UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD I

Aplicación Android para la gestión de la información en el diagnóstico de la migración de los servicios telemáticos

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

Autor:

Alejandro Puerta Díaz

Tutores:

Ing. Abel Ochoa Izquierdo

Msc. Briseis Angeles Godinez Valdés

Msc. Yurisbel Vega Ortiz

La Habana, Cuba.

Junio 2020

Declaración de autoría

Declaro por este medio que yo **Alejandro Puerta Díaz**, con carné de identidad **93031811140** soy el autor principal del trabajo titulado **Aplicación Android para la gestión de la información en el diagnóstico de la migración de los servicios telemáticos** 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.

_____

Firma del autor

_____ _____ _____

Firma del tutor **Firma del tutor** **Firma del tutor**

Dedicatoria

Agradecimientos

Resumen

El proceso de migración a software libre en Cuba es guiado por el Centro de Software Libre de la Universidad de las Ciencias Informáticas, principalmente en los Organismos de Administración Central del Estado. Este proceso se divide en tres etapas, Preparación o Diagnóstico, Ejecución y Consolidación, de las cuales depende la calidad y el éxito del proceso. En el diagnóstico de los servicios telemáticos se recolecta toda la información referente a los servidores, servicios telemáticos y la topología de red de la entidad. La presente investigación tiene como propósito desarrollar una aplicación para dispositivos móviles con sistema operativo Android, para mejorar la gestión de la información recolectada en dicha etapa. Durante el proceso de desarrollo de la propuesta de solución se utilizó como metodología de desarrollo de software AUP en su variante para la Universidad de las Ciencias Informáticas, el lenguaje de programación Java y como entorno de desarrollo integrado Android Studio en su versión 3.5.3.0. Como resultado se obtuvo una aplicación Android que permite mejorar la gestión de la información. Además, genera un informe con la información recolectada. Se realizaron las pruebas de software, donde se obtuvo la aceptación por parte del cliente.

Palabras clave: Android, aplicación, diagnóstico, Migración, software libre.

Índice

Índice de Figuras

Índice de Tablas

Introducción

Con el paso de los años el desarrollo de las tecnologías ha avanzado aceleradamente y con ello los sistemas operativos, por lo que la clasificación de los mismos sigue haciéndose más extensa. Los sistemas operativos son categorizados por las clasificaciones tradicionales que están orientadas a la estructura, servicios o la forma de ofrecer los servicios del sistema operativo, o a las más recientes que están orientadas por el tipo de adquisición y de derechos sobre el sistema operativo o el dispositivo que lo contiene. De esta última surgen los términos software libre o código abierto y software comercial o privativo.

Los derechos otorgados al usuario bajo una licencia privativa, son insuficientes para las necesidades operativas de los Estados. El software libre ofrece ventajas de índole económica, social, operativa y de seguridad nacional que hacen imperativo su uso en forma exclusiva en todas las áreas de la administración pública, especialmente en los centros educacionales (Feal Delgado, y otros, 2014). En los inicios el software era parte del equipo informático, no se consideraba como un elemento separado y las instituciones que los adquirían podían compartirlo con toda facilidad y sin restricciones. Cuando comienza la era de las computadoras personales a principios de los años 80, inicia el proceso de privatización del software.

En esta época aparecen las primeras empresas en Estados Unidos que producen código con el fin de comercializarlo, presionando al gobierno para que apruebe leyes que restrinjan los derechos de los usuarios, conociéndose entonces como software comercial o privativo (Medrano, 2014). Durante muchos años, el uso del software privativo se realizó de forma masiva a nivel mundial, mayormente por los países desarrollados, para mantener informatizada su infraestructura tecnológica. En el caso de los países subdesarrollados, esto suponía un reto, ya que la utilización de tecnología basada en software privativo; provocaba el desembolso de grandes sumas de dinero en el pago de licencias de uso, para mantener actualizadas sus dependencias estatales y mantener un correcto funcionamiento de las Tecnologías de la Información y las Comunicaciones (TIC).

