FACULTAD 1

CENTRO DE SOFTWARE LIBRE

Aplicación móvil para el monitoreo pasivo de Nova Servidores

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

Autor:

Reimer Darian Malleza Romero.

Tutores:

Ing. Hanny Valdés Hernández

Ing. Gustavo Quezada Arévalo.

LA HABANA, JUNIO 2019

AÑO 61 DE LA REVOLUCIÓN

“En la tierra hacen falta personas que trabajen más y critiquen menos, que construyan más y destruyan menos, que prometan menos y resuelvan más, que esperen recibir menos y dar más, que digan mejor ahora que mañana.

Ernesto Che Guevara

DECLARACIÓN DE AUTORÍA

Declaro por este medio que yo Reimer Darían Malleza Romero, con carné de identidad 94063031322 soy el autor principal del trabajo titulado Aplicación móvil para el monitoreo pasivo de Nova Servidores y autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Para que así conste firman la presente a los __ días del mes de junio del año 2019.

_____

Reimer Darían Malleza Romero

Autor

_____ _____

Ing. Gustavo Quesada Arévalo

Tutor

Ing. Hanny Valdés Hernández

Tutora

Datos del autor

Nombre y apellidos: Reimer Darían Malleza Romero

Correo electrónico: rdmalleza@estudiantes.uci.cu

Situación laboral: Estudiante.

Institución a la que pertenece: Universidad de las Ciencias Informáticas.

Dirección: Carretera a San Antonio de los Baños, Torrens, Municipio Boyeros, Ciudad de la Habana, Cuba, Código Postal 19370.

Datos de los tutores

Nombre y apellidos: Hanny Valdés Hernández

Correo electrónico: hanny@uci.cu

Especialidad de graduación: Ingeniero en Ciencias Informáticas.

Institución a la que pertenece: Universidad de las Ciencias Informáticas.

Dirección: Carretera a San Antonio de los Baños, Torrens, Municipio Boyeros, Ciudad de la Habana, Cuba, Código Postal 19370**.**

Nombre y apellidos: Gustavo Quesada Arévalo

Correo electrónico: gquezada@uci.cu

Especialidad de graduación: Ingeniero en Ciencias Informáticas.

Institución a la que pertenece: Universidad de las Ciencias Informáticas.

Dirección: Carretera a San Antonio de los Baños, Torrens, Municipio Boyeros, Ciudad de la Habana, Cuba, Código Postal 19370**.**

Dedicado a mi familia, especialmente a mi mamá y a mi hermano que siempre me apoyaron en todos estos 5 años.

Agradecer a mi familia por todo el apoyo incondicional todo este tiempo.

A mis amigos que se convirtieron en otra familia para mi.

A mis tutores por toda la paciencia que tuvieron para guiarme en este proceso.

A todos los profesores que tuve la oportunidad de conocer ya cada uno aporto algo a mi formación.

Resumen

Como parte del proceso de desarrollo de software y la migración nacional a software libre surge Nova Servidores, variante de la distribución cubana GNU/Linux Nova desarrollada en el Centro de Software Libre de la Universidad de las Ciencias Informáticas para servidores. En dicha variante existen una herramienta llamada zabbix encargada del monitoreo y control de los recursos, aplicaciones y servicios de los servidores. El presente trabajo de diploma tuvo como objetivo desarrollar una aplicación móvil para el monitoreo pasivo de Nova Servidores. Para ello se analizaron herramientas móviles utilizadas actualmente para realizar el monitoreo de servidores con zabbix. Además, se documentaron las tecnologías, herramientas, lenguajes a utilizar en la construcción de la aplicación y la definición de los elementos necesarios para el exitoso desarrollo de la misma, así como los artefactos requeridos por la metodología de desarrollo Variación del Proceso Unificado Ágil para la Universidad de las Ciencias Informáticas. Además esta aplicación fue sometida a pruebas de software para verificar su calidad y correcto funcionamiento. Al concluir la investigación se obtuvo una aplicación móvil para el monitoreo pasivo de Nova Servidores, que permite alertar a los administradores de red ante fallas en los servidores y visualizar el estado de los recursos y servicios.

Palabras clave: aplicación móvil, ionic, monitoreo pasivo, Nova Servidores, servidores, zabbix.

Índice

Introducción

En la actualidad las Tecnologías de la Información y las Comunicaciones (TIC) son cada vez más usadas para el apoyo y automatización de todas las actividades de las empresas. Gracias a ellas, las organizaciones han conseguido obtener importantes beneficios, entre los que caben mencionar la mejora de sus operaciones, alcance a una mayor cantidad de clientes, la optimización de sus recursos, la apertura a nuevos mercados, un conocimiento más profundo acerca de las necesidades existentes para brindarles un servicio de mejor calidad y una comunicación más fluida. En pocas palabras, las TIC permiten aumentar considerablemente la eficiencia de los procesos que se realizan en las empresas (Aniel, 2013).

Aquellas con un amplio número de miembros adoptan prácticas asociadas a la prestación de determinados servicios como correo, mensajería instantánea y publicación de información utilizando servidores centrales. En la actualidad existen dos factores que han dado lugar a un aumento en el número de servidores a administrar, el crecimiento de la información almacenada de manera virtual y del uso masivo de la computación. Incluso en pequeñas y medianas empresas, se puede encontrar una persona que administre entre uno o más servidores, números que podrían aumentar en dependencia de la cantidad de servicios que se ofrezcan.

Con ese aumento que hoy se percibe en los servidores no conocer la acerca del tráfico que atraviesa la red, qué enlace está saturando el ancho de banda o que servicio está haciendo que la carga de los mismos sea elevada hace imposible tener una red de telecomunicaciones óptima ya que en cualquier momento los servidores pueden caerse y detener servicios de vital importancia para la comunicación de la empresa. Por lo que se hace necesario realizar un monitoreo constante y preciso que permita aprovechar al máximo los recursos de hardware, prevenir incidencias y detectar los problemas cuanto antes, además de ahorrar costes y tiempo, monitoreo tanto con *software* automático como también mediante chequeos manuales.

El monitoreo de servidores es una tarea imprescindible para que el cliente/usuario pueda estar tranquilo, sabiendo que hay alguien que se está encargando de velar por su servidor, sin importar el horario o la fecha. Existen una gran cantidad de herramientas para realizar el monitoreo, estas pueden ser sistemas *web,* aplicaciones de escritorio y aplicaciones móviles. Estas últimas muy utilizadas en la actualidad debido a su portabilidad que brindan los dispositivos móviles y el intercambio constante de estos dispositivos con las personas.

Cuba, en aras de ganar en soberanía tecnológica, así como garantizar la informatización de todas las esferas de la sociedad, inició su incursión en el año 2002 en el desarrollo del Proyecto Futuro, nombre dado por el Comandante en Jefe Fidel Castro Ruz a la Universidad de las Ciencias Informáticas (UCI). En la actualidad la Universidad cuenta con un total de 14 centros de desarrollo de software, dentro de ellos se encuentra el Centro de *Software* Libre (CESOL), cuyo objetivo es el desarrollo de la Distribución Cubana de GNU/Linux Nova. La selección de Nova para la migración del país responde a las necesidades de la informatización segura de la sociedad cubana.

La distribución cubana de GNU/Linux Nova tiene como una de sus variantes Nova Servidores orientada a servidores usando estándares abiertos y adaptada a los entornos de las empresas cubanas. Entre sus características se encuentran la configuración fácil e intuitiva a través de la herramienta *nova-manager* destinada a la administración de los servicios telemáticos y la compatibilidad con *hardware* obsoleto en el entorno empresarial (Nova, 2016). CESOL ha desempeñado un papel fundamental en el proceso de migración debido que sus especialistas son los encargados de llevarlo a cabo. Para ello realizan un diagnóstico para determinar el estado tecnológico en las instituciones, luego proceden a la instalación de Nova y sus herramientas, entre ellas Zabbix, la cual garantiza seguimiento de los recursos y servicios de los servidores.

En las empresas cubanas donde los especialistas de CESOL han desplegado Nova Servidores se han detectado que los servidores están distribuidos en diferentes áreas de la entidad y algunas cuentan con pocos administradores de red. En empresas de gran infraestructura existen varios administradores de red encargados de vigilar el estado los servidores y la seguridad informática, además un grupo de soporte técnico que se encarga de las tareas de mantenimiento y control de los activos físicos tecnológicos, cableado de red, computadoras e impresoras. En entidades de baja infraestructura el administrador de red cumple las funciones que le corresponden y también se desempeña como soporte técnico por lo que no puede estar pendiente al estado de los servidores a tiempo completo. Actualmente los administradores que utilizan Zabbix para realizar el monitoreo de los recursos y servicios en servidores, han demostrado que si no se encuentran físicamente en la estación de trabajo no pueden detectar la ocurrencia de fallas de algunos de estos servicios ya que no cuentan con una herramienta que les permita el monitoreo remoto de sus servidores. Esta situación provoca:

- Falta de control de los recursos de los servidores.

- Falta de disponibilidad de servicios e información.
- Demora en la solución de problemas relacionados con los servidores.

Lo antes planteado ocasiona afectaciones directas en los procesos fundamentales de las instituciones lo que trae consigo pérdidas de tiempo y baja productividad de las mismas.

Teniendo en cuenta la situación anterior como punto de partida surge el siguiente **problema de investigación**: ¿Cómo contribuir al monitoreo pasivo de Nova Servidores desde dispositivos móviles?
Para dar respuesta a la problemática planteada la presente investigación centra su **objeto de estudio** en el proceso de monitoreo pasivo en servidores, delimitándose como **campo de acción** el proceso de monitoreo pasivo en Nova Servidores desde dispositivos móviles.
Se establece entonces como **objetivo general**: desarrollar una aplicación móvil para el monitoreo pasivo de Nova Servidores.
Este se desglosa en los siguientes **objetivos específicos**:

1. Elaborar el marco teórico de la investigación sobre el proceso de monitoreo pasivo de servidores.
2. Diseñar una aplicación móvil para el monitoreo pasivo de Nova Servidores.
3. Implementar una aplicación móvil para el monitoreo pasivo de Nova Servidores.
4. Evaluar la aplicación móvil para el monitoreo pasivo de Nova Servidores, mediante pruebas de software y un estudio de caso.

Con el propósito de dar cumplimiento a lo previamente planteado se definen las siguientes **tareas de investigación**:

1. Análisis de las características de las soluciones informáticas existentes para el monitoreo pasivo de servidores.
2. Selección de las herramientas, estándares, metodología y tecnologías que se necesitan para el desarrollo de la propuesta de solución.
3. Elaboración de los artefactos requeridos por la metodología de desarrollo de software.
4. Implementación de las funcionalidades de la aplicación móvil para el monitoreo pasivo de Nova Servidores.
5. Evaluación mediante una estrategia de pruebas de software y un estudio de caso, de la aplicación

móvil para el monitoreo pasivo de Nova Servidores.

Para el desarrollo de la presente investigación se hace uso de los siguientes **métodos científicos**:

Métodos teóricos

- **Analítico-Sintético**: El análisis permite la división mental del fenómeno en sus múltiples relaciones y la síntesis establece mentalmente la unión entre las partes previamente analizadas, posibilita descubrir sus características generales y las relaciones esenciales entre ellas (Hernández León, et al., 2011). La utilización de este método permite realizar el análisis teórico e identificar los principales conceptos a incluir en la fundamentación teórica y el análisis de la información, permitiendo extraer los elementos importantes relacionados con el monitoreo pasivo de servidores.
- **Inductivo-Deductivo:** Se empleó para arribar a razonamientos que puedan ser aplicables al problema a resolver luego de adquirir una serie de elementos relacionados con el monitoreo pasivo en Nova Servidores.

Métodos empíricos:

- **Entrevista:** La entrevista es una conversación planificada entre el investigador y el entrevistado para obtener información (Hernández León, et al., 2011). Se emplea con miembros del grupo de desarrollo sobre diferentes temas relacionados con la investigación para obtener ideas, referencias e información útil para elaborar los requisitos.
- **Observación:** La observación científica es la percepción planificada dirigida a un fin y relativamente prolongada de un hecho o fenómeno. Es el instrumento universal del científico, se realiza de forma consciente y orientada a un objetivo determinado (Hernández León, et al., 2011). Su empleo permite observar y estudiar el proceso de monitoreo pasivo de servidores determinando las principales deficiencias y ventajas de su empleo.

El presente trabajo de diploma se encuentra dividido en tres capítulos estructurados de la siguiente forma:

Capítulo I. Fundamentación teórica: Se definen los principales conceptos relacionados con el tema de investigación para lograr un mejor entendimiento del problema a resolver, lo que incluye un análisis de las tecnologías homólogas existentes que contribuyen proceso de monitoreo pasivo de servidores, resultando en la selección de una de éstas para llevar a cabo la solución propuesta. Además, se realiza una descripción de la metodología, las herramientas y tecnologías a emplear durante el desarrollo de la propuesta de solución.

Capítulo II: **Análisis y diseño de la aplicación móvil para el monitoreo pasivo de Nova Servidores**. En este capítulo se describen elementos como: la propuesta de solución de la aplicación a desarrollar, el modelo de dominio, técnicas de captura y validación de requisitos, requisitos funcionales y no funcionales, diseño arquitectónico a emplear en el desarrollo de la solución, diagrama de clases del diseño según el marco de trabajo que se utiliza y patrones de diseño.

