cost) project, a low-cost monitoring system for processes and mechanical systems. This project is primarily, though not exclusively, aimed at small and medium-sized companies, where the use of alternatives available in the world market is not financially feasible. The present investigation arises from the need of the directors, supervisors and personnel that use the ROFLEXIN/LC system, to visualize the information gathered by said system in mobile devices and tablets. The main objective is to develop a mobile application capable of real-time visualization of ROFLEXIN/LC system information. For the development of the proposed solution an analysis is made of the main tools, technologies and methodologies used in the construction of a software. The process was guided by the use of the following tools and technologies: Visual Paradigm as a CASE tool, UML as a modeling language, JSON as a data exchange format and Java as a programming language. Developed with the Eclipse IDE with the extension ADT. To validate that the results achieved were as expected, a set of tests was carried out: acceptance, compatibility, usability, field, performance tests and user satisfaction tests. It was possible to correct all detected nonconformities which means that the proposed solution is ready to be deployed. Finally, a Visualizer was obtained for mobile devices and tablets with an Android operating system capable of representing in real time the information coming from the ROFLEXIN/LC system.

Keyworks: Android, Human-Machine Interface, Monitoring system.

Tabla de Contenido

Introducción

En poco tiempo las Tecnologías de la Información y las Comunicaciones (TIC) han evolucionado considerablemente, hasta el punto de volverse casi imprescindible para algunos sectores de nuestra sociedad. La automatización de los procesos industriales se ha convertido en un gran avance para el sector empresarial, con su uso, se han revolucionado las producciones a diferentes escalas, reduciendo el peligro de catástrofes, además de obtener grandes ahorros en tiempo, dinero y recursos.

Dentro de esa gama de productos que han revolucionados se encuentra los sistemas de monitoreo los cuales contribuyen significativamente al aumento de la productividad y la calidad, además de garantizar una mayor seguridad para el equipamiento y los operadores de procesos. Sin embargo, los altos costos de estos sistemas, junto a la necesidad de importarlos, hacen prohibitiva su aplicación, especialmente para pequeñas y medianas empresas.

La informatización de la sociedad cubana es una voluntad política del gobierno que está expresada en los Lineamientos de la Política Económica y Social, aprobados en el Sexto Congreso del Partido Comunista de Cuba (CubaDebate, 2017) . En varios sectores de la industria y la economía cubana se necesita un sistema robusto, flexible y de bajo costo para el monitoreo de sistemas y procesos mecánicos, que sea posible de implementar con un mínimo de componentes importados.

El Centro de Estudios de Fabricación Avanzada y Sostenible (CEFAS) de la Universidad de Matanzas vinculado al Programa Nacional de Ciencia, Tecnología e Innovación desarrolla el proyecto ROFLEXIN/LC un sistema de monitoreo de bajo costo para procesos y sistemas mecánicos.

Dicho sistema está compuesto por varios módulos que interactúan entre sí, comenzando el proceso por los controladores de dispositivos y finalizando en la Interfaz Hombre-Máquina o HMI por la abreviación de *Human-Machine Interface* .

El HMI está compuesto por un ambiente de edición o editor, que es donde se realizan las configuraciones para el monitoreo, y el ambiente de ejecución o visualizador, que permite al operador supervisar el sistema o los procesos mecánicos.

El ROFLEXIN/LC está orientado al trabajo en estaciones fijas como una aplicación de escritorio, el cual debe ser instalado sobre el sistema operativo de cada ordenador destinado a la monitorización de los procesos.

Las condiciones planteadas anteriormente provocan que los directivos, supervisores y el personal autorizado de la empresa, no puedan visualizar la información de los procesos en tiempo real mediante el Ambiente de Ejecución del HMI en estaciones de trabajo que no tengan instalado el sistema, limitando la disponibilidad de la aplicación. Esto ocasiona que los directivos necesiten trasladarse hacia las oficinas para poder supervisar los procesos, estando estas en muchas ocasiones alejadas de las entidades administrativas, provocando además gastos innecesarios.

Teniendo en cuenta la situación planteada con anterioridad se define el siguiente **problema de investigación**: ¿Cómo lograr la disponibilidad del visualizador del HMI del ROFLEXIN/LC en los dispositivos móviles?

Planteandosé como **hipótesis**: Mediante el uso de las herramientas y tecnologías actuales, es posible desarrollar el ambiente ejecución o visualizador del HMI del sistema ROFLEXIN/LC, para dispositivos móviles.

En concordancia con lo anterior se propone como **objetivo general**: Desarrollar una aplicación para dispositivos móviles que permita la disponibilidad del el ambiente ejecucción o visualizador del HMI del ROFLEXIN/LC.

Para dar cumplimiento a los objetivos de esta investigación se definieron las siguientes **tareas investigativas**:

1. Establecimiento de los fundamentos teórico-metodológicos para el desarrollo de sistemas informáticos de monitoreo en el entorno móvil.

2. Análisis del estado del arte de las principales tecnologías, metodologías y herramientas para el desarrollo de aplicaciones móviles en tiempo real.

3. Análisis y diseño de la solución informática.

4. Implementación de la solución informática.
5. Validación de los resultados obtenidos mediante la realización de pruebas.
6. Valoración cualitativa de los resultados obtenidos en la validación de la solución propuesta.

Durante la investigación se llevan a cabo varios métodos y técnicas en la búsqueda y procesamiento de la información como son: Métodos Teóricos

- **Histórico-lógico:** Para la comprensión de los antecedentes y las tendencias actuales referidas la evolución en el mundo de los sistemas de monitoreo desde dispositivos moviles.

Métodos Empíricos

- **Consulta bibliográfica:** Empleada para consultar las fuentes de información relacionados con los tipos de los sumarios y gráficos visualizados en los entornos móviles de los HMI en sistemas de monitoreo.

La estructura del documento se resume en los siguientes acápites:

Capítulo 1. Fundamentación teórica: Abarca conceptos fundamentales asociados al sistema ROFLEXIN/LC. De igual manera se analizan aplicaciones de monitoreo desde dispositivos móviles. Además se exponen las características principales de estos sistemas. Se analizan las principales tendencias, tecnologías, metodologías y softwares utilizados en la actualidad para el desarrollo de aplicaciones móviles . A su vez se analizan y se fundamenta la selección de estas para el desarrollo de la solución propuesta.

Capítulo 2. Análisis y diseño del sistema: Se reflejan las actividades realizadas en los procesos de análisis y diseño de la solución propuesta; proceso que será guiado por la metodología de desarrollo seleccionada. En el mismo se realiza el modelo de dominio donde se describen las entidades que intervienen con el objetivo de facilitar la comprensión de los principales conceptos que se utilizarán en el proceso de negocio identificado. Se exponen

los artefactos más importantes que describen el flujo normal de eventos que ocurren en el sistema, se realiza una descripción de la solución propuesta, planteándose los requisitos funcionales y no funcionales. Se define la arquitectura que tendrá la solución propuesta.

Capítulo 3. Implementación y pruebas: En este capítulo se aborda las tareas de implementación a través de las Tareas de Ingeniería. Se describe los estándares de codificación para el desarrollo de la solución propuesta a tener en cuenta, así como los resultados de las pruebas aplicadas para comprobar que las Historias de Usuario implementadas estén correctas a través de las pruebas de aceptación.

Capítulo 4. Análisis de los resultados: En este capítulo se hará un análisis sobre los resultados obtenidos por las pruebas realizadas para la correcta validación del sistema. De igual forma son abordados aquellos aspectos que presentaron dificultades y las decisiones tomadas para solucionarlos.

Capítulo 1: Fundamentación Teórica

En el presente capítulo se engloban conceptos fundamentales asociados al sistema ROFLEXIN/LC. De igual manera se analizan aplicaciones de monitoreo desde dispositivos móviles. Además se exponen las características principales de estos sistemas. Se analizan las principales tendencias, tecnologías, metodologías y softwares utilizados en la actualidad para el desarrollo de aplicaciones móviles . A su vez se analizan y se fundamenta la selección de estas para el desarrollo de la solución propuesta.

1.1 ROFLEXIN/LC

El proyecto ROFLEXIN/LC se enfoca a la obtención de un sistema de monitoreo de bajo costo para procesos y sistemas mecánicos. El mismo se basa en la utilización de tecnologías de hardware más baratas que las convencionales con un software desarrollado con herramientas libres y código abierto bajo una arquitectura abierta que solo use aquellos módulos que sean necesarios en dependencia de las necesidades de clientes.

La arquitectura que se propone para la conexión del hardware se muestra en la figura, la misma está compuesta por un sistema de acondicionamiento de señal que está formado por una placa con elementos circuitales que trasforman la salida de los sensores conectados al sistema, señal la cual se conecta al conversor análogo-digital (ADC) el cual se encuentra conectado por el bus i2c a la Raspberry Pi, la cual se encarga de la recolección de los datos para su envió a través de un interface *ethernet* que se encuentra conectada a un servidor en el cual una aplicación de escritorio recepciona los datos y brinda opciones de visualización de gráficos de históricos y tendencias, y permite además la configuración del hardware de manera remota, así como la obtención de modelos del proceso a través de la utilización de algoritmos de modelación basados en elementos de inteligencia artificial. Además, dicho servidor también brinda la posibilidad de realizar la supervisión vía web mediante un servidor web con el cual se pueden comunicar todos los ordenadores conectados a su red a

través del protocolo de trasferencia de hipertexto (http) (Jaén, 2017).

1.2 Enfoques existentes

El módulo de HMI móvil para sistemas de monitoreo está compuesto un cliente móvil y un servidor HMI web, que se comunican haciendo uso del protocolo de comunicación HTTP (HyperText Transfer Protocol). Éste módulo brinda las funcionalidad convencionales de un sistema de monitoreo, permite a los usuarios autorizados estar en contacto directo con el sistema, realizar el monitoreo del proceso en general; permitiendo así una alta disponibilidad de los recursos solicitados donde se tenga acceso a Internet, fácil integración de datos de diferentes fuentes y su respectiva centralización, racionalización del trabajo permitiendo la reducción del tiempo y dinero destinado al mantenimiento, menos requerimientos de memoria en comparación con programas instalados localmente.
En la actualidad existe una gran variedad de sistemas de monitero o *Supervisory Control And Data Acquisition* (SCADA) que poseen una Interfaz Hombre-Máquina que incluyen una extensión para la supervisión y control de los procesos mediante dispositivos móviles.

1.2.1 Slicetex Virtual HMI (Virtual HMI)

Virtual HMI es un panel HMI virtual que le permite controlar el *Programmable Logic Controller* (PLC) remotamente y al mismo tiempo visualizar mensajes provenientes del PLC en un dispositivo Android con conexión a la red *ethernet*. La aplicación Virtual HMI en conjunto con una *tablet*, puede ser una alternativa económica frente a los terminales HMI *touch* existentes en el mercado. Está diseñada para mostrar múltiples configuraciones visuales prediseñadas. Cada pantalla se llama panel. Se puede elegir diferentes paneles de acuerdo a su aplicación. Por ejemplo un panel que combine un monitor de pantalla de cristal líquido y teclas, o un panel que solo contenga teclas o barras deslizantes. Necesita de una red Wi-Fi para funcionar, no es posible utilizarlo con redes de datos móvil (GPRS, Edge, 3G, 4G, etc.) (Estudiez, 2015).

1.2.2 EasyAccess 2.0

EasyAccess 2.0 es una herramienta de acceso remoto para su máquina o HMI industrial. Le permite monitorear o actualizar los controladores o proyectos conectados de HMI. Cuenta con su versión para dispositivos móviles con sistema operativo Android con una versión 4.1 o superior. Aunque las funcionalidades en un dispositivo Android y su interfaz de usuario pueden ser ligeramente diferentes a las de la PC (Weintek Labs., 2014), (Maple Systems, 2015).

1.2.3 EZ RMC Remote HMI App

La aplicación EZ RMC Remote HMI es una aplicación diseñada para que sus dispositivos Android habiliten la supervisión y el control de sus HMI EZTouch desde EZAutomation.net. La aplicación permite el acceso directo y total a su HMI EZTouch. Permite la visualización de los datos como hojas de cálculo, gráficos de barra o de líneas (EZAutomation.net, 2018), (Bernauer, 2015) . Entre sus principales características podemos mencionar:

1. Visualización y control en tiempo real de su panel HMI EZTouch.
2. Zoom de estilo *pellizco* para una visión más profunda de su proyecto HMI.
3. Guardar capturas de pantalla directamente desde la aplicación.

1.2.4 TeslaSCADA2 Runtime

TeslaSCADA2 Runtime es el ambiente de visualización para proyectos desarrollados en TeslaSCADA IDE. El mismo requiere del sistema operativo Android en su versión 3.0 o superior. Cuenta con una interfaz gráfica para la configuración de la herramienta. Para visualizar los proyectos en la aplicación se cuenta con dos formas. La primera a través de la red, conectandosé a una red Wi-Fi y descargando el proyecto. La segunda copiando el fichero del proyecto dentro del directorio del dispositivo (Tesla, 2011), (2017) .

1.3 Sistemas operativos para móviles

Un sistema operativo (SO) o, frecuentemente, *OS* del inglés *Operating System* es un programa o conjunto de programas de un sistema informático que gestiona los recursos de hardware y provee servicios a los programas de aplicación, ejecutándose en modo privilegiado respecto de los restantes (aunque puede que parte de él se ejecute en espacio de usuario) (Niño, 2011), (Llaven, 2015). Los mismos pueden ser diseñados o especializados para computadoras tanto personal como de escritorio, servidores, sistemas empotrados o para *smartphone* de los cuales se abordarán los predominantes a continuación.

1.3.1 Android

Android es el SO basado en Linux desarrollado por *Googl*e, sigue una filosofía de software libre y código abierto. Está desarrollado en Java, permite ejecutar procesos en segundo plano, soporta gráficos en 2D y 3D, ofrece una buena interfaz de usuario y nos proporciona una base de datos SQLite embebida. El código de Android está liberado bajo licencia Apache v2.0 (a excepción de las modificaciones del kernel que han sido liberadas bajo la GNU GPLv2)

Android también ha incursionado en los *tablets* de la mano de compañías como HP, Dell, Notion INC o MSI (por citar algunos), *eBooks* (como el Alex de Grammata o el Nook de Barnes&Noble) e incluso en *Google TV*. Algunas de sus características son un *kernel* basado en Linux con un *framewor*k de aplicaciones que permite reutilizar y remplazar sus componentes. Posee un navegador web integrado basado en *Webkit*. Su gráficos están optimizados en el caso de 2D con librerías propias y el 3D basada en OpenGL. Para el almacenamiento de datos usa SQLite. Brinda soporte para los formatos más utilizados de sonido, video e imagen, para la telefonía GSM, Bluetooth, EDGE, 3G, Wifi, cámara, GPS, compás, acelerómetro y un entorno de desarrollo que incluye: documentación, emulador de dispositivos, herramientas de debug y análisis de uso de memoria/CPU, plugin para el entorno de desarrollo Eclipse y varias utilidades complementarias (Brahler, 2010), (Google, 2012), (Android, 2014).

1.3.2 *iOS*

Sistema operativo de *Apple* para sus dispositivos *IPhone*, *iPod Touch*, e *iPad*. *Apple* no permite la instalación de *iOS* en hardware de terceros. *iOS* se deriva de *Mac OS X*, que a su vez está basado en Darwin BSD, y por lo tanto es un sistema operativo Unix. Para todo tipo de movimientos se necesita *iTunes* ya sea a la hora de actualizar el SO como sincronización de archivos, creación de aplicaciones propias, etc. Posteriormente el 11 de julio apareció la *App Store* oficial de *Apple* que supuso una revolución para las aplicaciones móviles (Smyth, 2011), (Inc., 2014).

1.3.3 *Symbian*

Symbian OS fue producto de la alianza de varias empresas de telefonía móvil, entre las que se encuentran *Nokia, Sony Ericsson, PSION, Samsung, Siemens, Arima, Benq, Fujitsu, Lenovo, LG, Motorola, Mitsubishi Electric, Panasonic, Sharp*.
En 2008, Nokia decidió comprar *Symbian* con la idea de establecer la Fundación *Symbian* y convertir este sistema operativo en una plataforma abierta, pero lo acaba haciendo bajo la licencia *Nokia Symbian*, poniendo el desarrollo en manos principalmende de *Accenture*. *Symbian* tendrá soporte hasta el año 2016, dejando a día de hoy ya de presentar terminales. En Nokia 803(sucesor del Nokia N8) podría ser el último con este SO. *Symbian* tuvo una época dorada en la que lideró el mercado para dar paso a una nueva etapa con *Windows Phone* como protagonista.
El lenguaje habitual para programación en *Symbian* es C++, pero se han ido desarrollando herramientas para programar en Java, Python, OPL, etc. por lo que hay un gran abanico de posibilidades a la hora de desarrollar para Symbian OS (Nokia, 2010)

1.3.4 *Blackberry OS*

Blackberry OS es el sistema operativo creado por RIM (*Research In Motion*), es un sistema multitarea enfocado principalmente a un uso profesional con sus servicios asociados de mensajería y gestión de correo dando una gran garantía de seguridad. Es un sistema de código cerrado con licencia propietaria. La compañía fabricante y promotora de Blackberry,

desarrolla su propio software para sus dispositivos usando C++, C y la tecnología Java (Limited, 2012).

1.3.5 *Windows Phone*

Windows Phone 7 aparece en octubre de 2010 como un sistema operativo móvil desarrollado por *Microsoft* (modelo propietario), como sucesor de la plataforma *Windows Mobile*. Este no será compatible con *Windows Mobile* 6 y estará menos orientada al mercado empresarial. Con *Windows Phone* 7 *Microsoft* ofrece una nueva interfaz de usuario e integra varios servicios en el sistema operativo.

Se basa en el núcleo del sistema operativo *Windows CE* y cuenta con un conjunto de aplicaciones básicas utilizando las API de *Microsoft Windows* y estéticamente está diseñado para ser similar a las versiones de escritorio de *Windows*. El lenguaje empleado para hacer desarrollos para este SO es C# (Microsoft, 2010).

1.3.6 Comparativa

Android, al contrario que otros sistemas operativos para dispositivos móviles como *iOS* o *Windows Phone*, se desarrolla de forma abierta y se puede acceder tanto al código fuente como a la lista de incidencias donde se pueden ver problemas aún no resueltos y reportar problemas nuevos. Android, permite que cualquier desarrollador pueda realizar aplicaciones para el sistema operativo, lo que le otorga una enorme cantidad de aplicaciones disponibles, algunas gratuitas y otras de pago, lo que aventaja enormemente a *iOS* que sólo permite utilizar aplicaciones desarrolladas por *Apple* o por grandes compañías de software. *Windows Phone*, sin embargo, a pesar de llevar casi la misma política de *iOS*, ha lanzado una enorme cantidad de aplicaciones, que pudieran otorgarle un segundo lugar en este sentido.

El desarrollo de aplicaciones para Android no requiere aprender lenguajes complejos de programación. Todo lo que se necesita es un conocimiento aceptable de Java y estar en posesión del kit de desarrollo de software o SDK provisto por Google el cual se puede descargar gratuitamente.

Sin duda, el crecimiento exponencial de Android ha ocurrido, en gran medida, por sus

cuestiones comerciales más que de rendimiento o popularidad mediática. Y es que el hecho de que cualquier fabricante pueda tomar Android como sistema operativo base para sus equipos, convirtiéndolos así en *smartphones*, ha llevado a una verdadera revolución en lo que a dispositivos móviles se refiere, con distintos modelos y equipos variados inundando los mercados internacionales. La mayor parte de los dispositivos que se adquieren por parte de los usuarios son de gama media o baja y vienen equipados con Android.
Si se tiene en cuenta los datos correspondientes a las cuotas de mercado (Pascual, 2018), (Nieto, 2018), (Moscaritolo, 2017) y de la alta actividad en los desarrollos de aplicaciones, de entre todas las plataformas más populares Android parece ser las más indicada para realizar desarrollos de aplicaciones. Está en pleno auge tomando el terreno a las demás plataformas y llevando un buen ritmo de crecimiento.

1.4 Aplicaciones nativas vs Aplicaciones Web

A nivel de desarrollo, existen varias formas de programar las aplicaciones. Cada una de la formas, tiene diferentes características y limitaciones, sobre todo desde el punto de vista técnico. Aunque a primera vista esto no parezca incumbencia del diseñador, la realidad es que el tipo de aplicación que se elija, condicionará el diseño visual y la interacción.
En la actualidad se está dando un gran enfrentamiento para ver cual es la forma más sencilla y económica de lograr desarrollar aplicaciones para los diferentes dispositivos móviles. Con la aparición de HTML5 y todas las posibilidades que incorpora el mercado de las aplicaciones móviles sufrió algunos cambios. Muchos son los que se debaten entre un desarrollo más complejo y en varios lenguajes de programación dependiendo del sistema operativo móvil pero logrando una aplicación totalmente en armonía con el dispositivo, o por el contrario reducir costes y buscar un mecanismo multiplataforma para llegar a todos ellos pero perdiendo muchas de sus ventajas. También han ido apareciendo algunas alternativas híbridas entre ambas.

1.4.1 Aplicaciones nativas

Lo más ideal dado un dispositivo parece desarrollar una aplicación nativa pensada totalmente para el mismo sabiendo sus posibilidades y explotándolas de la mejor manera para

conseguir el objetivo que se desee.

Lo primero a destacar de las aplicaciones nativas es que permiten explotar al máximo las prestaciones del dispositivo teniendo un control absoluto sobre el mismo. Esto dará una mejor experiencia de usuario más ligada al dispositivo, además la interfaz no tiene que cargarse junto con el resto de datos.

Las notificaciones han cobrado gran relevancia en todos los SO móviles ya que permiten una interacción más directa con el usuario pudiendo informarle de novedades, cualquier información relevante, etc. Solo se tiene control sobre las *Push Notifications* si desarrolla la aplicación de forma nativa.

Como desventaja se requieren diferentes herramientas de desarrollo. Para cada plataforma de desarrollo se utilizan diferentes herramientas de programación o SDK. De igual manera el código fuente no es reutilizable para cada plataforma. Como hemos indicado anteriormente, Android se desarrolla en Java, IOS en Objective C, Window Phone en C# etc. Con lo que el código fuente no es reutilizable entre diferentes plataformas.

1.4.2 Aplicaciones web

Las aplicaciones móviles web o también llamadas *webapps* tienen HTML como lenguaje base de programación conjuntamente con JavaScript y CSS, que son herramientas ya mayormente conocidas por los programadores web. En este caso se programa de manera independiente al sistema operativo (en el cual se utilizará la aplicación) y no se emplea un SDK en concreto. Las aplicaciones web no necesitan instalarse, ya que se visualizan usando el navegador del teléfono como un sitio web normal, por eso estas aplicaciones pueden ser fácilmente utilizadas en diferentes plataformas sin mayores inconvenientes y sin necesidad de desarrollar un código diferente para cada caso particular. Por tanto el uso de HTML5 supondrá una serie de ventajas importantes:

La principal relevancia de utilizar aplicaciones web HTML5 es que son multiplataforma, se podrá crear una aplicación una vez y distribuirlo hacia todas las plataformas. Probablemente sean necesarios algunos ajustes para el correcto funcionamiento en todos los dispositivos pero no implicará un desarrollo por cada plataforma.

Además de programar la aplicación una sola vez, la tecnología es más sencilla que la ne-

cesaria para desarrollar aplicaciones nativas, pudiendo utilizar lenguajes como HTML5, JavaScript o CSS. Por tanto está mucho mas extendida y es más sencillo encontrar programadores para estas tecnologías que para las concretas de cada SO móvil.
Por otro lado a pesar de los grandes avances de HTML5 al no estar haciendo un desarrollo nativo con las SDKs y APIs disponibles para cada plataforma estamos inevitablemente perdiendo una parte importante de la funcionalidad del dispositivo así como en apariencia, siendo complicado que la interfaz se sienta tan ligada al teléfono como en casos nativos. También puede ser complicado lograr un solo diseño que se adapte perfectamente a todos los dispositivos a la vez.

1.4.2.1 Progressive Web Apps (PWA)

Las aplicaciones progresivas son una nueva generación de aplicaciones potenciadas por Google. Están en expansión y por el momento no son muy conocidas. Se basan en aplicaciones desarrolladas con tecnologías web y usan el navegador como motor de la aplicaciones. Por eso son accesibles desde cualquier dispositivo. Visualmente es la misma experiencia que la de una aplicación nativa.

1.4.3 Híbridas

Este tipo de aplicaciones es una especie de combinación entre las dos anteriores. La forma de desarrollarlas es similar a la de una aplicación web (usando HTML, CSS y JavaScript), aunque a diferencia de las aplicaciones web, éstas permiten acceder (usando librerías) a las capacidades del teléfono, tal como lo haría una aplicación móvil nativa. Y una vez que la aplicación está terminada, se compila o empaqueta de forma tal, que el resultado final es como si se tratara de una aplicación nativa. Uno de sus inconvenientes es el diseño visual no siempre relacionado con el sistema operativo en el que se muestre. Ya que no se tratan de aplicaciones móviles específicas para cada sistema, pueden existir ciertas incompatibilidades para cada caso.

1.4.4 Comparativa

Es evidente que el coste que supone el desarrollo nativo será considerablemente más alto, por lo que es bueno analizar el tipo de aplicación antes de decidir por qué tipo de desarrollo decantarse. Si la aplicación tiene mucho contenido web y no necesita un control minucioso de las funciones del dispositivo se puede valorar el crear una aplicación web HTML5 o bien híbrida. Con esto en vez de tener que desarrollar una aplicación distinta para cada plataforma móvil, utilizando lenguajes distintos y necesitando a personal distinto especializado en cada sistema operativo, podrían desarrollar una sola aplicación HTML5 que valiese para todas las plataformas. A pesar de esto será muy difícil que las aplicaciones web alcancen un potencial tan grande que realmente pueda desbancar a las nativas, existen una serie de puntos a destacar en las aplicaciones nativas frente a HTML5:

- Si lo que se desea es una aplicación potente con acceso a muchas funcionalidades del teléfono y de una forma lo más eficiente posible habrá que desarrollar de forma nativa, haciendo una aplicación por cada sistema operativo.
- Tampoco se puede olvidar que la persistencia también estará más limitada en HTML5.
- Con HTML5 obtendremos interfaces más pobres y menos ligadas al teléfono. Uno de los puntos fuertes de las aplicaciones nativas es la incorporación de gestos multitoque que dan mayor usabilidad.
- No todos los dispositivos soportan de la misma forma las mismas cosas por lo que la idea de hacer la misma cosa para todos ellos siempre se quedará detrás de un desarrollo totalmente pensado para ese dispositivo.

Y es por estos elementos planteados y los reflejados por Terrera (Terrera, 2014) y Jiménez (Jiménez, 2018) los que permiten decidir optar por el desarrollo de una aplicación nativa de Android.

1.5 Metodologías para el desarrollo de software

Las metodologías de desarrollo de software se definen como un conjunto de herramientas, técnicas, procedimientos y soporte documental para el diseño de sistemas de información.

Las metodologías de desarrollo de software se divide en dos grandes grupos: las metodologías tradicionales o pesadas que se centran en el control del proceso mediante una rigurosa definición de roles, actividades, artefactos, herramientas y documentación detallada para el modelado donde el resultado final sería un proceso de desarrollo más complejo, entre las metodologías tradicionales se encuentran RUP (*Rational Unified Process*) y MSF (*Microsoft Solution Framework*) siendo RUP la metodología más utilizada.
Por otra parte las metodologías ágiles que se basan en el desarrollo iterativo e incremental, teniendo presente los cambios y respondiendo a estos mediante la colaboración de un grupo de desarrolladores auto- organizados y multidisciplinares, dentro de este grupo se encuentra la metodología *eXtreme Programming* (XP) o Programación Extrema.

1.5.1 Metodología de desarrollo RUP

Se caracteriza por ser dirigida por casos de uso, estar centrada en la arquitectura y ser iterativa e incremental. Entre sus principios se encuentran adaptar el proceso a la necesidad del cliente, equilibrar prioridades, demostrar valor iterativamente, fomentar la colaboración entre equipos, elevar el nivel de abstracción y enfocarse en la calidad (Kruchten, 2004). Sin embargo, RUP presenta algunos inconvenientes para su aplicación en algunos proyectos como son: límite de tiempo demasiado ajustado, la documentación al ser tan exhaustiva hace mucho énfasis en la planificación, no siempre se puede llevar a cabo la implementación del software de acuerdo con su planificación pues aparecen cambios los cuales solo pueden ser percibidos en el momento de la codificación . RUP cuenta con cuatro fases de desarrollo como se muestra en la figura:

- **Conceptualización (Concepción o Inicio):** Se describe el negocio y se delimita el proyecto describiendo sus alcances con la identificación de los casos de uso del sistema.

- **Elaboración:** Se define la arquitectura del sistema y se obtiene una aplicación ejecutable que responde a los casos de uso que la comprometen. A pesar de que se desarrolla a profundidad una parte del sistema, las decisiones sobre la arquitectura se hacen sobre la base de la comprensión del sistema completo y los requerimientos

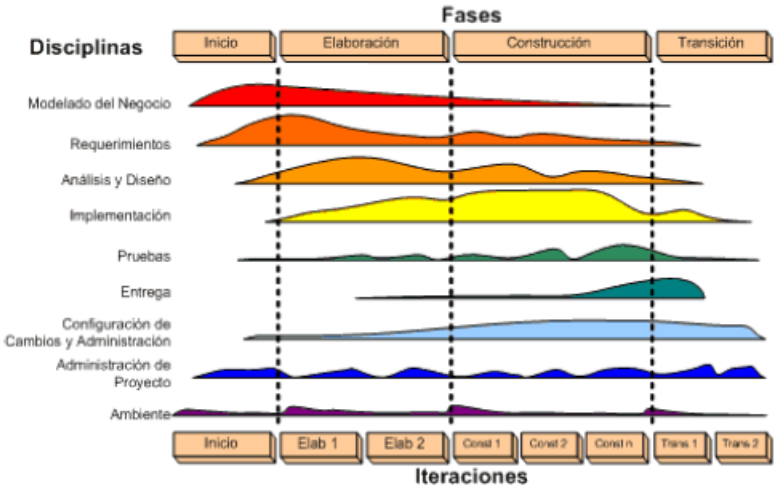


Figura 1.1: Ciclo de vida RUP (Muñoz, Rozas y Herrera, 2015)

(funcionales y no funcionales) identificados de acuerdo al alcance definido.