Las TIC son el resultado de la interrelación de componentes, uno de ellos es el software que es controlado por grandes monopolios empresariales que obtienen todos los años miles de millones de dólares por concepto de pagos de licencias de uso pues, sus clientes están obligados a depender de ellos porque restringen el conocimiento de su funcionamiento y no venden un producto sino el derecho a utilizarlo. Por

este motivo, la independencia tecnológica, es una preocupación actual de muchos gobiernos y organizaciones que quieren mantener el control sobre las bases tecnológicas en las que se asientan las TIC (Feal Delgado, y otros, 2014).

Esto evidencia que es una necesidad imperativa para países subdesarrollados, encaminarse a la migración a software libre, por temas económicos, de seguridad nacional y soberanía tecnológica, siendo esta una vía estable y con grandes beneficios, dado el tamaño de la comunidad de software libre y el gran soporte tecnológico que provee. La migración a software libre es un conjunto de actuaciones cuya finalidad es la sustitución de infraestructuras tecnológicas apoyadas en software propietario por otras con funciones equivalentes basadas en software libre.

La migración a software libre y código abierto constituye una prioridad de vital importancia para Cuba como país subdesarrollado, pero esto requiere un cambio en la manera de pensar, debido a la fuerte tradición de empleo de las herramientas y plataformas de software privativo. Cuba promueve, como parte de su programa de informatización, que no puede apostarse por los sistemas operativos privativos como un camino viable para el desarrollo tecnológico y que a la vez conduzca a la independencia en este mismo ámbito (Montano, 2015). Una de las formas de alcanzar la independencia tecnológica es mediante la migración a software libre, la cual es una vía factible para Cuba por ser además un país bloqueado, lo que constituye una alternativa para superarse económicamente.

Cuba cuenta, desde su creación, con la Universidad de las Ciencias Informáticas (UCI) para el desarrollo de software, para sustentar la infraestructura tecnológica del país, basado en tecnologías libres. La UCI tiene entre sus centros de investigación y desarrollo el Centro de Software Libre (CESOL), principal impulsor de esta tarea, teniendo como objetivos, desarrollar la distribución cubana de GNU/Linux Nova y apoyar las directrices, lineamientos y soluciones que guiarán la migración nacional. Este centro creó la “Guía Cubana de Migración a Software Libre” en el 2009, documento que desde entonces ha servido de consulta para todas las instituciones y personas en el país que ejecutan procesos de migración (Pérez Guevara , y otros, 2014).

El centro CESOL es el encargado de llevar a cabo el proceso de migración a software libre, principalmente en los Organismos de la Administración Central del Estado (OACE), utilizando un conjunto de herramientas informáticas para que este se realice de forma rápida y eficiente. Este proceso consta de tres etapas:

Preparación o Diagnóstico, Ejecución y Consolidación, que vistas como un sistema influyen en la calidad y el éxito del proceso. En la primera etapa del proceso se desarrolla el diagnóstico de la organización, en la segunda etapa es donde se realiza la migración total o parcial de la entidad y en la tercera etapa se brinda soporte técnico y capacitación del personal (Villazon, y otros, 2018).

Las tecnologías móviles son muy populares hoy en día debido a los beneficios que brindan, tales como: la portabilidad, el acceso a datos y conexión inalámbrica en todo momento, como el fácil acceso a las mismas. CESOL cuenta con una aplicación para dispositivos móviles con sistema operativo Android, que sirve de apoyo a los jefes de equipos de migración para llevar a cabo las tareas que se realizan en la Etapa de ejecución de la migración en las estaciones de trabajo. Sin embargo para el proceso de diagnóstico de los servicios telemáticos de las entidades, los especialistas no cuentan con un apoyo tecnológico que mejore el trabajo a realizar.