Capítulo III: **Implementación y prueba de la aplicación móvil aplicación móvil para el monitoreo pasivo de Nova Servidores**. En este capítulo se comienza con el diseño de casos de prueba para realizar la validación de la aplicación desarrollada, se muestran los resultados de las pruebas realizadas a la solución informática obtenidos mediante las estrategias de prueba. Las mismas se dividen en niveles de pruebas y cada nivel tiene un método de prueba. De estos métodos se explica su funcionamiento y resultados a través de las pruebas realizadas al software para detectar errores relacionados con sus funcionalidades y para comprobar la veracidad del código.

Capítulo 1.Fundamentación teórica

En el presente capítulo se analizan los conceptos y elementos teóricos relacionados con el proceso monitoreo pasivo de servidores. Se plasma además el resultado del estudio realizado a las principales soluciones que resuelven de forma parcial el problema de investigación planteado. Por último, se describe la metodología seleccionada para guiar el proceso de desarrollo de software y las tecnologías que se seleccionaron para el desarrollo de la solución.

1.1 Conceptos asociados al dominio del problema

Con el objetivo de realizar un correcto proceso de investigación se detallan a continuación los principales conceptos asociados al dominio del problema.

1.1.1 Monitoreo

El monitoreo es el proceso de mantener la vigilancia sobre la existencia y magnitud de cambio de estado y flujo de datos en un sistema informático. Tiene como objetivo identificar fallas y ayudar en su posterior eliminación. Las técnicas utilizadas en el control de la información de los sistemas informáticos se cruzan con los campos de procesamiento en tiempo real, estadísticas y análisis de datos. Un conjunto de componentes de software utilizados para la recopilación de datos, su procesamiento y presentación es llamado un sistema de monitoreo (LIGUS, 2013).

Monitoreo pasivo:

Este enfoque se basa en la obtención de datos a partir de recolectar y analizar el tráfico que circula por la red. Se emplean diversos dispositivos como *sniffers*[1], ruteadores, computadoras con software de análisis de tráfico y en general dispositivos con soporte para SNMP[2], RMON[3] y Netflow[4]. Este enfoque no agrega

[1] Programa informático que registra la información que envían los periféricos, así como la actividad realizada en un determinado ordenador.

[2] Protocolo de la capa de aplicación que facilita el intercambio de información de administración entre dispositivos de red. Es parte de la familia de protocolos TCP/IP.

[3] Es un estándar que define objetos actuales e históricos de control, permitiendo que usted capture la información en tiempo real a través de la red entera

[4] Herramienta de monitorización de ancho de banda basado en tecnología web.

tráfico a la red como lo hace el activo y es utilizado para caracterizar el tráfico en la red y para contabilizar su uso (Romero, 2018).

Técnicas de monitoreo pasivo:

1. **Solicitudes remotas:**

Mediante SNMP:

Esta técnica es utilizada para obtener estadísticas sobre la utilización de ancho de banda en los dispositivos de red, para ello se requiere tener acceso a dichos dispositivos. Al mismo tiempo, este protocolo genera paquetes llamados *traps* que indican que un evento inusual se ha producido.

2. **Otros métodos de acceso:**

Se pueden realizar scripts[5] que tengan acceso a dispositivos remotos para obtener información importante a monitorear.

Captura de tráfico:

Se puede llevar a cabo de dos formas:

1) Mediante la configuración de un puerto espejo en un dispositivo de red, el cual hará una copia del tráfico que se recibe en un puerto hacia otro donde estará conectado el equipo que realizará la captura.
2) Mediante la instalación de un dispositivo intermedio que capture el tráfico, el cual puede ser una computadora con el software de captura o un dispositivo extra. Esta técnica es utilizada para contabilizar el tráfico que circula por la red.

Análisis del tráfico:

Se utiliza para caracterizar el tráfico de red, es decir, para identificar el tipo de aplicaciones que son más utilizadas. Se puede implementar haciendo uso de dispositivos *probe* que envíen información mediante RMON o a través de un dispositivo intermedio con una aplicación capaz de clasificar el tráfico por aplicación, direcciones IP origen y destino, puertos origen y destino.

[5] En informática, un *script* es un guión o conjunto de instrucciones que permiten la automatización de tareas creando pequeñas utilidades.

Flujos:

También utilizado para identificar el tipo de tráfico utilizado en la red. Un flujo es un conjunto de paquetes con:

- La misma dirección
- El mismo puerto TCP origen y destino
- El mismo tipo de aplicación.

Los flujos pueden ser obtenidos de ruteadores o mediante dispositivos que sean capaces de capturar tráfico y transformarlo en flujos. También es usado para tareas de facturación.

Estrategias de monitoreo:

Antes de implementar un esquema de monitoreo se deben tomar en cuenta los elementos que se van a monitorear, así como las herramientas que se utilizarán para esta tarea.

¿Qué monitorear?

Una consideración muy importante es delimitar el espectro sobre el cual se va a trabajar. Existen muchos aspectos que pueden ser monitoreados, los más comunes son los siguientes:

1. Utilización de ancho de banda
2. Consumo de CPU[6].
3. Consumo de memoria.
4. Estado físico de las conexiones.
5. Tipo de tráfico.
6. Alarmas.
7. Servicios (Web, correo, bases de datos, *proxy*).

1.1.2 Monitoreo de red

Sistema que realiza un control constante de una red de ordenadores, intentando detectar defectos y anomalías; en caso de encontrar algún desperfecto, envía un informe a los administradores (Ana Gardey, Julián Pérez Porto, 2013).

[6] Por sus siglas del inglés Central Processor Unit, al español *Unidad Central de Procesamiento*, es un componente básico de la computadora.

1.1.3 Alerta

Es la capacidad de un sistema de monitoreo para detectar y notificar a los operadores sobre eventos significativos que denotan un grave cambio de estado. La notificación se conoce como alerta y es un mensaje simple que puede adoptar múltiples formas: correo electrónico, SMS[7], mensaje instantáneo o una llamada telefónica. La alerta se transmite al destinatario apropiado, es decir, una parte responsable de tratar el evento. La alerta a menudo se registra en forma de boleto en un Sistema de Seguimiento de Problemas (Ligus, 2013).

1.1.4 Aplicación Móvil

Son programas diseñados para ser ejecutados en teléfonos, tabletas y otros dispositivos móviles, que permiten al usuario realizar actividades profesionales, acceder a servicios, mantenerse informado, entre otro universo de posibilidades (Servisoftcorp, 2019).

1.2 Estudio de herramientas homólogas a la investigación

En el presente epígrafe se realiza un análisis de las soluciones que resuelven de forma parcial el problema de investigación, desarrolladas a nivel internacional. Todas las aplicaciones analizadas fueron desarrolladas para Zabbix, ya que es la herramienta que trae Nova Servidores para realizar el monitoreo. Durante todo el proceso se identificaron 25 soluciones informáticas, todas fueron aplicaciones móviles, 14 de estas soluciones no contaban con código fuente disponible y muy poca documentación por lo que se desechan de la investigación, otras 6 eran de carácter privativo por lo que no se tenía acceso ni a documentación ni a código. En resumen se determinaron para la investigación 5 aplicaciones móviles de las cuales se tienen en cuenta las siguientes características:

- Uso de la API de Zabbix.
- Soporte para zabbix
- Tipo de autenticación.
- Disponibilidad de código fuente.

[7] El servicio de mensajes cortos o servicio de mensajes simples, más conocido como SMS, es un servicio disponible en los teléfonos móviles que permite el envío de mensajes cortos, conocidos como mensajes de texto, entre teléfonos móviles

- Funcionamiento fuera de línea
- Plataformas.

1.2.1 ZAX Zabbix *Systems Monitoring*

ZAX es una interfaz móvil para el sistema de monitoreo empresarial Zabbix. Es gratis y no tiene límites. Esta aplicación muestra no solo los activadores activos, los eventos y los datos más recientes de los artículos, sino que también tiene algunas características más:

- Muestra problemas actuales.
- Muestra eventos.
- Permite filtrar eventos y problemas por grupo de servidores y/o severidad.
- Muestra los últimos artículos.
- Muestra pantallas.
- Historial de artículos (gráficos).
- HTTP[8] autenticación para proxy.
- Conexión HTTPS[9] / SSL[10] (confiable y opcionalmente no confiable).
- Soporte para Zabbix 2.x y 3.

1.2.2 ZabbixAlert

Sencilla aplicación para obtener notificaciones de activadores de Zabbix.
Características:

- Mostrar solo los activadores activos.
- Nivel de alerta mínimo configurable para recibir notificación.
- Sonidos configurables para diferentes niveles de alerta.

[8] *Hypertext Transfer Protocol*, 'protocolo de transferencia de hipertextos', que se utiliza en algunas direcciones de internet.

[9] Hypertext Transfer Protocol Secure, 'Protocolo seguro de transferencia de hipertexto', que se utiliza en algunas direcciones de internet.

[10] Secure Sockets Layer es un protocolo diseñado para permitir que las aplicaciones para transmitir información de ida y de manera segura hacia atrás

- Widget[11] 4x4 para un fácil monitoreo.
- No requiere ningún servicio de envío de notificaciones externo.
- Soporta múltiples servidores Zabbix.
- Soporta HTTP y HTTPS.
- Alarma de repeticiones programables.
- Zabbix 2.0 o superior.

1.2.3 Andzabbix

Cliente nativo de Androide para el sistema de monitoreo Zabbix. El Andzabbix le ayuda a saber siempre el estado de los desencadenantes. La mejor aplicación de Androide para el sistema de monitorización Zabbix.

Características:

- Muestra una lista de activadores activos
- Verificación de fondo activadores activos
- Notificación sobre nuevos activadores activos
- Actualización automática lista de desencadenantes activos
- Control de activadores simples (Deshabilitar, eliminar, activar)
- Mostrar lista de eventos para disparadores. Reconocer eventos.
- Las pantallas son listas de hosts con resumen de elementos
- Mostrar todos los gráficos
- Soporte para servidor múltiple
- Conexiones SSL
- Autenticación básica
- Soporte completo para Zabbix 2.0.x
- Gráficos simples en la página Resumen de todos los artículos.

También características de la versión Pro:

- Mostrar mapas
- Filtro de host y clasificación para Descripción

[11] Son pequeñas aplicaciones cuyo objetivo es dotar de información visual y facilitar el acceso a las funciones que se utilizan de forma frecuente.

- Crear, modificar, borrar usuarios
- Crear, modificar, borrar hosts
- Crear, modificar, eliminar grupos de usuarios
- Soporte algunas manipulaciones con scripts, acciones, pantallas.
- Widgets para pantallas de inicio y bloqueo.
- Ejecutar scripts en el mapa
- Comentarios de eventos (mostrar y agregar nuevos)

1.2.4 Zbx. Zabbix *Monitoring Client*

Zbx es un cliente nativo para el sistema de monitoreo Zabbix. Un cliente móvil, sencillo y gratuito. Zbx trabaja directamente con la API de Zabbix.

Características:

- Atractiva pantalla de inicio con información directa.
- Muestra una lista de activadores activos.
- Notificaciones completas en problemas desencadenantes.
- Verificaciones de antecedentes de activadores activos, con notificaciones directas.
- Soporta la autenticación htaccess[12].
- Mostrar toda la información sobre un host.
- Gráficos simples para todos los artículos.
- Búsqueda avanzada de hosts.

1.2.5 Zabbkit

ZabbKit es una aplicación cliente para el sistema de monitoreo Zabbix. Siempre lo mantendrá actualizado sobre las condiciones de su servidor mediante las notificaciones de GCM[13].
Con ZabbKit puede recibir de forma remota toda la información necesaria y, lo que es más importante, poder resolver problemas problemáticos inmediatamente después de que suceda algo. La aplicación es rápida de

[12] Abreviatura de Hypertext Access. Se trata de un archivo de configuración utilizado por servidores web basados en apache.

[13] *Global System for Mobile communications* (Sistema Global para las comunicaciones Móviles), es el sistema de teléfono móvil digital más utilizado y el estándar de facto para teléfonos móviles en Europa.

personalizar y fácil de usar. La aplicación admite un número ilimitado de servidores Zabbix con validación de URL[14] obligatoria.

Las principales características de la aplicación son:

- Visualización del estado de los disparadores;
- Visualización de la lista de eventos, datos del host y gráficos;
- Creación de listas de activadores, activadores favoritos y gráficos favoritos;
- Creación de gráficos para hosts separados y para grupos de hosts;
- Compatibilidad con certificados SSL auto firmados;
- Recibiendo notificaciones *push* de su servidor Zabbix.

1.2.6 Comparaciones entre las herramientas estudiadas.

Como resultado del análisis anterior y a modo resumen se muestra la *Tabla 1* donde las anteriores aplicaciones son comparadas.