- **Construcción:** Se obtiene un producto listo para la utilización, que está documentado y tiene manual de usuario. Se obtienen tantas versiones de entrega como determinan las pruebas.

- **Transición:** Con la liberación, ya el producto está listo para su instalación en condiciones reales. Puede implicar la reparación de errores como parte del proceso de soporte.

1.5.2 Metodología de desarrollo XP

Metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promueve el trabajo en equipo, se preocupa por el aprendizaje de los desarrolladores, y propicia un buen clima de trabajo. Se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios (Beck y Molina, 2002). XP propone cuatro fases de desarrollo que son:

1. **Planificación y estimación:** Se analiza el negocio, se identifican los requisitos y se establecen los planes de entrega.

2. **Análisis y diseño:** Se diseña el sistema informático, se establece la arquitectura y se modelan las clases del sistema.

3. **Implementación:** Se desarrolla el sistema a partir de las clases modeladas.

4. **Pruebas:** Se prueba la solución y se verifica el correcto funcionamiento del sistema. A partir de las pruebas y cuando se valida la solución, comienza una nueva iteración con nuevas funcionalidades.

El proceso de desarrollo de XP sigue un ciclo, el cual consiste en los siguientes pasos como se muestra en la figura:

Figura 1.2: Ciclo de vida XP (Hechavarría, 2015)

1. El cliente define el valor del negocio a implementar.

2. El programador estima el esfuerzo necesario para su implementación.

3. El cliente selecciona qué construir, de acuerdo con sus prioridades y las restricciones de tiempo.

4. El programador construye ese valor del negocio.

5. Vuelve al paso uno.

1.5.3 Elección de la metodología de desarrollo de software

Se selecciona XP como metodología de desarrollo de software porque suprime la extensa documentación que proponen las metodologías tradicionales y asegura mediante sus principales valores, que el desarrollador sea versátil y conozca las necesidades elementales del cliente siendo este último parte del equipo de desarrollo, la documentación que se genera es suficiente y comprensible por desarrolladores y clientes, es ideal para proyectos donde el ciclo de vida es relativamente corto por lo que puede adaptarse a la solución propuesta ya que tiene un tiempo de duración de 6 meses, los cambios a los requerimientos son bienvenidos, aún en fases tardías del desarrollo, se centra fundamentalmente en entregar un producto de software en el menor tiempo posible.

1.6 Tecnologías y herramientas a utilizar

Las herramientas y tecnologías escogidas para el desarrollo de un proyecto deben ser seleccionadas cautelosamente, ya que pueden suponer el fracaso de éste o pueden aumentar su complejidad, para lo cual se deben conocer cuáles son las distintas alternativas y las necesidades del proyecto.

1.6.1 Lenguaje de modelado

Es un lenguaje artificial usado para expresar información, conocimiento, o sistemas, en una estructura definida por un conjunto de reglas consistentes. Desde el punto de vista de la ingeniería de software proporciona una forma estandar de describir el diseño y la funcionalidad de un sistema (Parra, 2018) .

1.6.1.1 Lenguaje de UML

Se selecciona como lenguaje el Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, *Unified Modeling Language*), debido a que se emplea para visualizar, especificar, construir y documentar los artefactos de la solución propuesta. Es un lenguaje estándar, fácil de aprender, y ofrece un conjunto de notaciones y diagramas estándar para mostrar la solución propuesta desde varias perspectivas. Está especialmente diseñado para apoyar

un estilo de desarrollo iterativo e incremental y presenta tecnología orientada a objetos. Ayuda al usuario a entender la realidad de la tecnología y tiene una notación gráfica muy expresiva que permite representar todas las fases de un proyecto informático (Rumbaugh, Booch y Jacobson, 2007).

1.6.2 Herramienta CASE

Las herramientas CASE (*Computer Aided Software Engineering*) fueron diseñadas para aumentar la productividad en el desarrollo de software, reduciendo el costo de estos en términos de tiempo y dinero. Corresponden a diversas aplicaciones informáticas, que incluyen un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de un proyecto a desarrollar. Además facilitan el uso de las distintas metodologías propias de la ingeniería del software.

1.6.2.1 *Visual Paradigm*

Se seleccionó como herramienta CASE el *Visual Paradigm* porque soporta el ciclo de vida completo del desarrollo de software, permite modelar los diagramas de la solución propuesta, realizar ingeniería tanto directa como inversa.
Soporta múltiples usuarios trabajando sobre el mismo proyecto, permite el control de versiones y generar la documentación automáticamente en formatos como web o PDF. Además este software de modelado UML ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor coste. A su vez es una herramienta libre y multiplataforma.
Soporta multitud de estándares de modelado como UM, SysML, ERD, DFD, BPMN. Es una herramienta que facilita y agiliza el desarrollo de los proyectos basados en experiencias de usuario apoyándose en la identificación de los casos, requisitos o flujos de acontecimientos (Oscar, 2013).

1.6.3 Formatos de comunicación

1.6.3.1 JSON

JSON (*JavaScript Object Notation*) es un formato ligero para el intercambio de datos. Es sencillo de leer y de escribir así como de interpretar por máquinas. Esta basado en un subconjunto de JavaScript. El formato de JSON es ampliamente reconocido por una gran variedad de lenguajes como Java, PHP, JavaScript, C++, C# y muchos otros. Esto hace que JSON sea un lenguaje de gran potencial para el intercambio de datos.
JSON es una forma de codificar objetos, arrays o cualquier otra serie de datos en un *string* y, posteriormente, poder descodificarlo sin muchos problemas. Es especialmente útil para enviar información entre cliente y servidor de una forma muy sencilla, puesto que cada componente descodifica la información según le convenga, indistintamente del resto del sistema (Keller, 2016),(Academy, 2017).

1.6.3.2 XML

XML proviene de *eXtensible Markup Language* (Lenguaje de Marcas Extensible). Se trata de un metalenguaje extensible de etiquetas que fue desarrollado por el *Word Wide Web Consortium (W3C)* (Vlist, 2002). Los servicios web son componentes de la red que brindan la posibilidad de realizar una serie de variada de operaciones, a través de métodos concretos que aprovechan el metalenguaje XML para sus comunicaciones, gracias a lo cual cualquier plataforma puede hacer uso de sus ventajas.

1.6.3.3 Comparativa

Estabilidad: Con JSON, se está limitado a almacenar sólo datos clásicos como texto y números. Sin embargo, XML le permite almacenar cualquier tipo de dato que se le pueda ocurrir. La capacidad para extender los atributos de los datos almacenados en los ficheros XML es lo que le permite ser más flexible que JSON. Sin embargo, también lo hace más difícil de leer. Esto hace a XML más extensible, pero puede que no sea algo bueno. Esto depende del tipo de la información que trata de transferir. Los documentos requieren extensibilidad para gestionar imágenes, tablas, gráficos, y otros elementos de formato. Sin

embargo, los datos clásicos no requieren esta extensibilidad y pueden beneficiarse de la simplicidad de JSON.

Legibilidad: Los ficheros JSON son más restrictivos y, por lo tanto, ligeramente más legibles. Esto se debe a que el número de formatos de datos permitidos por JSON es mucho menor que XML. Además, la estructura de los datos está más estandarizada con los ficheros JSON debido al hecho de que existen menos opciones cuando se compara con el formato XML.

Integración con otros los formatos: Con XML es posible adjuntar cualquier fichero de cualquier formato, esto puede parecer algo bueno al principio, también puede ser peligroso. Eso es porque podría incluir un fichero ejecutable que podría tener peligrosas consecuencias para la seguridad. Por otro lado, JSON sólo soporta formatos de datos tradicionales. La simplicidad de las estructuras de datos que JSON soporta hace imposible los ataques usando este formato.

Compartir datos tradicionales: JSON es la mejor herramienta para compartir datos. Esto se debe a que los datos están almacenados en vectores y registros, mientras que XML almacena los datos en árboles. Ambos tienen sus ventajas, pero las transferencias de datos son mucho más fáciles cuando los datos se almacenan en una estructura que está familiarizada a los lenguajes orientados a objetos. Esto hace que sea muy sencillo importar datos desde un fichero JSON a Perl, Ruby, JavaScript, Python, y otros muchos lenguajes.

1.6.3.4 Resultados

A partir de los parámetros evaluados anteriormente y apoyado en los resultados obtenidos y expuestos por Nurzhan Nurseitov (Nurseitov y col., 2009) y Sumaray (Sumaray y Makki, 2012) se decide utilizar JSON como formato de comunicación.

1.6.4 Lenguajes de programación

Un lenguaje de programación es aquel elemento dentro de la informática que permite crear programas mediante un conjunto de instrucciones, operadores y reglas de sintaxis; que

pone a disposición del programador para que este pueda comunicarse con los dispositivos hardware y software existentes. Tiene la capacidad de especificar, de forma precisa, cuáles son los datos que debe trabajar un equipo informático, de qué modo deben ser conservados o transferidos dichos datos y qué instrucciones debe poner en marcha la computadora ante ciertas circunstancias.

Las aplicaciones nativas para la plataforma Android se desarrollan utilizando el lenguaje de programación Java

1.6.4.1 Lenguaje de programación Java

Java es un lenguaje de programación orientado a objetos, desarrollado por *Sun Microsystems* a principios de los años 90. El lenguaje es muy parecido en sintaxis a C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel como la manipulación directa de punteros a memoria. La memoria es gestionada mediante un recolector de basura lo que aísla al programador de la necesidad de hacer liberaciones explícitas de la memoria asignada dinámicamente y, de esta manera, contribuye a eliminar las fugas de memoria que comúnmente afectan a programas desarrollados en C y C++

Una de las características más peculiares de este lenguaje es la posibilidad que ofrece de crear aplicaciones independientes a la plataforma donde va a ser ejecutada. Esta peculiaridad, viene dada por la existencia de una máquina virtual denominada máquina virtual de Java o JVM (*Java Virtual Machine*). Al compilar un programa en Java no se ejecuta un binario directamente sobre el hardware, sino un código intermedio llamado bytecode en ficheros cuya extensión es .class.

La JVM es la encargada de traducir estos bytecodes convirtiéndolos al código particular del hardware utilizado. Para cada hardware debe haber una JVM específica, ya sea un teléfono móvil, un ordenador con Windows, o un equipo electrodoméstico etc.

Para programar es necesario el kit de desarrollo JDK (*Java Develoment Kit*); un conjunto de herramientas como son: un compilador de línea de comandos javac, la máquina virtual de Java con la que se puede ejecutar aplicaciones java, una herramienta de documentación javadoc , y una herramienta para empaquetar proyectos *jar* (Backiel y Broucke, 2015).

1.6.5 Entornos de desarrollo

Un entorno de desarrollo integrado, conocido también como IDE (*Integrated Development Environment*) por sus siglas en inglés, es un programa informático compuesto por un conjunto de herramientas de programación que facilitan el desarrollo de aplicaciones. Un IDE puede denominarse como un entorno de programación, esto significa que consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Los IDE proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación (Muñoz, 2012), (Jiménez, 2016).

1.6.5.1 *Android Studio*

Android Studio es un entorno de desarrollo integrado para la plataforma Android. Fue anunciado el 16 de mayo de 2013 en la conferencia Google I/O (Dobie, 2013), y reemplazó a Eclipse como el IDE oficial para el desarrollo de aplicaciones para Android. La primera versión estable fue publicada en diciembre de 2014.
Está basado en el software *IntelliJ IDEA* de *JetBrains*, y es publicado de forma gratuita a través de la Licencia Apache 2.0. Está disponible para las plataformas *Microsoft Windows*, *Mac OS X* y *GNU/Linux* (Hagos, 2018), (Yener y Dundar, 2016)

1.6.5.2 Eclipse

Eclipse fue desarrollado originalmente por IBM como el sucesor de su familia de herramientas para VisualAge. Eclipse es ahora desarrollado por la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios (Burd, 2004).
Eclipse es un entorno de desarrollo compuesto por herramientas de programación de código abierto multiplataforma. Se le añaden plugins para programar en Java por ejemplo *Java Development Tools, JDT* (Gallardo, Burnette y McGovern, 2003), o , en el caso de Android (*Android Development Tools, ADT*). Bundle ADT contiene los componentes esenciales de desarrollo Android. Diseñado para ampliar las características de Eclipse y permitir configurar y codificar nuevos proyectos Android.

ADT amplía las capacidades de Eclipse para que pueda configurar rápidamente nuevos proyectos de Android, se crea una interfaz de usuario de aplicación, se agregan los paquetes basados en la API de Android, se depuran las aplicaciones utilizando las herramientas del SDK de Android, e incluso permite exportar el archivo *apk* con firma con el fin de distribuir la aplicación.

1.6.5.3 Comparativa

Android Studio al ser el IDE oficial del Google para Android requiere una frecuente actualización por parte de los desarrolladores lo cual es un inconveniente producto a las condiciones de Internet con que se cuenta para llevar a cabo la presente investigación. A pesar los requerimientos *hardware* especificados por los desarrolladores de *Android Studio* se ha comprobado que para un buen desempeño de la herramienta las cuotas necesarias de *hardware* son superiores a las especificadas por los desarrolladores, todo lo contrario a Eclipse el cual necesita cuotas inferiores para un óptimo desempeño. Otro elemento que atenta contra *Android Studio* es el proceso inicial de configuración para comenzar el desarrollo la cual es muy compleja sobre todo por la configuración del *Gradle* en contraste a la Eclipse, la cual en pocos pasos ya se está listo para desarrollar.
Por todo los elementos anteriores se selecciona Eclipse con el plugin ADT como entorno de desarrollo integrado para ser utilizado en la elaboración de la solución.

1.6.6 Android SDK

El Android SDK *Android Software Development Kit*, contiene herramientas necesarias y gratuitas para el desarrollo Android en Eclipse. Combinado con el *Bundle* ADT antes mencionado forman la combinación perfecta para este tipo de desarrollo.
La plataforma integral de desarrollo soportada oficialmente es Eclipse, junto con el complemento ADT (*Android Development Tools plugin*). Aunque también puede utilizarse 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. Además, pueden controlarse dispositivos Android que estén conectados. Contiene emuladores y un depurador bastante potente. Además incorpora la herramienta

Manager, que permite descargar e incorporar librerías de la versión deseada al proyecto, así como todo tipo de documentación.
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 (Ramnath, Crawfis y Sivilotti, 2011), (DiMarzio, 2008).

1.7 Conclusiones parciales del capítulo

Una vez finalizado el presente capítulo, se ha podido arribar a las siguientes conclusiones parciales:

1. El sistema operativo Android es el más utilizado en dispositivos móviles y *tablets*.
2. Para un mejor aprovechamiento de las potenciales de los dispositivos móviles, *tablets* y disminuir los posibles errores de interfaz visual y comportamiento es preferible el desarrollo de aplicaciones nativas antes que aplicaciones web y híbridas.
3. Se optó por utilizar las siguientes tecnologías y herramientas:
   - UML y el *Visual Paradigm* como herramienta CASE para visualizar, construir y documentar los artefactos del sistema.
   - Android como sistema operativo.
   - JSON para el intercambio de datos.
   - Java como lenguaje de programación.
   - Eclipse con el *plugin* ADT como entorno de desarrollo integrado.

   Ya que están mas acordes a las necesidades y condiciones en la que se realiza la presente investigación.

Capítulo 2: Análisis y diseño del sistema

En el presente capítulo se reflejan las actividades realizadas en los procesos de análisis y diseño de la solución propuesta; proceso que será guiado por la metodología de desarrollo seleccionada previamente. En el mismo se realiza el modelo de dominio donde se describen las entidades que intervienen con el objetivo de facilitar la comprensión de los principales conceptos que se utilizarán en el proceso de negocio identificado. Se exponen los artefactos más importantes que describen el flujo normal de eventos que ocurren en el sistema, se realiza una descripción de la solución propuesta, planteándose los requisitos funcionales y no funcionales. Se define la arquitectura que tendrá la solución propuesta.

2.1 Propuesta del sistema

Se propone un sistema llamado Visualizador, utilizando para su desarrollo herramientas libres y que sea capaz de brindar todas las funcionalidades requeridas para la visualización desde dispositivos móviles y tablets con sistema operativo Android la información monitoreada por el sistema ROFLEXIN/LC.

2.2 Modelo de dominio

El modelo de dominio es la representación visual de los conceptos u objetos más importantes de un negocio, sus características y las relaciones entre dichos conceptos. Es el mecanismo fundamental para comprender el dominio del problema y para establecer conceptos comunes. Es un diccionario visual del dominio del problema (Oldfield, 2002).
A continuación se presentan las clases del dominio perteneciente a la solución:
En la monitorización de los procesos, el Usuario interactúa con el Visualizador , el cual solicita los datos al HMIServer y éste le envía la información solicitada. Los datos se pueden visualizar de las siguientes maneras:

- Formato Tabular mediante Sumarios: Pueden ser de Variables.

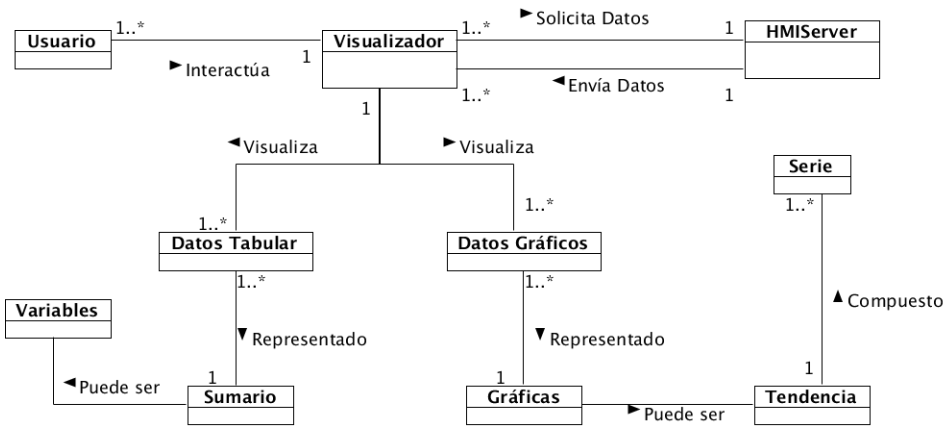


Figura 2.1: Modelo de dominio

- Formato Gráfico mediante Gráficas: Pueden ser de Tendencia.

2.2.1 Descripción de los conceptos del dominio

- **Usuario:** Persona capacitada para interactuar con el Visualizador.
- **Visualizador:**Herramienta encargada de la visualización de los datos.
- **HMIServer:** Herramienta encargada en gestionar las peticiones provenientes del Visualizador.
- **Datos Tabular:** Representación de la información en forma de tuplas.
- **Datos Gráficos:** Representación de la información de forma gráfica.
- **Sumario:** Conjunto de elementos representados en tabla.
- **Variables:** Es la representación de un parámetro medible de un dispositivo de campo al cual está conectado un sensor el cual es el encargado de cuantificar o medir el comportamiento de dicho parámetro en el dispositivo de campo.
- **Gráfica:** Es un tipo de representación de datos, generalmente numéricos, mediante recursos gráficos (líneas, vectores, superficies o símbolos), para que se manifieste visualmente la relación matemática o correlación estadística que guardan entre sí.

- **Tendencia:** Gráfica que muestra un patrón de comportamiento de los elementos de un entorno particular durante un período de tiempo. Es representada en un gráfico bidimensional donde se refleja el comportamiento histórico de una variable adquirida en una de las coordenadas, y el tiempo en la otra.

- **Serie:** Conjunto de mediciones asociadas a una variable. Utilizado para representar el comportamiento de una variable a través del tiempo en una gráfica de tendencia.

2.3 Fase de exploración y planificación

En la fase de exploración y planificación los clientes describen sus necesidades en las Historias de Usuario que son los requisitos funcionales del sistema y establecen las prioridades de cada una. Al mismo tiempo, se define el tiempo de desarrollo de cada Historia de Usuario y se familiariza con las tecnologías, herramientas y prácticas que se utilizarán en el desarrollo del Visualizador. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema. La fase de exploración toma poco tiempo, dependiendo de la capacidad del programador con la tecnología y el alcance del proyecto.

2.3.1 Requisitos funcionales

Los requisitos funcionales son aquellos que describen qué debe hacer el sistema, desde el punto de vista de las necesidades del usuario, son capacidades o condiciones que debe cumplir el sistema y que están fuertemente ligados a las opciones del programa (Cardozzo y Academy, 2016). Los requisitos funcionales (RF) del sistema propuesto se identificaron de acuerdo a las capacidades o condiciones que este debe cumplir.

1. **RF-1:** Iniciar la aplicación.

   a) **RF-1.1:**Comprobar conexión del dispositivo.

   b) **RF-1.2:**Cargar configuración definida para la terminal.

2. **RF-2:** Visualizar datos de la terminal.

3. **RF-3:** Visualizar fecha y hora actual.

4. **RF-4:** Visualizar estado actual de la batería del dispositivo.
5. **RF-5:** Interacción con el módulo de Seguridad.
    a) **RF-5.1:**Autenticar terminal.
    b) **RF-5.2:**Cerrar sesión.
6. **RF-6:** Visualizar sumario de variables.
7. **RF-7:** Visualizar detalles de una variable.
8. **RF-8:** Visualizar comportamiento de una variable en un gráfico de tendencia.
9. **RF-9:** Visualizar gráfico de tendencia.
    a) **RF-9.1:**Adicionar serie a la gráfica.
    b) **RF-9.2:**Eliminar serie de la gráfica.
    c) **RF-9.3:**Mostrar u ocultar serie de la gráfica.
    d) **RF-9.4:**Limpiar gráfico de tendencia.
    e) **RF-9.5:**Invertir los colores de texto y fondo de la gráfica.
    f) **RF-9.6:**Exportar el gráfico a formato de imagen.
    g) **RF-9.7:**Mostrar u ocultar leyenda de las series representadas en la gráfica.
    h) **RF-9.8:**Editar propiedades de una serie.
10. **RF-10:** Visualizar estado actual de conexión del dispositivo.

2.3.2 Requisitos no funcionales

Los requerimientos no funcionales (RFN) son requisitos que imponen restricciones en el diseño o la implementación. Son propiedades o cualidades que el producto debe cumplir. Sin embargo estos requisitos no definen el éxito del producto, pero influyen considerablemente en la evaluación del mismo (Ramos y col., 2017). Teniendo en cuenta las características del sistema se definieron los siguientes requerimientos no funcionales:

1. **RNF-1 Interfaz:**

   a) Clara y concisa. No debe dar lugar a la confusión del usuario y debe seguir los estándares de diseño de interfaces de Google.

   b) Las interfaces visuales del sistema deben ser diseñadas para ser visualizadas en modo *landscape*.

   c) La interfaz de usuario del sistema deberá ser diseñada de forma tal que permita el aprovechamiento del espacio.

   d) En la pantalla principal se tendrá acceso a la mayoría de las funcionalidades con que contará el sistema de manera sencilla y con un solo clic.

   e) Los componentes Android de interfaz de usuario para la entrada de datos, de ser posible, deberán ser configurados de manera que el riesgo de entrada de datos inválidos por parte del usuario disminuya.

2. **RNF-2 Estabilidad:** El sistema debe ser capaz de manejar los errores ocurridos durante la ejecución de la misma y avisando de la naturalidad del error.

3. **RNF-3 Rendimiento:** El sistema debe desempeñar su función de una manera fluida. Se debe buscar la experiencia de uso más agradable para el usuario.

4. **RNF-4 Optimización:** El consumo de batería y de datos debe ser adecuado, y nunca dejar procesos sueltos que consuman memoria y batería. El tiempo de ejecución debe ser mínimo, para mejorar los tiempos de respuesta y la experiencia de uso del usuario.

5. **RNF-5: Usabilidad**

   a) El sistema deberá visualizarse correctamente en cualquier dispositivo Android con las siguientes dimensiones de pantalla: pequeña, normal o grande.

   b) La interfaz visual del sistema debe ser atractiva y sencilla, permitiendo al usuario facilidad de uso y entrenamiento.

   c) El sistema deberá ser soportada en la mayor cantidad posible de versiones del SO Android.

d) El sistema deberá ser capaz de brindar información al usuario mediante mensajes, para ayudarlo y guiarlo durante su interacción con la aplicación.

6. **RNF-6: Ayuda y documentación:** Se brindarán manuales de ayuda que documenten cómo trabajar de forma adecuada con el sistema.

2.3.3 Historias de Usuario

Las Historias de Usuario se escriben en el lenguaje del cliente, representa un requisito que se debe satisfacer con la implementación del sistema y debe ser lo suficientemente sencilla para que el programador pueda saber que va a implementar. Si la Historia de Usuario cuando el cliente la escribe el programador entiende que no es lo completamente sencilla como para implementarla como una funcionalidad, entonces se divide en dos o más Historias de Usuario (Letelier y Penadés, 2012) . A continuación se muestra un ejemplo de las Historias de Usuarios del sistema

HISTORIA DE USO			
Orden	HU_15	**Nombre**	Limpiar gráfico de tendencia
Riesgo	Bajo	**Prioridad**	Media
Iteración	8	**Puntos estimados**	1
Descripción	El sistema debe dar la posibilidad al usuario de limpiar la gráfica de tendencia. Entre los botones presentes en la barra de herramientas de la gráfica de tendencia debe existir uno que permita limpiar o borrar todas las series que han sido adicionadas por el usuario a la gráfica sin importar sin estas están visibles o no en ese momento.		
Observación			

Descripción de los campos que componen las Historias de Usuario:

- **Orden:** Está constituido por dos partes. La primera está referido al nomenclador HU

(Historia de Usuario y la segunda corresponde el número de la funcionalidad que representa.

- **Nombre:** Nombre que identificará a la Historia de Usuario.
- **Riesgo:** Es el grado de incertidumbre en el desarrollo que se asocia a la Historia de Usuario. Determina la posibilidad real de implementarse o no con las condiciones previstas por el equipo de desarrollo (tiempo, recursos, personal). Puede ser Bajo, Medio o Alto.
- **Prioridad:** La prioridad la define el cliente, y es el grado de importancia que le concede a la funcionalidad.
- **Iteración:** Es el número de la fase en la cual se define la Historia de Usuario.
- **Puntos estimados:** Es un número entero que representa la cantidad de semanas que se dispone para el desarrollo de la Historia de Usuario. Las Historias de Usuario con altos puntos estimados deben ser separadas en varias tareas. Un punto es una semana efectiva de desarrollo.
- **Descripción:** Se escribe una fundamentación de lo que hace la funcionalidad.
- **Observación:** Se escribe los elementos o detalles que se deben tener en cuenta para la implementación de la misma.

A partir de la solución propuesta se identificaron 20 requisitos funcionales agrupados en las siguientes Historias de Usuario:

1. Comprobar conexión del dispositivo.	2. Cargar configuración definida para la terminal.
3. Visualizar fecha y hora actual.	4. Visualizar estado actual de la batería del dispositivo.
5. Autenticar terminal.	6. Cerrar sesión.
7. Visualizar datos de la terminal.	8. Visualizar sumario de variables.

9. Visualizar detalles de una variable.	10. Visualizar comportamiento de una variable en un gráfico de tendencia.
11. Visualizar gráfico de tendencia.	12. Adicionar serie a la gráfica de tendencia
13. Eliminar serie de la gráfica de tendencia	14. Mostrar u ocultar serie de la gráfica de tendencia.
15. Limpiar gráfico de tendencia.	16. Invertir los colores de texto y fondo de la gráfica de tendencia.
17. Exportar el gráfico de tendencia a formato de imagen.	18. Mostrar u ocultar leyenda de las series representadas en la gráfica
19. Visualizar estado actual de conexión del dispositivo.	20. Editar propiedades de una serie.

La descripción del resto de las Historias de Usuarios que definen el sistema están adjunta en los anexos del documento.

2.3.4 Estimación de esfuerzo por Historias de Usuario

Las estimaciones de esfuerzo asociado a la implementación de las Historias de Usuario se realizan con el objetivo de lograr una planificación real en el desarrollo del sistema Visualizador y llevar un registro de la velocidad de desarrollo, basándose principalmente en la suma de puntos correspondientes a las Historias de Usuario.