Tabla 1: Comparación de herramientas similares a la propuesta de solución (Fuente: Elaboración propia)

	ZAX Zabbix Systems Monitoring	**ZabbixAlert**	**Andzabbix**	**Zbx. Zabbix Monitoring Client**	**Zabbkit**
Soporte para zabbix	2x y 3	2.0 o superior	2.0.x	2.0.x	3.0
Funcionamiento fuera de línea	No	No	Si	No	No
Uso api de Zabbix	No	No	No	Si	No
Disponibilidad de código fuente	No	Si	No	No	Si
Plataformas	Android y iOs	Android	Android	Android	Android y iOs
Tipo de autenticación	HTTP	Soporta HTTP y HTTPS	Básica	htaccess	Soporta HTTP y HTTPS

[14] En inglés de *Uniform Resource Locator*, que en español significa Localizador Uniforme de Recursos**,** es la dirección específica que se asigna a cada uno de los recursos disponibles en la red

Una vez realizado el análisis comparativo, se puede concluir que:

- Son herramientas que han sido desarrolladas por programadores independientes para entornos específicos
- **Andzabbix** y **Zabbkit** son las únicas que se encuentra disponible en idioma español.
- **Zbx. Zabbix Monitoring Client** y **Zabbkit** están disponibles para *android* 4.4 o superior, las restantes no soportan las versiones superior a 4.2.
- **Andzabbix** y **Zbx. Zabbix Monitoring Client** no tienen soporte para zabbix superior a 2.0.x.

Por tanto se decide la creación de una herramienta que responda a las dificultades expuestas en la problemática, además de contribuir al desarrollo de aplicaciones propias y eliminar la dependencia de software externo.

Se extraen de las aplicaciones estudiadas funcionalidades claves para el desarrollo de la solución como son: visualización de recursos, tales como, carga de CPU, RAM, uso de disco, actividad de red y envió de notificaciones.

1.3 Metodología de desarrollo de software

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos de software. Van indicando paso a paso todas las actividades a realizar para lograr el producto informático deseado, señalando qué personas deben participar en el desarrollo de las actividades y qué papel deben tener. Además, detallan la información que se debe producir como resultado de una actividad y la información necesaria para comenzarla (BARZANALLA, 2016). A continuación, se describe la metodología de desarrollo de software que se va a emplear para llevar a cabo la presente investigación.

Variación de AUP para la UCI es una variación de la metodología Proceso Unificado Ágil (AUP, del inglés *Agile Unified Process*), esta se adapta al ciclo de vida definido para la actividad productiva de la UCI. Esta metodología se apoya en la Integración de Modelos de Madurez de Capacidades para Desarrollo (CMMI-DEV, del inglés *Capability Maturity Model Integration Development*). El cual constituye una guía para aplicar las mejores prácticas en una entidad desarrolladora (RODRÍGUEZ, 2014).

Esta metodología se adapta al ambiente de trabajo y le permite al cliente acompañar al equipo de desarrollo para convenir los requisitos y así poder implementarlos. El escenario escogido para esta investigación es el

número cuatro, pues la solución no posee un negocio, este escenario se utiliza para modelar el sistema. Además, se puede concluir que esta metodología se ajusta a cualquier proyecto productivo de la UCI.

1.4 Herramientas

Para llevar a cabo la solución que se propone, dar cumplimiento a los objetivos de esta investigación, y teniendo en cuenta las características del producto que se desea obtener; es necesario realizar una descripción de las herramientas y tecnologías implicadas en el proceso de desarrollo del software. A continuación, se describen las tecnologías y herramientas de desarrollo que se utilizarán para realizar la propuesta de solución.

1.4.1 Zabbix

Zabbix es un software de nivel empresarial diseñado para monitoreo de alta disponibilidad y rendimiento de toda la infraestructura informática. Es de código abierto y no tiene costo. Con Zabbix es posible reunir virtualmente ilimitadamente información de la red. Alto rendimiento de monitoreo en tiempo real de miles de servidores, máquinas virtuales y dispositivos de red pueden ser monitorizados simultáneamente. Junto con el almacenamiento de datos, están disponibles características de visualización (mapas, gráficos).También esta herramienta contiene un API (*Application Programming Interface*) para permitir la unión con terceros, o sea permite que se integren soluciones alternativas para la especificación y control de variables de monitoreo (Zabbix, 2018).

¿Qué se puede hacer con Zabbix?

- Agregar y monitorear servidores, equipos, servicios, aplicaciones específicas, dispositivos físicos como impresoras, *routers*.
- Reporte en tiempo real a través de gráficas, datos y alertas visuales que muestran el estado y rendimiento de los servicios y equipos monitoreados.
- Inventario de equipos para mantener al día la infraestructura tecnológica.
- Mapas de la red de la empresa.
- Configuración de notificaciones vía correo electrónico.
- Perfiles de usuarios para el uso del administrador.

Ventajas:

- Interfaz basada en la web.
- Reportes detallados.
- Fácil configuración.
- Estadísticas en tiempo real del estado de los servidores.
- Reduce los costos de operación al evitar el tiempo de inactividad.

1.4.1.1 API de Zabbix

La API de Zabbix comienza a desempeñar un papel importante, especialmente cuando se trata de la integración de Zabbix con software de terceros como la configuración y los sistemas de gestión de incidentes, así como para la automatización de tareas rutinarias.

La API se introdujo en Zabbix 1.8 y ya se usa ampliamente. Todos los clientes móviles de Zabbix se basan en la API, incluso la interfaz web nativa. El software intermedio API hace que la arquitectura sea más modular y ayuda a evitar llamadas directas a la base de datos.

La API de Zabbix proporciona dos funciones principales:

- Gestión remota de la configuración de Zabbix.
- Recuperación remota de datos históricos y de configuración.

1.4.2 Lenguaje de modelado

Se utiliza UML en su versión 2.0 como lenguaje de modelado, ya que el mismo permite al equipo de desarrollo visualizar, especificar, construir y documentar los artefactos del sistema. Esto proporciona una forma estándar de escribir los planos del mismo y cubrir tanto las cosas conceptuales, tales como procesos del negocio y funciones del sistema, como las cosas concretas, tales como las clases escritas en un lenguaje de programación específico, esquemas de bases de datos y componentes software reutilizables (JACOBSON, et al., 2016)

1.4.3 Herramienta CASE

Es una herramienta CASE (*Computer Aided Software Engineering*) Ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas (Pressman, 2011).

Características:

- Disponibilidad en múltiples plataformas (Windows, Linux).
- Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Diagramas de Procesos de Negocio - Proceso, Decisión, Actor de negocio, Documento.

1.4.4 Lenguaje de programación

TypeScript

Es un lenguaje de programación de código abierto con herramientas de programación orientada a objetos, muy favorable si se tienen proyectos grandes. Convierte su código en JavaScript común. Un lenguaje basado en el original, ofreciéndonos grandes beneficios como el descrito anteriormente, aunque existen otros beneficios. Es muy similar a JavaScript y a C# gracias a que su creador posee conocimientos de ambos lenguajes (Caceres, 2018).

HTML5

HTML surgió en el año 1991 como un documento que detallaba elementos utilizados para construir páginas web. Muchos de estos elementos eran para describir el contenido de una página web, como encabezados, párrafos y listas. Las características y capacidades de HTML han aumentado a medida que el lenguaje ha evolucionado con la introducción de otros elementos y ajustes a sus reglas. HTML5, la versión más reciente, es una evolución natural de versiones anteriores y se enfoca en reflejar las necesidades de los sitios web actuales y futuros. Un principio clave de diseño de HTML5 lo constituye ser compatible con varios navegadores. Esta tecnología incorpora nuevas características. Dentro de las más sencillas se encuentran elementos adicionales (article, section, figure) que se emplean para describir el contenido. Las más complejas ayudan a crear eficaces aplicaciones web, como la reproducción nativa de audio y video (Castro, et al., 2012).

CSS3

CSS es un lenguaje de estilos empleado para definir la presentación, el formato y la apariencia de un documento de marcado, sea HTML, XML[15]. Comúnmente es empleado para proporcionar formato visual a documentos HTML y XHTML[16] (Collell Puig).

CSS3 es más potente que las versiones anteriores de CSS, abarcando diseño, estilos web, forma y movimiento de los elementos. La especificación de CSS3 es presentada en módulos que permiten a la tecnología proveer una especificación estándar por cada aspecto involucrado en la presentación visual del documento. El empleo de CSS3 permite lograr transformaciones y reposicionamiento de los elementos presentados en la pantalla y la creación de esquinas redondeadas y sombras (Gauchat, 2012).

Sass

Sass (acrónimo de *Syntactically Awesome StyleSheets*) es una extensión de CSS que añade características muy potentes y elegantes a este lenguaje de estilos. Sass permite el uso de variables, reglas CSS anidadas, *mixins*, importación de hojas de estilos y muchas otras características, al tiempo que mantiene la compatibilidad con CSS.

Características:

- 100% compatible con CSS3.
- Permite el uso de variables, anidamiento de estilos y *mixins*.
- Incluye numerosas funciones para manipular con facilidad colores y otros valores.
- Permite el uso de elementos básicos de programación como las directivas de control y las librerías.
- Genera archivos CSS bien formateados y permite configurar su formato.

1.4.5 Marco de trabajo

Un marco de trabajo (del inglés, *framework)* 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. En la presente investigación se escogieron dos *framework* teniendo en cuenta sus características: presentan una curva de aprendizaje fácil, tiempo de desarrollo corto, son de código abierto (ROUSE, 2019).

[15] Del inglés eXtensible Markup Language, traducido como "Lenguaje de Marcado Extensible".

[16] Acrónimo de eXtensible Hypertext Markup Language.

Ionic 4

Ionic es framework de código abierto para el desarrollo de aplicaciones móviles híbridas. La versión inicial fue presentada en el año 2013 y construida sobre Angular y Apache Cordova. Dentro de las versiones más recientes se encuentran: Ionic 2 construido sobre Angular 2 e Ionic 3 basado en Angular 4. Ionic proporciona herramientas y servicios para desarrollar aplicaciones móviles híbridas empleando tecnologías web como CSS3, HTML5 y Sass[17]. Las aplicaciones se distribuyen a través de tiendas de aplicaciones nativas para que se instalen en dispositivos aprovechando Cordova. En 2015, se crearon más de 1,3 millones de aplicaciones Ionic (Dylan Schiemann, 2019).

Angular 7

La primera versión de Angular proporcionó a los programadores las herramientas para desarrollar y diseñar aplicaciones de JavaScript a gran escala. Angular 7 mejoró la funcionalidad de Angular 1.x y la hizo más rápida, escalable y moderna. Angular 7 es una actualización de Angular 4 que implica una mejora en el rendimiento de la aplicación, reduciendo el código generado y acelerando el desarrollo de la misma. Angular 7 fue escrito en TypeScript 2.2, un superconjunto de JavaScript que implementa nuevas características (Jimenez Castela, 2017).

1.4.6 Entorno de desarrollo

Visual Studio Code v1.33

Es un **editor de código,** soporta una serie de lenguajes como son: **C#, F# y Visual Basic,** PHP**, Python, Perl, SQL,** shell scripting en **Bash y Java.** Ofrece a los desarrolladores soporte integrado para múltiples idiomas. El editor presenta todas las herramientas estándar de un editor de código moderno, incluyendo resaltado de sintaxis, enlaces de teclado personalizables, coincidencia de corchetes y fragmentos. Control integrado de versiones a través de Git (Medina, 2015).

Android Studio v3.3

IDE oficial para el desarrollo de aplicaciones Android, desarrollado por Google publicado de forma gratuita bajo la licencia de código abierto Apache 2.0. Está basado en el IDE Intellj IDEA de la compañía JetBrains

[17] Acrónimo de Syntactically Awesome Stylesheets es un lenguaje de hoja de estilos.

y está disponible para las plataformas Windows, MacOS y GNU/Linux (Jamal Eason, 2019).

1.4.7 Sistema control de versiones

Git 2.7.4

Es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente. Su propósito es llevar registro de los cambios en archivos de computadora y coordinar el trabajo que varias personas realizan sobre archivos compartidos.

1.5 Conclusiones parciales

En este capítulo se describieron los principales conceptos relacionados con el tema de investigación. Luego de un estudio sobre el estado actual y funcionamiento del proceso de monitoreo pasivo de servidores se demostró la necesidad de crear una aplicación móvil que contribuya al monitoreo pasivo en Nova Servidores que se adapte a las necesidades de los administradores de red. Se realizó además un estudio, análisis y comparación de las diferentes soluciones existentes para el monitoreo pasivo en servidores; obteniéndose como resultado de este estudio el desarrollo de una aplicación móvil para el monitoreo pasivo en Nova Servidores. Se realizó una descripción de la metodología, herramientas y tecnologías seleccionadas para el desarrollo, definiendo la Variación AUP como metodología a utilizar Visual Studio Code como plataforma de desarrollo, Visual Paradigm 8.0 como herramienta de modelado, el empleo de Angular constituye algunas de las principales características que convierten a Ionic en la opción más viable para la implementación de la solución y JavaScript como lenguaje de programación para el desarrollo de la aplicación.

Capítulo 2: Análisis y diseño de la propuesta de solución

En el presente capítulo se realiza el análisis y diseño de la aplicación móvil para el monitoreo pasivo de Nova Servidores. Se definen sus principales características a través de la descripción de los requisitos funcionales y no funcionales. Se presenta una especificación de los requisitos funcionales utilizando las Historias de Usuario. Además, se define la arquitectura de software y los patrones de diseño.

2.1 Descripción de la propuesta de solución

A continuación, se presenta la propuesta de solución, aplicación móvil para el monitoreo pasivo de Nova Servidores, con el objetivo de darle respuesta al problema de investigación planteado anteriormente y una descripción del contexto del problema.