La planificación se puede realizar basándose en el tiempo. La velocidad de desarrollo es utilizada para establecer cuántas Historias de Usuario se pueden implementar antes de una fecha determinada o cuánto tiempo tomará implementar un conjunto de éstas. Se establece como medida el punto estimado. Un punto estimado, equivale a una semana ideal de programación. Las Historias de Usuario generalmente valen de 1 a 3 puntos. Teniendo en cuenta lo expuesto anteriormente la estimación de esfuerzo de las Historias de Usuario queda como se muestra:

Tabla 2.3: Estimación de esfuerzo por Historia de Usuario

Historia de Usuario	Estimación de esfuerzo
Comprobar conexión del dispositivo	1 semana
Cargar configuración definida para la terminal	2 semanas
Visualizar fecha y hora actual	1 semana
Visualizar estado actual de la batería del dispositivo	1 semana
Autenticar terminal	1 semana
Cerrar sesión	1 semana
Visualizar datos de la terminal	1 semana
Visualizar sumario de variables	3 semanas
Visualizar detalles de una variable	2 semanas
Visualizar comportamiento de una variable en un gráfico de tendencia	2 semanas
Visualizar gráfico de tendencia	3 semanas
Adicionar serie a la gráfica de tendencia	1 semana
Eliminar serie de la gráfica de tendencia	1 semana
Mostrar u ocultar serie de la gráfica de tendencia	1 semana
Limpiar gráfico de tendencia	1 semana
Invertir los colores de texto y fondo de la gráfica de tendencia	1 semana
Mostrar u ocultar leyenda de las series representadas en la gráfica	1 semana
Exportar el gráfico de tendencia a formato de imagen	2 semanas
Visualizar estado actual de conexión del dispositivo	1 semana
Editar propiedades de una serie	1 semana
Total de Historias de Usuario: 20	**Total de esfuerzo:** 28 semanas

A partir de la suma de los puntos de estimación de esfuerzo por cada Historia de Usuario, se calcula que el desarrollo del sistema tendrá una duración de 28 semanas.

2.3.5 Plan de iteraciones

Para lograr una mejor organización del trabajo y proporcionar un desarrollo iterativo e incremental, se crea el plan de iteraciones donde se planifica el orden de desarrollo de las Historias de Usuario. Se definió realizar 10 iteraciones, su orden está determinada según las prioridades de las Historias de Usuario y las dependencias existentes entre ellas. La duración total de cada iteración dependerá de los puntos estimados de las Historias de Usuario que en él se desarrollan.

Tabla 2.4: Plan de iteraciones

Iteración	Historia de Usuario
1	Comprobar conexión del dispositivo Cargar configuración definida para la terminal
2	Autenticar terminal Cerrar sesión Visualizar datos de la terminal
3	Visualizar sumario de variables
4	Visualizar gráfico de tendencia
5	Visualizar detalles de una variable Visualizar fecha y hora actual
6	Visualizar comportamiento de una variable en un gráfico de tendencia Visualizar estado actual de la batería del dispositivo
7	Adicionar serie a la gráfica de tendencia Eliminar serie de la gráfica de tendencia Mostrar u ocultar serie de la gráfica de tendencia
8	Limpiar gráfico de tendencia Invertir los colores de texto y fondo de la gráfica de tendencia Mostrar u ocultar leyenda de las series representadas en la gráfica
9	Exportar el gráfico de tendencia a formato de imagen
Continúa en la siguiente página	

Tabla 2.4 Continuación de la página anterior

	Visualizar estado actual de conexión del dispositivo
10	Editar propiedades de una serie

2.3.6 Plan de entrega

El plan de entrega es un documento que especifica con exactitud qué Historias de Usuario serán implementadas en cada entrega del sistema y sus prioridades, de modo que también permita conocer con claridad qué Historias de Usuario serán implementadas en la próxima iteración. Debe ser negociado y elaborado en forma conjunta entre el cliente y el equipo de desarrollado durante las reuniones de planificación de entregas, la idea es hacer entregas frecuentes para obtener una mayor retroalimentación.
A continuación se muestra en el plan de entrega definido para el ciclo de desarrollo.

Tabla 2.5: Plan de entregas

Iteración	Historia de Usuario	Fecha de entrega
1	2	20 de enero del 2018
2	3	10 de febero del 2018
3	1	3 de marzo del 2018
4	1	24 de marzo del 2018
5	2	14 de abril del 2018
6	2	5 de mayo del 2018
7	3	26 de mayo del 2018
8	3	16 de junio del 2018
9	2	7 de julio del 2018
10	1	14 de julio del 2018

2.4 Estimación del costo

Entre los aspectos que no se puede dejar abordar dentro de este capítulo de análisis esta el del análisis económico de la solución propuesta. A continuación se realizará un desglose

del coste de los elementos necesarios en esta investigación. Dichos elementos incluyen costes de personal, de hardware y de software.
La investigación se realizará entre el 3 de enero del 2018 al 14 de julio de 2018 por lo tanto han sido 7 meses de trabajo. Teniendo en cuenta una jornada laboral de 8 horas tendremos un total de 1344 horas de trabajo, distribuidas entre diferentes tareas y diferentes roles profesionales que las llevan a cabo.

2.4.1 Coste de personal

La metodología de software escogida propone un equipo de desarrollo pequeño donde cada integrante tiene su rol y funciones bien definidas. Para determinar el coste del personal involucrado se va desglosar el equipo de acuerdo a la categoría de cada uno así como en la fase donde participa quedando el desglose del coste como se aprecia en la siguiente tabla.

Tabla 2.6: Coste de personal

Fase	Categoría	Horas	Coste/hora	Coste
Documentación	Analista	120	40 CUP	4800 CUP
Análisis	Analista	147	40 CUP	5880 CUP
Diseño	Diseñador	241	30 CUP	7230 CUP
Diseño gráfico	Diseñador gráfico	94	30 CUP	2820 CUP
Implementación	Programador	648	20 CUP	12960 CUP
Pruebas	Programador	94	20 CUP	2820 CUP
Total		**1344**		**36150 CUP**

2.4.2 Coste de hardware

Para el hardware calcularemos el coste según el período de amortización teniendo en cuenta una duración del proyecto de 7 meses. El equipo esta formado por ordenadores y tablet, necesitando uno de cada uno de estos dispositivos para el desarrollo.

Tabla 2.7: Coste de hardware

Equipo	Coste	Coste de amortizado
CPU	657.50 CUP	219.17 CUP
Monitor	154.24 CUP	15.42 CUP
Teclado	4.50 CUP	0.65 CUP
Mouse	3.46 CUP	1.73 CUP
Tablet	213.33 CUP	106.67 CUP
Computadora portatil	500.00 CUP	166.67 CUP
Total	1533.03 CUP	510.31 CUP

2.4.3 Coste de software

En la realización de esta investigación se ha optado por utilizar software libre por lo que no tenemos ningún coste asociado al software.

2.4.4 Coste total

A partir del coste de cada de los elementos necesarios para la investigación se puede llegar al coste total, como se aprecia en la siguiente tabla.

Tabla 2.8: Coste total

Tipo de coste	Total
Coste de personal	36150 CUP
Coste de hardware	510.31 CUP
Coste de software	0 CUP
Total	36660.31 CUP

Por tanto el coste total para la presente investigación asciende a: 36660.31 CUP

2.5 Diagrama de paquetes

Aunque la metodología XP no comprende el trabajo con diagrama de paquetes, se realizó el diagrama de paquetes de la solución para detallar su funcionamiento.
Este diagrama es el encargado de representar las dependencias entre los paquetes que componen un modelo. Es decir, muestra cómo un sistema está dividido en agrupaciones lógicas y las dependencias entre ellas (Jiménez, 2016).
Cuando se crea un proyecto Android utilizando Eclipse se genera automáticamente la estructura de carpetas necesaria para poder generar posteriormente la aplicación. Esta estructura será común a cualquier aplicación, independientemente de su tamaño y complejidad. En la siguiente imagen se muestra dicha estructura.

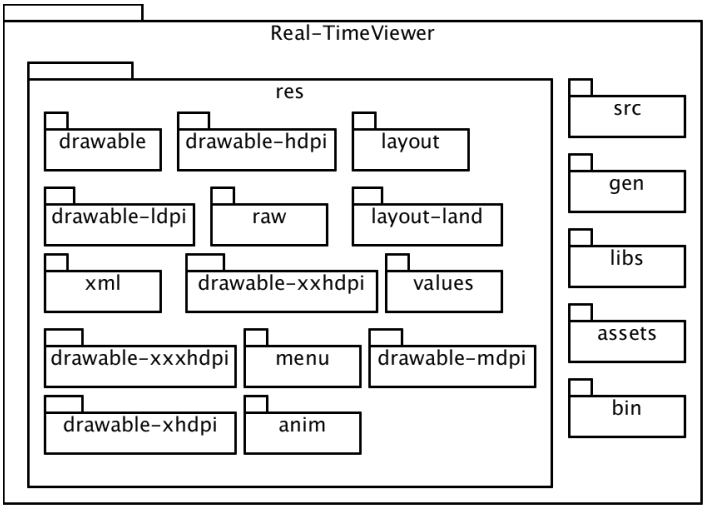


Figura 2.2: Estructura de carpetas de una solución Android con Eclipse

Toda esta estructura es contenida dentro de la carpeta del proyecto que en nuestro caso es *Real-TimeViewer* La misma se organiza de la siguiente manera:

- *gen:* Contiene una serie de elementos de código generados automáticamente al compilar el proyecto. Cada vez que generamos nuestro proyecto, la maquinaria de compilación de Android genera por nosotros una serie de ficheros fuente en java dirigidos al control de los recursos de la aplicación.

- *assets:* Contiene todos los demás ficheros auxiliares necesarios para la aplicación (y

que se incluirán en su propio paquete), como por ejemplo ficheros de configuración, de datos, etc.

- *bin:* Esta carpeta contiene los archivos binarios del proyecto generados a partir de los archivos fuente. Al igual que la carpeta *src*, se mantiene la misma jerarquía de subcarpetas que la representada en el nombre del paquete.
- *libs:* Contiene todas las librerías externas que se utilizarán en el desarrollo del proyecto.
- *res:* Contiene todos los ficheros de recursos necesarios para el proyecto: imágenes, vídeos, cadenas de texto, etc. Los diferentes tipos de recursos de deberán distribuir entre las siguientes carpetas:
  - *drawable:* Contienen las imágenes de la aplicación. Para utilizar diferentes recursos dependiendo de la resolución del dispositivo se suele dividir en varias carpetas como son *drawable-hdpi*, *drawable-ldpi*, *drawable-xxhdpi*, *drawable-xxxhdpi*, *drawable-xhdpi*, y *drawable-mdpi*.
  - *layout:* Contienen los ficheros de definición de las diferentes pantallas de la interfaz gráfica. Para definir distintos layouts dependiendo de la orientación del dispositivo se puede dividir en dos carpetas *layout* y *layout-land*.
  - *anim:* Contiene la definición de las animaciones utilizadas por la aplicación.
  - *menu:* Contiene la definición de los menús de la aplicación.
  - *values:* Contiene otros recursos de la aplicación como por ejemplo cadenas de texto (strings.xml), estilos (styles.xml), colores (colors.xml), etc.
  - *xml:* Contiene los ficheros XML utilizados por la aplicación.
  - *raw:* Contiene recursos adicionales, normalmente en formato distinto a XML, que no se incluyan en el resto de carpetas de recursos.
- *src:* Contiene todo el código fuente del proyecto, código de la interfaz gráfica, clases auxiliares, etc. Inicialmente, Eclipse creará por nosotros el código básico de la pantalla principal de la aplicación, siempre bajo la estructura del paquete java definido.

La estructura del código fuente del sistema está organizado dentro la carpeta *src* tal y como muestra la siguiente figura.

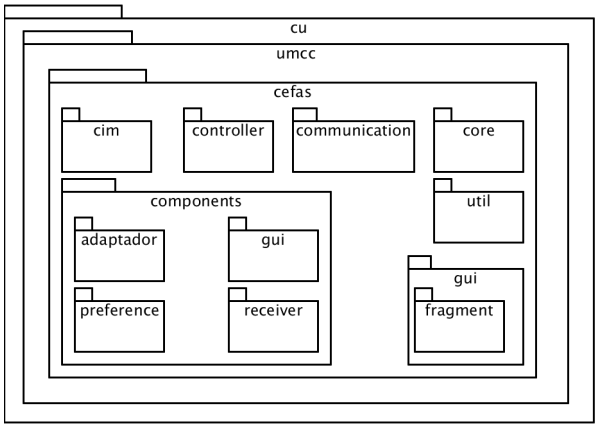


Figura 2.3: Diagrama de paquetes del sistema.

La misma se organiza de la siguiente manera:

- Paquetes *cu*,*umcc* y *cefas*, encierran toda la solución. La primera hace referencia al código del país, mientras la segunda y la tercera hacen referencia a la entidad y dentro de esta el grupo que lleva a cabo el desarrollo del sistema.

- Paquete *cim:* Contiene la entidades que representan el modelo común de información que van compartir el sistema con el servidor.

- Paquete *controller:* Contiene las entidades encargadas de controlar la lógica del negocio del sistema.

- Paquete *communication:* Agrupa las entidades encargadas de establecer comunicación con el servidor, así como de elaborar las peticiones y procesar las respuestas.

- Paquete *core:* Contiene aquellas entidades que sirven de interfaces, los enumerativos que se utilizarán en el sistema.

- Paquete *util:* En este paquete estan aquellas entidades que no persiguen o tienen un objetivo específico sino con sus funcionalidades ayudan a otras a cumplir sus propositos u objetivos.

- Paquete *gui:* Compuesta por entidades que representa las pantallas o interfaces gráficas del sistema y el subpaquete *fragments*.
    - Paquete *fragments:* Entidades que heredan de la entidad *Fragment* propia de Android y representan fragmentos de las interfaces visuales del sistema.
- Paquete *components:* Integrada por cuatro subpaquetes que albergan aquellas entidades que heredan y redefinen comportamientos de entidades propias de Android.
    - Paquete *gui:* Conformado por las entidades que heredan y redefinen el comportamiento de determinados componentes visuales propios de Android.
    - Paquete *receiver:* Almacena las entidades que heredan y redefinen el comportamiento de la clase *BroadcastReceiver* propia de Android.
    - Paquete *preference:* Constituida por las entidades que heredan y redefinen el comportamiento de las clases destinadas a la confección de las preferencias de las aplicaciones desarrolladas con Android.
    - Paquete *adaptador:* Alberga las entidades que heredan y redefinen el comportamiento de las clases *ArrayAdapter* y *BaseExpandableListAdapter* propias de Android.

2.6 Patrones de Arquitectura

La selección de un patrón arquitectónico es una decisión fundamental de diseño en el desarrollo de software ya que provee un conjunto de subsistemas predefinidos, especifica sus responsabilidades e incluye reglas y pautas para la organización de las relaciones entre ellos.

En el desarrollo de un software, se hace necesario seleccionar diferentes patrones los cuales ayudan a que esté presente una buena estructura y organización que hace eficiente su funcionamiento. Estos patrones son una guía para cometer una determinada acción. Especifican un conjunto predefinido de subsistemas con sus responsabilidades y una serie de recomendaciones, para organizar los distintos componentes.

En el caso de la solución propuesta en este documento vamos a partir de que forma parte de una arquitectura cliente servidor tal y como se muestra en la figura.

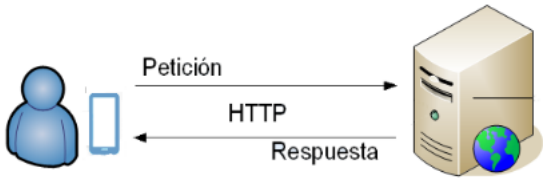


Figura 2.4: Arquitectura cliente servidor

La arquitectura cliente servidor es una arquitectura de aplicación distribuida en la que un prestador de servicio, el servidor, responde a las peticiones de los clientes. Una máquina servidora es una máquina que está ejecutando uno o más programas servidores que comparten sus recursos con los clientes. Un cliente no comparte sus recursos sino que solicita al servidor su contenido o servicio. Por tanto son los clientes quienes inician el proceso de comunicación. En nuestro caso el Visualizador constituirá la parte cliente. El mismo presenta una arquitectura basada en capas.

2.6.1 Arquitectura N-Capas

La arquitectura N-Capas se enfoca en dividir una aplicación en capas separadas que desempeñan diferentes roles y funcionalidades. Su principal objetivo es separar la lógica del negocio de la lógica de diseño. Una aplicación N-Capas tradicional incluye la capa de presentación, la capa de negocios y la capa de datos. Al estar los componentes separados y localizados, es más sencillo darle mantenimiento al software. Además, la capacidad de realizar pruebas mejora considerablemente debido a que las capas solo interactúan entre ellas mediante interfaces bien definidas y es posible añadir implementaciones alternativas a cada capa.

En el caso del sistema el mismo se divide en dos capas tal y como se muestra en la figura. Una primera capa Aplicación donde va estar todo lo referente a la lógica de negocio definida para el sistema, así como las entidades que sirven de apoyo para esto. Mientras en una segunda capa esta Comunicación capa encargada de permitir la comunicación entre las

Figura 2.5: Arquitectura N-Capas en el sistema.

entidades que componen la capa de Aplicación y el servidor HMI, con el uso de interfaces bien definidas en la capa de Comunicación.

2.7 Patrones de diseño

Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software, es decir brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares (Iturralde, 2016). Esto provee las siguientes ventajas: reduce los esfuerzos de desarrollo y mantenimiento, mejora la seguridad informática, eficiencia y consistencia del diseño, y proporciona un considerable ahorro en la inversión. A continuación se describe los patrones de diseño utilizados y como se aplican en el desarrollo de la estructura del sistema.

2.7.1 Instancia Única (*Singleton*)

Proporciona una forma de agrupar el código en una unidad lógica, que se puede acceder a través de una sola variable. Al asegurar que sólo existe un objeto único, se sabe que todo el código hace uso de los mismos recursos globales. El sistema desarrollado hace uso de este patrón específicamente en la clase *ControllerRequest* encargada de controlar las peticiones que se le realiza al servidor (Gamma, 2002).

2.7.2 Cajón de navegación (*Navigation Drawer*)

Con el fin de conseguir una aplicación intuitiva se va a implementar el patrón de diseño *Navigation Drawer*. Es un panel lateral que se desliza desde el borde izquierdo de la pantalla y muestra las principales opciones de navegación de la aplicación (Smyth, 2017).

El *Navigation Drawer* es una vista que actúa como menú para cambiar entre los distintos *Fragments* o pantallas de la aplicación. El *drawer* se oculta automáticamente cuando el usuario está usando la aplicación, por lo que permite a los *Fragments* ocupar el mayor tamaño de pantalla posible.

2.7.3 Modelo Vista Controlador

El patrón de diseño MVC es uno de los patrones existentes más citados. Desde su creación ha jugado un papel influyente en la mayoría de los frameworks de interfaz de usuario y en la manera de pensar acerca del diseño de la interfaz de usuario.
Este patrón define tres componentes principales: el Modelo, la Vista y el Controlador. A continuación se describen brevemente la funcionalidad de cada componente (Fowler, 2012), (Sommerville, 2007):

- **El Modelo:** Maneja el sistema de datos y las operaciones asociadas a dichos datos.
- **La Vista:** Representa la visualización del Modelo. Define y maneja como los datos serán mostrados al usuario.
- **El Controlador:** Maneja las interacciones del usuario, manipula el Modelo y causa la actualización de la Vista.

El *Framework* de Interfaz de Usuario de Android está organizado alrededor del patrón MVC. Este *framework* provee la estructura y herramientas para construir un Controlador que trate las entradas del usuario (ej. toques de pantalla) y una Vista que muestre de forma gráfica información en la pantalla (Gargenta y Nakamura, 2014)

2.7.4 *ViewHolder*

Éste patrón es usado en un caso muy particular en el desarrollo para dispositivos Android: los *ListView*. El patrón *ViewHolder* tiene un único propósito en su implementación: mejorar el desempeño y rendimiento. Y es que cuando se habla de aplicaciones móviles, el rendimiento es un tema de gran importancia (Calvo, 2015).
Lo que hace el *ViewHolder* es mantener una referencia a los elementos del *ListView* mientras el usuario realiza *scrolling* en la aplicación. Así que cada vez que se obtiene la vista de

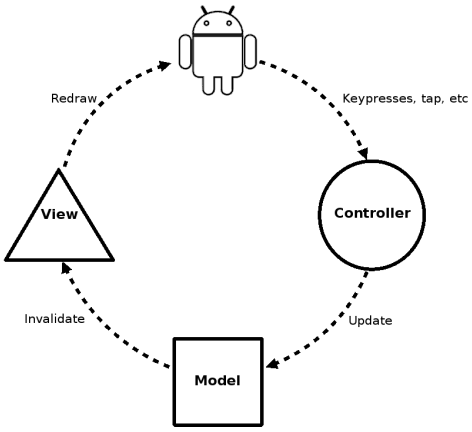


Figura 2.6: Diagrama del MVC en Android (Diagram, 2018)

un *item*, evitamos las frecuentes llamadas a *findViewById*, la cuál se realizaría únicamente la primera vez y el resto llamaría a la referencia en el *ViewHolder*, ahorrando procesamiento. Se evidencia en el sistema en cada vista donde se liste de forma visual diferentes elementos siendo la vista del sumario de variables un ejemplo claro.

2.7.5 Observador (*Observer*)

Es un patrón de diseño de comportamiento que define una dependencia de uno-a-muchos o de uno-a-uno entre objetos, de forma que cuando un objeto cambie de estado se notifique y actualicen automáticamente todos los objetos que dependen de él (Gamma, 2002).
Dentro del sistema es utilizado para que determinadas entidades sean notificadas cuando el usuario realiza determinadas acciones sobre algunos de los componentes visuales que integran las interfaces visuales del sistema. Permite establecer la lógica programable de las interfaces visuales y de los componentes que la integran.

2.7.6 Adaptador (*Adapter*)

Es un patrón de diseño estructural que se encarga de adaptar una interfaz para que pueda ser utilizada por una clase que de otro modo no podría utilizarla. Este patrón es conocido como Envoltorio (*Wrapper*) (Ruiz, 2015), (Gamma, 2002). En el sistema determinadas entidades implementa este patrón para que entidades del modelo del negocio puedan ser visualizadas en componentes gráficos como *ListView*, *GridView*, ect.

2.7.7 Fachada (*Facade*)

Este es un patrón de diseño estructural que provee de una interfaz unificada simple para acceder a una interfaz o grupo de interfaces de un subsistema (Gamma y col., 1994) .
Este patrón es utilizado en el sistema para proveer un único canal de intercambio de datos entre los dos subsistemas o capas que componen el sistema.

2.8 Tareas en segundo plano Android

Uno de los aspectos y funcionalidades más importantes del sistema, es el poder realizar peticiones http a un servidor externo sin que el rendimiento de la aplicación se vea afectado. Es decir, sin que el usuario experimente bloqueos.
Si se desarrolla aplicaciones complejas como tener que realizar una conexión con el servidor y recibir datos, como es el caso del sistema, es obligatorio en las últimas versiones del sistema operativo Android, que esto se haga sin interrumpir el hilo principal, ya que no será permitido, produciéndose múltiples errores de código en el sistema. El hilo principal del usuario es el principal de la aplicación. Normalmente es el único que se utiliza en aplicaciones sencillas, sin embargo, existen aplicaciones complejas en las que se tiene múltiples hilos que se ejecutan a la vez. Como por ejemplo, una en la que utilizamos el hilo principal para moverse por la interfaz de pantalla mientras que, en segundo plano, se están recibiendo datos de un servidor.
Para resolver esto normalmente se hace uso de los *Threads* (hilos). Para ello, se crea un hilo adicional para la tarea y se continua con el hilo principal para el resto de la aplicación. El problema es que la interfaz gráfica (UI) de Android no permite llamadas desde otros hilos que no sea el suyo, por lo que si realiza alguna actualización, como añadir información al sistema se cerrará. Es por eso que es importante analizar que opciones se tiene para solventar dicha situación.

2.8.1 *Threads* (hilos) de java

Mediante el uso de hilos, se pueden crear aplicaciones que responden cuando se está trabajando con procesos que requieren mucho tiempo. En Java se puede crear una clase que

extiende de la clase *Thread* y anula el método *public void run ()*, después se llama al método *start ()* para ejecutar el proceso que consume más tiempo. Si la clase ya extiende de otra clase, se puede implementar la interfaz *Runnable*.
Si se intenta actualizar la interfaz gráfica de usuario desde el *Thread* (no desde el *Thread* principal) la aplicación se bloqueará. Pero se puede resolver si se usa un controlador, u objeto de la clase *Handler*. A este objeto le pasaremos un objeto *Runnable* mediante el método *Handler. post ()*, que lo añade al hilo principal (Oaks y Wong, 2004).

2.8.2 Clase *AsyncTask*

La clase *AsyncTask* permite realizar tareas en *background* y publicar los resultados en la interfaz de usuario sin necesidad de crear hilos y sincronizarlos con la Ul. Extendiendo de *AsyncTask* (tarea asíncrona) se puede crear una clase que tenga entre 1 y 4 métodos según se necesite. Además, la clase AsyncTask dispone de tres tipos de parámetros distintos, que se tiene que especificar cuando se declara la clase (MacLean, Komatineni y Allen, 2015).
public class MiTarea extends AsyncTask<Params, Progress, Result> {}

- **Params:** Datos que se le pasarán al comenzar la tarea.
- **Progress:** Parámetros que se necesita para actualizar la UI.
- **Result:** Dato que se devolverá una vez terminada la tarea.

Si alguno de los parámetros no se necesita, se puede sustituir por *Void*.
Los métodos que se tienen que redefinir en la clase son los siguientes, aunque solo es obligatorio el primero:

1. *protected Result doInBackground(Params... p)* : Este método será el encargado de realizar la tarea en segundo plano. Recibe un número cualquiera de parámetros del tipo Params, así que se debe tratar a p como un arreglo.Este método se ejecuta en otro hilo, por lo que no se puede modificar la UI desde él. Para ello, se debe usar los tres métodos siguientes.

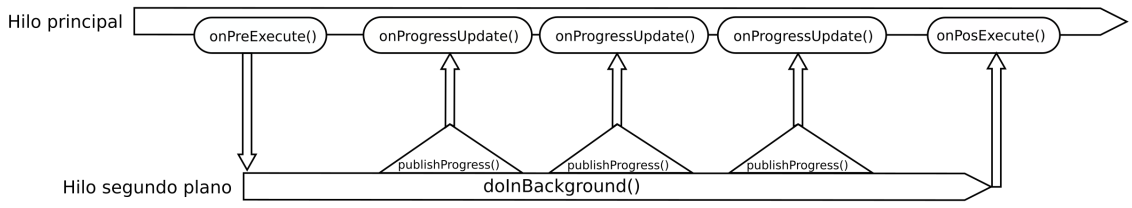


Figura 2.7: Ciclo de vida de un objeto de la clase *AsyncTask* de Android (Invarato, 2013)

2. *protected void onPreExecute()*: Este método se ejecutará antes de *doInBackground*, por el cual se puede modificar la interfaz para indicar el comienzo de la tarea (colocar un cargando, desactivar botones, ect).

3. *protected void onProgressUpdate (Progress ... values)* : Este método permitirá actualizar la interfaz mientras se ejecuta la tarea asíncrona. Para ello, desde *doInBackground* se debe llamar a *publishProgress* y pasarle los parámetros oportunos.

4. *protected void onPostExecute (Result result)*:Este último método se ejecuta tras terminar *doInBackground* y recibe como parámetro lo que este devuelva.

2.9 Tarjetas CRC

Las características más sobresalientes de las tarjetas CRC son su simpleza y adaptabilidad. Se define a una tarjeta CRC como una ficha de papel o cartón que representa a una entidad del sistema, las cuales permiten también que el equipo completo contribuya en la tarea del diseño. Estas tarjetas se utilizan para estructurar las clases y a su vez definir las responsabilidades sobre las mismas, así como la simulación de escenarios en el sistema (Goytia y González, 2014). El nombre de la clase se coloca a modo de título en la tarjeta, las responsabilidades se colocan en la parte izquierda, y las clases que se implican en cada responsabilidad en la parte derecha.

Clase: Es cualquier persona, evento, concepto, pantalla o reporte.

Responsabilidades: Las responsabilidades de una clase son las entidades que conoce y las que realizan sus atributos y métodos.

Colaboradoras: Los colaboradoras de una clase son las demás clases con las que trabaja en conjunto para llevar a cabo sus responsabilidades.
A continuación se muestra unas de las tarjetas CRC obtenidas del sistema

Nombre de Clase: ***Variable***	
Superclase: *Data*	**Subclases:**
Responsibilidades	**Colaboradoras**
Entidad que representa una variable monitoreada por el sistema.	

La descripción del resto de tarjetas CRC que definen el sistema están adjunta en los anexos del documento.

2.10 Conclusiones parciales del capítulo

Una vez finalizado el presente capítulo, se ha podido arribar a las siguientes conclusiones parciales:

1. Se definió el modelo del dominio, el cual refleja el punto de partida de la solución propuesta.

2. Se especificaron además los requisitos del sistema que permitieron identificar las funcionalidades con las que este contará y que darán respuesta a las necesidades del usuario.

3. Se elaboraron los principales artefactos que guían la metodología de desarrollo, lo que permitió definir los aspectos necesarios para el desarrollo del sistema.

4. Se delimitó la arquitectura que tendrá el sistema guiado por el patrón arquitectónico N-Capas, así como los patrones de diseño los cuales posibilitaron definir una mejor arquitectura.

Capítulo 3: Implementación y prueba

En este capítulo se aborda las tareas de implementación a través de las Tareas de Ingeniería. Se describe los estándares de codificación para el desarrollo de la solución propuesta a tener en cuenta, así como la confección de las pruebas aplicadas para comprobar que las Historias de Usuario implementadas estén correctas.

3.1 Tarea de Ingeniería

Las Tareas de Ingeniería son actividades que los programadores conocen que el sistema debe hacer. Deben ser estimables, su tiempo de implementación debe ser corto, aproximadamente entre uno y tres días, su objetivo es resolver las Historias de Usuario. Una Historia de Usuario puede tener una o varias Tareas de Ingeniería en dependencia de la funcionalidad a desarrollar. Pueden existir también tareas de ingeniería técnicas, que son aquellas que aunque no derivan directamente de una Historia de Usuario, es necesaria su consideración para que el sistema funcione.
En la siguiente tabla se muestra el formato utilizado para la confección de las tareas de ingeniería:

Tabla 3.1: Ejemplo de Tarea de Ingeniería.