Actualmente Nova Servidores cuenta con la herramienta Zabbix encargada del monitoreo. Esta herramienta cuenta con una interfaz de comunicación, de donde obtiene la información de los recursos que este posee (CPU, RAM, discos duros, tráfico de la red, servicios), además genera alertas de monitoreo de servicios y recursos. Para que los administradores de red logren un control total de la herramienta deben estar físicamente en la estación de trabajo, lo cual es una limitante ya que estos cumplen otras actividades y tareas dentro de la institución. Por tanto se propone el desarrollo de una aplicación móvil para el monitoreo pasivo de Nova Servidores, la cual garantiza que el administrador pueda estar en otras áreas de la institución, incluso fuera de esta, seguirá recibiendo alertas del monitoreo y observar consumo de recursos y estados de servicios.

2.2 Requisitos

Según el estándar 1233 de la IEEE[18]: Guía para el desarrollo de Especificaciones de Requerimientos de Sistemas, un requisito se define como:

- Condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo.
- Condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente.

[18] IEEE: Del inglés *Institute of Electrical and Electronics Engineers*. Traducido al español Instituto de Ingeniería Eléctrica y Electrónica.

Es posible concluir que los requisitos de software son características y funcionalidades que debe poseer un sistema y están enfocados hacia lo que debe hacer el software. Además, pueden ser clasificados en funcionales y no funcionales.

2.2.1 Fuentes para la obtención de requisitos

Las fuentes para la obtención de requisitos utilizadas fueron:

- Especialistas del Centro de Software Libre asociados al proyecto Nova específicamente en la variante para servidores.
- Aplicaciones móviles diseñadas como terceros de la herramienta de monitoreo Zabbix.
- Entrevista: la entrevista es una técnica de recogida de información que además de ser una de las estrategias utilizadas en procesos de investigación, tiene ya un valor en sí misma.

2.2.2 Requisitos Funcionales (RF)

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir, cómo debe comportarse en situaciones específicas. En algunos casos también pueden plantear explícitamente qué no debe hacer el sistema (Somerville, 2011).

Con el objetivo de satisfacer las necesidades del cliente, se definieron los siguientes requisitos funcionales:

- Requisito 1. Autenticar usuario
- Requisito 2. Mostrar uso de RAM.
- Requisito 3. Mostrar uso de CPU.
- Requisito 4. Mostrar tráfico RED.
- Requisito 5. Mostrar estado de servicios
- Requisito 6. Mostrar alertas
- Requisito 7. Listar alertas.
- Requisito 8. Mostrar almacenamiento en disco.
- Requisito 9. Mostrar almacenamiento en swap.

2.2.3 Requisitos no funcionales

Los requisitos no funcionales son aquellos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades de este como fiabilidad, tiempo de respuesta y la capacidad de almacenamiento. Incluyen además restricciones de tiempo, sobre el proceso de desarrollo y

estándares (Somerville, 2011). A continuación, se definen los requerimientos no funcionales que debe cumplir la aplicación basándose en los establecido por las normas ISO 25000 Calidad del Producto de Software, específicamente la ISO/IEC 25010 que define las características de calidad que se tienen en cuanta al evaluar las propiedades de un producto de software:

Portabilidad:

1. La aplicación deberá funcionar correctamente en versiones superiores a 4.4 del sistema operativo *Android*.

Usabilidad:

2. Utilizar una iconografía en correspondencia con la utilizada por el *framework Ionic*.

Disponibilidad

3. Para que la aplicación funcione debe estar activa y en línea la herramienta Zabbix.

Rendimiento

4. La aplicación deberá funcionar correctamente en dispositivos con memoria RAM superior a 512MB.

2.3 Historias de usuario

Teniendo en cuenta la metodología de desarrollo escogida y las particularidades de la propuesta de solución se utilizará el escenario 4 de la misma ya que esta tiene como característica que es para soluciones que no modelen su negocio. Este escenario propone las Historias de Usuario donde se describen brevemente y en un lenguaje natural, las características que el sistema debe poseer. Cada Historia de Usuario debe ser lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en un corto período de tiempo (Canós, 2006). A continuación, se presentan tres de estas historias de usuario correspondientes a la solución, las restantes historias de usuario de pueden encontrar en los Anexo 1.

Tabla 2 : Historia de usuario correspondiente a requisito funcional RF_1 (Fuente: Elaboración propia).

<table>
<tr><th colspan="3">Historia de Usuario</th></tr>
<tr><td>Número: 1</td><td colspan="2">Nombre del requisito: Autenticar usuario</td></tr>
<tr><td colspan="2">Programador: Reimer D. Malleza Romero</td><td>Iteración Asignada: 1</td></tr>
<tr><td colspan="2">Prioridad: Alta</td><td>Tiempo Estimado: 8 horas</td></tr>
<tr><td colspan="2">Riesgo en Desarrollo: N/A</td><td>Tiempo Real: 2 horas</td></tr>
<tr><td colspan="3">Descripción: Los usuarios deben insertar la dirección del servidor, su nombre de usuario y contraseña.
➢ Dirección servidor: Obligatorio, y se espera una url (ejemplo: http/10.8.52.20/zabbix).
➢ Usuario: Obligatorio, se espera cadenas de caracteres de la a-z, se permiten mayúsculas.
➢ Contraseña: Obligatorio, se espera cadenas de caracteres de al a-z, números y caracteres especiales.</td></tr>
<tr><td colspan="3">Observaciones:
1. El servidor debe estar en línea
2. Si el usuario introduce la información de forma correcta, automáticamente entra a la aplicación.
3. Si el usuario introduce la información de forma incorrecta, se notificando el error.
4. Si el usuario deja algún campo vacío, el botón “Entrar” permanece inactivo.</td></tr>
<tr><td colspan="3">Prototipo elemental de interfaz gráfica de usuario:
 
</td></tr>
</table>

Tabla 3 : Historia de usuario correspondiente a requisitos funcionales RF_2, RF_3 y RF_4

(Fuente: Elaboración propia).

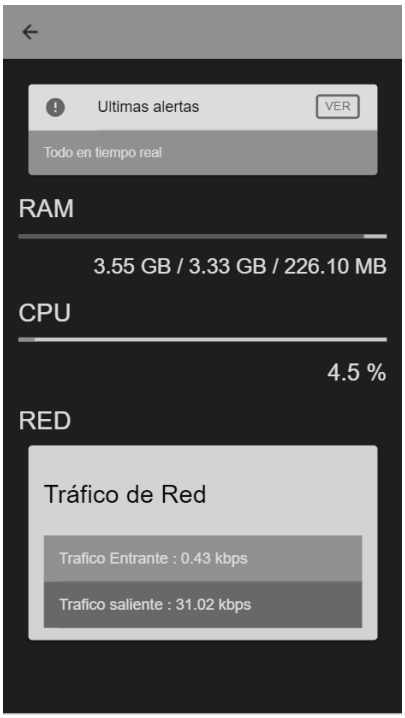
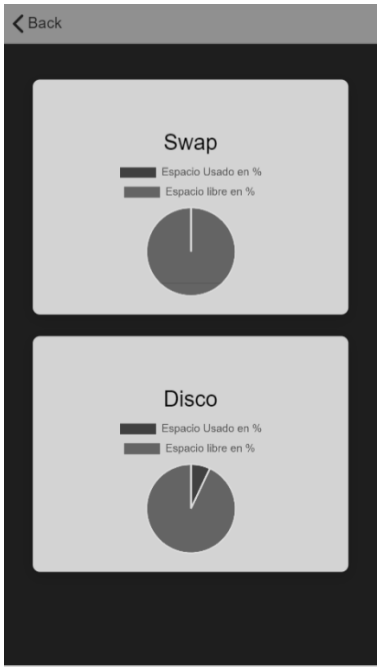
Historia de Usuario	
Número: 2	**Nombre del requisito:** Mostrar uso de RAM , CPU y RED
Programador: Reimer D. Malleza Romero	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 8 horas
Riesgo en Desarrollo: N/A	**Tiempo Real:** 2 horas
Descripción: La aplicación debe mostrar el uso de RAM, CPU y RED	
Observaciones:	
Prototipo elemental de interfaz gráfica de usuario: 	

Tabla 4 : Historia de usuario correspondiente a requisito funcional RF_6 (Fuente: Elaboración propia).

Historia de Usuario	
Número: 1	**Nombre del requisito:** Mostrar almacenamiento de disco y swap.
Programador: Reimer D. Malleza Romero	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 8 horas
Riesgo en Desarrollo: N/A	**Tiempo Real:** 2 horas
Descripción: La aplicación debe mostrar el estado de almacenamiento tanto de Disco como Swap en porcientos.	
Observaciones:	
Prototipo elemental de interfaz gráfica de usuario: 	

2.4 Análisis y diseño

El papel del diseño en el ciclo de vida de un software es facilitar la comprensión de su funcionamiento y proveer una representación o modelo del mismo con el propósito de definirlo con los suficientes detalles como para permitir su realización física. El modelo de diseño provee una representación arquitectónica del

software que sirve de punto de partida para las tareas de implementación.

2.4.1 Descripción de la arquitectura

El estándar IEEE define la arquitectura de software como la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución («ISO/IEC/IEEE 42010: Defining «architecture»» 2011). Según Roger Pressman: "En su forma más simple, la arquitectura del software es la estructura u organización de los componentes del programa, la manera en que estos interactúan y la estructura de datos que utilizan" (Pressman, 2011)

La arquitectura de software es una forma de representar sistemas mediante el uso de la abstracción, de forma que aporte el más alto nivel de comprensión de los mismos. Esta representación incluye los componentes fundamentales del software, su comportamiento y formas de interacción para satisfacer los requisitos del sistema.

Los patrones arquitectónicos se acercan a un problema de aplicación específica dentro de un contexto dado y sujeto a limitaciones y restricciones. El patrón propone una solución arquitectónica, esta solución sirve como base para el diseño de la arquitectura (Pressman, 2011). El patrón arquitectónico utilizado en la investigación fue Modelo-Vista-Controlador.

El patrón arquitectónico utilizado por Ionic es Modelo-Vista-Controlador (MVC), en la Figura 1 se muestra la representación de este patrón en el componente Mostrar uso recursos. El patrón MVC surge con el objetivo de reducir el esfuerzo de programación, necesario en la implementación de sistemas múltiples y sincronizados de los mismos datos, a partir de estandarizar el diseño de las aplicaciones. El patrón MVC es un paradigma que divide las partes que conforman una aplicación en el Modelo, las Vistas y los Controladores, permitiendo la implementación por separado de cada elemento. A partir del uso de *frameworks* basados en el patrón MVC se puede lograr una mejor organización del trabajo y mayor especialización de los desarrolladores y diseñadores (González, et al., 2012).

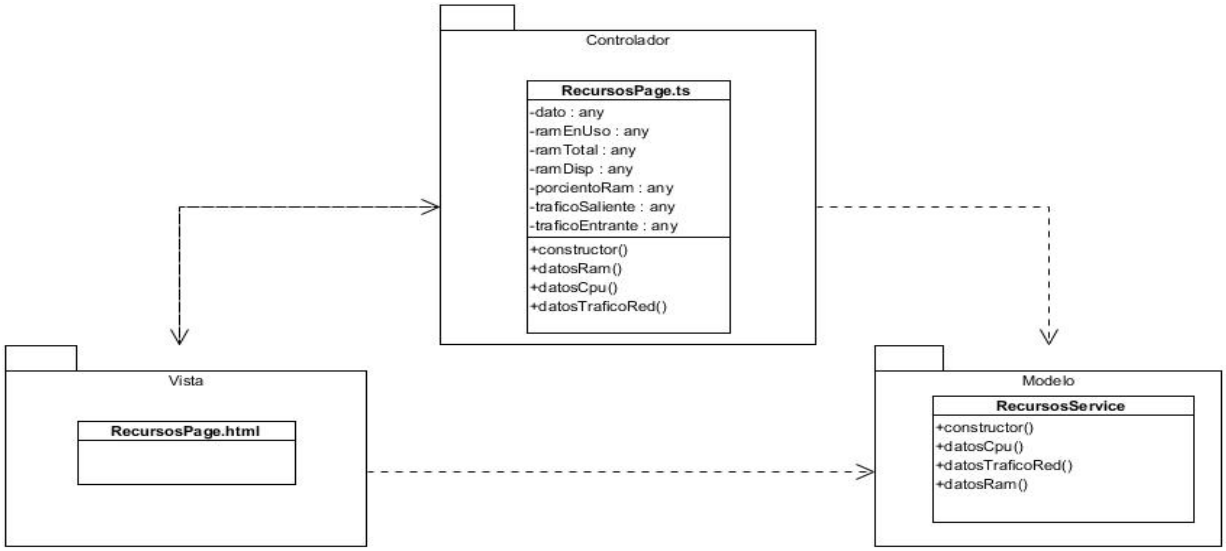


Figura 1 : Patrón arquitectónico Modelo-Vista-Controlador (Fuente: Elaboración propia).

El **Modelo** es el objeto que representa los datos del programa. Maneja los datos y controla todas sus transformaciones. En la aplicación desarrollada el modelo se evidencia en las clases empleadas para almacenar la información proveniente del api de zabbix en forma de objetos para su posterior utilización.

La **Vista** es el objeto que maneja la presentación visual de los datos representados por el Modelo. En este caso las vistas serían las plantillas empleadas para la representación visual de la aplicación. Las plantillas en el *framework Ionic* son almacenadas con la extensión html. Las plantillas constituyen una combinación de código *html* y un conjunto de directivas provenientes del *framework* Angular.

El **Controlador** es el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el Modelo, centra toda la interacción entre la Vista y el Modelo. En la solución propuesta los controladores son clases escritas en *TypeScript*, encargados de vincular las plantillas con los modelos y gestionar las peticiones de los usuarios.

2.5 Diagrama de clases del diseño

Los diagramas de clases son un tipo de diagrama de estructura estática que describe la estructura de un sistema mostrando las clases del sistema, sus atributos, operaciones (o métodos), y las relaciones entre los objetos; las cuales pueden ser asociativas, de herencia, de uso y de contenido. Expresan la estructura u organización del software en términos de clases. Además, constituyen el pilar básico del modelado con

UML, siendo utilizados tanto para mostrar lo que el sistema puede hacer, como para mostrar cómo puede ser construido.

La Figura 2 muestra el diagrama de clases del diseño correspondiente a los requisitos “Mostrar uso de RAM, Mostrar uso de CPU y Mostrar uso de RED” ya que están agrupados en la misma clase e interfaz. Este diagrama está basado en la arquitectura de software utilizada durante la investigación, MVC.

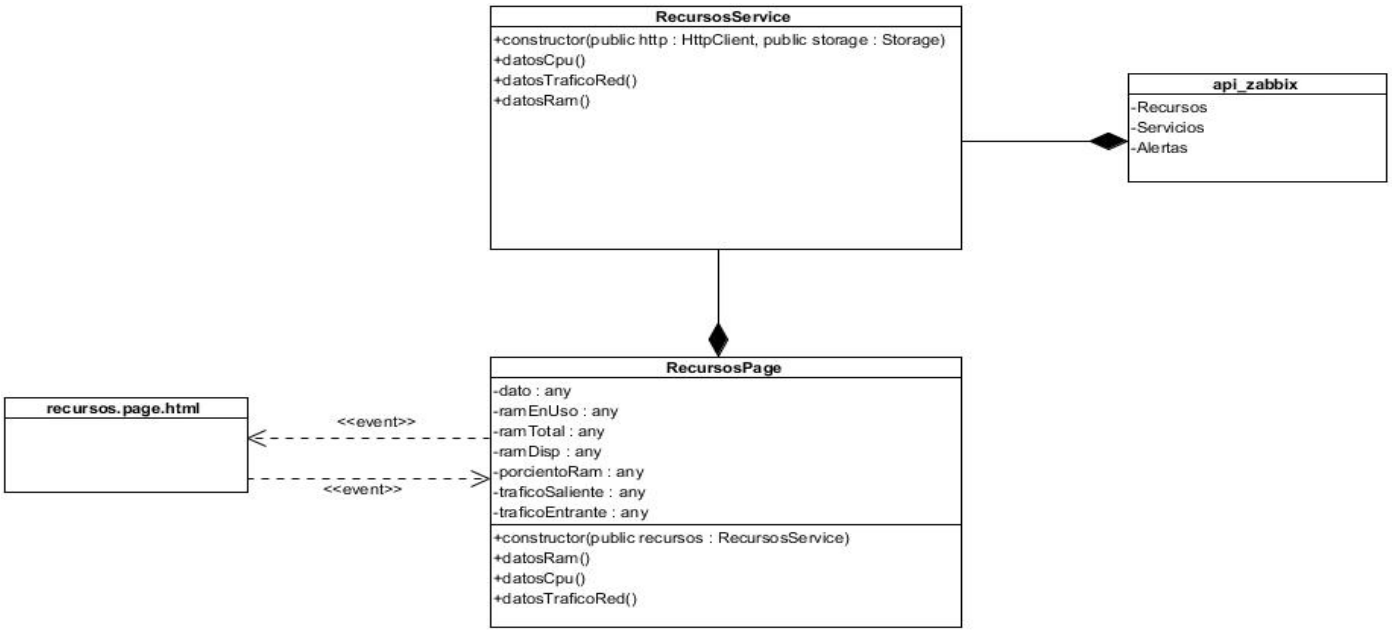


Figura 2 : Diagrama de clases correspondiente a requisitos funcionales RF_2, RF_3 y RF_4

(Fuente: Elaboración propia).

Descripción de las clases

recursos.page.html: Página encargada de mostrar la interfaz al usuario correspondiente a la vista recursos.

RecursosPage: Encargada de la interacción entre la Vista y el Modelo, es una clase escrita en *TypeScript*, encargada de vincular las plantillas con los modelos y gestionar las peticiones de los usuarios.

RecursosService: En la propuesta de solución esta clase es la encargada de comunicarse con el api de zabbix y obtener los datos necesarios para responder las peticiones del usuario.

api_zabbix: Constituye el proveedor principal de información de la aplicación. Establece la comunicación entre la aplicación móvil y el servidor de zabbix.

2.6 Patrones de diseño

Un patrón de diseño describe una estructura que resuelve un problema particular del diseño dentro de un contexto específico. El objetivo de cada patrón de diseño es proporcionar una descripción que permita a un diseñador determinar si el patrón es aplicable al trabajo en cuestión, si puede ser reutilizado y si sirve como guía para desarrollar un patrón distinto en funciones o estructura (Pressman).

Ionic está desarrollado utilizando como tecnología base el *framework* Angular. El empleo de Angular 7 proporciona ventajas relacionadas con la estructura del proyecto, el trabajo con buenas prácticas, el uso de patrones de diseño de software variados y una amplia gama de componentes y directivas.

- **Dependency Injection (inyección de dependencias):** Es un patrón de diseño de software empleado en la Programación Orientada a Objetos, en el que se suministran objetos a una clase en lugar de ser la propia clase la que cree el objeto. Expresa que los objetos nunca deben construir aquellos otros objetos que necesitan para funcionar. Esa parte de creación de los objetos se debe hacer en otro lugar diferente a la inicialización de un objeto. La Figura 3 muestra el empleo de este patrón.

```
@Injectable({
  providedIn: 'root'
})
export class AutenticacionService {
  token: any;
  autenticacionEstado = new BehaviorSubject(false);


  constructor(private storage: Storage, private plt: Platform, private http: HttpClient,
    public alertController: AlertController, public loadingController: LoadingController , public router : Router) {
    this.plt.ready().then(() => {
      this.checkToken();
    });
  }
```

Figura 3 : Empleo del patrón Inyección de dependencias (Fuente: Elaboración propia).

2.6.1 Patrones GRASP

Los Patrones Generales de Software de Asignación de Responsabilidades (GRASP, del inglés *General Responsibility Assignment Software Patterns*) constituyen un apoyo para la enseñanza, ayudan a entender el diseño de objetos y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable. Este enfoque para la comprensión y utilización de los principios de diseño se basa en los patrones de asignación de responsabilidades (Larman, 1999) .Los patrones GRASP utilizados en esta propuesta de solución fueron:

- **Creador:** El empleo de este patrón asigna a determinado objeto la responsabilidad de crear instancias de otros objetos. El uso de este patrón se puede evidenciar en el componente Recursos de tiempo cuando se le solicita información a la clase recursosService y esta se encarga de crear las instancias.
- **Alta Cohesión:** Verifica que la información que almacena una clase debe ser coherente y debe estar, en la medida de lo posible, relacionada con la clase. En la solución propuesta las clases clientes y las servidoras, se encargan cada una de realizar una función específica, permitiendo que las mismas no se saturen y a través de la asignación de responsabilidades se garantiza la alta cohesión en las clases existentes.
- **Bajo Acoplamiento:** El acoplamiento mide el grado en que una clase está conectada, tiene conocimiento o depende de otra. En el desarrollo de la aplicación se evidencia al realizar modificaciones en alguna de sus clases, producto a que las mismas no están fuertemente relacionadas, por tanto, la repercusión sobre las demás será mínima.
- **Controlador:** Un controlador es un objeto responsable del manejo de los eventos del sistema, que no pertenece a la interfaz del usuario. La clase AlertasPage del componente Alertas evidencia el uso de este patrón, debido a que es la encargada de brindar la respuesta adecuada a cada evento.

2.6.2 Patrones GoF

Son patrones usualmente aplicados en el trabajo orientado a objetos. Los mismos se clasifican de acuerdo a sus funciones en tres grupos: creacionales, estructurales y de comportamiento. De ellos se utilizan los siguientes:

- **Singleton (instancia única)**: Está diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. Su intención consiste en garantizar que una clase

sólo tenga una instancia y proporcionar un punto de acceso global a ella. En Ionic este patrón se evidencia mediante el empleo de servicios[19].

```
providers: [
  StatusBar,
  SplashScreen,
  AutenticacionService,
  {
    provide: RouteReuseStrategy,
    useClass: IonicRouteStrategy
  }
],
bootstrap: [AppComponent]
)
```

Figura 4 : Providers utilizados en Ionic (Fuente: Elaboración propia).

- **Decorator (envoltorio):** Responde a la necesidad de añadir dinámicamente funcionalidad a un Objeto. Esto permite no tener que crear sucesivas clases que hereden de la primera incorporando la nueva funcionalidad, sino otras que la implementan y se asocian a la primera. En la Figura 5 se evidencia el empleo de este patrón.

```
@Component({
  selector: 'app-recursos',
  templateUrl: './recursos.page.html',
  styleUrls: ['./recursos.page.scss'],
})
```

Figura 5 : Evidencia del patrón Decorator en Ionic (Fuente: Elaboración propia).

2.7 Conclusiones parciales del capítulo.

La definición de los requisitos funcionales y no funcionales del sistema permitió tener una visión más clara sobre las características y restricciones que debe cumplir el mismo. El diseño de los prototipos de interfaz de usuario permite guiar el proceso de desarrollo de las interfaces funcionales. Las características

[19] Los servicios en Ionic son conocidos como providers.

abordadas sobre el diseño del sistema permitieron establecer la comprensión de cada elemento (arquitectura, clases) que componen al mismo. Se realizaron los diagramas de clases del diseño y se establecieron los patrones que se utilizaron en la implementación de la aplicación. Además, la realización del diseño de la arquitectura permitió organizar y estructurar la solución informática enfocada en los patrones Modelo – Vista – Controlador.

Capítulo 3.Implementación y prueba de la propuesta de solución.

Se exponen en el presente capítulo las especificaciones asociadas a la implementación de la aplicación. Se describen las pautas de codificación utilizadas y el servidor de prueba utilizado para apoyar la construcción de la solución. Al término de esta, la aplicación resultante será sometida a un proceso de pruebas con el objetivo de validar el cumplimiento de los requerimientos especificados anteriormente.

3.1 Implementación

La codificación de la solución propuesta tiene lugar una vez que se han definido las historias de usuario y se ha concluido el diseño de la aplicación. Está encaminada a desarrollar de forma iterativa e incremental un producto completo listo para el despliegue, obteniendo versiones útiles de forma rápida, las que paulatinamente completan el desarrollo de la aplicación.

Los diagramas de componentes son utilizados para estructurar el modelo de la implementación. Permiten modelar una vista estática del sistema, muestran la organización y las dependencias lógicas entre un conjunto de componentes del *software*, que pueden ser librerías, binarios, ejecutables y códigos fuentes.

3.1.1 Diagrama de compontes

El diagrama de componentes es un tipo de diagrama del Lenguaje Unificado de Modelado. Representa la organización lógica de la implementación de un sistema, a través de los componentes y las relaciones de dependencia entre ellos (Pressman).La Figura 6 muestra el diagrama de componentes de la propuesta de solución.

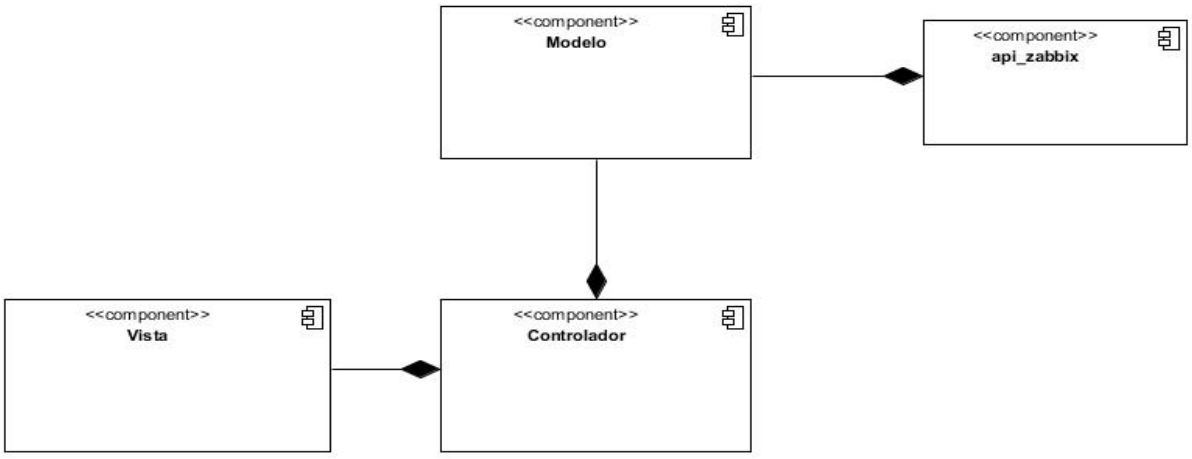


Figura 6 : Diagrama de componentes (Fuente: elaboración propia).