Tarea de Ingeniería	
Número tarea: 04	**Número de Historia de Usuario:** 19
Nombre tarea: Implementar mecanismo de actualización del estado de conexión del dispositivo.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 3
Fecha inicio: 4 de julio del 2018	**Fecha fin:** 6 de julio del 2018
Continúa en la siguiente página	

Tabla 3.1 Continuación de la página anterior

Programador responsable: Luis Andrés Valido Fajardo
Descripción: Implementar un mecanismo que cada un segundo actualice el estado de conexión del dispositivo y lo muestre gráficamente en la barra de título utilizando el ícono que se corresponde con el estado.

Los campos de la tarjeta de las Tareas de Ingeniería reflejan lo siguiente:

- **Número tarea:** Representa el número por el que se identifica a la tarea. Cada tarea tiene un único número que la identifica.
- **Número Historia de Usuario:** Es el número de la Historia de Usuario a la que responde la tarea.
- **Nombre de tarea:** Define el nombre o funcionalidad concreta a la que se dedica la tarea, debe estar expresado en forma infinitiva.
- **Tipo de tarea:** Información del tipo de tarea a realizar, la misma puede ser:
  - **Desarrollo:** Tarea que se realizará por primera vez.
  - **Corrección:** Tarea que se realiza a partir de una anterior que no se realizó correctamente, es decir, que no pasó los casos de prueba satisfactoriamente.
  - **Mejora:** Tarea que se realiza a partir de una anterior incorporándole nuevos requerimientos.
  - **Otra:** Tarea que no corresponde con una de las anteriores, en este caso es necesario especificar el tipo de tarea o realizar una descripción más profunda de esta.
- **Puntos estimados:** Tiempo de duración de la tarea. El tiempo estimado es reflejado en días. La suma de los puntos estimados de las tareas de ingeniería de una Historia de Usuario no puede superar la cantidad de puntos estimados definidos para la Historia de Usuario.

- **Fecha inicial:** Fecha en la que se inicia el desarrollo de la tarea de ingeniería.
- **Fecha final:** Fecha en la que se concluye el desarrollo de la tarea de ingeniería.
- **Programador responsable:** Nombre del responsable de la realización de la tarea.
- **Descripción:** Es una breve descripción sobre lo que la tarea debe hacer o resolver.

La descripción del resto de Tareas de Ingeniería que definen el sistema están adjunta en los anexos del documento.
Durante la etapa de implementación se realizaron un total de 70 Tareas de Ingeniería, las cuales responden al número de Historias de Usuario implementadas. El resumen de las tareas de ingenierías implementadas por cada Historia de Usuario están adjunta en los anexos del documento.

3.2 Modelo de implementación

Para la composición física de la implementación del sistema se realiza el modelo de implementación, el cual está integrado por un conjunto de componentes, entre los que se encuentran ejecutables y librerías. Este modelo permitió definir una organización del código en términos de los subsistemas de implementación organizados en capas, implementar los elementos de diseño en términos de elementos de implementación, es decir programas ejecutables, finalmente probar y desarrollar los componentes como unidades. Para conformar el modelo de implementación se realizan dos diagramas fundamentales, el diagrama de componentes y el diagrama de despliegue.

3.2.1 Diagrama de Componentes

El diagrama de componentes es otro de los artefactos importantes que incluye el Modelo de Implementación. El mismo muestra elementos del modelo, tales como, los componentes y sus relaciones. Se utiliza para modelar la vista estática del sistema y muestra la organización y las dependencias lógicas entre los componentes de software, sean estos de código fuente, binarios o ejecutables (Jiménez, 2016),(Palomo y Gil, 2014).

- **android:** Es una biblioteca referenciada. Solo es usada durante la compilación.

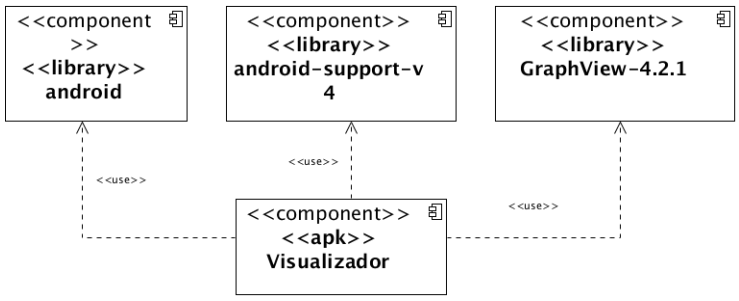


Figura 3.1: Diagrama de componentes del sistema Visualizador.

- **android-support-v4:**Es la biblioteca de soporte para Android que brinda compatibilidad con versiones anteriores para que los desarrolladores utilicen las nuevas características de nivel API en dispositivos móviles que no brindan esa función.

- **GraphView-4.2.1:** Biblioteca de Android para la representación de gráficos de código abierto. Utilizado en la solución para la visualización del comportamiento de una o varias variables en una gráfica de tendencia.

- **Visualizador:** Es un fichero donde se empaqueta el sistema que se propone como solución de esta investigación para su posterior instalación en algún dispositivo con sistema operativo Android.

3.2.2 Diagrama de Despliegue

Con el objetivo de proveer una descripción de la distribución física del sistema se realiza el modelo de despliegue, mediante el cual se muestran las relaciones físicas de los distintos nodos que componen el sistema. Los diagramas de despliegue son los complementos de los diagramas de componentes que, unidos, proporcionan la vista de implementación del sistema. A su vez describen la topología del sistema, es decir la estructura de los elementos de hardware y el software que consumen. Además muestran las relaciones físicas de los distintos nodos que componen un sistema (Jiménez, 2016),(Palomo y Gil, 2014). A continuación se muestra el modelo de despliegue del sistema en donde los nodos representados mediante estereotipos se conectan mediante soportes bidireccionales.

- **TerminalA y TerminalB:** Representa los dispositivos móviles o *tablets* con sistema

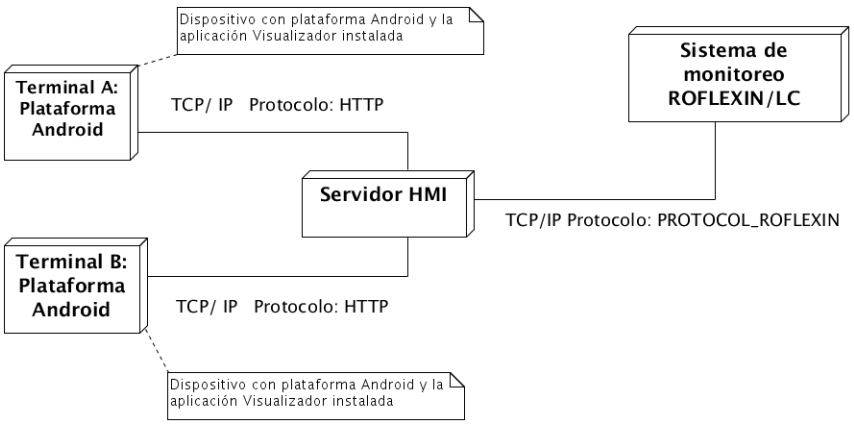


Figura 3.2: Diagrama de despliegue de la solución.

operativo Android donde se encuentra instalado el Visualizador

- **Servidor HMI:** Servidor que le provee la información a los clientes conectados a este de los datos adquiridos por el sistema ROFLEXIN/LC. Los clientes interactúan con este a través de servicios web.

- **Sistema de monitoreo ROFLEXIN/LC:** Sistema encargado de adquirir y almacenar datos provenientes del monitoreo de algún proceso o sistema mecánico

3.3 Estándares de codificación

Las convenciones o estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código. La metodología XP enfatiza la comunicación de los programadores a través del código, con lo cual es necesario que se sigan ciertos estándares de programación. Para la implementación del sistema se siguieron normas y estándares desarrollados, que se relacionan a continuación:

- El código será escrito en inglés, y la documentación en español.

- La identación por bloques será de tres (3) espacios, no se permiten tabulaciones, debido a que su representación depende de la configuración del software empleado visualizar el código.

- Mantener las líneas menores de 80 caracteres, insertar rupturas de línea manuales si es necesario.
- Ninguna función debe contener más de 200 líneas de código fuente.
- Los parámetros que recibe una función su nombre debe comenzar con un carácter *underscore* o guión bajo.

3.4 Pruebas

La metodología XP enfatiza en la realización de pruebas a lo largo de todo el desarrollo del software, con el fin de lograr un producto con calidad, reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su corrección. En este proceso no solo participa el desarrollador, también es importante la colaboración del cliente, sobre todo en las pruebas de aceptación. En XP las pruebas se dividen en dos grupos: pruebas unitarias encargadas de verificar el código y pruebas de aceptación orientadas a probar las funcionalidades del sistema

3.4.1 Pruebas de aceptación

Las pruebas de aceptación son las especificaciones para el comportamiento deseado y la funcionalidad de un sistema. Muestra por una Historia de Usuario dada, cómo el sistema se encarga de ciertas condiciones y con qué tipo de resultados. Los clientes junto a un miembro del equipo de desarrollo son los responsables de verificar que los resultados de estas pruebas sean los correctos para así tomar decisiones acerca de las mismas. Una Historia de Usuario no se puede considerar terminada hasta que no pase las pruebas de aceptación. Es recomendable publicar los resultados de las pruebas de aceptación, para que todo el equipo de desarrollo esté al tanto de esta información. Al sistema además de las pruebas de funcionalidad, se le realizaron Pruebas de Regresión para comprobar que las no conformidades detectadas habían sido resueltas y que no se había afectado otras funcionalidades (Ellingwood, 2017).

3.4.2 Casos de prueba

Los casos de prueba son evidencias de pruebas funcionales o unitarias que se realizan al sistema para comprobar su funcionamiento. Las pruebas funcionales son validaciones escritas desde la perspectiva del cliente, y las pruebas unitarias son validaciones desde la perspectiva del programador. El objetivo general es tener una forma para decirle al cliente que la Historia de Usuario está lista. Las pruebas funcionales o pruebas de aceptación, son las más importantes, ya que representan la medida de satisfacción del cliente para una funcionalidad que el sistema debe tener. Los casos de pruebas fueron definidos para cada Historia de Usuario establecida. Se comprueba el funcionamiento de cada una de las funcionalidades implementadas que responden a la misma. El formato utilizado para la confección de casos de pruebas se muestra a continuación en la tabla.

Tabla 3.2: Ejemplo de los casos de prueba.

Caso de prueba de aceptación	
Código: PA01_HU05	**Historia de Usuario:** HU05
Nombre: Auntenticar terminal con errores I	
Descripción: Se intenterá autenticar la terminal dejando los campos de anfrition y puerto del servidor vacío.	
Condiciones de ejecución: El dispositivo no debe haber iniciado sesión o ternerla cerrada en el servidor.	
Pasos de ejecución: Se selecciona la opción de auntenticar ya sea por el menú de navegación o en la vista principal del sistema. Se presiona el botón Aceptar del cuadro de díalogo con los campos anfitrión y puerto vacíos.	
Resultados esperados: Se debe visualizar un cuadro notificación que alerte de que existen campos vacíos. Se debe señalar de forma gráfica aquellas caja de texto que están vacías.	
Evaluación de la prueba: Satisfactoria	

Campos del caso de prueba

- *Código:* Identificador del caso de prueba. Dividido en dos partes. La primera representa la inicial del artefacto y la segunda representa el número con que se identifica la prueba.
- *Historia de Usuario:* Es el número de la Historia de Usuario a la que responde el caso de prueba.
- *Descripción:* Es una breve descripción del propósito de la prueba.
- *Condiciones de ejecución:* Condiciones especiales que deben tenerse en cuenta para ejecutar el caso de prueba.
- *Entradas / pasos de ejecución:* Entradas o funciones que deben ejecutarse para realizar el caso de prueba.
- *Resultado esperado:* Salida u objetivo que debe cumplir la funcionalidad a la que se le realiza el caso de prueba.
- *Evaluación:* Evaluación de éxito del caso de prueba. Prueba satisfactoria en caso de éxito o prueba insatisfactoria en caso de fallo.

Los casos de prueba son agregados a los artefactos de entrega que se realiza al cliente al terminar cada fase o iteración del proyecto. Las Historias de Usuario con evaluación insatisfactoria, serán corregidas en la próxima iteración a partir de nuevas tareas de ingeniería. Los casos de pruebas realizados al sistema se encuentran adjuntos en los anexos del documento.

3.4.3 Pruebas de compatibilidad

Se asegura que las aplicaciones móviles funcionan como se pretende, con los dispositivos seleccionados, con diferentes tamaños de pantalla, resolución y versión de sistema operativo. Pueden ser de inspección manual o automatizada mediante la captura de pantallas que son posteriormente revisadas por el probador (Naik y Tripathy, 2010). El sistema está

diseñada para funcionar en cualquier *smartphone* o *tablet* con sistema operativo Android 4.2.2 – Jellybean o superior, por tanto, se realizaron pruebas en distintos dispositivos. Los dispositivos utilizados en la pruebas de compatibilidad, así como sus características están detalladas en los anexos del documento.

3.4.4 Pruebas de usabilidad

Las pruebas de usabilidad son un servicio de aseguramiento de calidad que consiste en invitar a profesionales, cuyo perfil se adapta al de su público objetivo, a probar el producto y proporcionar comentarios valiosos sobre su facilidad de uso y eficiencia. El objetivo principal de las pruebas de usabilidad es identificar los problemas de usabilidad, recolectar comentarios pertinentes y mejorar la satisfacción de sus usuarios (García, 2015).
Una de las pruebas de usabilidad más frecuentemente realizada, son las *heurísticas de Nielsen* (Nielsen, 1995). Nielsen, tras analizar varios artículos y libros plantea diez heurísticas que permiten evaluar la usabilidad de un producto. Dichas heurísticas plantea:

1. Visibilidad del estado del sistema. El sistema siempre debe mantener a los usuarios informados sobre lo que está pasando, a través de una retroalimentación adecuada dentro de un tiempo razonable.

2. Correspondencia entre el sistema y el mundo real. El sistema debe hablar el idioma de los usuarios, con palabras, frases y conceptos familiares para el usuario, en lugar de términos orientados al sistema. Haciendo que la información aparezca en un orden natural y lógico.

3. Control del usuario y libertad. Los usuarios suelen elegir funciones del sistema por error y necesitarán una *salida de emergencia* claramente marcada para dejar el estado no deseado sin tener que pasar por un diálogo extendido. La posibilidad de deshacer y rehacer acciones.

4. Consistencia y estándares. Los usuarios no deberían preguntarse si diferentes palabras, situaciones o acciones significan lo mismo. Siga las convenciones de la plataforma.

5. Prevención de errores. Incluso mejor que los buenos mensajes de error, es un diseño cuidadoso que impide que se produzca un problema en primer lugar. Elimine las condiciones propensas a errores o compruébelo y presente a los usuarios una opción de confirmación antes de comprometerse con la acción.

6. Reconocimiento en lugar de recordar. Minimice la carga de memoria del usuario haciendo visibles los objetos, las acciones y las opciones. El usuario no debe tener que recordar la información de una parte del diálogo a otra. Las instrucciones de uso del sistema deben ser visibles o fácilmente recuperables cuando sea apropiado.

7. Flexibilidad y eficiencia de uso. Los aceleradores no vistos por el usuario principiante pueden a menudo acelerar la interacción para el usuario experto, de manera que el sistema pueda atender a usuarios inexpertos y experimentados. Permite a los usuarios adaptar acciones frecuentes.

8. Diseño estético y minimalista. Los diálogos no deben contener información irrelevante o raramente necesaria. Cada unidad extra de información compite con las unidades relevantes de información y disminuye la visibilidad relativa.

9. Ayuda a los usuarios a reconocer, diagnosticar y recuperar errores. Los mensajes de error deben expresarse en lenguaje sencillo (sin códigos), indicar con precisión el problema y sugerir constructivamente una solución.

10. Ayuda y documentación. A pesar de que es mejor si el sistema puede utilizarse sin documentación, puede ser necesario proporcionar ayuda y documentación. Cualquier información de este tipo debe ser fácil de buscar, centrada en la tarea del usuario, enumerar los pasos concretos que se deben llevar a cabo, y no ser demasiado grande.

3.4.5 Pruebas de campo

El sistema debe obtener la información que se encuentra en un servidor por tanto la conectividad es el mayor reto con el que se debe enfrentar. Se sometieron pruebas de este tipo en distintos dispositivos en diferentes zonas de cobertura (Excelente, Buena, Aceptable, Mala) de la red inalámbrica (Wi-Fi).

3.4.6 Pruebas de rendimiento

Unas de las pruebas ejecutadas sobre el sistema corresponden a las de rendimiento. Para que estas pruebas sean más objetivas, se han ejecutado sobre diferentes dispositivos Android (coinciden con los dispositivos utilizados en las pruebas de compatibilidad). Siempre con señal de cobertura Excelente en zonas red inalámbrica (Wi-Fi) y siendo el único dispositivo conectado al servidor.
Para esta prueba se ha medido el tiempo de respuesta de distintas funcionalidades del sistema. A continuación el listado de pruebas realizadas y bajo las condiciones realizadas.

1. Visualización del sumario de variables con un total del 100 variables durante 5 minutos. Esta cantidad de variables es superior a la cantidad de variables máxima que monitorea el sistema ROFLEXIN/LC. Donde el intervalo de actualización de la información que visualiza el sistema se debe hacer cada 500 milísegundos (0.5 segundos)

2. Visualización de la gráfica de tendencia con la graficación del comportamiento de 16 variables durante 5 minutos. Esta cantidad es la máxima que puede ser representada en la gráfica de tendencia por restricciones impuesta por el cliente. Donde el intervalo de actualización de la información que visualiza el sistema se debe hacer cada 500 milísegundos (0.5 segundos)

Se escojen estas pruebas por ser las de mayor trasnferencia y procesamiento de datos entre el servidor y el sistema desarrollado.

3.4.7 Pruebas de satisfacción de usuarios

Otro factor a analizar para saber si el sistema cumple con los requisitos necesarios de cara al cliente es realizar pruebas para determinar el agrado del usuario. La población utilizada para estas pruebas constaba de un equipo de 6 personas que enviaban comentarios y sugerencias para el sistema. Para este cometido se ha realizado una encuesta a estos usuarios, donde se preguntaban sobre las siguientes cuestiones:

1. El sistema es fácil de entender para el usuario.

2. La interfaz del sistema es atractiva y amigable.

3. El tiempo de respuesta es correcto bajo ciertas condiciones.

4. En términos generales, el sistema cumple su cometido y con buen rendimiento.

5. ¿Qué es lo que más le ha gustado de la aplicación?.

    - Interfaz.
    - Facilidad de uso.
    - Funcionalidades.
    - Tiempo de respuesta / Rendimiento

3.5 Conclusiones parciales del capítulo

Una vez finalizado el presente capítulo, se ha podido arribar a las siguientes conclusiones parciales:

1. Se definió todas la tareas de ingenierías para solucionar cada una de las Historias de Usuario identificadas para el desarrollo del sistema.

2. Se determinó la manera en que se implantará el sistema propuesto en un ambiente real.

3. Se definieron las pruebas a que se someterá el software, según la metodología de software escogida en conjunto con otros tipos de pruebas para la validación del sistema desarrollado.

Capítulo 4: Análisis de los resultados

En este capítulo se hará un análisis sobre los resultados obtenidos por las pruebas realizadas para la correcta validación del sistema. De igual forma son abordados aquellos aspectos que presentaron dificultades y las decisiones tomadas para solucionarlos.

4.1 Resultados de las pruebas de aceptación

Como resultado de las pruebas de aceptación Se detectaron un total de 8 no conformidades. A medida que se fue avanzando en las iteraciones disminuyeron el número de no conformidades hasta no quedar ninguna, demostrándose de esta manera que el sistema estaba listo para ser utilizado. En la siguiente figura se resume las no conformidades detectadas en cada una de las iteraciones.

4.2 Resultados de las pruebas de compatibilidad

En los *smartphones* probados se pudo apreciar como el sistema se ajusta de manera automática a las variaciones de tamaño de pantalla y densidad de pixeles, además al instalarse en distintas versiones del sistema operativo la aplicación respondió como se esperaba ya que al instalarla en dispositivos con sistema operativo Android inferior a la versión 5.0 –

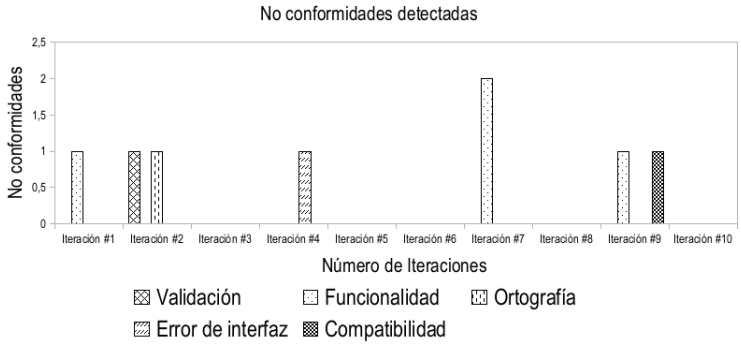


Figura 4.1: Resultados de las no conformidades detectadas

Lollipop, se distingue ligeramente el cambio en la apariencia de los componentes, aunque no afecta directamente la estética del sistema.
En los *tablets* el sistema se comporta de manera debida variando la forma en que se visualiza el contenido, para ajustarse a pantallas con mayor tamaño se utilizaron *layouts* diferentes con el objetivo de aprovechar las ventajas de tener más espacio para mostrar información en dispositivos más grandes. También se probó en versiones distintas de Android para evaluar su comportamiento, el cual fue exitoso.

4.3 Resultados de las pruebas de usabilidad

Para llegar a un o varios resultados que puedan arrojar las pruebas de usuabilidad debemos partir del cumplimiento en el sistema de las diez heurísticas que permiten evaluar la usabilidad de un producto según Nielsen.

1. Este primer aspecto es cumplido por el sistema el cual muestra en todo momento en la barra de título en la esquina derecha la fecha y hora, así como el estado de conexión y de batería actualizado del dispositivo.

2. Los mensajes de error aparecen en el momento que el usuario se está autenticando o adicionando una serie a la gráfica de tendencia en el sistema, estos son concisos y no utilizan palabras técnicas o ambiguas que puedan confundir al usuario del por qué el error.

3. En el sistema no es necesario una salida de emergencia ya que los procesos que pueden tardar en terminar están implementados en hilos diferentes al de ejecución, por lo que el usuario puede navegar libremente por la aplicación sin interrupción.

4. El sistema utiliza componentes estándares a los que los usuarios ya están acostumbrados a interactuar con ellos y a su vez estos responden de manera intuitiva a posibles interacciones. Las palabras utilizadas en botones, etiquetas, títulos y otros componentes guardan relación con la acción asociada.

5. El sistema utiliza pantallas en las que el usuario tiene que escribir en varios campos de texto que pueden contener errores que repercutan de alguna forma en el funcio-

namiento del sistema. Para evitar esto el sistema le hace una propuesta de valor para dichos campos al usuario, así como la restrición de de entrada a la cajas de texto de acuerdo al tipo de valor que estas se van a introducir.

6. El sistema cuenta con varios componentes interactivos, todos están visibles y situados de forma que minimice el riesgo de que el usuario pueda realizar acciones equivocas accidentalmente. En todo momento se indica dónde se encuentra el usuario y la información necesaria en dependencia a la vista en la que se encuentre.

7. Este parámetro no es evaluado ya que el sistema no hace uso de aceleradores, acceso directos o atajos del teclado esto se debe a que los dispositivos para los que fue diseñada no incluyen teclados que puedan facilitar su uso mediante acceso directo a funcionalidades por combinaciones de teclas.

8. El diseño de la aplicación está basado en los principios de *Material Design*, de esta forma se garantiza que el diseño sea minimalista y acorde con la plataforma, en cuanto a la estética se tuvo en cuanto los tamaños de las fuentes utilizadas en correspondencia con la función que el texto desempeña, los controles alineados y uniformes que se adaptan de igual forma para cualquier dispositivo sin importar tamaño o versión del sistema operativo.

9. Los mensajes de error son precisos sin llegar a una formalidad que desinterese a los usuarios, pero tampoco con una informalidad que los incomode; e indicando qué hacer para solucionar el problema.

10. Una documentación o una ayuda cargada de texto para una aplicación móvil puede resultar abrumador para el usuario, por este motivo se optó por una guía rápida que en unas pocas imágenes y texto brinde información sobre el funcionamiento en general del sistema.

4.4 Resultados de las pruebas de campo

Para comprobar cuanto puede influir la conectividad en la inmediatez o actualizacion de la información visualizada en el sistema se realizaron dos pruebas en varios dispositivos

para diferentes niveles de cobertura. En ambas pruebas se hizo enfasís en el tiempo que transcurre entre el momento que el sistema realiza una petición al servidor y el momento en que el sistema recibe la respuesta del servidor. Se toma las peticiones que realiza el sistema para actualizar la información visualizadas en el sumario de variables y el gráfico de tendencia por ser las peticiones de mayor tráfico de información que existe entre el sistema y el servidor. Para esto se considera un sumario que visualiza 100 variables y una gráfica de tendencia con 16 variables. La siguiente tabla muestra los resultados arrojados por las pruebas.

Tabla 4.1: Resultados de las pruebas de campo para los estados Excelente y Buena.

Estado	**Excelente**			**Buena**		
Tiempo(ms)	**Pro**	**Max**	**Min**	**Pro**	**Max**	**Min**
Sum. Var.	58.28	75	27	63.05	429	37
Gráf. Tend.	74.61	111	29	77.72	158	35

Tabla 4.2: Resultados de las pruebas de campo para los estados Aceptable y Mala.

Estado	**Aceptable**			**Mala**		
Tiempo(ms)	**Pro**	**Max**	**Min**	**Pro**	**Max**	**Min**
Sum. Var.	1278.70	64110	88	N.C	N.C	N.C
Gráf. Tend.	3029.21	81076	24	N.C	N.C	N.C

Los resultados mostrados en la anterior tabla muestran como puede influir de forma inversa la conectividad en la inmediatez o actualizacion de la información visualizada en el sistema incluso para un estado de conexión no se logra establecer conexión entre el sistema y el servidor. A una mejor conectividad o estado conexión del dispositivo con la red menor son los tiempos de respuestas del servidor.

Todo lo anterior indica que para poder monitorear un proceso cuyo intervalos de actuali-

zación son pequeños se requiere para un uso adecuado del sistema una conexión a red con estado *Buena* o superior.

4.5 Resultados de las pruebas de rendimiento

Como se mencionó anteriormente fueron escogidas dos pruebas para comprobar el rendimiento del sistema las cuales involucran la mayor trasnferencia y procesamiento de datos entre el servidor y el sistema. Las mismas fueron divididas en varios momentos para medir el tiempo de ejecucción en los diferentes dispositivos. A continuación la explicación detallada de cada una de ellas.

4.5.1 Visualización de 100 variables en el sumario de variables por 5 minutos con intervalos de actualización cada 500 milísegundos

En esta prueba vamos a fragmentarla en varios momentos para medir tiempos de ejecucción. A continuación las descripción de los momentos:

- **Intervalo A:** Tiempo que transcurre entre la conformación de la petición de actualización de la información en la vista del sistema y la llegada de esta a la capa de comunicación del sistema para ser solicitada al servidor.
- **Intervalo B:** Tiempo que transcurre entre el envió de la petición del sistema al servidor y la respuesta de este a la petición.
- **Intervalo C:** Tiempo que transcurre entre la llegada de la respuesta del servidor y el procesamiento de esta.
- **Intervalo D:** Tiempo que transcurre entre el procesamiento de la respuesta del servidor y la actualización de la vista con la nueva información.
- **Intervalo E:** Tiempo que transcurre entre la conformación de la petición de actualización de la información en la vista del sistema y la actualización de la vista con la nueva información.
- **Intervalo F:** Tiempo que transcurre entre la última actualización de la vista y la próxima petición de actualización.

Los resultados obtenidos se muestran el siguiente tabla.

Tabla 4.3: Resultados de la prueba de rendimiento No.1.

Dispositivos	A	B	C	D	E	F
Marca rockchip modelo PLT7050	2.33	46.72	2.33	5.71	57.09	447.39
Marca Allwinner-Tablet modelo X7	18.35	86.87	2.11	6.95	114.28	417.67
Marca MID modelo M721S	2.33	75.31	1.49	5.88	85.01	419.77
Marca zte modelo ZTE V765M fabricante zte	1.40	24.94	1.08	3.14	30.56	473.23
Marca BLU modelo BLU STUDIO 5.0 C fabricante BLU	4.65	52.37	2.85	6.48	66.35	445.20
Marca HUAWEI modelo CPN-L09 fabricante HUAWEI	4.94	43.21	4.66	5.85	58.67	512.31
Marca Samsung modelo Samsumg Galaxy J3 Prime fabricante Samsumg	3.95	78.54	3.49	4.57	90.54	418.42

4.5.2 Visualización de 16 variables en la gráfica de tendencia por 5 minutos con intervalos de actualización cada 500 milísegundos

Para esta prueba se va a proceder de forma similar a la anterior. Se va a fragmentar en varios momentos para medir tiempos de ejecucción. A continuación las descripción de los momentos:

- **Intervalo A:** Tiempo que transcurre entre la conformación de la petición de actualización de la información en el controlador y la llegada de esta a la capa de comunicación del sistema para ser solicitada al servidor.
- **Intervalo B:** Tiempo que transcurre entre el envió de la petición del sistema al servidor y la respuesta de este a la petición.
- **Intervalo C:** Tiempo que transcurre entre la llegada de la respuesta del servidor y el procesamiento de esta.