3.1.2 Estándares de codificación

Los estándares de codificación son un conjunto de convenciones (denominaciones, formatos) establecidas para la escritura de código. Estos estándares varían en dependencia del lenguaje de programación y son necesarios para los programadores y miembros del equipo de trabajo. Generalmente, el software no es mantenido por el/los autor(es) original(es) por ende debe ser revisado y/o manejado por otros. Los estándares permiten una mejor lectura e interpretación del software y su empleo permite aplicar un conjunto de lineamientos (Ampuero, et al., 2010). Los siguientes estándares de codificación son empleados durante el desarrollo de la aplicación:

- **Tamaño máximo de línea:** La longitud máxima recomendable para una línea de código es de 80 caracteres.
- **Clases:** El nombre de la clase debe estar en concordancia con el nombre del archivo y debe declararse en notación PascalCase.
- **Funciones y métodos:** Los nombres de funciones deben empezar siempre con una letra minúscula, utilizando el formato camelCase.
- **Variables:** Los nombres de variables pueden contener caracteres alfanuméricos. Los números están permitidos en los nombres de variable, pero no se aconseja en la mayoría de los casos. El formato definido para los nombres de variables debe ser camelCase.
- **Declaración de clases:** La llave "{" debe escribirse siempre seguido del nombre de la clase, ver Figura 7.

```
export class RecursosService {

}
```

Figura 7 : Declaración de clases (Fuente: Elaboración propia).

- **Declaración de funciones y métodos:** La llave "{" debe escribirse siempre seguido del nombre.
- **Uso de Funciones y Métodos:** Los argumentos de la función tienen que estar separados por un único espacio posterior después del delimitador coma.

- **Sentencias de Control:** Las sentencias de control basadas en las construcciones if y elseif deben tener un solo espacio en blanco antes del paréntesis de apertura del condicional y un solo espacio en blanco después del paréntesis de cierre. La llave de apertura "{" se escribe en la misma línea que la sentencia condicional. La llave de cierre "}" se escribe siempre en su propia línea.
- **Operadores:** Se debe colocar un espacio antes y después de los operadores binarios (+, -, +=,!=, ==, etc.).
- **Comentarios:** Los comentarios deben ser añadidos de forma que resulten prácticos, para explicar el flujo del código y el propósito de las funciones y variables. Los comentarios de una solo línea deben emplear los caracteres "//" al inicio. Los comentarios que ocupen múltiples líneas deben comenzar con los caracteres "/*" y concluir con "*/".

3.1.3 Creación y despliegue de un proyecto Ionic.

El Diagrama de Despliegue es un tipo de diagrama del Lenguaje Unificado de Modelado que se utiliza para modelar la disposición física de los componentes que integran al sistema. Este diagrama describe el despliegue físico de información generada por el programa de software en los componentes de hardware. A continuación se presenta el diagrama de despliegue de la solución, donde se muestran los requerimientos que debe tener la PC_Cliente utilizada por el desarrollador para implementar la solución.

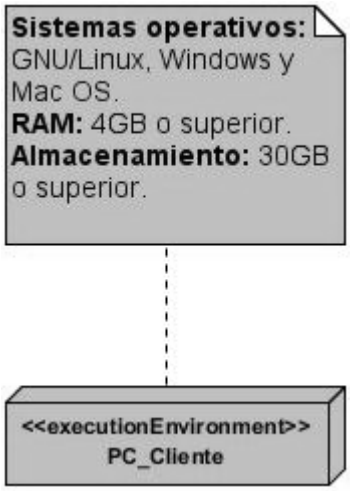


Figura 8 : Diagrama de despliegue (Fuente: Elaboración propia).

La utilización del framework Ionic requiere la instalación de varias aplicaciones o utilidades de consola en el ambiente de desarrollo. Los pasos a seguir para crear proyectos Ionic en el entorno de trabajo son (ionicframework, 2017):

- Instalar NodeJS[20] en correspondencia con el sistema operativo utilizado.
- Instalar una versión de Ionic CLI y Cordova. El comando que posibilita la instalación de ambas herramientas de manera global es: *npm install -g ionic cordova*.
- **Descargar una aplicación básica** de ejemplo para iniciar el proceso de desarrollo. El comando *ionic start [parámetro 1] [parámetro 2]* posibilita la descarga de las aplicaciones predeterminadas. El primer parámetro se refiere al nombre de la aplicación, mientras que el segundo lo hace al tipo de aplicación a descargar. Existen tres tipos básicos: blank (para una estructura básica y en blanco), tabs (con la navegación por tabs incluida) o sidemenu (para incluir un menú lateral). Un ejemplo con el tipo sidemenu es: *ionic start MiAplicacion sidemenu*.
- **Visualizar el resultado en el navegador**. Para ello Ionic proporciona el siguiente comando: *ionic serve*. Esta acción crea un servidor web que muestra el resultado de la ejecución de la aplicación en el navegador predeterminado. Mediante las herramientas de emulación es posible comprobar, en algunos navegadores, la apariencia de la aplicación en los sistemas operativos móviles iPhone y Android. Otra de las facilidades de desarrollo brindadas por el framework es la recarga automática al realizar alguna modificación en los ficheros del proyecto.
- **Desplegar en el dispositivo** la aplicación. Para el despliegue en Android se debe ejecutar el comando: *ionic platform add Android* que permite generar el código del proyecto para la plataforma Android. Este comando incluye en el proyecto las librerías necesarias en el lenguaje Java y los recursos propios de una aplicación nativa.
- **Realizar la construcción** (build) y compilación (run) del proyecto. Para cada uno de estos pasos Ionic brinda un comando específico. Sin embargo, el segundo incluye al primero, así que solamente es necesario ejecutar: *ionic run android*.

[20] Consultar la página: https://nodejs.org/es/

3.2 Pruebas de software

La prueba del software es un elemento crítico para la garantía de calidad del software y representa una revisión de las especificaciones, del diseño y de la codificación. Una vez generado el código fuente es necesario probar el software para descubrir y corregir la mayor cantidad de errores posibles antes de ser entregado. Su objetivo es diseñar una serie de casos de prueba que tengan una alta probabilidad de encontrar errores. Estas se dividen principalmente en los siguientes niveles: pruebas de unidad, de integración, de validación, de sistema y de aceptación (Pressman, 2011).

3.2.1 Pruebas funcionales

Las pruebas funcionales se basan en la ejecución, revisión y retroalimentación de las funcionalidades previamente diseñadas para el software. El objetivo principal de las pruebas funcionales es analizar el producto terminado y determinar si hace todo lo que debería hacer y si lo hace correctamente. El método utilizado para llevar a cabo este tipo de prueba es caja negra (Pressman, 2002).

Método de caja negra

Las pruebas de caja negra, también llamadas pruebas de comportamiento, se enfocan en los requerimientos funcionales del software. Las técnicas de prueba de caja negra permiten derivar conjuntos de condiciones de entrada que revisarán los requerimientos funcionales para un programa (Pressman, 2011). Entre las técnicas de pruebas existentes en el método de caja negra se encuentran: partición de equivalencia, análisis de valores límites y grafos de causa-efecto.

En el presente trabajo se aplicó el método de caja negra, específicamente la técnica partición de equivalencia. Los casos de pruebas, empleados durante la validación, fueron generados a partir de la aplicación de dicha técnica sobre las distintas interfaces que responden a los requisitos funcionales.

Resultados de las pruebas funcionales

A continuación se presenta el diseño de caso de prueba (DCP) del requisito “Autenticar usuario” donde se analizan las variables y condiciones que pueden determinar la respuesta del sistema.

Tabla 5: Caso de prueba de RF_1 "Autenticar usuario" (Fuente: Elaboración propia).

Escenario	Descripción	Variables			Respuesta del sistema	Flujo Central
		url	Usuario	contraseña		
EC 1 Autenticar con datos válidos	El sistema permite autenticar al usuario	V	V	V	El sistema permite acceder a la página principal	El usuario entra los datos de autenticación de manera correcta y selecciona el botón de entrada.
EC 2 Autenticar con datos inválidos	El sistema no debe permitir autenticar al usuario	V	I	I	El sistema envía una alerta donde notifica de errores en los valores de entrada.	El usuario entra los datos de autenticación de manera incorrecta y selecciona el botón entrada.

Tabla 6: Variables del caso de prueba "Autenticar usuario" (Fuente: Elaboración propia).

No.	Variable	Valor Nulo	Descripción
1	url	No	Es un campo de entrada para registrar la dirección del servidor.
2	Usuario	No	Es un campo de entrada para registrar el usuario administrador.
3	Contraseña	No	Es un campo de entrada para registrar la contraseña de administrador.

Durante la realización de las pruebas de caja negra se detectaron un conjunto de no conformidades relacionadas con errores de validación y funcionalidad. Los resultados se muestran en la Figura 9, donde se muestran la cantidad de casos de pruebas usados, los casos de pruebas con no conformidades y las no conformidades identificadas en cada iteración. Se realizaron tres iteraciones, durante la primera iteración se analizaron seis casos de pruebas, de los cuales cuatro resultaron tener un total de tres no conformidades. En la segunda iteración a través de las pruebas de regresión se verificó que las no conformidades anteriores

estuviesen solucionadas sobre los casos de prueba que incidían; de estas pruebas se obtuvo una única no conformidad, quedando resuelta en la tercera iteración y cumpliéndose correctamente los requisitos funcionales.

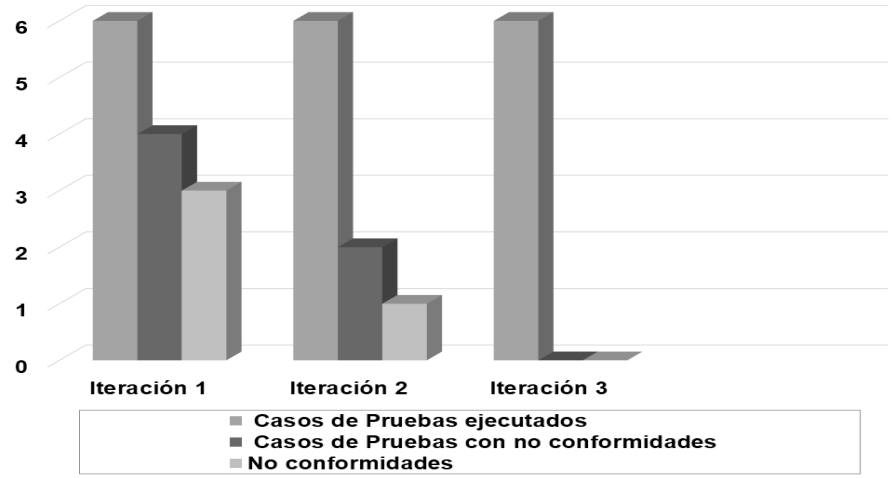


Figura 9 : Resultados de las pruebas de caja negra (Fuente: Elaboración propia).

3.2.2 Pruebas unitarias

Las pruebas unitarias son una forma de probar pequeñas e individuales porciones de código. A través de ellas se puede verificar que determinado módulo o funcionalidad se ejecuta dentro de los parámetros y especificaciones concretadas en los requisitos. El objetivo principal de una prueba unitaria es comprobar el correcto funcionamiento de una unidad de código. El método utilizado para realizar este tipo de prueba se denomina caja blanca (Pressman, 2002).

Método de caja blanca

La prueba de caja blanca, en ocasiones llamada prueba de caja de vidrio, consiste en probar el código. Es una filosofía de diseño de casos de prueba que usa la estructura de control descrita como parte del diseño a nivel de componentes para derivar casos de prueba (Pressman). Entre las técnicas de pruebas existentes en caja blanca se encuentran: prueba del camino básico, prueba de condición, prueba de flujo de datos y prueba de bucles. En la investigación se utiliza la prueba del camino básico. La aplicación de esta técnica permite: generar el grafo de flujo, calcular la Complejidad Ciclomática (CC), determinar los caminos

linealmente independientes y diseñar los casos de prueba para forzar la ejecución de cada camino del conjunto básico (Pressman, 2011).

Resultados de las pruebas unitarias

La técnica del camino básico fue aplicada al método login1() de la clase AutenticacionService, que se encarga de autenticar el usuario a través del api de zabbix. Luego de la determinación de los nodos y flujos de control del código se obtuvo el grafo de flujo y se calculó la complejidad ciclomática del algoritmo según muestra la Tabla 7.

Tabla 7 : Cálculo de la complejidad ciclomática del método login() (Fuente: Elaboración propia).

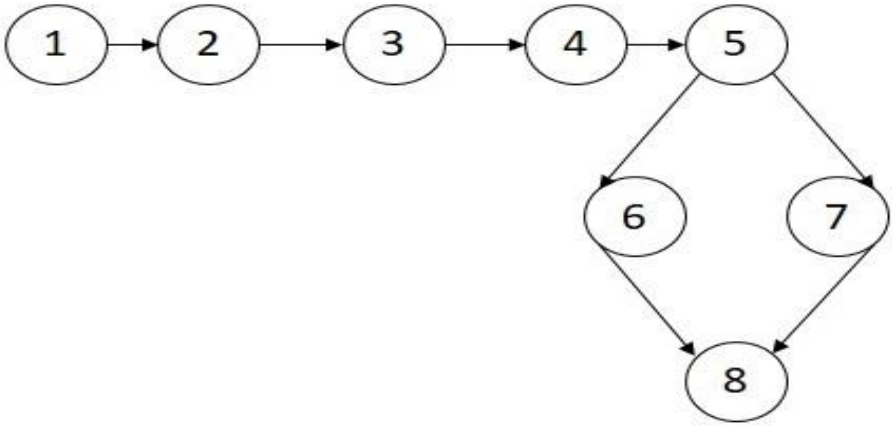
Método:

```
login(url, user, pass) {

  return new Promise((resolve) => {
   this.http.post(url, JSON.stringify({
    "jsonrpc": "2.0",
    "method": "user.login",
    "params": {
     "user": user,
     "password": pass
    },
    "id": 0,
    "auth": null
   }),
    {
     headers: new HttpHeaders().set('Content-Type',
'application/json'),
    }).subscribe(res => {
     resolve(res);
     this.token = res;
     if (this.token.result == undefined) {
      this.autenticacionEstado.next(false);
      this.presentLoading();
      this.router.navigate(['login']);
     }
     else {
      this.presentLoading2();
```

```
      this.autenticacionEstado.next(true);
      this.storage.set(TOKEN_KEY, this.token.result);
      this.storage.set(dir, url)
      this.router.navigate(['inicio']);
     }
    });|
  });
  }
```

Grafo resultante:



Complejidad Ciclomática:	V (G)= A-N+2	V (G)= P+1
V (G)= # de regiones	V (G)= 8-8+2	V (G)= 1+1
V (G)= 2	V (G)= 0+2	V (G)= 2
	V (G)= 2	

La complejidad ciclomática es igual a dos, lo que significa que existen dos posibles caminos linealmente independientes y hay que diseñar un mínimo de dos casos de prueba para el algoritmo. La Tabla 8 muestra los caminos existentes.

Tabla 8 : Caminos del grafo de flujo (Fuente: Elaboración propia).

No.	Camino
1	1,2,3,4,5,6,8
2	1,2,3,4,5,7,8

Los casos de prueba para las pruebas de caja blanca por la técnica de camino básico se ejecutan por cada camino independiente que se determine en un algoritmo específico. A continuación se muestra el caso de prueba para el camino básico independiente 2 del algoritmo, el resto se encuentra en el Anexo 2.

Tabla 9 : Caso de Prueba para el camino básico 2 (Fuente: Elaboración propia).

Proceso:
Autenticar usuario
Caso de Prueba:
Autenticar usuario
Camino independiente:
1,2,3,4,5,7,8
Entradas
Dirección url del servidor y usuario y contraseña de administrador
Resultados esperados:
• Realiza la conexión con el api • Guarda la dirección url y obtiene token de autenticación • Muestra la interfaz principal de la aplicación
Condiciones de ejecución:
El servidor de zabbix debe estar en línea.

Con la realización de los casos de prueba diseñados se probó la ejecución de cada sentencia del código al menos una vez, teniendo en cuenta todas las condiciones lógicas en sus variantes verdaderas y falsas. Los resultados del método de caja blanca fueron satisfactorios. La obtención de la complejidad ciclomática de valor dos permitió determinar que el algoritmo es de baja complejidad.

3.2.3 Pruebas de aceptación

Las pruebas de aceptación son aquellas ejecutadas por el propio usuario final para comprobar que el sistema realiza lo que el cliente especificó de manera correcta. Dichas pruebas aseguran la validez y conformidad de los clientes con el producto que se les está entregado en base a lo que se acordó inicialmente. De manera general las pruebas de aceptación pueden afrontarse mediante dos tipos de procedimiento para realizarlas: pruebas alfa y pruebas beta (José Ponce González, 2014).

Dentro del nivel de aceptación se decide aplicar las pruebas alfa ya que en ellas se le entrega a un usuario final todo el producto terminado, junto a su documentación correspondiente. Pues el cliente es el encargado de ejecutarlas en presencia del desarrollador y en entornos previamente preparados para la ejecución de las mismas. Una vez que el cliente detecta errores los va informando al desarrollador para su corrección.

Se realizaron 2 iteraciones del método alfa, detectándose un total de 2 no conformidades (NC). En la primera iteración se detectaron 2 NC y en la segunda iteración no se detectaron NC, obteniéndose un producto libre de errores. El cliente se mostró conforme con la herramienta desarrollada y emitió una carta de aceptación.

3.3 Conclusiones parciales

Al finalizar el presente capítulo quedó desarrollada satisfactoriamente la propuesta de solución, logrando así la implementación de la aplicación móvil para el monitoreo pasivo de Nova Servidores. La utilización de los estándares de codificación, permitió la obtención del código de forma organizada, facilitando su comprensión por los desarrolladores que requieran su uso. La ejecución de las pruebas de software posibilitó detectar las no conformidades existentes y solucionarlas.

Conclusiones generales

Una vez realizado el presente trabajo de diploma quedaron cumplidos los objetivos propuestos por lo que se concluye que:

- La realización de un análisis sobre el proceso de monitoreo pasivo de servidores posibilitó la comprensión de sus características y funcionamiento.
- El análisis de las características de aplicaciones móviles para el monitoreo pasivo de servidores posibilitó obtener funcionalidades para el desarrollo de la solución.
- Las tecnologías y herramientas seleccionadas permitieron guiar el proceso de desarrollo de la aplicación móvil para el monitoreo pasivo de Nova Servidores.
- El empleo de AUP-UCI permitió documentar el ciclo de vida del desarrollo de la solución.
- La implementación de la solución desarrollada permitió dar cumplimiento a la totalidad de los requisitos de software identificados con el cliente durante la etapa de análisis.
- La ejecución de pruebas de software permitió detectar y corregir errores no identificados durante la implementación, posibilitando el cumplimiento de las especificaciones requeridas.

Recomendaciones

Se recomienda:

- En futuras versiones del producto identificar y desarrollar nuevas funcionalidades para contribuir al monitoreo activo.
- Ampliar el rango de recursos a monitorizar por la aplicación.

Referencias

Ampuero, Margarita André y Trujillo, Yucely López. 2010. Creando un profesional con disciplina en el proceso de desarrollo de software. La Habana : Ingeniería Industrial, 2010. Vol. 27, 1.

Aniel. 2013. *https://www.aniel.es/importancia-de-las-tic-para-la-gestion-empresarial/.* [En línea] 7 de Agosto de 2013.

Caceres, Martin. 2018. ¿Qué es TypeScript. *DevCode.* [En línea] 2018. http://www.devcode.com.

Canós, José, Letelier, Patricio, Panadés, Ma Carmen. 2006. Metodologías Ágiles en el Desarrollo de Software. *ResearchGate.* [En línea] enero de 2006. [Citado el: 11 de 2 de 2019.] https://www.researchgate.net/publication/26428496_Metodologias_agiles_para_el_desarrollo_de_software_eXtreme_Programming_XP.

Castro, Elizabeth y Hyslop, Bruce. 2012. *HTML5 and CSS3, Seventh Edition: Visual QuickStart Guide.* San Francisco : s.n., 2012. ISBN-13: 978-0-321-71961-4 .

Collell Puig, Jordi. *CSS3 y Javascript avanzado.* Catalunya : s.n. PID_00176160.

Dylan Schiemann. 2019. Ionic Framework. *Ionic Framework.* [En línea] 23 de enero de 2019. https://ionicframework.com/press/release/2019/ionic-framework-4-release.

Eguíluz Pérez, Javier. 2008. *Introducción a JavaScript.* 2008.

Gauchat, Juan Diego. 2012. *El gran libro de HTML5, CSS3 y Javascript.* Barcelona : MARCOMBO, S.A. , 2012. ISBN: 978-84-267-1782-5.

González, Yanette Díaz y Romero, Yenisleidy Fernández. 2012. *Patrón Modelo-Vista-Controlador.* s.l. : Revista Telem@tica, 2012. vol. 11.

Hernández León, Rolando Alfredo y Coello González, Sayda. 2011. *El proceso de Investigación Científica.* La Habana : Universitaria, 2011.

ionicframework. 2017. IONIC. [En línea] 2017. [Citado el: 13 de Octubre de 2017.] https://ionicframework.com/.

JACOBSON, Ivar y BOOCH, Grady. 2016. *El Lenguaje Unificado de Modelado.* s.l. : Manual de Referencia, 2016.

Jamal Eason. 2019. Android Developers Blog. *Android Developers Blog.* [En línea] 2019. https://android-developers.googleblog.com/2019/01/android-studio-33.html.

Jimenez Castela, Pedro. 2017. *Angular 4 desde Cero.* 2017.

José Ponce González, Francisco José Dominguez Mayo,Javier Jesús Gutiérrez Rodríguez, Maria José Escalona Cuaresma. 2014. *Pruebas de aceptación orientadas al usuario:contexto ágil para un proyecto de gestión.* Sevilla : s.n., 2014.

Kuzmina, N.V. 1970. *Metódicas investigativas de la actividad pedagógica.* s.l. : Leningrado, 1970.

Larman, Craig. 1999. *UML y patrones: Una introduccion al analisis y diseño orientado a objetos y al proceso unificado.* México : Prentice Hall, 1999. ISBN 970-17-0261 -1.

Medina, Eduardo. 2015. Visual Studio Code, editor de código de Microsoft para Windows, OS X y GNU_Linux » MuyLinux. *MuyLinux.* [En línea] 30 de abril de 2015. http://www.muylinux.com.

Pressman, Roger S. 2002. *Ingeniería de Software, un enfoque práctico. Quinta Edición.* s.l. : McGraw-Hill Companies, 2002. ISBN: 8448132149.

—. 2011. *Ingeniería de Software,un enfoque práctico.* s.l. : McGraw-Hill Companies, 2011.

—. *Ingeniería del software: Un enfoque práctico. Séptima edición.* México : McGRAW-HILL INTERAMERICANA EDITORES. ISBN: 978-607-15-0314-5.

Romero, M.Sc. Gerardo Junco. 2018. Monografias.com. *Monografias.com.* [En línea] 2018. https://www.monografias.com/trabajos95/recursos-red-y-su-monitoreo/recursos-red-y-su-monitoreo.shtml.

Servisoftcorp. 2019. Softcorp. *Softcorp.* [En línea] 16 de 02 de 2019. https://www.servisoftcorp.com/definicion-y-como-funcionan-las-aplicaciones-moviles/.

Somerville, Ian. 2011. *Software Engineering.* 2011.

Bibliografía

Alonso, Lexys Manuel Díaz. 2017. Módulo para el monitoreo en tiempo real de clientes ligeros desde NovaLTSP. La Habana : s.n., 2017.

Altamirano, Carlos Alberto Vicente. 2005. Monitoreo de recursos de red. 2005.

Ampuero, Margarita André y Trujillo, Yucely López. 2010. Creando un profesional con disciplina en el proceso de desarrollo de software. La Habana : Ingeniería Industrial, 2010. Vol. 27, 1.

Ana Gardey, Julián Pérez Porto. 2013. Definicion.de. [En línea] 2013. https://definicion.de/monitoreo/.

Aniel. 2013. *https://www.aniel.es/importancia-de-las-tic-para-la-gestion-empresarial/.* [En línea] 7 de Agosto de 2013.

Caceres, Martin. 2018. ¿Qué es TypeScript. *DevCode.* [En línea] 2018. http://www.devcode.com.

Canós, José, Letelier, Patricio, Panadés, Ma Carmen. 2006. Metodologías Ágiles en el Desarrollo de Software. *ResearchGate.* [En línea] enero de 2006. [Citado el: 11 de 2 de 2019.] https://www.researchgate.net/publication/26428496_Metodologias_agiles_para_el_desarrollo_de_software_eXtreme_Programming_XP.

Castro, Elizabeth y Hyslop, Bruce. 2012. *HTML5 and CSS3, Seventh Edition: Visual QuickStart Guide.* San Francisco : s.n., 2012. ISBN-13: 978-0-321-71961-4 .

Collell Puig, Jordi. *CSS3 y Javascript avanzado.* Catalunya : s.n. PID_00176160.

Dylan Schiemann. 2019. Ionic Framework. *Ionic Framework.* [En línea] 23 de enero de 2019. https://ionicframework.com/press/release/2019/ionic-framework-4-release.

Ecured. 2018. Tecnología. *Ecured.* [En línea] 2018.

Eguíluz Pérez, Javier. 2008. *Introducción a JavaScript.* 2008.

Gauchat, Juan Diego. 2012. *El gran libro de HTML5, CSS3 y Javascript.* Barcelona : MARCOMBO, S.A. , 2012. ISBN: 978-84-267-1782-5.

González, Yanette Díaz y Romero, Yenisleidy Fernández. 2012. *Patrón Modelo-Vista-Controlador.* s.l. : Revista Telem@tica, 2012. vol. 11.