- **Intervalo D:** Tiempo que transcurre la conformación de la petición de actualización de la información del controlador y la actualización en el controlador con la nueva información.
- **Intervalo E:** Tiempo que transcurre entre la última actualización y la próxima petición de actualización.

Los resultados obtenidos se muestran el siguiente tabla.

Tabla 4.4: Resultados de la prueba de rendimiento No.2.

Dispositivos	A	B	C	D	E
Marca rockchip modelo PLT7050	4.37	68.78	2.44	75.58	432.40
Marca Allwinner-Tablet modelo X7	24.71	91.30	3.34	119.34	427.77
Marca MID modelo M721S	2.62	94.87	2.98	100.48	405.19
Marca zte modelo ZTE V765M fabricante zte	2.91	91.25	2.47	96.63	408.85
Marca BLU modelo BLU STUDIO 5.0 C fabricante BLU	4.76	88.74	2.48	96.07	412.66
Marca HUAWEI modelo CPN-L09 fabricante HUAWEI	5.11	44.68	4.00	53.79	459.30
Marca Samsung modelo Samsumg Galaxy J3 Prime fabricante Samsumg	4.66	42.68	3.91	50.99	458.37

4.6 Resultados de las pruebas de satisfacción de usuarios

Para las cuatro primeras pruebas de satifacción se les solicitó a los usuarios involucrados en estas pruebas que dieran una valoración de 1 a 5. Donde 1 significa estar en desacuerdo, 5 si esta complemetamente de acuerdo.

4.6.1 La aplicación es fácil de entender para el usuario

Tabla 4.5: Resultados de la pregunta 1 de las pruebas de satisfacción de usuarios.

Valoración	Votos	Por ciento
Valoración de 1	0	0.00 %
Valoración de 2	0	0.00 %
Valoración de 3	0	0.00 %
Valoración de 4	3	50.00 %
Valoración de 5	3	50.00 %

Como se puede observar la mayoría de los usuarios que probaron el sistema quedaron satisfechos con la disposición de sus funcionalidades y su facilidad de uso.

4.6.2 La interfaz del sistema es atractiva y amigable

Tabla 4.6: Resultados de la pregunta 2 de las pruebas de satisfacción de usuarios.

Valoración	Votos	Por ciento
Valoración de 1	0	0.00 %
Valoración de 2	0	0.00 %
Valoración de 3	0	0.00 %
Valoración de 4	2	33.33 %
Valoración de 5	4	66.66 %

Como se puede observar, los usuarios consideran el diseño de la interfaz de la aplicación atractiva y amigable. Se puede considerar que el patrón de diseño ofrecido en la aplicación ha gustado entre los usuarios ya que la aplicación parece atractiva y fácil de usar.

4.6.3 El tiempo de respuesta es correcto bajo ciertas condiciones

Tabla 4.7: Resultados de la pregunta 3 de las pruebas de satisfacción de usuarios.

Valoración	Votos	Por ciento
Valoración de 1	0	0.00 %
Valoración de 2	0	0.00 %
Valoración de 3	0	0.00 %
Valoración de 4	1	16.66 %
Valoración de 5	5	83.33 %

En esta pregunta pudimos comprobar como los usuarios consideraban notable el tiempo de respuesta del sistema.

4.6.4 En términos generales, el sistema cumple su cometido y con buen rendimiento

Tabla 4.8: Resultados de la pregunta 4 de las pruebas de satisfacción de usuarios.

Valoración	Votos	Por ciento
Valoración de 1	0	0.00 %
Valoración de 2	0	0.00 %
Valoración de 3	0	0.00 %
Valoración de 4	2	33.33 %
Valoración de 5	4	66.66 %

Observando esta pregunta podemos ver que los usuarios están contentos con la funcionalidad obtenida en la aplicación. Se podría decir que se ha obtenido un rendimiento y una cantidad de funcionalidades que todos los usuarios esperaban. Han valorado positivamente el tiempo de respuesta y el rendimiento por lo tanto podemos concluir que el sistema cumple con los requisitos no funcionales expuestos en la fase de exploración.

4.6.5 ¿Qué es lo que más le ha gustado de la aplicación?

A modo de buscar un punto fuerte del sistema, se preguntó a los usuarios que creían más atractivo del sistema. Esto sirve al equipo de desarrollo para determinar que punto del sistema causa más impacto en los usuarios y mejorarlo.

Tabla 4.9: Resultados de la pregunta 5 de las pruebas de satisfacción de usuarios

Indicador	Votos	Por ciento
Interfaz	0	0.00 %
Facilidad de uso	2	33.33 %
Funcionalidades	3	50.00 %
Tiempo de respuesta / Rendimiento	1	16.66 %

Como se puede ver, las opciones más marcadas en la encuesta fueron las funcionalidades y la facilidad de uso. Una vez más, gracias a la encuesta se pudo observar que se debe trabajar en la interfaz del sistema ya que fue el indicador menos votados.

4.7 Conclusiones parciales del capítulo

Una vez finalizado el presente capítulo, se ha podido arribar a las siguientes conclusiones parciales:

1. Las fallas detectadas por las diferentes pruebas realizadas fueron solucionadas.

2. El sistema Visualizador está listo para ser desplegado y ser utilizado como la interfaz Hombre-Máquina del ambiente de ejecución o visualización del sistema ROFLE-XIN/LC desde dispositivos móviles y *tablets*.

Conclusiones

A través del presente trabajo, se ha podido arribar a las siguientes conclusiones generales:

1. Se definió una metodología de desarrollo para el ambiente de ejecución del HMI del sistema ROFLEXIN/LC, de acuerdo al estado del arte de las principales tecnologías, metodologías y herramientas para el desarrollo de aplicaciones móviles en tiempo real, y se especificaron los requisitos del sistema que permitieron identificar las funcionalidades que dan respuesta a las necesidades del usuario.

2. Se implementó el ambiente de ejecución del HMI del sistema ROFLEXIN/LC, para sistema operativo Android, según los presupuestos y requisitos previamente definidos, haciendo uso de herramientas de software libre y código abierto.

3. Se validó el funcionamiento del ambiente de ejecución del HMI del sistema ROFLEXIN/LC, mediante la realización de las pruebas necesarias, según la metodología XP, y el análisis de los resultados que arrojaron las mismas.

Recomendaciones

Dentro de las posibles mejoras y recomendaciones que se le hacen al sistema obtenido como parte de la solución expuesta en la presente investigación están:

1. Extrapolar esta solución para que sea compatible en otros dispositivos móviles que tengan un sistema operativo diferente Android.

2. Mejorar la interfaz gráfica, con ayuda de uno o varios especialistas con vista a que el sistema tenga un acabado visual más profesional acorde su clasificación.

3. Android es un sistema operativo que se actualiza continuamente, por lo que debería programarse una tarea de mantenimiento para estudiar nuevas versiones y adaptar el sistema, evitando de este modo comportamientos erróneos del sistema en futuras versiones de Android.

4. Cambiar el protocolo HTTP por HTTPS para realizar las peticiones al servidor.

5. Adicionar a la representación gráfica de la variables en las gráficas de tendencia la posibilidad de visualizar los límites de control de la variable, así como en la vista de detalles de la variable.

Referencias Bibliográficas

Academy, I. (2017). *Json for Beginners: Your Guide to Easily Learn Json in 7 Days*. Amazon Digital Services LLC - Kdp Print Us. ISBN: 9781549578458.

Android (2014). *Android*. Inglés. Consultado: 2019-01-19. URL: https://www.android.com.

Backiel, A. y S. vanden Broucke (2015). *Beginning Java Programming: The Object-Oriented Approach*. EBL-Schweitzer. Wiley. ISBN: 9781118739495.

Beck, K. y J. Molina (2002). *Una explicación de la programación extrema: aceptar el cambio*. Programación extrema. Pearson Educación. ISBN: 9788478290550.

Bernauer, T. (2015). *EZ RMC Remote HMI App Application Guide for Android Devices*.

Brahler, S. (2010). "Analysis of the android architecture". En: *Karlsruhe institute for technology* 7.8.

Burd, B. (2004). *Eclipse For Dummies*. –For dummies. Wiley. ISBN: 9780764589447.

Calvo, A. (2015). *Beginning Android Wearables: With Android Wear and Google Glass SDKs*. Apress. ISBN: 9781484205174.

Cardozzo, D. e I. Academy (2016). *Desarrollo de Software: Requisitos, Estimaciones y Análisis. 2 Edición*. CreateSpace Independent Publishing Platform. ISBN: 9781530088614.

CubaDebate (2017). *Lineamientos de la política económica y social del partido y la revolución para el período 2016-2021*.

Diagram, W. (2018). *Mvc Diagram Best Of Model View Controller Mvc Pattern – wiring diagram*. Inglés. Consultado: 2019-01-03. URL: http://pinnacleeventswnc.com/mvc-diagram/mvc-diagram-best-of-model-view-controller-mvc-pattern/.

DiMarzio, J. (2008). *ANDROID A PROGRAMMERS GUIDE*. McGraw-Hill professional. McGraw-Hill Education. ISBN: 9780071599894.

Dobie, A. (2013). *Google I/O 2013 keynote recap*. Ed. por A. Central. Consultado: 2019-01-18. URL: https://www.androidcentral.com/google-io-2013-day-one-recap.

Ellingwood, J. (2017). *An Introduction to Continuous Integration, Delivery, and Deployment*. Ed. por D. Ocean. Consultado: 2019-01-18. URL: https://www.digitalocean.com/community/tutorials/an-introduction-to-continuous-integration-delivery-and-deployment.

Estudiez, B. (2015). *Slicetex Virtual HMI para Android. Manual de Usuario para Android*. SLICETEX ELECTRONICS.

EZAutomation.net (2018). *EZ RMC Remote HMI for Remote Monitoring and Control for iOS and Android*. Inglés. Consultado: 2019-01-10. URL: https://www.ezautomation.net/apps/index.htm.

Fowler, M. (2012). *Patterns of Enterprise Application Architecture: Pattern Enterpr Applica Arch*. Addison-Wesley Signature Series (Fowler). Pearson Education. ISBN: 9780133065213.

Gallardo, D., E. Burnette y R. McGovern (2003). *Eclipse in Action: A Guide for Java Developers*. In Action Series. Manning. ISBN: 9781930110960.

Gamma, E. (2002). *Patrones de diseño: elementos de software orientado a objetos reutilizable*. Addison-Wesley professional computing series. Pearson Educación. ISBN: 9788478290598.

Gamma, E. y col. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional Computing Series. Pearson Education. ISBN: 9780321700698.

García, X. (2015). *UF1843 - Aplicaciones técnicas de usabilidad y accesibilidad en el entorno cliente*. Ediciones Paraninfo, S.A. ISBN: 9788428397810.

Gargenta, M. y M. Nakamura (2014). *Learning Android: Develop Mobile Apps Using Java and Eclipse*. O'Reilly Media. ISBN: 9781449336264.

Google (2012). *Android Developers*. Inglés. Consultado: 2019-01-19. URL: http://developer.android.com.

Goytia, L. y Á. González (2014). *Programación Orientada a Objetos C++ y Java*. Ingeniería y Ciencia Básicas. Grupo Editorial Patria. ISBN: 9786074389333.

Hagos, T. (2018). *Learn Android Studio 3: Efficient Android App Development*. Apress. ISBN: 9781484231562.

Hechavarría, Y. C. (2015). *Sistema para la gestión de la información de los Expedientes Técnicos en el Centro de Informática Industrial*. Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas. La Habana (Cuba). Universidad de las Ciencias Informáticas.

Inc., A. (2014). *Apple*. Inglés. Consultado: 2019-01-19. URL: https://www.apple.com.

Invarato, R. (2013). *AsyncTask en Android*. Ed. por Jarroba.com. Consultado: 2019-01-18. URL: https://jarroba.com/asynctask-en-android/.

Iturralde, O. (2016). *Introducción a Los Patrones de Diseño: Un Enfoque Práctico*. CreateSpace Independent Publishing Platform. ISBN: 9781539619215.

Jaén, A. V. (2017). *Sistema de monitoreo de bajo costo para procesos y sistemas mecánicos.* Tesis de Maestría en Ingeniería Asistida por Computadora. Matanzas (Cuba). Universidad de Matanzas.

Jiménez, J. (2016). *UF2406 - El cliclo de vida del desarrollo de aplicaciones*. Editorial Elearning, S.L.

Jiménez, L. (2018). *Comparativa de desarrollo de aplicaciones móviles*. Ed. por Rubrika. Consultado: 2019-01-18. URL: https://rubrika.es/comparativa-desarrollo-aplicaciones/.

Keller, S. (2016). *Json Book: Easy Learning of Javascript Standard Object Notation*. CreateSpace Independent Publishing Platform. ISBN: 9781541228122.

Kruchten, P. (2004). *The Rational Unified Process: An Introduction*. The Addison-Wesley object technology series. Addison-Wesley. ISBN: 9780321197702.

Letelier, P. y M. C. Penadés (2012). "Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP)". En:

Limited, B. (2012). *BlackBerry Software - Secure UEM, Mobile Productivity and Collaboration*. Inglés. Consultado: 2019-01-21. URL: https://www.blackberry.com.

Llaven, D. (2015). *Sistemas Operativos: Panorama para ingeniería en computación e informática*. Elibro Catedra. Larousse - Grupo Editorial Patria. ISBN: 9786077442677.

MacLean, D., S. Komatineni y G. Allen (2015). *Pro Android 5*. Apress. ISBN: 9781430246817.

Maple Systems, I. (2015). *EasyAccess 2.0 Manual*. Maple Systems, Inc.

Microsoft (2010). *Microsoft - Official Home Page*. Inglés. Consultado: 2019-01-23. URL: https://www.microsoft.com.

Moscaritolo, A. (2017). *El 99.6 % del mercado móvil le pertenece a Android y iOS*. Ed. por P. Latam. Consultado: 2019-01-16. URL: https://latam.pcmag.com/sistemas-operativos-moviles/18490/el-996-del-mercado-movil-le-pertenece-a-android-y-ios.

Muñoz, I., S. Rozas y S. Herrera (2015). *Proyecto Ajedrez Amigo*. Consultado: 2019-01-18. Universidad Tecnológica de Chile. URL: https://slideplayer.es/slide/1675333/.

Muñoz, V. (2012). *Aprendiendo a programar paso a paso con C*. Vicente Javier Eslava Muñoz. ISBN: 9788468610610.

Naik, K. y P. Tripathy (2010). *SOFTWARE TESTING AND QUALITY ASSURANCE: THEORY AND PRACTICE*. Wiley India Pvt. Limited. ISBN: 9788126525935.

Nielsen, J. (1995). "10 usability heuristics for user interface design". En: *Nielsen Norman Group* 1.1.

Nieto, J. G. (2018). *Así es como Android se ha comido el mercado en diez años*. Ed. por X. Móvil. Consultado: 2019-01-17. URL: https://www.xatakamovil.com/sistemas-operativos/asi-como-android-se-ha-comido-mercado-diez-anos.

Niño, J. (2011). *Introducción a los sistemas operativos (Sistemas operativos monopuesto)*. Ciclos Formativos. Editorial Editex. ISBN: 9788490030486.

Nokia (2010). *Nokia Corporation*. Inglés. Consultado: 2019-01-20. URL: https://www.nokia.com.

Nurseitov, N. y col. (2009). "Comparison of JSON and XML data interchange formats: A case study". En: págs. 157-162.

Oaks, S. y H. Wong (2004). *Java Threads*. Nutshell handbooks. O'Reilly Media. ISBN: 9780596007829.

Oldfield, P. (2002). "Domain modelling". En: *Appropriate Process Group*.

Oscar, S. (2013). *Visual Paradigm for Uml*. International Book Market Service Limited. ISBN: 9786139166534.

Palomo, S. y E. Gil (2014). *Aproximación a la ingeniería del software*. INGENIERÍA Y CIENCIAS. Editorial Universitaria Ramón Areces. ISBN: 9788499610931.

Parra, J. (2018). *UF2218 - Desarrollo de un CMS*. Editorial Elearning, S.L.

Pascual, J. A. (2018). *Android vs iPhone: la guerra de los smartphones en cifras*. Ed. por ComputerHoy. Consultado: 2019-01-21. URL: https://computerhoy.com/reportajes/industria/android-vs-iphone-guerra-smartphones-cifras-271447.

Ramnath, R., R. Crawfis y P. Sivilotti (2011). *Android 3 SDK Programming For Dummies*. –For dummies. Wiley. ISBN: 9781118146361.

Ramos, D. y col. (2017). *Curso de Ingeniería de Software: 2ª Edición*. CreateSpace Independent Publishing Platform. ISBN: 9781544132532.

Ruiz, A. (2015). *Mastering Android Application Development*. Packt Publishing. ISBN: 9781785887628.

Rumbaugh, J., G. Booch e I. Jacobson (2007). *El Lenguaje unificado de modelado: manual de referencia*. Addison-Wesley object technology series. Pearson Educación. ISBN: 9788478290871.

Smyth, N. (2017). *Kotlin / Android Studio 3.0 Development Essentials - Android 8 Edition*. CreateSpace Independent Publishing Platform. ISBN: 9781979493956.

Smyth, N. (2011). *iPhone iOS 5 Development Essentials*. ISBN: 9781466337275.

Sommerville, I. (2007). *Software Engineering*. International computer science series Software engineering. Addison-Wesley. ISBN: 9780321313799.

Sumaray, A. y S. K. Makki (2012). "A Comparison of Data Serialization Formats for Optimal Efficiency on a Mobile Platform". En: *Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication*. ICUIMC '12. ISBN: 978-1-4503-1172-4.

Terrera, G. (2014). *Diferencias entre aplicaciones web, nativas e hibridas*. Ed. por TestingBaires. Consultado: 2019-01-18. URL: https://testingbaires.com/

2014/06/18/diferencias-entre-aplicaciones-web-nativas-e-hibridas/.

Tesla, L. (2011). *Product TeslaSCADA2 - Multi-platform SCADA system - teslascada*. Inglés. Consultado: 2019-01-05. URL: http://teslascada.com/products/teslascada2.

Tesla, L. (2017). *TeslaSCADA2 Runtime(Android). User Manual*.

Vlist, E. van der (2002). *XML Schema: The W3C's Object-Oriented Descriptions for XML*. Safari electronic books. O'Reilly Media. ISBN: 9781449315375.

Weintek Labs., I. (2014). *Welcome to Weintek.com*. Inglés. Consultado: 2019-01-24. URL: https://www.weintek.com/globalw/software/easyaccess.aspx.

Yener, M. y O. Dundar (2016). *Expert Android Studio*. Wrox expert. Wiley. ISBN: 9781119089254.

Anexo A: Glosario de términos

API La interfaz de programación de aplicaciones, conocida también por la sigla API del inglés *application programming interface*, es un conjunto de subrutinas, funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Framework Un *framework*, entorno de trabajo o marco de trabajo es un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.

Smartphone: Anglicismo. Teléfono móvil construido sobre una plataforma informática móvil, con capacidad para almacenar datos y realizar actividades asemejándose a una minicomputadora.

HTTP: Protocolo usado para acceder a la Web (www). Se encarga de procesar y dar respuestas a las peticiones para visualizar una página web. Además sirve para el envío de información adicional como el envío de formularios con mensajes, etc.

SCADA: Acrónimo de Supervisory Control And Data Acquisition (Supervisión, Control y Adquisición de Datos) es un software para ordenadoresque 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 calidad, control de producción, almacenamiento de datos) y permite su gestión e intervención.

Sumario: Un sumario se define como un despliegue tipo tabla que permite obtener la información global a cerca de los recursos del sistema. Es una interfaz gráfica que concentra

las condiciones generales de las información visualizada sin referencia gráfica del proceso asociado.

Detalles: Se refiere a un despliegue de tipo ventana que permite obtener la información detallada sobre un dterminado recurso del sistema, como ejemplo las variables. Es una interfaz gráfica que concentra las condiciones particulares de la información visualizada sin referencia gráfica del proceso asociado.

***Branding*:** Es un anglicismo empleado en mercadotecnía que hace referencia al proceso de hacer y construir una marca (en inglés, *brand equity*) mediante la administración estratégica del conjunto total de activos vinculados en forma directa o indirecta al nombre y/o símbolo (logotipo) que identifican a la marca influyendo en el valor de la marca, tanto para el cliente como para la empresa propietaria de la marca. En este proceso se identifica o define los conceptos; supone el desarrollo creativo de una identidad.

***Fragments*:** Es una porción de interface de usuario o vista que se integra en una *activity*. Por lo tanto tendremos la posibilidad de combinar múltiples fragmentos en una sola actividad o incluso reutilizar fragmentos en otras actividades.

Raspberry Pi: Es un ordenador de placa reducida, ordenador de placa única u ordenador de placa simple (SBC) de bajo coste desarrollado en el Reino Unido por la Fundación Raspberry Pi, con el objetivo de estimular la enseñanza de informática en las escuelas.

Anexo B: Historias de Usuarios

A continuación la descripción de cada una de las Historias de Usuarios definidas para el sistema.

HISTORIA DE USO			
Orden	HU_01	**Nombre**	Comprobar conexión del dispositivo
Riesgo	Medio	**Prioridad**	Alta
Iteración	1	**Puntos estimados**	1
Descripción	Una vez iniciado el sistema por usuario el mismo debe comprobar la conexión a alguna red por parte del dispositivo así como el tipo de red de estar conectado donde se ejecuta el sistema. El estado de conexión del dispositivo debe ser informado al usuario mediante un cuadro de notificación emergente con los siguientes posibles textos. ▪ Si el dispositivo no está conectado a ninguna red el mensaje debe ser el siguiente: *Sin conexión a Internet*. ▪ Si el dispositivo está conenctado a una red Wifi el mensaje debe ser el siguiente: *Estás conectado a una red Wi-Fi* . ▪ Si el dispositivo está conenctado a una red creada por otro dispositivo móvil el mensaje debe ser el siguiente: *Estás conectado a una red móvil* .		
Observación			

HISTORIA DE USO			
Orden	HU_02	**Nombre**	Cargar configuración definida para la terminal
Riesgo	Alto	**Prioridad**	Alta
Iteración	1	**Puntos estimados**	2
Descripción	El sistema debe permitir cargar la configuración definida para la terminal desde un archivo con extensión *json*. Una vez instalado el sistema, cada que vez que sea ejecutado el sistema debe listar dentro de los directorios a los cuales tiene acceso todos los ficheros cuya extensión sea *json* para que el usuario escoja cual es el archivo donde esta guardado la configuración del sistema para esta terminal. Una vez que el usuario seleccione un archivo y sea correcta la interpretación del archivo por parte del sistema esta será la configuración por defecto del sistema. El archivo será movido de lugar hacia un directorio que solo el sistema tenga acceso.		
Observación	El procedimiento de listar los ficheros con extensión *json* y visualizarlo para que el usuario lo escoja, solo se hará hasta que el usuario escoja uno que se corresponda con un fichero de configuración. A partir de ese momento ya no realizará esta acción.		

HISTORIA DE USO			
Orden	HU_03	**Nombre**	Visualizar fecha y hora actual
Riesgo	Baja	**Prioridad**	Baja

Iteración	5	**Puntos estimados**	1
Descripción	Una vez iniciado el sistema por usuario el mismo debe visualizar en la barra de título del sistema en su parte derecha superior la fecha y hora con que cuenta el dispositivo. La misma debe actualizarse cada un segundo. Se debe visualizar con el siguiente formato *d-M-Y hh:mm:ss a* donde: ■ *d:* Día. Un numero entre 01 y 31. ■ *M:* Mes. Un numero entre 01 y 12. ■ *Y:* Año. Un numero de cuatro valores. ■ *hh:* Hora. Un numero entre 00 y 12. ■ *minutos:* Minutos. Un numero entre 00 y 59. ■ *segundos:* Segundos. Un numero entre 00 y 59. ■ *a:* Cadena de texto de longitud 2 que indica si es antemeridiano *am* o pasado meridiano *pm*.		
Observación			

HISTORIA DE USO			
Orden	HU_04	**Nombre**	Visualizar estado actual de la batería del dispositivo
Riesgo	Medio	**Prioridad**	Baja

Iteración	6	**Puntos estimados**	1
Descripción	Una vez iniciado el sistema por usuario el mismo debe visualizar en la barra de título del sistema en su parte derecha inferior el estado de la batería del dispositivo. La vizualización de dicha información debe realizarse tanto textual como gráfica. La representación textual de la información va ser mediante un texto que indicará el porciento de carga del dispositivo, por ejemplo *50 %*. En el caso de la representación gráfica será mediante la utilización de una imagen de una batería la cual su indicador de carga cambiará en correspondencia con el nivel de carga del dispositivo. A continuación se muestra posibles representaciones gráficas para cuando la batería esta al 100 % , 50 % y 10 % respectivamente.  De igual forma si el dispositivo estuviera cargandosé la representación gráfica debe reflejarlo como por ejemplo se muestra en la siguiente gráfica para cuando la batería esta al 100 % , 50 % y 10 % respectivamente. 		
Observación			

HISTORIA DE USO			
Orden	HU_05	**Nombre**	Autenticar terminal
Riesgo	Medio	**Prioridad**	Alta
Iteración	2	**Puntos estimados**	1

Descripción	El sistema debe brindar la funcionalidad de auntenticar el dispositivo como cliente de un servidor HMI. El sistema debe representar dicha funcionalidad al usuario mediante algún elemento gráfico. Una vez que el usuario interactue con dicho elemento, digase una acción de tocar el elemento el sistema debe responder de la siguiente manera: 1. El sistema comprueba que el dispositivo se encuentra conectado alguna red, de no estarlo lo notifica mediante un cuadro emergente y finaliza el procedimiento sino realiza el siguiente paso. 2. El sistema visualiza un cuadro de díalogo que en su diseño debe contener dos cajas de entrada de texto y dos botones. La primera caja de entrada de texto es para que el usuario entre la dirección IP de la estación de trabajo donde se encuentra alojado el servidor al cual desea conectarse. La segunda caja de entrada de texto es para que el usuario defina el puerto por el cual escucha el servidor. Los botones son para darle la opción de cancelar la operación o de continuar con la misma. Los campos de entrada texto deben mostrar los valores definidos por el usuario en las preferencias del sistema si este así lo definió en dicha vista. Sino los campos deberán vizualizarse vacíos. *a*) Si presiona el botón *Cancelar* finaliza el procedimiento. *b*) Si presiona el botón *Aceptar* el sistema debe tomar los datos entrados por el usuario y validarlo. Sin son incorrectos notificarlo al usuario mediante un cuadro emergente y detener el procedimiento hasta que el usuario corrija los errores o cancele la operación. Si son correctos los datos se le envía la petición de autenticación al servidor. 3. Una vez enviada la petición al servidor el sistema debe visualizar la respuesta de este mediante un cuadro emergente.

Observación	El sistema no puede permitir que el usuario se autentique dos veces concecutivas. Para que el usuario pueda autenticar nuevamente la terminal en el servidor debe previamente cerrar sesión (Ver HU_06). Una vez ejecutada con exito la funcionalidad, esta deja de estar disponible en el sistema hasta que no se ejecute con exito la funcionalidad de cerrar sesión (Ver HU_06).

HISTORIA DE USO			
Orden	HU_06	**Nombre**	Cerrar sesión
Riesgo	Medio	**Prioridad**	Alta
Iteración	2	**Puntos estimados**	1

Descripción	El sistema debe brindar la funcionalidad de auntenticar el dispositivo como cliente de un servidor HMI. El sistema debe representar dicha funcionalidad al usuario mediante algún elemento gráfico. Una vez que el usuario interactue con dicho elemento, digase una acción de tocar el elemento el sistema debe responder de la siguiente manera 1. El sistema debe comprobar que el dispositivo esta conectado alguna red. De no estarlo debe ser notificado al usuario mediante un cuadro emergente. 2. De estar conectado el sistema toma los datos del servidor HMI al cual el sistema se auntenticó como cliente y le envía la petición de cierre de sesión de cliente. 3. El sistema visualiza la pantalla principal del sistema. 4. El sistema visualiza la repuesta del servidor mediante un cuadro emergente.
Observación	Esta funcionalidad solo puede estar disponible en el sistema una vez que se haya realizado con exito la funcionalidad de autenticar la terminal (Ver HU_05). Una vez ejecutado la funcionalidad con exito la misma deja de estar disponible en el sistema. En caso que se detenga el sistema por el cierre del mismo por parte del usuario y el sistema este auntenticado en un servidor se debe realizar la funcionalidad antes de cerrar el sistema completamente .

HISTORIA DE USO

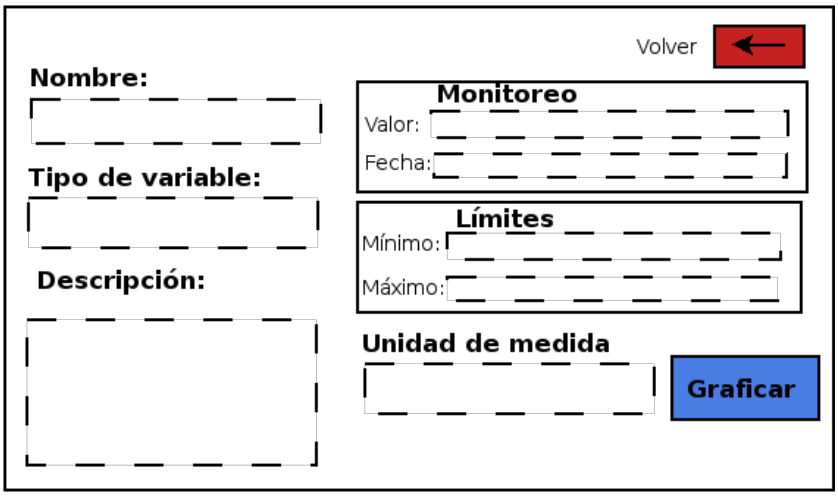
Orden	HU_07	**Nombre**	Visualizar datos de la terminal
Riesgo	Bajo	**Prioridad**	Baja
Iteración	2	**Puntos estimados**	1
Descripción	Una vez iniciado el sistema por usuario el mismo debe visualizar los datos del dispositivo sobre el cual esta siendo ejecutado. El único dato que debe visualizar es la dirección MAC del dispositivo. En caso de no determinar dicho dato porque el dispositivo no cuenta con una tarjeta de red que defina dicho parámetro en el dispositivo el valor mostrado debe ser *hh:hh:hh:hh:hh:hh* el cual es un valor inválido para este parámetro.		
Observación			

HISTORIA DE USO			
Orden	HU_08	**Nombre**	Visualizar sumario de variables
Riesgo	Alto	**Prioridad**	Alta
Iteración	3	**Puntos estimados**	3

Descripción	El sistema debe ser capaz de visualizar todas o algunas de las variables que participan en el proceso que se monitorea con el sistema ROFLEXIN/LC de acuerdo a la configuración inicial definida para la terminal (Ver HU_02) de forma tabular donde cada tupla hará referencia una variable. De cada variable se visualizará en la tupla los siguientes datos: nombre, valor de la última medición que debe actualizarse cada un cierto intervalo de tiempo de acuerdo a la configuración, la unidad de medida definida para la variable, la fecha y hora del ultimo valor tomado por la variable y los primeros 20 carácteres de la descripción de la variable. La distribución de dichos datos en la tupla se hará de acuerdo al siguiente diagrama:  Donde: 1. Visualizará el nombre de la variable. 2. Visualizará la descripción de la variable. Si este dato tiene un longitud mayor de 20 carácteres se visualizarán los primeros 17 mas tres sucesivos concatenados, sino se visualizará la cadena completa. 3. Visualizará el último valor que va tomando la variable concatenada con la unidad de medida de la variable. 4. Visualizará la fecha y hora del último valor tomado por la variable. El color de fondo (*background*) de cada tupla será negro mientras el color de texto (*foreground*) será verde. El sumario debe brindar la posibilidad de desplegar un menú contextual una vez que el usuario seleccione una tupla con las funcionalidad de ver los detalles de la variable visualizada en la tupla selecionada o ver el comportamiento de la variable en un gráfico de tendencia.

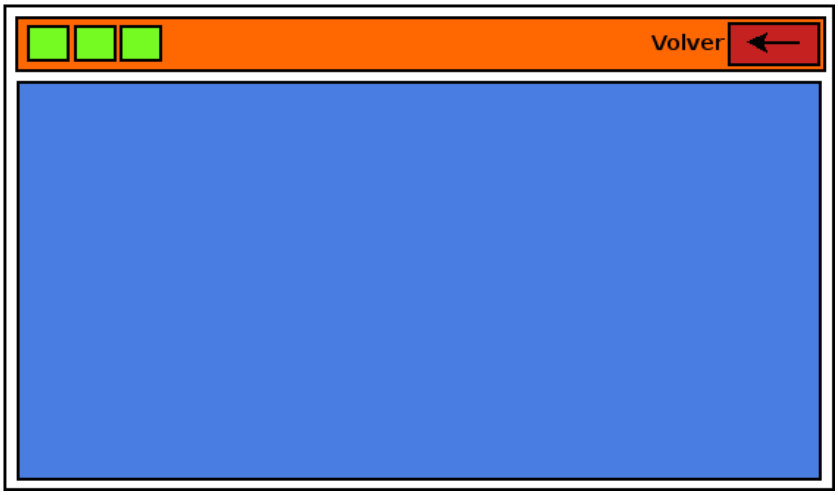
Observación	El usuario solo podrá acceder a la vista de esta funcionalidad una vez que se haya iniciado sesión correctamente del cliente móvil en el servidor.

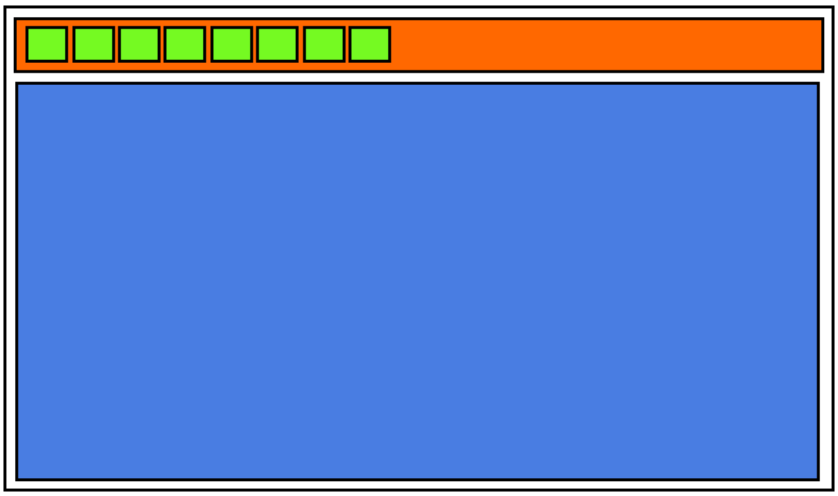
HISTORIA DE USO			
Orden	HU_09	**Nombre**	Visualizar detalles de una variable
Riesgo	Medio	**Prioridad**	Alta
Iteración	5	**Puntos estimados**	2

Descripción	El sistema debe brindar la funcionalidad de visualizar todos los datos referentes a una variable que interviene en el proceso que es monitoreado al usuario. Es por eso que el sistema debe contar con despliegue de tipo ventana que permita obtener toda la información actualizada en cada instante de tiempo de una determinada variable. El diseño de la ventana es como se muestra en el siguiente esquema:  Donde en los recuadros representados por líneas discontinuas será visualizado el atributo referente actualizado de la variable. La ventana debe contar con la funcionalidad de retornar al sumario de variables tal y se muestra en el diagrama (recuadro rojo) de igual forma contar con algún elemento visual como un botón que le permita al usuario visualizar el comportamiento de la variable en una gráfica de tendencia.
Observación	Esta funcionalidad solo es accesible al usuario desde un menú contextual que se visualiza una vez que el usuario seleciona una tupla del sumario de variables.

HISTORIA DE USO

Orden	HU_10	**Nombre**	Visualizar comportamiento de una variable en un gráfico de tendencia.
Riesgo	Alta	**Prioridad**	Alta
Iteración	6	**Puntos estimados**	2

Descripción	El sistema debe brindar la funcionalidad de visualizar el comportamiento de una variable en una gráfica de tendencia. Dicha vista debe tener el siguiente diseño visual:  La cual se divide en dos áreas fundamentales, el recuadro azul destinada a la visualización del comportamiento de la variable en una gráfica de tendencia. Mientras la otra área (recuadro naranja) representa la barra de herramientas de la vista donde el usuario podrá acceder a la funcionalidades de mostrar y ocultar leyenda, invertir los colores de fondo y texto y salvar en un formato de imagen el estado de la gráfica(recuadro verdes). En dicha barra de herramienta se dará la posibilidad de volver al sumario de variables o a la vista de detalles de una variable (recuadro rojo) en dependencia de donde fue invocada dicha vista.
Observación	Esta funcionalidad solo es accesible al usuario desde un menú contextual que se visualiza una vez que el usuario seleciona una tupla del sumario de variables o desde la vista de detalles de una variable. El rango de tiempo que se debe visualizar en la gráfica va estar en dependencia de la configuración hecha para el sistema.

<table>
<tr><th colspan="4">HISTORIA DE USO</th></tr>
<tr><td>Orden</td><td>HU_11</td><td>Nombre</td><td>Visualizar gráfico de tendencia</td></tr>
<tr><td>Riesgo</td><td>Alto</td><td>Prioridad</td><td>Alta</td></tr>
<tr><td>Iteración</td><td>4</td><td>Puntos estimados</td><td>3</td></tr>
<tr><td>Descripción</td><td colspan="3">El sistema debe brindar la funcionalidad de visualizar el comportamiento de varias variables en una gráfica de tendencia. La vista debe tener el siguiente diseño visual:
 
La cual se divide en dos áreas fundamentales, el recuadro azul destinada a la visualización del comportamiento de la variable en una gráfica de tendencia. Mientras la otra área (recuadro naranja) representa la barra de herramientas de la vista donde el usuario podrá acceder a la funcionalidades adicionar,editar, mostrar/ocultar y eliminar serie, mostrar y ocultar leyenda, limpiar la gráfica de tendencia, invertir los colores de fondo y texto y salvar en un formato de imagen el estado de la gráfica (recuadro verdes).</td></tr>
</table>

Observación	El rango de tiempo que se debe visualizar en la gráfica va estar en dependencia de la configuración hecha para el sistema.

HISTORIA DE USO			
Orden	HU_12	**Nombre**	Adicionar serie a la gráfica de tendencia
Riesgo	Medio	**Prioridad**	Alta
Iteración	7	**Puntos estimados**	1
Descripción	El sistema debe dar la posibilidad al usuario de adicionar una variable a la gráfica de tendencia para visualizar el comportamiento de esta a través de una serie. En la barra de herramienta de la vista de tendencia debe existir algún elemento visual que le permita al usuario acceder a dicha funcionalidad. Una vez que el usuario acceda a dicha funcionalidad el sistema debe desplegar un cuadro de diálogo donde el usuario escogerá la variable, la etiqueta y el color que le serán asociadas a la serie que se representará en la gráfica de tendencia. Una vez que el usuario confirme los datos y los mismo estén correcto. El sistema adiciona la serie a la gráfica en la cual se debe comenzar visualizar con el color escogido.		
Observación	Esta funcionalidad solo podra ser efectiva si el usuario ya inicio sesión en el servidor. En caso de que el usuario no tenga la sesión iniciada en el servidor el sistema debe notificarselo a través de un cuadro de notificación emergente. El sistema no puede adicionar una serie que tenga asociada una variable que ya este asociada a otra serie que esta siendo representada en la gráfica de tendencia.		

HISTORIA DE USO

Orden	HU_13	**Nombre**	Eliminar serie de la gráfica de tendencia
Riesgo	Medio	**Prioridad**	Alta
Iteración	7	**Puntos estimados**	1
Descripción	El sistema debe dar la posibilidad al usuario de eliminar una o algunas de las variables que su comportamiento están siendo visualizadas en la gráfica de tendencia como una serie. Para esto en la barra de herramientas de la vista de la gráfica de tendencia debe existir algún elemento visual que le permita al usuario acceder a esta funcionalidad. Una vez que el usuario presione el elemento visual que representa dicha funcionalidad el sistema debe visualizar en cuadro de diálogo una lista de la variables que están siendo visualizadas en la gráfica. Una vez que el usuario seleccione la variables que desea eliminar y confirme la operación el sistema eliminará de la gráfica de tendencia aquellas series que se relacionan con las variables seleccionadas.		
Observación	De no existir ninguna variable previamente adicionada como serie a la gráfica el sistema no ejecuta la acción y notifica al usuario mediante un cuadro de notificación emergente de dicha situación. Las variables cuyas series están ocultas en ese momento también deben ser listadas para su posible eliminación.		

HISTORIA DE USO			
Orden	HU_14	**Nombre**	Mostrar u ocultar serie de la gráfica
Riesgo	Medio	**Prioridad**	Media

Iteración	7	**Puntos estimados**	1
Descripción	El sistema debe dar la posibilidad al usuario de ocultar o mostrar una o algunas de las variables que su comportamiento están siendo visualizadas en la gráfica de tendencia como una serie. Para esto en la barra de herramientas de la vista de la gráfica de tendencia debe existir algún elemento visual que le permita al usuario acceder a esta funcionalidad. Una vez que el usuario presione el elemento visual que representa dicha funcionalidad el sistema debe visualizar en cuadro de diálogo una lista de la variables que han sido adicionada en la gráfica como serie. Aquellas variables que seleccione el usuario serán las permanecerán visibles en las gráfica mientras las no estén selecionadas se ocultarán.		
Observación			

HISTORIA DE USO			
Orden	HU_15	**Nombre**	Limpiar gráfico de tendencia
Riesgo	Bajo	**Prioridad**	Media
Iteración	8	**Puntos estimados**	1
Descripción	El sistema debe dar la posibilidad al usuario de limpiar la gráfica de tendencia. Entre los botones presentes en la barra de herramientas de la gráfica de tendencia debe existir uno que permita limpiar o borrar todas las series que han sido adicionadas por el usuario a la gráfica sin importar sin estas están visibles o no en ese momento.		
Observación			

HISTORIA DE USO			
Orden	HU_16	**Nombre**	Invertir los colores de texto y fondo de la gráfica de tendencia.
Riesgo	Bajo	**Prioridad**	Baja
Iteración	8	**Puntos estimados**	1
Descripción	El sistema debe dar la posibilidad al usuario de invertir los colores de texto y fondo de la gráfica de tendencia. Entre los botones presentes en la barra de herramientas de la gráfica de tendencia debe existir uno que permita invertir los colores de texto y fondo de la gráfica de tendencia. Los colores de texto y fondo por defecto de la gráfica de tendencia cada vez que sea visualizada es blanco y negro respectivamente. Cada vez que el usuario presiones el botón correspondiente a la funcionalidad estos colores deben intercambiarse, donde es utilizado el negro se utilizará el blanco y donde es utilizado blanco se utilizará el negro. Tener en cuenta que dicha funcionalidad puede ser utilizada varias veces durante la visualización de la gráfica de tendencia. Lo que significa que si es utilizada una cantidad impar de veces por el usuario la gráfica debe visualizarse con el texto en color negro y el fondo color blanco. Mientras si es utilizada una cantidad par de veces debe visualizarse con el texto de color blanco y el fondo color negro.		
Observación			

HISTORIA DE USO			
Orden	HU_17	**Nombre**	Exportar el gráfico de tendencia a formato de imagen.

Riesgo	Alto	**Prioridad**	Media
Iteración	9	**Puntos estimados**	2
Descripción	El sistema debe dar la posibilidad al usuario de exportar el gráfico de tendencia en su estado actual a un formato de imagen. Entre los botones presentes en la barra de herramientas de la gráfica de tendencia debe existir uno que permita exportar el gráfico de tendencia en su estado actual a un formato de imagen (de preferencia jpg o png). La imagen exportada debe ser guardada en un fichero con el el siguiente formato como nombre *Visualizador_trend_<HH-mm-ss-S dd-MM-yyyy>.extensión* en el directorio externo del dispositivo. Una vez concluida la operación se le debe notificar al usuario el directorio donde fue salvada la imagen mediante un cuadro emergente.		
Observación			

HISTORIA DE USO			
Orden	HU_18	**Nombre**	Mostrar u ocultar leyenda de las series representadas en la gráfica.
Riesgo	Medio	**Prioridad**	Alta
Iteración	8	**Puntos estimados**	1

Descripción	El sistema debe dar la posibilidad al usuario de mostrar u ocultar la leyenda de las series representadas en la gráficas de tendencia. Entre los botones presentes en la barra de herramientas de la gráfica de tendencia debe existir uno que permita al usuario visualizar u ocultar según sea el caso una leyenda con las series que están siendo visualizadas en ese momento en la gráfica de tendencia. Por defecto dicha leyenda estará oculta. Su comportamiento es el siguiente cada vez que el usuario presione el elemento gráfico para acceder a la funcionalidad la leyenda va a cambiar su estado de visibilidad de oculta-visible o visible-oculta según sea el caso, siendo su estado inicial el de oculta. Se va a visualizar en un cuadro o dialogo emergente el cual debe ser semitransparente su fondo y ubicarse en una de las esquina de la pantalla (de preferencia en la esquina izquierda superior o en la esquina izquierda inferior). En la leyenda aparecerá el nombre definido para la serie, el valor actual de la misma y el color con que es representada la misma en el gráfico.
Observación	En caso de que no exista ningun serie adicionada al gráfico o las series esten ocultas en ese momento no se muestra la leyenda y se notifica al usuario mediante un cuadro emergente.

HISTORIA DE USO			
Orden	HU_19	**Nombre**	Visualizar estado actual de conexión del dispositivo
Riesgo	Medio	**Prioridad**	Baja
Iteración	9	**Puntos estimados**	1

Descripción	El sistema debe ser capaz de monitorear constamente la conexión del dispositivo y notificar el estado de dicha conexión de forma gráfica. En la barra de título del sistema en la parte inferior derecha se debe mostrar a través de un ícono el estado de dicha conexión. Cada vez que exista un cambio se debe notificar a través de un cuadro emergente al usuario de dicho cambio.
Observación	

HISTORIA DE USO			
Orden	HU_20	**Nombre**	Editar propiedades de una serie
Riesgo	Bajo	**Prioridad**	Baja
Iteración	10	**Puntos estimados**	1
Descripción	El sistema debe dar la posibilidad al usuario de editar las propiedades de una serie que ha sido adicionada a la gráfica de tendencia. Entre los botones presentes en la barra de herramientas de la gráfica de tendencia debe existir uno que permita editar las propiedates (color con que es visualizada y etiqueta) de una serie adicionada a la gráfica de tendencia.Una vez que el usuario acceda a dicha funcionalidad el sistema debe desplegar un cuadro de diálogo donde el usuario escogerá la variable asociada a la serie que editará. Una vez seleccionada la variable en los campos referentes al color y etiqueta de la serie aparecerán los valores que tienen dichos atributos para la modificación de los mismo por parte del usuario. Una vez hecha la modificación por parte del usuario y el mismo confirme el cambio el sistema debe validar y actualizar dicha serie en caso de que todo este correcto sino mostrarle una notificación al usuario del error cometido en la edición.		

Observación	De no existir ninguna variable previamente adicionada como serie a la gráfica el sistema no ejecuta la acción y notifica al usuario mediante un cuadro de notificación emergente de dicha situación. Las variables cuyas series están ocultas en ese momento también deben ser listadas para su posible edición.

Anexo C: Tarjetas CRC

A continuación la descripción de cada una de las tarjetas CRC definidas para el sistema. Entidades pertenecientes al paquete *cim*.

Nombre de Clase: ***Data***	
Superclase: *Object*	**Subclases:** *Measurement*,*Serie*,*Variable*
Responsibilidades	**Colaboradoras**
Entidad genérica para representar cualquier dato que gestione el sistema.	

Nombre de Clase: ***Item***	
Superclase: *Object*	**Subclases:**
Responsibilidades	**Colaboradoras**
Entidad modelo que representa un elemento de acceso a las funcionalidades, visualizado lo mismo en el menú de navegación o en la ventana principal.	

Nombre de Clase: ***Measurement***	
Superclase: *Data*	**Subclases:**
Responsibilidades	**Colaboradoras**
Entidad encargada de representar una medición que esta asociada a una variable monitoreada por el sistema.	

Nombre de Clase: ***Serie***

Superclase: *Data*	**Subclases:**
Responsibilidades	**Colaboradoras**
Entidad que representa una serie visualizada en la gráfica de tendencia y que esta asociada a una variable monitoreada por el sistema.	

Nombre de Clase: ***Variable***	
Superclase: *Data*	**Subclases:**
Responsibilidades	**Colaboradoras**
Entidad que representa una variable monitoreada por el sistema.	

Entidades pertenecientes al paquete *communication*.

Nombre de Clase: ***BuildUrlRequest***	
Superclase:	**Subclases:**
Responsibilidades	**Colaboradoras**
Entidad encargada de construir el identificador de recursos uniforme perteneciente a una determinada petición que se le realizará al servidor.	

Nombre de Clase: ***ControllerRequest***	
Superclase:	**Subclases:**
Responsibilidades	**Colaboradoras**
Entidad encargada de controlar las peticiones realizadas desde el sistema al servidor.	*Request* *HttpRequestAsyncTask* *Session*

Nombre de Clase: ***HttpRequestAsyncTask***	
Superclase: *AsyncTask*	**Subclases:**
Responsibilidades	**Colaboradoras**
Entidad encargada de realizar una petición http al servidor y esperar la respuesta de este en un hilo ejecucción diferente al hilo principal sobre el cual se ejecuta el sistema.	

Nombre de Clase: ***MessageError***	
Superclase:	**Subclases:**
Responsibilidades	**Colaboradoras**
Entidad encargada de conformar el mensaje que se le mostrará al usuario en caso de que ocurra algún error con la conexión del servidor.	

Nombre de Clase: ***Request***	
Superclase:	**Subclases:**
Responsibilidades	**Colaboradoras**
Entidad que sirve de interfaz entre la capa de Aplicación y la capa de Comunicación a traves de ella las diferentes entidades de la capa Aplicación conforman solicitudes para realizar al servidor que son procesadas por la capa de Comunicación.	

Nombre de Clase: ***Session***	
Superclase: Object	**Subclases:**

Responsibilidades	**Colaboradoras**
Entidad que representa y almacerna los datos referentes a sesión abierta desde el dispositivo en el servidor.	

Entidades pertenecientes al paquete *components*.

Nombre de Clase: *CustomLabelFormatterDate*	
Superclase: DefaultLabelFormatter	**Subclases:**
Responsibilidades	**Colaboradoras**
Entidad encargada de definir como se visualizarán las etiquetas de tipo fecha en la gráfica de tendencia.	

Nombre de Clase: *CustomLegendRenderer*	
Superclase: LegendRenderer	**Subclases:**
Responsibilidades	**Colaboradoras**
Entidad encargada de definir como se visualizará la leyenda de la gráfica de tendencia.	

Nombre de Clase: *CustomLineGraphSeries*	
Superclase: LineGraphSeries	**Subclases:**
Responsibilidades	**Colaboradoras**
Entidad encargada de definir como se visualizarán las series de tipo línea en la gráfica de tendencia.	

Nombre de Clase: *CustomGraphView*

Superclase: GraphView	**Subclases:**
Responsibilidades	**Colaboradoras**
Entidad encargada de definir como se visualizarán la gráfica de tendencia.	

Entidades pertenecientes al paquete *components* en el subpaquete *adaptador*.

Nombre de Clase: ***AdaptadorFile***	
Superclase: BaseAdapter	**Subclases:**
Responsibilidades	**Colaboradoras**
Entidad que define como se visualizarán los diferentes ficheros con extensión json presente dentro de los directorios de archivos a los cuales el sistema tiene acceso.	

Nombre de Clase: ***AdaptadorItem***	
Superclase: ArrayAdapter	**Subclases:**
Responsibilidades	**Colaboradoras**
Entidad que define como se visualizarán los diferentes items que componen el menú de opciones o funcionalidades del sistema de acuerdo al componentes gráfico que lo visualize.	

Nombre de Clase: ***AdaptadorSerie***	
Superclase: BaseAdapter	**Subclases:**
Responsibilidades	**Colaboradoras**

Entidad encargada de definir como se visualizarán las series como items del componente gráfico *ListView*.	

Nombre de Clase: ***AdaptadorVariable***	
Superclase: ArrayAdapter	**Subclases:**
Responsibilidades	**Colaboradoras**
Entidad encargada de definir como si visualizará un variable como tupla o fila del sumario de variables.	

Nombre de Clase: ***ExpandableListAdapter***	
Superclase: BaseExpandableListAdapter	**Subclases:**
Responsibilidades	**Colaboradoras**
Entidad que define como se visualizarán las diferentes secciones que componen la vista de ayuda del sistema.	

Nombre de Clase: ***AdaptadorSpinnerSerie***	
Superclase: ArrayAdapter	**Subclases:**
Responsibilidades	**Colaboradoras**
Entidad encargada de definir como se visualizarán las series en el componente gráfico *Spinner*	

Nombre de Clase: ***AdaptadorSpinnerVariable***	
Superclase: ArrayAdapter	**Subclases:**
Responsibilidades	**Colaboradoras**

Entidad encargada de definir como se visualizarán las variables en el componente gráfico *Spinner*	

Entidades pertenecientes al paquete *components* en el subpaquete *gui*.

Nombre de Clase: *CustomFragment*	
Superclase: Fragment	**Subclases:**
Responsibilidades	**Colaboradoras**
Entidad genérica que representa un fragmento de vista dentro de un *activity*	MainActivity

Nombre de Clase: *ColorPicker*	
Superclase:	**Subclases:**
Responsibilidades	**Colaboradoras**
Entidad encargada de representar graficamente una paleta de colores.	

Entidades pertenecientes al paquete *components* en el subpaquete *preference*.

Nombre de Clase: *IntEditTextPreference*	
Superclase: EditTextPreference	**Subclases:**
Responsibilidades	**Colaboradoras**
Entidad encargada de visualizar aquellos campos de entrada de los atributos de tipo número entero definidos en la vista de las preferencias del sistema.	

Nombre de Clase: *RangeTrendPreference*

Superclase: DialogPreference	**Subclases:**
Responsibilidades	**Colaboradoras**
Entidad que tiene la responsabilidad de visualizar el atributo de intervalo de visualización de la gráfica de tendencia en la vista de preferencias del sistema.	

Nombre de Clase: ***RadioButtonPreference***	
Superclase: CheckBoxPreference	**Subclases:**
Responsibilidades	**Colaboradoras**
Entidad encargada de visualizar aquellos campos de entrada de los atributos de tipo lógico que su representación sea a través de radio button definidos en la vista de las preferencias del sistema.	

Entidades pertenecientes al paquete *components* en el subpaquete *receiver*.

Nombre de Clase: ***StatusBatteryReceiver***	
Superclase: *BroadcastReceiver*	**Subclases:**
Responsibilidades	**Colaboradoras**
Entidad encargada de recibir y procesar la notificaciones que envía el dispositivo móvil o tablet acerca del estado de la batería.	

Nombre de Clase: ***StatusNetworkReceiver***	
Superclase: *BroadcastReceiver*	**Subclases:**
Responsibilidades	**Colaboradoras**

Entidad encargada de recibir y procesar la notificaciones que envía el dispositivo móvil o tablet acerca del estado de la conexión alguna red.	

Entidades pertenecientes al paquete *controller*

Nombre de Clase: *ManagerSerie*	
Superclase:	**Subclases:**
Responsibilidades	**Colaboradoras**
Entidad que inicia el proceso de actualización del valor de las varibles que se esta representando en la interfaz gráfica asociada a esta entidad a traves de las series. Entidad encargada de procesar la respuesta del servidor ante una petición de actualización o solicitud de valores de las variables visualizadas en la interfaz. Entidad de gestionar todas las series que son visualizadas en la gráfica de tendencia.	*Util*, *Serie*, *Measurement*, *Request*

Entidades pertenecientes al paquete *core*.

Nombre de Clase: *RequestView*	
Superclase:	**Subclases:**
Responsibilidades	**Colaboradoras**
Interfaz de la cual deben implentar todas aquellas entidades que necesitan saber la respuesta que dio el servidor ante una petición iniciada en dicha entidad.	

Entidades pertenecientes al paquete *gui*.

Nombre de Clase: *MainActivity*	
Superclase: *Activity*	**Subclases:**
Responsibilidades	**Colaboradoras**
Entidad encargada de visualizar las diferentes vista que posee el sistema así como de controlar las transiciones entre estas. Entidad encargada de visualizar el estado conexión del dispositvo, asi como el estado de la bateria	*TrendVariableFragment*, *TrendFragment*, *PreferenceFragment*, *SummaryVariableFragment*, *MainFragment* , *DetailVariableFragment*, *HelpFragment*.

Entidades pertenecientes al paquete *gui* en el subpaquete *fragment*.

Nombre de Clase: *DetailVariableFragment*	
Superclase: CustomFragment	**Subclases:**
Responsibilidades	**Colaboradoras**
Entidad encargada de representar la vista de detalles de una variable. Entidad que inicia el proceso de actualización de la información de la varible que se esta representando en la interfaz gráfica asociada a esta entidad. Entidad encargada de procesar la respuesta del servidor ante una petición de actualización o solicitud de los detalles de una variable. Entidad encargada de actualizar los componentes gráficos de la iterfaz de detalles de una variable a partir de la información que recibe.	*MainActivity*, *Request*

Nombre de Clase: ***HelpFragment***	
Superclase: CustomFragment	**Subclases:**
Responsibilidades	**Colaboradoras**
Entidad encargada de representar la vista de ayuda al usuario del sistema.	

Nombre de Clase: ***MainFragment***	
Superclase: CustomFragment	**Subclases:**
Responsibilidades	**Colaboradoras**
Entidad encargada de visualizar la vista principal que le muestra el sistema al usuario	

Nombre de Clase: ***PreferenceFragment***	
Superclase: PreferenceFragment	**Subclases:**
Responsibilidades	**Colaboradoras**
Entidad encargada de visualizar al usuario las preferencias con que cuenta el sistema. Gestiona los valores que define el usuario para los diferentes atributos que integran las preferencias del sistema.	