Hernández León, Rolando Alfredo y Coello González, Sayda. 2011. *El proceso de Investigación Científica.* La Habana : Universitaria, 2011.

ionicframework. 2017. IONIC. [En línea] 2017. [Citado el: 13 de Octubre de 2017.] https://ionicframework.com/.

JACOBSON, Ivar y BOOCH, Grady. 2016. *El Lenguaje Unificado de Modelado.* s.l. : Manual de Referencia, 2016.

Jamal Eason. 2019. Android Developers Blog. *Android Developers Blog.* [En línea] 2019. https://android-developers.googleblog.com/2019/01/android-studio-33.html.

Jimenez Castela, Pedro. 2017. *Angular 4 desde Cero.* 2017.

José Ponce González, Francisco José Dominguez Mayo,Javier Jesús Gutiérrez Rodríguez, Maria José Escalona Cuaresma. 2014. *Pruebas de aceptación orientadas al usuario:contexto ágil para un proyecto de gestión.* Sevilla : s.n., 2014.

Kuzmina, N.V. 1970. *Metódicas investigativas de la actividad pedagógica.* s.l. : Leningrado, 1970.

Larman, Craig. 1999. *UML y patrones: Una introduccion al analisis y diseño orientado a objetos y al proceso unificado.* México : Prentice Hall, 1999. ISBN 970-17-0261 -1.

Medina, Eduardo. 2015. Visual Studio Code, editor de código de Microsoft para Windows, OS X y GNU_Linux » MuyLinux. *MuyLinux.* [En línea] 30 de abril de 2015. http://www.muylinux.com.

Pressman, Roger S. 2010. Ingeniería del software.Un enfoque práctico. [En línea] 2010. [Citado el: 10 de diciembre de 2018.] http://cotana.informatica.edu.bo/downloads/Id-Ingenieria.de.software.enfoque.practico.7ed.Pressman.PDF.

—. 2009. *Software Engineering. A practitioner´s Approach. New York.* s.l. : McGraw Hill, 2009. ISBN 978-0-07-33759.

Pressman, Roger S. 2002. *Ingeniería de Software, un enfoque práctico. Quinta Edición.* s.l. : McGraw-Hill Companies, 2002. ISBN: 8448132149.

—. 2011. *Ingeniería de Software,un enfoque práctico.* s.l. : McGraw-Hill Companies, 2011.

—. *Ingeniería del software: Un enfoque práctico. Séptima edición.* México : McGRAW-HILL INTERAMERICANA EDITORES. ISBN: 978-607-15-0314-5.

Pressman, Roger S. 2009. *Software Engineering. A practitioner´s Approach.* New York : McGraw Hill, 2009. ISBN 978-0-07-337597-7.

Romero, M.Sc. Gerardo Junco. 2018. Monografias.com. *Monografias.com.* [En línea] 2018. https://www.monografias.com/trabajos95/recursos-red-y-su-monitoreo/recursos-red-y-su-monitoreo.shtml.

Servisoftcorp. 2019. Softcorp. *Softcorp.* [En línea] 16 de 02 de 2019. https://www.servisoftcorp.com/definicion-y-como-funcionan-las-aplicaciones-moviles/.

Somerville, Ian. 2011. *Software Engineering.* 2011.

Sommerville , Ian. 2011. *Ingenieria de Software 9na s.l.* s.l. : Adison-Wesley, 2011, 2011. ISBN 978-0-13-703515-1.

Zabbix. 2018. Server Monitoring. *Zabbix.* [En línea] 2018. http://www.zabbix.com.

Anexos

Anexo 1

Historias de usuario

Tabla 10 : Historia de usuario correspondiente al requisito funcional RF_7

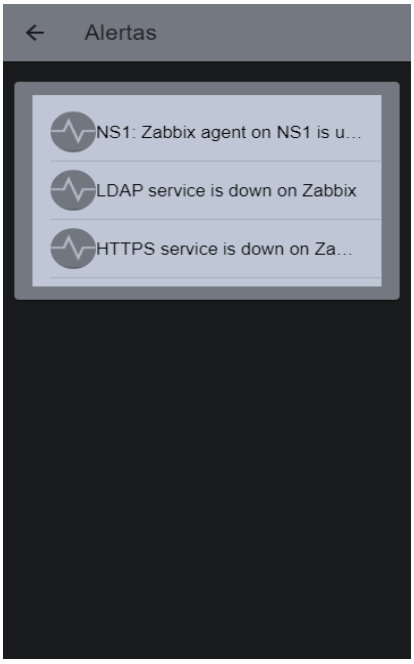
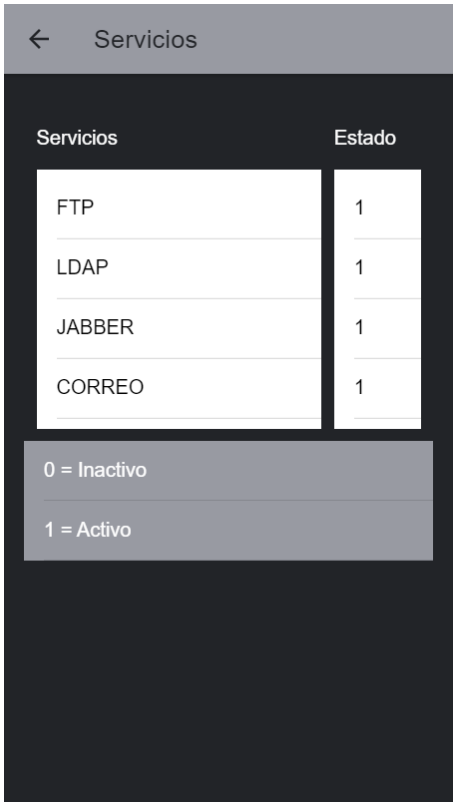
Historia de Usuario	
Número: 2	**Nombre del requisito:** Listar alertas
Programador: Reimer D. Malleza Romero	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 8 horas
Riesgo en Desarrollo: N/A	**Tiempo Real:** 2 horas
Descripción: La aplicación debe mostrar el historial de alertas de monitoreo provenientes del servidor de sabbix	
Observaciones: **1.** La aplicación toma las alertas exactamente como están configuradas en el servidor de zabbix.	
Prototipo elemental de interfaz gráfica de usuario: 	

Tabla 11 : Historia de usuario correspondiente al requisito funcional RF_5.

Historia de Usuario	
Número: 2	**Nombre del requisito:** Mostrar estado de servicios
Programador: Reimer D. Malleza Romero	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 8 horas
Riesgo en Desarrollo: N/A	**Tiempo Real:** 2 horas
Descripción: La aplicación debe mostrar el estado de los servicios que monitorea el servidor de zabbix, solo se muestra si está activo o inactivo.	
Observaciones: Los servicios que mostrara la aplicación son los que deben estar añadidos en el servidor de zabbix.	
Prototipo elemental de interfaz gráfica de usuario:	



Anexo 2

Casos de pruebas para las pruebas unitarias.

Tabla 12 : Caso de Prueba para el camino básico 1 (Fuente: Elaboración propia).

Proceso:
Autenticar usuario
Caso de Prueba:
Autenticar usuario
Camino independiente:
1,2,3,4,5,6,8
Entradas
Dirección url del servidor y usuario y contraseña de administrador
Resultados esperados:
• No se realiza conexión con el api de zabbix. • Se emite una alerta indicando que hay un error en los datos proporcionados o algún error de conectividad del dispositivo. • La aplicación indica y permite entrar nuevamente los datos.
Condiciones de ejecución:
El servidor de zabbix debe estar en línea.

Anexo 3

Casos de pruebas para las pruebas funcionales.

Tabla 13 : Caso de prueba funcional del RF_2, RF_3 y RF_4 (Fuente: elaboración propia).

Escenario	**Descripción**	**Variables**	**Respuesta del sistema**	**Flujo Central**
EC 1 Mostrar uso de RAM, Mostrar	El administrador de red se autentica en la aplicación y accede a la vista principal donde selecciona la opción		El sistema muestra una barra de progreso para	El administrador de red se autentica en la aplicación, luego

uso de CPU y Mostrar tráfico de RED	"Recursos" que lo lleva hacia la vista recursos donde puede observar toda la información referente al uso de RAM, CPU y tráfico de RED. En cuanto a RAM y CPU se utilizaron barra de progreso y para el tráfico de RED cuadro informativo sobre tráfico saliente y entrante.		representar el consumo de RAM y presenta la información de la siguiente manera (ramTotal/ramUso/ramLibre) en su parte inferior. Muestra para CPU una barra de progreso con el uso en porciento y por ultimo muestra la información referente al tráfico de salida y de entrada.	selecciona la opción Recursos que lo redirecciona a la vista recursos donde se observa información de uso de RAM, CPU y tráfico de RED proveniente del api de zabbix.

Tabla 14 : Caso de prueba funcional del RF_5 "Mostrar estado de servicios" (Fuente: elaboración propia).

Escenario	Descripción	Variables	Respuesta del sistema	Flujo Central
EC 1 Mostrar estado de servicios	El administrador de red se autentica en la aplicación y accede a la vista principal donde selecciona la opción "Servicios" que lo lleva hacia la vista servicios donde puede observar los servicios que están siendo monitorizados por zabbix.		El sistema muestra una lista de los servicios y un valor (0 o 1) asociado a cada uno, si es 0 el servicio está inactivo por el contrario si está en 1 el servicio está activo.	El administrador de red se autentica en la aplicación, luego selecciona la opción "Servicios" que lo redirecciona a la vista servicios. A medida que se añaden servicios al servidor la aplicación los mostrara.

Tabla 15 : Caso de prueba funcional de RF_7 "Listar alertas" (Fuente: elaboración propia)

Escenario	Descripción	Variables	Respuesta del sistema	Flujo Central
EC 1 Listar alertas	El administrador de red se autentica en la aplicación y accede a la vista principal donde selecciona la opción "Alertas" que lo lleva hacia la vista alertas donde puede observar las últimas notificaciones de alertas.		El sistema muestra una lista de notificaciones sobre las alertas ocurridas en el servidor.	El administrador de red se autentica en la aplicación, luego selecciona la opción alertas que lo redirecciona a la vista alertas. A medida que se producen alertas en el servidor se van añadiendo a la lista de alertas en la aplicación.

Tabla 16 : Caso de prueba funcional de RF_8 y RF_9 "Mostrar almacenamiento en Disco", "Mostrar almacenamiento en Swap" (Fuente: elaboración propia)

Escenario	Descripción	Variables	Respuesta del sistema	Flujo Central
EC 1 Mostrar almacenamiento en disco, Mostrar almacenamiento de swap.	El administrador de red se autentica en la aplicación y accede a la vista principal donde selecciona la opción "Almacenamiento" que lo lleva hacia la vista almacenamiento donde puede observar toda la información referente al uso de Disco y Swap a través de graficas de pastel, se muestra el espacio usado y libre en porciento.		El sistema muestra graficas de pastel para representar espacio libre y usado de Disco y Swap en porciento.	El administrador de red se autentica en la aplicación, luego selecciona la opción Almacenamiento que lo redirecciona a la vista almacenamiento donde se observa información de uso de almacenamiento de Disco y Swap proveniente del api de zabbix.

Anexo 4

Entrevista realizada a miembros del grupo de desarrollo de Nova Servidores sobre diferentes temas relacionados con la investigación para obtener ideas, referencias e información útil.

Estimado(a) compañero(a):

La presente entrevista tiene como objetivo conocer, analizar y determinar debilidades que presente Nova Servidores respecto al monitoreo y control de sus recursos y servicios. El resultado de esta entrevista resulta de vital importancia para la presente investigación debido a que contribuirá a la correcta elaboración de la propuesta de solución.

Datos generales del entrevistado:

Título universitario:

Categoría científica:

Categoría docente:

Instrucciones:

1. ¿A qué tipo de clientes (usuarios) va dirigido Nova Servidores? ¿En qué entorno es desplegado Nova Servidores?
2. ¿Existe algún mecanismo o herramienta para el monitoreo de Nova Servidores? ¿Cuál o cuáles?
3. ¿Cuáles son los aspectos fundamentales a tener en cuenta para en el monitoreo? ¿Por qué?
4. ¿Se han detectado problemas en el monitoreo de Nova Servidores? ¿Cuáles?

Muchas gracias.

Anexo 5

Acta de aceptación de productos de trabajo

UCI
Universidad de las Ciencias Informáticas

ACTA DE ACEPTACIÓN DE PRODUCTOS DE TRABAJO

En cumplimiento del **Convenio de colaboración** establecido entre el **Centro de Software Libre (CESOL)** y la estudiante **Reimer Darian Malleza Romero** de la Facultad 1 de la Universidad de las Ciencias Informáticas y en función de la ejecución del proyecto**: Aplicación móvil para el monitoreo pasivo de Nova Servidores**, se hace entrega del producto que se relaciona a continuación:

- Aplicación móvil para el monitoreo pasivo de Nova Servidores

La parte Cliente, luego de haber revisado el producto de trabajo relacionado anteriormente procede a firmar la aceptación de los mismos en total conformidad.

Entrega	Recibe
Nombre y Apellidos:	**Nombre y Apellidos:**
Reimer Darian Malleza Romero	Hanny Valdés Hernández
Cargo: Estudiante Facultad 1	**Cargo:** Jefe de Departamento
Firma:	**Firma:**

UCI Universidad Facultad 1

Fecha: 25 de marzo de 2019