Nombre de Clase: ***SummaryVariableFragment***	
Superclase: CustomFragment	**Subclases:**
Responsibilidades	**Colaboradoras**

Entidad encargada de visualizar la vista referente al sumario de variables Entidad que inicia el proceso de actualización de la información de las varibles que se esta representando en la interfaz gráfica asociada a esta entidad. Entidad encargada de procesar la respuesta del servidor ante una petición de actualización o solicitud de información de las variables visualizadas en la interfaz. Entidad encargada de actualizar los componentes gráficos de la interfaz de sumario de variables a partir de la información que recibe.	*MainActivity*, *Request*, *AdaptadorVariable*

Nombre de Clase: *TrendFragment*	
Superclase: CustomFragment	**Subclases:**
Responsibilidades	**Colaboradoras**
Entidad encargada de visualizar la vista con la gráfica de tendencia donde el usuario puede graficar el comportamiento de las variables durante el transcurso del tiempo Gestiona las series que se desean visualizar el gráfica de tendencia. Implementa las diferentes funcionalidades que posee la vista.	*MainActivity*, *ManagerSerie*, *ColorPicker* , *Util*, *Serie*, *Variable*

Nombre de Clase: ***TrendVariableFragment***	
Superclase: TrendFragment	**Subclases:**
Responsibilidades	**Colaboradoras**
Entidad encargada de visualizar la vista con la gráfica de tendencia donde el usuario puede graficar el comportamiento de una variable durante el transcurso del tiempo	

Nombre de Clase: ***LoadConfigFragment***	
Superclase: CustomFragment	**Subclases:**
Responsibilidades	**Colaboradoras**
Entidad encargada de visualizar la vista donde se despliega los diferentes ficheros que pueden contener una configuración predeterminada para el sistema.	*MainActivity*, *AdaptadorFile*

Entidades pertenecientes al paquete *util*.

Nombre de Clase: ***Util***	
Superclase:	**Subclases:**
Responsibilidades	**Colaboradoras**
Entidad que contiene un grupo de funcionalidades que contribuyen a que el resto de las entidades que componen el sistema puedan realizar sus responsabilidades.	

Nombre de Clase: ***ComparatorPoint***	
Superclase:	**Subclases:**

Responsibilidades	**Colaboradoras**
Entidad que implementa y define como se deben comparar dos variables de tipo *PointF*.	

Anexo D: Tareas de Ingeniería

A continuación la descripción de cada una de las Tareas de Ingeniería definidas para el sistema.

Tarea de Ingeniería	
Número tarea: 01	**Número de Historia de Usuario:** 01,19
Nombre tarea: Incorporar los permisos de Internet al sistema	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 1 de enero del 2018	**Fecha fin:** 1 de enero del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Incorporar los permisos pertinentes al sistema para que el mismo pueda acceder al estado de conexión del dispositivo. Los permisos a incorporar son los siguientes: ▪ android.permission.INTERNET ▪ android.permission.ACCESS_NETWORK_STATE ▪ android.permission.ACCESS_WIFI_STATE ▪ android.permission.CHANGE_WIFI_STATE ▪ android.permission.CHANGE_NETWORK_STATE ▪ android.permission.CHANGE_WIFI_MULTICAST_STATE	

Tarea de Ingeniería	
Número tarea: 02	**Número de Historia de Usuario:** 01,19
Nombre tarea: Implementar mecanismo de consulta de conexión	

Tipo de tarea: Desarrollo	**Puntos estimados:** 4
Fecha inicio: 2 de enero del 2018	**Fecha fin:** 5 de enero del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Implementación de un mecanismo que permita consultar el estado conexión del dispositivo y que informe de forma visual de dicho estado.	

Tarea de Ingeniería	
Número tarea: 03	**Número de Historia de Usuario:** 19
Nombre tarea: Iconografía referente a la HU_19	
Tipo de tarea: Diseño	**Puntos estimados:** 2
Fecha inicio: 2 de julio del 2018	**Fecha fin:** 3 de julio del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Diseñar los íconos que represente los diferentes estados de conexión por los que puede presentar el dispositivo. Los iconos deben ser de las siguientes resoluciones 192x192, 144x144, 96x96, 72x72, 48x48 y 36x36 pixeles.	

Tarea de Ingeniería	
Número tarea: 04	**Número de Historia de Usuario:** 19
Nombre tarea: Implementar mecanismo de actualización del estado de conexión del dispositivo.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 3
Fecha inicio: 4 de julio del 2018	**Fecha fin:** 6 de julio del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Implementar un mecanismo que cada un segundo actualice el estado de conexión del dispositivo y lo muestre gráficamente en la barra de título utilizando el ícono que se corresponde con el estado.	

Tarea de Ingeniería

Número tarea: 05	**Número de Historia de Usuario:** 15
Nombre tarea: Iconografía referente a la HU_15	
Tipo de tarea: Diseño	**Puntos estimados:** 2
Fecha inicio: 28 de mayo del 2018	**Fecha fin:** 29 de mayo del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Diseñar el ícono que represente graficamente la funcionalidad de eliminar todas las series que han sido añadidas a la gráfica de tendencia.	

Tarea de Ingeniería	
Número tarea: 06	**Número de Historia de Usuario:** 15
Nombre tarea: Implementación mecanismo de eliminación de todas series.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 3
Fecha inicio: 30 de mayo del 2018	**Fecha fin:** 1 de junio del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Diseñar e implementar un mecanismo que permita eliminar todas las series adicionadas a la gráfica de tendencia.	

Tarea de Ingeniería	
Número tarea: 07	**Número de Historia de Usuario:** 13
Nombre tarea: Iconografía referente a la HU_13	
Tipo de tarea: Diseño	**Puntos estimados:** 2
Fecha inicio: 14 de mayo del 2018	**Fecha fin:** 15 de mayo del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Diseñar el ícono que represente graficamente la funcionalidad de eliminar de las series añadidas a la gráfica de tendencia aquellas que el usuario seleccione.	

Tarea de Ingeniería

Número tarea: 08	**Número de Historia de Usuario:** 13
Nombre tarea: Diseño visual de la interfaz para elminar series.	
Tipo de tarea: Diseño	**Puntos estimados:** 1
Fecha inicio: 16 de mayo del 2018	**Fecha fin:** 16 de mayo del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Diseñar la interfaz visual que le mostrará el sistema al usuario para que este seleccione la o las series que desea eliminar.	

Tarea de Ingeniería	
Número tarea: 09	**Número de Historia de Usuario:** 13
Nombre tarea: Implementación del visual de la interfaz para elminar series.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 17 de mayo del 2018	**Fecha fin:** 17 de mayo del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Implementar la lógica programable de los componentes que integran la interfaz visual para eliminar series de la gráfica de tendencia.	

Tarea de Ingeniería	
Número tarea: 10	**Número de Historia de Usuario:** 13
Nombre tarea: Implementación del mecanismo para elimnar una serie de la gráfica de tendencia.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 18 de mayo del 2018	**Fecha fin:** 18 de mayo del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Implementar un mecanismo que permite eliminar una o varias series que estan añadidas a la gráfica de tendencia dado sus identificadores.	

Tarea de Ingeniería

Número tarea: 11	**Número de Historia de Usuario:** 14
Nombre tarea: Iconografía referente a la HU_14	
Tipo de tarea: Diseño	**Puntos estimados:** 2
Fecha inicio: 21 de mayo del 2018	**Fecha fin:** 22 de mayo del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Diseñar el ícono que represente graficamente la funcionalidad de cambiar la visibilidad de las series añadidas a la gráfica de tendencia.	

Tarea de Ingeniería	
Número tarea: 12	**Número de Historia de Usuario:** 14
Nombre tarea: Diseño visual de la interfaz para mostrar y ocultar series.	
Tipo de tarea: Diseño	**Puntos estimados:** 1
Fecha inicio: 23 de mayo del 2018	**Fecha fin:** 23 de mayo del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Diseñar la interfaz visual que le mostrará el sistema al usuario para que este seleccione la o las series que desea cambiar su estado de visibilidad.	

Tarea de Ingeniería	
Número tarea: 13	**Número de Historia de Usuario:** 14
Nombre tarea: Implementación del visual de la interfaz para elminar series.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 24 de mayo del 2018	**Fecha fin:** 24 de mayo del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Implementar la lógica programable de los componentes que integran la interfaz visual para seleccionar las series que se desean cambiar su visibilidad.	

Tarea de Ingeniería	
Número tarea: 14	**Número de Historia de Usuario:** 14

Nombre tarea: Implementación del mecanismo para cambiar el estado de visibilidad de una serie de la gráfica de tendencia.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 25 de mayo del 2018	**Fecha fin:** 25 de mayo del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Implementar un mecanismo que permite cambiar la visibilidad una o varias series que estan añadidas a la gráfica de tendencia dado sus identificadores.	

Tarea de Ingeniería	
Número tarea: 15	**Número de Historia de Usuario:** 12
Nombre tarea: Iconografía referente a la HU_12	
Tipo de tarea: Diseño	**Puntos estimados:** 2
Fecha inicio: 7 de mayo del 2018	**Fecha fin:** 8 de mayo del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Diseñar el ícono que represente graficamente la funcionalidad de adicionar series a la gráfica de tendencia.	

Tarea de Ingeniería	
Número tarea: 16	**Número de Historia de Usuario:** 12
Nombre tarea: Diseño visual de la interfaz para adicionar series.	
Tipo de tarea: Diseño	**Puntos estimados:** 1
Fecha inicio: 9 de mayo del 2018	**Fecha fin:** 9 de mayo del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Diseñar la interfaz visual que le mostrará el sistema al usuario para que este adicione series a la gráfica de tendencia.	

Tarea de Ingeniería	
Número tarea: 17	**Número de Historia de Usuario:** 12

Nombre tarea: Implementación del visual de la interfaz para adicionar series.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 10 de mayo del 2018	**Fecha fin:** 10 de mayo del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Implementar la lógica programable de los componentes que integran la interfaz visual para adicionar series a la gráfica de tendencia.	

Tarea de Ingeniería	
Número tarea: 18	**Número de Historia de Usuario:** 12
Nombre tarea: Implementación del mecanismo para adicionar una serie de la gráfica de tendencia.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 11 de mayo del 2018	**Fecha fin:** 11 de mayo del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Implementar un mecanismo que permite una adicionar una serie a la gráfica de tendencia con los datos proporcionados por el usuario.	

Tarea de Ingeniería	
Número tarea: 23	**Número de Historia de Usuario:** 16
Nombre tarea: Iconografía referente a la HU_16	
Tipo de tarea: Diseño	**Puntos estimados:** 2
Fecha inicio: 4 de junio del 2018	**Fecha fin:** 5 de junio del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Diseñar el ícono que represente graficamente la funcionalidad de invertir los colores de texto y fondo de la gráfica de tendencia.	

Tarea de Ingeniería	
Número tarea: 24	**Número de Historia de Usuario:** 16

Nombre tarea: Implementación mecanismo de invertir los colores de la gráfica de tendencia.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 3
Fecha inicio: 6 de junio del 2018	**Fecha fin:** 8 de junio del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Diseñar e implementar un mecanismo que permita invertir los colores de texto y fondo de la gráfica de tendencia. Por defecto el fondo será negro y el texto de de color blanco. El mecanismo debe permitir cambiar de negro a blanco o de blanco a negro.	

Tarea de Ingeniería	
Número tarea: 25	**Número de Historia de Usuario:** 17
Nombre tarea: Iconografía referente a la HU_17	
Tipo de tarea: Diseño	**Puntos estimados:** 2
Fecha inicio: 18 de junio del 2018	**Fecha fin:** 19 de junio del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Diseñar el ícono que represente graficamente la funcionalidad de exportar la gráfica de tendecia a una imagen.	

Tarea de Ingeniería	
Número tarea: 26	**Número de Historia de Usuario:** 17
Nombre tarea: Exportar gráfica de tendencia a una imagen	
Tipo de tarea: Desarrollo	**Puntos estimados:** 3
Fecha inicio: 20 de junio del 2018	**Fecha fin:** 22 de junio del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Diseñar e implementar un mecanismo que permita exportar hacia una imagen la gráfica de tendencia con su color de fondo y texto , asi como sus ejes y los intervalos de estos.	

Tarea de Ingeniería	
Número tarea: 27	**Número de Historia de Usuario:** 17
Nombre tarea: Exportar series de la gráfica de tendencia a una imagen	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 25 de junio del 2018	**Fecha fin:** 26 de junio del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Diseñar e implementar un mecanismo que permita exportar hacia una imagen todas las series visibles en la gráfica de tendencia.	

Tarea de Ingeniería	
Número tarea: 28	**Número de Historia de Usuario:** 17
Nombre tarea: Salvar imagen en fichero	
Tipo de tarea: Desarrollo	**Puntos estimados:** 3
Fecha inicio: 27 de junio del 2018	**Fecha fin:** 29 de junio del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Diseñar e implementar un mecanismo que permita salvar una imagen en un fichero dentro del directorio destinado para la aplicación en el dispositivo.	

Tarea de Ingeniería	
Número tarea: 29	**Número de Historia de Usuario:** 20
Nombre tarea: Iconografía referente a la HU_20	
Tipo de tarea: Diseño	**Puntos estimados:** 2
Fecha inicio: 9 de julio del 2018	**Fecha fin:** 10 de julio del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Diseñar el ícono que represente graficamente la funcionalidad de editar series representadas en la gráfica de tendencia.	

Tarea de Ingeniería

Número tarea: 30	**Número de Historia de Usuario:** 20
Nombre tarea: Diseño visual de la interfaz para editar series.	
Tipo de tarea: Diseño	**Puntos estimados:** 1
Fecha inicio: 11 de julio del 2018	**Fecha fin:** 11 de julio del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Diseñar la interfaz visual que le mostrará el sistema al usuario para que este eddite las series visualizadas en la gráfica de tendencia.	

Tarea de Ingeniería	
Número tarea: 31	**Número de Historia de Usuario:** 20
Nombre tarea: Implementación del visual de la interfaz para editar series.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 12 de julio del 2018	**Fecha fin:** 12 julio del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Implementar la lógica programable de los componentes que integran la interfaz visual para editar las series visualizadas en la gráfica de tendencia.	

Tarea de Ingeniería	
Número tarea: 32	**Número de Historia de Usuario:** 20
Nombre tarea: Implementación del mecanismo para editar una serie de la gráfica de tendencia.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 13 de julio del 2018	**Fecha fin:** 13 de julio del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Implementar un mecanismo que permite editar una serie de la gráfica de tendencia con los datos proporcionados por el usuario.	

Tarea de Ingeniería

Número tarea: 33	**Número de Historia de Usuario:** 18
Nombre tarea: Iconografía referente a la HU_18	
Tipo de tarea: Diseño	**Puntos estimados:** 2
Fecha inicio: 11 de junio del 2018	**Fecha fin:** 12 de junio del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Diseñar el ícono que represente graficamente la funcionalidad de mostrar/ocultar la leyenda de la gráfica de tendencia.	

Tarea de Ingeniería	
Número tarea: 34	**Número de Historia de Usuario:** 18
Nombre tarea: Mecanismo de visualización de la leyenda	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 13 de junio del 2018	**Fecha fin:** 14 de junio del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Diseñar e implementar el mecanismo para mostrar la leyenda asociada a la gráfica de tendencia.	

Tarea de Ingeniería	
Número tarea: 35	**Número de Historia de Usuario:** 18
Nombre tarea: Mecanismo de pintando de la leyenda	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 15 de junio del 2018	**Fecha fin:** 15 de junio del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Diseñar e implementar el mecanismo para la visualización de la leyenda en la gráfica de tendencia.	

Tarea de Ingeniería	
Número tarea: 36	**Número de Historia de Usuario:** 11

Nombre tarea: Diseño de la interfaz visual de la HU_11	
Tipo de tarea: Diseño	**Puntos estimados:** 3
Fecha inicio: 5 de marzo del 2018	**Fecha fin:** 7 de marzo del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Diseñar la interfaz visual de la gráfica de tendencia. Dicha interfaz visual se compone de dos secciones la primera la barra de herramientas y la segunda que debe ocupar la mayor área de la vista el área de graficación.	

Tarea de Ingeniería	
Número tarea: 37	**Número de Historia de Usuario:** 11
Nombre tarea: Lógica programable de la interfaz visual de la HU_11	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 8 de marzo del 2018	**Fecha fin:** 9 de marzo del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Diseñar e implementar la lógica programable de los componentes visuales que componen la interfaz visual de la grafica de tendencia.	

Tarea de Ingeniería	
Número tarea: 38	**Número de Historia de Usuario:** 11
Nombre tarea: Incorporar el *GraphView*	
Tipo de tarea: Desarrollo	**Puntos estimados:** 3
Fecha inicio: 12 de marzo del 2018	**Fecha fin:** 14 de marzo del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Incorporar a la vista de la interfaz de la gráfica de tendencia la entidad *GraphView* que representa la gráfica de tendencia.	

Tarea de Ingeniería	
Número tarea: 39	**Número de Historia de Usuario:** 11

Nombre tarea: Iconografía referente a la HU_11	
Tipo de tarea: Diseño	**Puntos estimados:** 2
Fecha inicio: 15 de marzo del 2018	**Fecha fin:** 16 de marzo del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Diseñar el ícono que represente graficamente la vista de la gráfica de tendencia.	

Tarea de Ingeniería	
Número tarea: 40	**Número de Historia de Usuario:** 11
Nombre tarea: Mecanismo para la actualización de la interfaz gráfica de la vista gráfica de tendencia	
Tipo de tarea: Desarrollo	**Puntos estimados:** 3
Fecha inicio: 19 de marzo del 2018	**Fecha fin:** 21 de marzo del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Diseñar y implementar un mecanismo que actualice cada un segundo en la gráfica de tendencia el intervalo de monitoreo según la configuración destinada para el dispositivo, tomando como extremo derecho del rango a visualizar la estampa de tiempo actual y como extremo izquierdo la estampa de tiempo actual menos el rango de tiempo que se desea monitorear en la gráfica.	

Tarea de Ingeniería	
Número tarea: 41	**Número de Historia de Usuario:** 11
Nombre tarea: Mecanismo para la actualización de los valores de las series	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 22 de marzo del 2018	**Fecha fin:** 23 de marzo del 2018
Programador responsable: Luis Andrés Valido Fajardo	

Descripción: Diseñar y implementar un mecanismo que solicite de forma períodica cada un intervalo de tiempo definido en la configuración del sistema al servidor el último valor adquirido por el sistema ROFLEXIN/LC de un grupo de determinadas variables y a su vez actualice la visualización de las series asociadas a estas variables con estos nuevos valores.

Tarea de Ingeniería	
Número tarea: 42	**Número de Historia de Usuario:** 10
Nombre tarea: Diseño de la interfaz visual de la HU_10	
Tipo de tarea: Diseño	**Puntos estimados:** 3
Fecha inicio: 16 de abril del 2018	**Fecha fin:** 18 de abril del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Diseñar la interfaz visual de la gráfica de tendencia donde visualizará el comportamiento de una variable. Dicha interfaz visual se compone de dos secciones la primera la barra de herramientas y la segunda que debe ocupar la mayor área de la vista el área de graficación.	

Tarea de Ingeniería	
Número tarea: 43	**Número de Historia de Usuario:** 10
Nombre tarea: Lógica programable de la interfaz visual de la HU_10	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 19 de abril del 2018	**Fecha fin:** 20 de abril del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Diseñar e implementar la lógica programable de los componentes visuales que componen la interfaz visual de la grafica de tendencia donde se visualizará el comportamiento de una variable.	

Tarea de Ingeniería	
Número tarea: 44	**Número de Historia de Usuario:** 10

Nombre tarea: Crear acceso a la funcionalidad asociada a la HU_10 I	
Tipo de tarea: Desarrollo	**Puntos estimados:** 3
Fecha inicio: 23 de abril del 2018	**Fecha fin:** 25 de abril del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Incorporar a la interfaz gráfica de la vista detalles de una variable un elemento gráfico que le permita al usuario acceder de desde esta vista a la vista de gráfica de tendencia de una variable.	

Tarea de Ingeniería	
Número tarea: 45	**Número de Historia de Usuario:** 10
Nombre tarea: Crear acceso a la funcionalidad asociada a la HU_10 II	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 26 de abril del 2018	**Fecha fin:** 27 de abril del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Incorporar al menú contextual del sumario de variables un elemento que permita acceder a la vista de la gráfica de tendencia de una variable.	

Tarea de Ingeniería	
Número tarea: 46	**Número de Historia de Usuario:** 05
Nombre tarea: Iconografía referente a la HU_05	
Tipo de tarea: Diseño	**Puntos estimados:** 1
Fecha inicio: 22 de enero del 2018	**Fecha fin:** 22 de enero del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Diseñar el ícono representativo de la funcionalidad *Autenticar terminal*.	

Tarea de Ingeniería	
Número tarea: 47	**Número de Historia de Usuario:** 05

Nombre tarea: Diseño de la interfaz visual de la HU_05	
Tipo de tarea: Diseño	**Puntos estimados:** 1
Fecha inicio: 23 de enero del 2018	**Fecha fin:** 23 de enero del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Diseñar la interfaz visual del cuadro de díalogo donde se introducirá los datos servidor a donde desea autenticarse.	

Tarea de Ingeniería	
Número tarea: 48	**Número de Historia de Usuario:** 05
Nombre tarea: Lógica programable de la HU_05	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 24 de enero del 2018	**Fecha fin:** 24 de enero del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Implementar la lógica programable de los componentes visuales presentes en la interfaz visual de la HU_05.	

Tarea de Ingeniería	
Número tarea: 49	**Número de Historia de Usuario:** 05
Nombre tarea: Mecanismo de autenticación	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 25 de enero del 2018	**Fecha fin:** 26 de enero del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Diseñar e implementar un mecanismo que envíe una solicitud de autenticación al servidor y reciba la respuesta de este.	

Tarea de Ingeniería	
Número tarea: 50	**Número de Historia de Usuario:** 06
Nombre tarea: Iconografía referente a la HU_06	

Tipo de tarea: Diseño	**Puntos estimados:** 1
Fecha inicio: 29 de enero del 2018	**Fecha fin:** 30 de enero del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Diseñar el ícono representativo de la funcionalidad *Cerrar sesión*.	

Tarea de Ingeniería	
Número tarea: 51	**Número de Historia de Usuario:** 06
Nombre tarea: Mecanismo de cierre de sesión	
Tipo de tarea: Desarrollo	**Puntos estimados:** 3
Fecha inicio: 31 de enero del 2018	**Fecha fin:** 2 de febrero del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Diseñar e implementar un mecanismo que envíe una solicitud de cierre de sesión al servidor y reciba la respuesta de este.	

Tarea de Ingeniería	
Número tarea: 52	**Número de Historia de Usuario:** 09
Nombre tarea: Diseño de la interfaz visual de la HU_09	
Tipo de tarea: Diseño	**Puntos estimados:** 2
Fecha inicio: 26 de marzo del 2018	**Fecha fin:** 27 de marzo del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Diseñar la interfaz visual de la vista detalles de una variable.	

Tarea de Ingeniería	
Número tarea: 53	**Número de Historia de Usuario:** 09
Nombre tarea: Lógica programable de la interfaz visual de la HU_09	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 28 de marzo del 2018	**Fecha fin:** 29 de marzo del 2018
Programador responsable: Luis Andrés Valido Fajardo	

Descripción: Diseñar e implementar la lógica programable de los componentes visuales que componen la interfaz visual de detalles de una variable.

Tarea de Ingeniería	
Número tarea: 54	**Número de Historia de Usuario:** 09
Nombre tarea: Crear acceso a la funcionalidad asociada a la HU_09	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 30 de marzo del 2018	**Fecha fin:** 30 de marzo del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Incorporar al menú contextual del sumario de variables un elemento que permita acceder a la vista de detalles de una variable.	

Tarea de Ingeniería	
Número tarea: 55	**Número de Historia de Usuario:** 09
Nombre tarea: Mecanismo para la actualización de la información de la variable	
Tipo de tarea: Desarrollo	**Puntos estimados:** 3
Fecha inicio: 2 de abril del 2018	**Fecha fin:** 4 de abril del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Diseñar y implementar un mecanismo que solicite de forma períodica cada un intervalo de tiempo definido en la configuración del sistema al servidor el último valor adquirido por el sistema ROFLEXIN/LC de una variable asi como toda la infomación referente a esta.	

Tarea de Ingeniería	
Número tarea: 56	**Número de Historia de Usuario:** 09
Nombre tarea: Mecanismo para la actualización de la interfaz gráfica de la vista detalles de una variable	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2

Fecha inicio: 5 de abril del 2018	**Fecha fin:** 6 de abril del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Diseñar y implementar un mecanismo que actualice la interfaz gráfica de la vista de detalles de una variable cada vez que llegue un cambio en algunos de los atributos de la variable.	

Tarea de Ingeniería	
Número tarea: 57	**Número de Historia de Usuario:** 08
Nombre tarea: Iconografía referente a la HU_08	
Tipo de tarea: Diseño	**Puntos estimados:** 1
Fecha inicio: 12 de febrero del 2018	**Fecha fin:** 12 de febrero del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Diseñar el ícono representativo de la vista sumario de las variables.	

Tarea de Ingeniería	
Número tarea: 58	**Número de Historia de Usuario:** 08
Nombre tarea: Diseño de la interfaz visual de la HU_08	
Tipo de tarea: Diseño	**Puntos estimados:** 2
Fecha inicio: 13 de febrero del 2018	**Fecha fin:** 14 de febrero del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Diseñar la interfaz visual de la vista sumario de variables.	

Tarea de Ingeniería	
Número tarea: 59	**Número de Historia de Usuario:** 08
Nombre tarea: Lógica programable de la HU_08 I	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 15 de febrero del 2018	**Fecha fin:** 16 de febrero del 2018
Programador responsable: Luis Andrés Valido Fajardo	

Descripción: Implementar la lógica programable de los componentes visuales presentes en la interfaz visual de la vista sumario de las variables.

Tarea de Ingeniería	
Número tarea: 60	**Número de Historia de Usuario:** 08
Nombre tarea: Diseño de la interfaz visual de la HU_08 II	
Tipo de tarea: Diseño	**Puntos estimados:** 2
Fecha inicio: 19 de febrero del 2018	**Fecha fin:** 20 de febrero del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Diseñar la interfaz visual de la tupla que representa una variable dentro del sumario de variables.	

Tarea de Ingeniería	
Número tarea: 61	**Número de Historia de Usuario:** 08
Nombre tarea: Lógica programable de la HU_08 II	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 21 de febrero del 2018	**Fecha fin:** 22 de febrero del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Implementar la lógica programable de los componentes visuales presentes en la interfaz visual de una tupla del sumario de variables.	

Tarea de Ingeniería	
Número tarea: 62	**Número de Historia de Usuario:** 08
Nombre tarea: Menú contextual del sumario de las variables	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 23 de febrero del 2018	**Fecha fin:** 23 de febrero del 2018
Programador responsable: Luis Andrés Valido Fajardo	

Descripción: Diseñar e implementar el menú contextual asociado al sumario de las variables.

Tarea de Ingeniería	
Número tarea: 63	**Número de Historia de Usuario:** 08
Nombre tarea: Mecanismo para la actualización de la interfaz gráfica de la vista sumario de variables	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 26 de febrero del 2018	**Fecha fin:** 27 de febrero del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Diseñar y implementar un mecanismo que actualice la interfaz gráfica de la vista de sumario de variables cada vez que cambie algunos de los atributos de las variables visualizadas.	

Tarea de Ingeniería	
Número tarea: 64	**Número de Historia de Usuario:** 08
Nombre tarea: Mecanismo para la actualización de la información visualizada en el sumario de variable	
Tipo de tarea: Desarrollo	**Puntos estimados:** 3
Fecha inicio: 28 de febrero del 2018	**Fecha fin:** 2 de marzo del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Diseñar y implementar un mecanismo que solicite de forma períodica cada un intervalo de tiempo definido en la configuración del sistema al servidor el último valor adquirido por el sistema ROFLEXIN/LC de las variables visualizadas en el sumario asi como toda la infomación referente a estas que se muestran en dicha vista.	

Tarea de Ingeniería	
Número tarea: 65	**Número de Historia de Usuario:** 04

Nombre tarea: Implementar mecanismo de consulta de la batería	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 1 de mayo del 2018	**Fecha fin:** 2 de mayo del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Implementación de un mecanismo que permita consultar el estado de la batería del dispositivo y que informe de forma visual de dicho estado.	

Tarea de Ingeniería	
Número tarea: 66	**Número de Historia de Usuario:** 04
Nombre tarea: Iconografía referente a la HU_04	
Tipo de tarea: Diseño	**Puntos estimados:** 1
Fecha inicio: 30 de abril del 2018	**Fecha fin:** 30 de abril del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Diseñar los íconos que represente los diferentes estados por los que puede presentar la batería del dispositivo. Los iconos deben ser de las siguientes resoluciones 192x192, 144x144, 96x96, 72x72, 48x48 y 36x36 pixeles.	

Tarea de Ingeniería	
Número tarea: 67	**Número de Historia de Usuario:** 04
Nombre tarea: Implementar mecanismo de actualización del estado de la batería del dispositivo.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 3 de mayo del 2018	**Fecha fin:** 4 de mayo del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Implementar un mecanismo que cada vez que ocurra un cambio de estado de la batería del dispositivo muestre gráficamente en la barra de título utilizando el ícono que se corresponde con el estado.	

Tarea de Ingeniería	
Número tarea: 68	**Número de Historia de Usuario:** 03
Nombre tarea: Implementar mecanismo de consulta de la fecha y hora	
Tipo de tarea: Desarrollo	**Puntos estimados:** 3
Fecha inicio: 9 de abril del 2018	**Fecha fin:** 11 de abril del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Implementación de un mecanismo que permita consultar la fecha y hora del dispositivo y que informe de forma visual de dichos datos .	

Tarea de Ingeniería	
Número tarea: 69	**Número de Historia de Usuario:** 04
Nombre tarea: Implementar mecanismo de actualización de la fecha y hora del dispositivo.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 12 de abril del 2018	**Fecha fin:** 13 de abril del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Implementar un mecanismo que cada un segundo actualice la hora y fecha visualizada en la barra de título del sistema.	

Tarea de Ingeniería	
Número tarea: 70	**Número de Historia de Usuario:** 07
Nombre tarea: Implementar mecanismo de consulta de datos de la terminal	
Tipo de tarea: Desarrollo	**Puntos estimados:** 3
Fecha inicio: 5 de febrero del 2018	**Fecha fin:** 7 de febrero del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Implementación de un mecanismo que permita consultar los datos del dispositivos (MAC de red del dispositivo) .	

Tarea de Ingeniería	
Número tarea:71	**Número de Historia de Usuario:** 07
Nombre tarea: Implementar mecanismo de visualizacion de los datos del dispositivo.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 8 de febrero del 2018	**Fecha fin:** 9 de febrero del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Implementar un mecanismo que visualice los datos del dispostivo.	

Tarea de Ingeniería	
Número tarea:72	**Número de Historia de Usuario:** 02
Nombre tarea: Implementar mecanismo de buscar ficheros con extensión JSON.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 8 de enero del 2018	**Fecha fin:** 9 de enero del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Implementar un mecanismo que busque en todos los directorios accesibles del sistema los ficheros cuya extensión sea .json .	

Tarea de Ingeniería	
Número tarea:73	**Número de Historia de Usuario:** 02
Nombre tarea: Diseñar interfaz de la HU_02.	
Tipo de tarea: Diseño	**Puntos estimados:** 2
Fecha inicio: 10 de enero del 2018	**Fecha fin:** 11 de enero del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Diseñar una interfaz gráfica que visualice todos los archivos de extensión .json y permita la selección de uno de ellos.	

Tarea de Ingeniería

Número tarea:74	**Número de Historia de Usuario:** 02
Nombre tarea:Lógica programable de la interfaz de la HU_02.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 12 de enero del 2018	**Fecha fin:** 12 de enero del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Implementar la lógica programable de la interfaz gráfica de la HU_02 y de los componentes visuales que la integran.	

Tarea de Ingeniería	
Número tarea:75	**Número de Historia de Usuario:** 02
Nombre tarea: Mecanismo de lectura de archivo de extensión *.json*.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 3
Fecha inicio: 15 de enero del 2018	**Fecha fin:** 17 de enero del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Implementar un mecanismo para leer todo el contenido de un archivo con extensión *.json* y guardar el contenido del fichero en una estructura de datos adecuada.	

Tarea de Ingeniería	
Número tarea:76	**Número de Historia de Usuario:** 02
Nombre tarea: Mecanismo de modificación de las preferencias del sistema.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 18 de enero del 2018	**Fecha fin:** 19 de enero del 2018
Programador responsable: Luis Andrés Valido Fajardo	
Descripción: Implementar un mecanismo que permita modificar los valores de las preferencias del sistema a partir de valores suministrados.	

Anexo E: Resumen de las Tareas de Ingeniería por Historias de Usuarios

A continuación el resumen de las tareas de ingenierías implementadas por cada Historia de Usuario.

Tabla E.1: Resumen de Tareas de Ingeniería por Historias de Usuario

Historia de Usuario	No. Tarea	Tarea de Ingeniería
Comprobar conexión del dispositivo	2	01, 02
Cargar configuración definida para la terminal	5	72, 73, 74, 75, 76
Mostrar fecha y hora actual	2	68, 69
Mostrar estado actual de la batería del dispositivo	3	65, 66, 67
Autenticar terminal	4	46, 47, 48, 49
Cerrar sesión	2	50, 51
Visualizar datos de la terminal	2	70, 71
Visualizar sumario de variables	8	57, 58, 59, 60, 61, 62, 63, 64
Visualizar detalles de una variable	5	52, 53, 54, 55, 56
Visualizar comportamiento de una variable en un gráfico de tendencia	6	40, 41, 42, 43, 44, 45
Visualizar gráfico de tendencia	6	36, 37, 38, 39, 40, 41
Adicionar serie a la gráfica de tendencia	4	15, 16, 17, 18
Eliminar serie de la gráfica de tendencia	4	07, 08, 09, 10

Continúa en la siguiente página

Tabla E.1 Continuación de la página anterior

Mostrar u ocultar serie de la gráfica de tendencia	4	11, 12, 13, 14
Limpiar gráfico de tendencia	2	05, 06
Invertir los colores de texto y fondo de la gráfica de tendencia	2	23, 24
Exportar el gráfico de tendencia a formato de imagen	4	25, 26, 27, 28
Mostrar u ocultar leyenda de las series representadas en la gráfica	3	33, 34, 35
Visualizar estado actual de conexión del dispositivo	4	01, 02, 03, 04
Editar propiedades de una serie	4	29, 30, 31, 32

Anexo F: Casos de pruebas

A continuación la descripción de cada uno de los casos de pruebas realizados al sistemas.

Caso de prueba de aceptación	
Código: PA01_HU05	**Historia de Usuario:** HU05
Nombre: Auntenticar terminal con errores I	
Descripción: Se intenterá autenticar la terminal dejando los campos de anfritión y puerto del servidor vacío.	
Condiciones de ejecución: El dispositivo no debe haber iniciado sesión o ternerla cerrada en el servidor.	
Pasos de ejecución: Se selecciona la opción de auntenticar ya sea por el menú de navegación o en la vista principal del sistema. Se presiona el botón Aceptar del cuadro de díalogo con los campos anfitrión y puerto vacíos.	
Resultados esperados: Se debe visualizar un cuadro notificación que alerte de que existen campos vacíos. Se debe señalar de forma gráfica aquellas caja de texto que están vacías.	
Evaluación de la prueba: Satisfactoria	

Caso de prueba de aceptación	
Código: PA02_HU01	**Historia de Usuario:** HU01
Nombre: Comprobar conexión del dispositivo conectado a una red wifi	
Descripción: Se ejecutará el sistema en el dispositivo, él cual debe estar conectado a red wifi previamente y se comprobará que el sistema es capaz de notificar del estado de conexión del dispositivo.	

Condiciones de ejecución: El dispositivo debe estar conectado a una red wifi e instalado el sistema entre sus aplicaciones.
Pasos de ejecución: Se busca el ícono del sistema dentro del menú de aplicaciones del dispositivo y se da click sobre este.
Resultados esperados: Una vez iniciado el sistema el mismo debe visualizar mediante una notificación emergente el estado de conexión del dispositivo. Para este caso debe notificar que esta conectado a una red wifi.
Evaluación de la prueba: Satisfactoria

Caso de prueba de aceptación	
Código: PA03_HU03	**Historia de Usuario:** HU03
Nombre: Comprobar visualización de la fecha y hora del dispositivo	
Descripción: Se ejecutará el sistema en el dispositivo y se comprobará que el sistema es capaz de visualizar la fecha y hora que posee el dispositivo y actualizar dichos valores cada un segundo.	
Condiciones de ejecución: El dispositivo debe tener instalado el sistema entre sus aplicaciones.	
Pasos de ejecución: Se busca el ícono del sistema dentro del menú de aplicaciones del dispositivo y se da click sobre este.	
Resultados esperados: Una vez iniciado el sistema el mismo debe visualizar en la barra de título del sistema en su parte derecha superior la fecha y hora del dipositivo el cual debe cumplir con el siguiente formato *d-M-Y hh:mm:ss*. Cada un segundo estos datos deberán actualizarse.	
Evaluación de la prueba: Satisfactoria	

Caso de prueba de aceptación	
Código: PA04_HU04	**Historia de Usuario:** HU04

Nombre: Comprobar visualización del estado de la batería del dispositivo.
Descripción: Se ejecutará el sistema en el dispositivo y se comprobará que el sistema es capaz de visualizar el estado de la batería que posee el dispositivo y actualizar dicho estado cuando ocurra un cambio.
Condiciones de ejecución: El dispositivo debe tener instalado el sistema entre sus aplicaciones. El dispositivo no debe estar conectado a ninguna fuente de alimentación ajena a su batería.
Pasos de ejecución: Se busca el ícono del sistema dentro del menú de aplicaciones del dispositivo y se da click sobre este. Una vez hecho esto se deja en ejecucción el sistema por un tiempo superior a los 15 minutos.
Resultados esperados: Una vez iniciado el sistema el mismo debe visualizar en la barra de título del sistema en su parte derecha inferior el estado de la batería del dipositivo el cual debe cumplir con lo especificado en la HU_04. El estado de la batería debe cambiar el porciento de carga después de 15 minutos y el sistema debe actualizar su representación
Evaluación de la prueba: Satisfactoria

Caso de prueba de aceptación	
Código: PA05_HU06	**Historia de Usuario:** HU06
Nombre: Cerrar sesión DCRSA	
Descripción: Se cerrará la sesión desde el dispositivo el cual está conectado a la misma red del servidor que se encuentra en ejecucción.	
Condiciones de ejecución: El dispositivo debe haber iniciado sessión en el servidor y este debe estar en ejecucción y ambos conectados a la misma red.	
Pasos de ejecución: Se selecciona la opción de cerrar sesión ya sea por el menú de navegación o en la vista principal del sistema.	

Resultados esperados: La visualización de un cuadro emergente donde se informa al usuario que se pudo cerrar la sesión en el servidor. Desaparece del menú de navegación y de la vista principal del sistema la opción de cerrar sesión y aparece la de autenticarse.
Evaluación de la prueba: Satisfactoria

Caso de prueba de aceptación	
Código: PA06_HU07	**Historia de Usuario:** HU07
Nombre: Visualización de los datos de la terminal	
Descripción: Se comprobará la visualización de los datos de la terminal I	
Condiciones de ejecución: El dispositivo debe tener instalado el sistema entre sus aplicaciones. El sistema debe haber sido iniciado.	
Pasos de ejecución: Se busca el ícono del sistema dentro del menú de aplicaciones del dispositivo y se da click sobre este. Una vez hecho esto se despliega el menú de navegación.	
Resultados esperados: En la parte superior del menú de navegación se visualizará la MAC del dispostivo.	
Evaluación de la prueba: Satisfactoria	

Caso de prueba de aceptación	
Código: PA07_HU16	**Historia de Usuario:** HU16
Nombre: Invertir colores de texto y fondo en la gráfica de tendencia.	
Descripción: Se comprobará el correcto funcionamiento de la funcionalidad de invertir los colores de texto y fondo de la gráfica de tendencia.	
Condiciones de ejecución: El dispositivo debe tener instalado el sistema entre sus aplicaciones. El sistema debe haber sido iniciado. Se debe estar visualizando la vista de gráfica de tendencia.	

Pasos de ejecución: Se busca el ícono del sistema dentro del menú de aplicaciones del dispositivo y se da click sobre este. Una vez hecho esto se despliega el menú de navegación o en la vista principal se da click sobre la opción de *Tendencia*. Dar click sobre el ícono en la barra de herramienta que representa la funcionalidad en la vista de la gráfica de tendencia.
Resultados esperados: Si el fondo es negro y los textos en blanco una vez dado un click sobre la funcionalidad los colores deben invertirse, el fondo blanco y los textos de negro, en caso de un nuevo click los colores se vuelven a invertirse y así sucesivamente tras cada click sobre el ícono de la funcionalidad.
Evaluación de la prueba: Satisfactoria

Caso de prueba de aceptación	
Código: PA08_HU15	**Historia de Usuario:** HU15
Nombre: Limpiar la gráfica de tendencia.	
Descripción: Se comprobará el correcto funcionamiento de la funcionalidad de limpiar la gráfica de tendencia.	
Condiciones de ejecución: El dispositivo debe tener instalado el sistema entre sus aplicaciones. El sistema debe haber sido iniciado. Se debe estar visualizando la vista de gráfica de tendencia. Se deben haber añadido al menos una variable en forma de serie a la gráfica y estar visualizandosé la misma.	
Pasos de ejecución: Se busca el ícono del sistema dentro del menú de aplicaciones del dispositivo y se da click sobre este. Una vez hecho esto se despliega el menú de navegación o en la vista principal se da click sobre la opción de *Tendencia*. Dar click sobre el ícono en la barra de herramienta que representa la funcionalidad en la vista de la gráfica de tendencia.	

Resultados esperados: Se debe eliminar todas las series representadas u ocultas en la gráficas dejando la gráfica.
Evaluación de la prueba: Satisfactoria

Caso de prueba de aceptación	
Código: PA09_HU18	**Historia de Usuario:** HU18
Nombre: Exportar el gráfico de tendencia a un formato de imagen.	
Descripción: Se comprobará el correcto funcionamiento de la funcionalidad de exportar el estado de la gráfica de tendencia a un formato de imagen.	
Condiciones de ejecución: El dispositivo debe tener instalado el sistema entre sus aplicaciones. El sistema debe haber sido iniciado. Se debe estar visualizando la vista de gráfica de tendencia.	
Pasos de ejecución: Se busca el ícono del sistema dentro del menú de aplicaciones del dispositivo y se da click sobre este. Una vez hecho esto se despliega el menú de navegación o en la vista principal se da click sobre la opción de *Tendencia*. Dar click sobre el ícono en la barra de herramienta que representa la funcionalidad en la vista de la gráfica de tendencia.	
Resultados esperados: Se debe visualizar en un díalogo emergente la dirección dentro del sistema de fichero la dirección donde fue guardada la imagen. Una vez abierta la imagen generada la misma debe representar el estado visual de la gráfica de tendencia tal y como estaba en el momento en que se ejecutó la funcionalidad.	
Evaluación de la prueba: Satisfactoria	

Caso de prueba de aceptación	
Código: PA10_HU17	**Historia de Usuario:** HU17
Nombre: Mostrar y ocultar leyenda con series visualizadas	

Descripción: Se comprobará el correcto funcionamiento de la funcionalidad de mostrar y ocultar la leyenda de la gráfica de tendencia.
Condiciones de ejecución: El dispositivo debe tener instalado el sistema entre sus aplicaciones. El sistema debe haber sido iniciado. Se debe estar visualizando la vista de gráfica de tendencia. Se deben haber añadido al menos una variable en forma de serie a la gráfica y estar visualizandosé la misma.
Pasos de ejecución: Se busca el ícono del sistema dentro del menú de aplicaciones del dispositivo y se da click sobre este. Una vez hecho esto se despliega el menú de navegación o en la vista principal se da click sobre la opción de *Tendencia*. Dar click sobre el ícono en la barra de herramienta que representa la funcionalidad en la vista de la gráfica de tendencia.
Resultados esperados: Si la leyenda está oculta se debe visualizar un cuadro de fondo gris en la esquina izquierda superior de la gráfica mostrando la etiqueta el valor actualizado de cada serie visualizada en ese instante en la gráfica, los valores mostrados deben cambiar a medida que se actualicen. Tanto la etiqueta y el valor de la serie el color de texto utilizado debe corresponderse con el utilizado para visualizar la serie en la gráfica. En caso de que la leyenda se muestre y se presiona el ícono que representa la funcionalidad, la leyenda debe ocultarse.
Evaluación de la prueba: Satisfactoria

Caso de prueba de aceptación	
Código: PA11_HU14	**Historia de Usuario:** HU14
Nombre: Mostrar y ocultar series de la gráfica de tendencia	
Descripción: Se comprobará el correcto funcionamiento de la funcionalidad de mostrar y ocultar series de la gráfica de tendencia.	

Condiciones de ejecución: El dispositivo debe tener instalado el sistema entre sus aplicaciones. El sistema debe haber sido iniciado. Se debe estar visualizando la vista de gráfica de tendencia. Se deben haber añadido al menos una variable en forma de serie a la gráfica y estar visualizandosé la misma.
Pasos de ejecución: Se busca el ícono del sistema dentro del menú de aplicaciones del dispositivo y se da click sobre este. Se autentica el dispositivo en el servidor. Una vez hecho esto se despliega el menú de navegación o en la vista principal se da click sobre la opción de *Tendencia*. Adicionar al menos una variable como serie a la gráfica de tendencia. Dar click sobre el ícono en la barra de herramienta que representa la funcionalidad en la vista de la gráfica de tendencia. Se despliega un cuadro de díalogo con un listado de todas las series adicionadas a la gráfica de tendencia. Aquellas que se están visualizando su casilla correspondiente estará marcada mientras las que están ocultas su casilla estarán desmarcadas. Para ocultar las series se deben desmarcar sus casillas con un click, mientras para mostrar se debe marcar sus casillas. Una vez listo se presiona el botón *Aceptar*
Resultados esperados: Una vez presionado el botón *Aceptar* se deben visualizar aquellas series que tenían sus casillas marcadas mientras deben ocultarse aquellas con casillas desmarcadas.
Evaluación de la prueba: Satisfactoria

Caso de prueba de aceptación	
Código: PA12_HU13	**Historia de Usuario:** HU13
Nombre: Eliminar series de la gráfica de tendencia	
Descripción: Se comprobará el correcto funcionamiento de la funcionalidad de eliminar series de la gráfica de tendencia.	

Condiciones de ejecución: El dispositivo debe tener instalado el sistema entre sus aplicaciones. El sistema debe haber sido iniciado. Se debe estar visualizando la vista de gráfica de tendencia. Se deben haber añadido al menos una variable en forma de serie a la gráfica y estar visualizandosé la misma.
Pasos de ejecución: Se busca el ícono del sistema dentro del menú de aplicaciones del dispositivo y se da click sobre este. Se autentica el dispositivo en el servidor. Una vez hecho esto se despliega el menú de navegación o en la vista principal se da click sobre la opción de *Tendencia*. Adicionar al menos una variable como serie a la gráfica de tendencia. Dar click sobre el ícono en la barra de herramienta que representa la funcionalidad en la vista de la gráfica de tendencia. Se despliega un cuadro de díalogo con un listado de todas las series adicionadas a la gráfica de tendencia. Aquellas series que se desean eliminar se deben marcar en su casilla. Una vez marcadas todas las series que deseamos eliminar se presiona el botón *Aceptar*
Resultados esperados: Una vez presionado el botón *Aceptar* se deben eliminar y por tanto dejar de visualizar en la gráfica de tendencia las series seleccionadas para eliminar.
Evaluación de la prueba: Satisfactoria

Caso de prueba de aceptación	
Código: PA13_HU07	**Historia de Usuario:** HU07
Nombre: Visualización de la gráfica de tendencia sin serie	
Descripción: Se comprobará la correcta visualización de la gráfica de tendencia sin ninguna serie añadida.	
Condiciones de ejecución: El dispositivo debe tener instalado el sistema entre sus aplicaciones.	

Pasos de ejecución: Se busca el ícono del sistema dentro del menú de aplicaciones del dispositivo y se da click sobre este. Una vez hecho esto se despliega el menú de navegación o en la vista principal se da click sobre la opción de *Tendencia*.
Resultados esperados: La visualización de la gráfica de tendencia con la barra de herramientas en la parte superior de la vista tal y como se describe en la HU. El fondo de la gráfica de color negro y el color de texto de las etiquetas de los ejes y los ejes en blanco. Cada un segundo y según el rango de tiempo visualizado en la gráfica se actualiza las etiquetas del eje X y las posiciones de estas.
Evaluación de la prueba: Satisfactoria

Caso de prueba de aceptación	
Código: PA14_HU06	**Historia de Usuario:** HU06
Nombre: Visualización del sumario de variable	
Descripción: Se comprobará la correcta visualización del sumario de variables con 100 variables.	
Condiciones de ejecución: El dispositivo debe tener instalado el sistema entre sus aplicaciones. Debe haber inciado sesión en un servidor que le ofrece 100 variables con un intervalo de tiempo de actualización de 500 ms.	
Pasos de ejecución: Se busca el ícono del sistema dentro del menú de aplicaciones del dispositivo y se da click sobre este. Se autentica el dispositivo en el servidor.Una vez hecho esto se despliega el menú de navegación o en la vista principal se da click sobre la opción de *Sumario de variable*.	

Resultados esperados: La visualización de 100 tuplas donde en cada tupla se ofrece la información resumida de cada variable. Los valores y la estampa de tiempo de las variables deben actualizarse o cambiar al menos una vez cada segundo.
Evaluación de la prueba: Satisfactoria

Caso de prueba de aceptación	
Código: PA15_HU08	**Historia de Usuario:** HU08
Nombre: Visualización de los detalles de la variable	
Descripción: Se comprobará la correcta visualización de los destalles de una variable.	
Condiciones de ejecución: El dispositivo debe tener instalado el sistema entre sus aplicaciones. Debe haber inciado sesión en un servidor que le ofrece al menos una variable con un intervalo de tiempo de actualización de 500 ms.	
Pasos de ejecución: Se busca el ícono del sistema dentro del menú de aplicaciones del dispositivo y se da click sobre este. Se autentica el dispositivo en el servidor.Una vez hecho esto se despliega el menú de navegación o en la vista principal se da click sobre la opción de *Sumario de variable. Una vez en esta vista se da un click extendido sobre una tupla del sumario. Presionar la opción* Detalles *en el menú contextual que se despliega*.	
Resultados esperados: La visualización de la vista detalles de una variable con toda información referente a la variable selecionada en el sumario de variables. El valor de la variable y la fecha de lectura del valor de la variable debe cambiar o actualizarse al menos una vez cada un segundo.	
Evaluación de la prueba: Satisfactoria	

Caso de prueba de aceptación	
Código: PA16_HU19	**Historia de Usuario:** HU19
Nombre: Comprobar visualización del estado de la conexión del dispositivo.	
Descripción: Se ejecutará el sistema en el dispositivo y se comprobará que el sistema es capaz de visualizar el estado de conexión que posee el dispositivo y actualizar dicho estado cuando ocurra un cambio.	
Condiciones de ejecución: El dispositivo debe tener instalado el sistema entre sus aplicaciones. El dispositivo no debe estar conectado a red.	
Pasos de ejecución: Se busca el ícono del sistema dentro del menú de aplicaciones del dispositivo y se da click sobre este. Una vez hecho esto se deja en ejecucción el sistema y se conecta el dispositivo a una red.	
Resultados esperados: Una vez iniciado el sistema el mismo debe visualizar en la barra de título del sistema en su parte derecha inferior el estado de la conexión del dipositivo el cual debe cumplir con lo especificado en la HU_19. Una vez conectado el dispositivo a una red la gráfica que representa el estado de conexión del sistema debe actualizar su representación y debe ser notificado al usuario del cambio del estado mediante un cuadro emergente.	
Evaluación de la prueba: Satisfactoria	

Caso de prueba de aceptación	
Código: PA17_HU02	**Historia de Usuario:** HU02
Nombre: Cargar configuración definida para el sistema desde un fichero	
Descripción: Se comprobará que el sistema sea capaz de leer la configuración definida en un fichero con extensión *json*	

Condiciones de ejecución: El dispositivo debe tener instalado el sistema entre sus aplicaciones. El sistema no debe haber cargado una configuración previa de un archivo. Dentro del dispositivo debe existir al menos archivo con una configuración predeterminada.
Pasos de ejecución: Se busca el ícono del sistema dentro del menú de aplicaciones del dispositivo y se da click sobre este. Se despliega una vista donde se listan todos los archivos con extensión *json* dentro de los directorios del dispositivos para escoger el que contenga la configuración. Una vez escogido el fichero se presiona el botón *Aceptar*.
Resultados esperados: Una vez que se acceda a la vista de *Preferencia* del sistema los valores defenidos para los campos deben coincidir con los valores de los campos defenidos en la configuración descrita en el fichero escogido.
Evaluación de la prueba: Satisfactoria

Caso de prueba de aceptación	
Código: PA18_HU10	**Historia de Usuario:** HU10
Nombre: Graficación del comportamiento de una variable en la gráfica de tendencia.	
Descripción: Se comprobará la correcta visualización del comportamiento de una variable en la gráfica de tendencia.	
Condiciones de ejecución: El dispositivo debe tener instalado el sistema entre sus aplicaciones. Debe haber inciado sesión en un servidor que le ofrece al menos una variable con un intervalo de tiempo de actualización de 500 ms.	

Pasos de ejecución: Se busca el ícono del sistema dentro del menú de aplicaciones del dispositivo y se da click sobre este. Se autentica el dispositivo en el servidor.Una vez hecho esto se despliega el menú de navegación o en la vista principal se da click sobre la opción de *Sumario de variable. Una vez en esta vista se da un click extendido sobre una tupla del sumario. Presionar la opción* Mostrar en tendencia *en el menú contextual que se despliega.*
Resultados esperados: La visualización del comportamiento de la variable como serie en una gráfica de tendencia que ofrezca las funcionalidades descritas en la HU10. El valor y representación de la variable debe cambiar al menos una vez cada un segundo.
Evaluación de la prueba: Satisfactoria

Caso de prueba de aceptación	
Código: PA19_HU12	**Historia de Usuario:** HU12
Nombre: Adicionar variable como serie a la gráfica de tendencia	
Descripción: Se comprobará que el sistema permitir adicionar una variable como serie a la gráfica de tendencia.	
Condiciones de ejecución: El dispositivo debe tener instalado el sistema entre sus aplicaciones. El sistema debe haber sido iniciado y haber inciado sesión en un servidor que le ofrece al menos una variable con un intervalo de tiempo de actualización de 500 ms.	

Pasos de ejecución: Se busca el ícono del sistema dentro del menú de aplicaciones del dispositivo y se da click sobre este. Se autentica el dispositivo en el servidor. Una vez hecho esto se despliega el menú de navegación o en la vista principal se da click sobre la opción de *Tendencia*. Dar click sobre el ícono en la barra de herramienta que representa la funcionalidad en la vista de la gráfica de tendencia. En el cuadro de díalogo escoger una variable, definir su etiqueta ,color y presionar el botón *Aceptar*.
Resultados esperados: En la gráfica de tendencia debe comenzar a visualizar el comportamiento de la variable seleccionada a través de un gráfico de líneas. El color de las líneas que representa los valores de esa variable se debe corresponder con el seleccionado previamente. Cada un segundo se debe representar al menos un nuevo valor de la variable.
Evaluación de la prueba: Satisfactoria

Caso de prueba de aceptación	
Código: PA20_HU20	**Historia de Usuario:** HU20
Nombre: Editar serie de la gráfica de tendencia	
Descripción: Se comprobará que el sistema permitir editar una serie de la gráfica de tendencia.	
Condiciones de ejecución: El dispositivo debe tener instalado el sistema entre sus aplicaciones. El sistema debe haber sido iniciado y haber inciado sesión en un servidor que le ofrece al menos una variable con un intervalo de tiempo de actualización de 500 ms. Se debe haber añadido al menos una serie a la gráfica de tendencia.	

Pasos de ejecución: Se busca el ícono del sistema dentro del menú de aplicaciones del dispositivo y se da click sobre este. Se autentica el dispositivo en el servidor. Una vez hecho esto se despliega el menú de navegación o en la vista principal se da click sobre la opción de *Tendencia*. Se adiciona una variable en forma de serie a la gráfica de tendencia. Dar click sobre el ícono en la barra de herramienta que representa la funcionalidad en la vista de la gráfica de tendencia. En el cuadro de díalogo escoger la serie que queremos editar, redefinir su etiqueta , color y presionar el botón *Aceptar*.
Resultados esperados: En la gráfica de tendencia el comportamiento de la serie editada debe visualizarse con el nuevo color definido. Si se muestra la leyenda de la gráfica la etiqueta de la serie debe coincidir con la definida en la edición y el color de texto debe ser igual al color selecionado en la edición.
Evaluación de la prueba: Satisfactoria

Anexo G: Dispositivos de prueba

A continuación los dispositivos utilizados en las pruebas de compatibilidad:

Dispositivos	
	-Marca rockchip modelo PLT7050. -CPU 2x ARM Cortex-A9 @ 1008 MHz. -Memoria RAM de 512 MB. -Almacenamiento interno 1003 MB -Pantalla táctil de 6.93 pulgadas en diagonal con una resolución de 800x400. -Android 4.4.2 - KitKat.
	-Marca Allwinner-Tablet modelo X7. -CPU 2x ARM Cortex-A7 @ 1536 MHz. -Memoria RAM de 512 MB. -Almacenamiento interno 1007 MB -Pantalla táctil de 6.96 pulgadas en diagonal con una resolución de 800x480. -Android 4.4.4 - KitKat.
	-Marca MID modelo M721S. -CPU 2x ARM Cortex-A7 @ 1536 MHz. -Memoria RAM de 512 MB. -Almacenamiento interno 1007 MB -Pantalla táctil de 6.98 pulgadas en diagonal con una resolución de 800x480. -Android 4.4.2 - KitKat.

	-Marca zte modelo ZTE V765M fabricante zte -CPU 2x ARM Cortex-A7 @ 1300 MHz. -Memoria RAM 512 MB -Almacenamiento interno 1304 MB -Pantalla táctil con una resolución de 480x800. -Android 4.2.2 - Jelly Bean.
	-Marca BLU modelo BLU STUDIO 5.0 C fabricante BLU -CPU 2x ARM Cortex-A7 @ 1300 MHz -Memoria RAM 512 MB LPDR2 -Almacenamiento interno 2485 MB -Pantalla táctil de 5.0 pulgadas en diagonal con una resolución de 480x854 -Android 4.4.2 - KitKat
	-Marca HUAWEI modelo CPN-L09 fabricante HUAWEI -CPU 4x ARM Cortex-A53 @ 1401 MHz, 4x ARM Cortex-A53 @ 1094 MHz -Memoria RAM 3 GB -Almacenamiento interno 23.96 GB -Pantalla táctil de 8.03 pulgadas en diagonal con una resolución de 1200x1920 -Android 7.0 - Nougat

	-Marca Samsung modelo Samsumg Galaxy J3 Prime fabricante Samsumg -CPU 4x ARM Cortex-A53 @ 1430 MHz -Memoria RAM 1.5 GB -Almacenamiento interno 11.95 GB -Pantalla táctil de 5.00 pulgadas en diagonal con una resolución de 1280x720 -Android 7.0 - Nougat