Actualmente la gestión de la información en el proceso de diagnóstico de los servicios telemáticos en las entidades a migrar, se realiza de forma manual, lo que hace ineficiente las tareas a realizar por los especialistas de migración. En este proceso es necesario recolectar toda la información referente a la entidad y hacer un levantamiento informático acerca de los servidores y los servicios telemáticos que brindan, además se necesita información de la topología de red presente. Debido a la cantidad de información que se necesita, en ocasiones, los especialistas incurren en la duplicación o pérdida de información, errores en la obtención de los datos, pérdida de tiempo en tareas de gestión de la información. Para obtener la información necesaria de cada servidor, el especialista de migración lo hace en formato duro, una vez obtenida, debe gestionarla y llenar un documento con la misma en su ordenador para completar el informe del diagnóstico a la entidad. Estas deficiencias atentan contra la calidad del proceso, ya que este depende de la calidad del diagnóstico realizado.

Por todo lo expuesto anteriormente se plantea como **problema de investigación**: ¿Cómo mejorar la gestión de la información en el proceso de diagnóstico de servicios telemáticos en los Organismos de la Administración Central del Estado?

Para dar cumplimiento al problema planteado se establece como **objetivo general**: Desarrollar una aplicación Android que permita mejorar la gestión de información en el proceso de diagnóstico de los servicios telemáticos de los Organismos de la Administración Central del Estado.

El **objeto de estudio** lo constituye: el proceso de gestión de la información en el diagnóstico de los servicios telemáticos de los OACE en la migración a software libre, el **campo de acción** se define en: las herramientas de apoyo al proceso de gestión de la información en el diagnóstico de los servicios telemáticos en los OACE.

El objetivo general se desglosa en los siguientes **objetivos específicos**:

1. Elaborar el marco teórico de la investigación sobre la gestión de información en el diagnóstico de los servicios telemáticos en los Organismos de la Administración Central del Estado.

2. Diseñar una aplicación Android que permita mejorar la gestión de información en el diagnóstico de los servicios telemáticos en los Organismos de la Administración Central del Estado.

3. Implementar una aplicación Android que permita mejorar la gestión de información en el diagnóstico de los servicios telemáticos en los Organismos de la Administración Central del Estado.

4. Evaluar la aplicación Android que permita mejorar la gestión de información en el diagnóstico de los servicios telemáticos en los Organismos de la Administración Central del Estado.

Como parte del desarrollo de la investigación surgen una serie de **preguntas científicas**, las cuales ayudan en el proceso de desarrollo de la misma:

1. ¿Cuáles son los fundamentos teóricos que sustentan la investigación sobre el proceso de gestión de la información en el diagnóstico de los servicios telemáticos en los OACE?

2. ¿Cuáles son los aspectos a tener en cuenta para realizar el diseño de una aplicación Android para mejorar la gestión de la información en el diagnóstico de los servicios telemáticos en los OACE?

3. ¿Qué funcionalidades se deben tener en cuenta para el desarrollo de una aplicación Android para mejorar la gestión de la información en el diagnóstico de los servicios telemáticos en los OACE?

4. ¿Qué pruebas de software aplicar para la evaluación de la aplicación Android para mejorar la gestión de la información en el diagnóstico de los servicios telemáticos en los OACE?

Los **métodos teóricos** utilizados en la investigación son:

El **Analítico-Sintético**, para identificar y analizar los principales conceptos y definiciones relacionados con el proceso de Diagnóstico de la migración a software libre.

La **Modelación** es utilizada para confeccionar los diagramas correspondientes a la representación de la propuesta de solución, al permitir la definición y descripción de las funcionalidades que la conforman. En la investigación se utiliza para hacer los diagramas de paquetes, clases y el modelo de datos.

Los **métodos empíricos** utilizados son:

La **Entrevista** (Anexo 1), permite recopilar información a través de los especialistas de migración para definir la situación problemática, el levantamiento de requisitos, y la determinación de las restricciones de software e implementación que deben tenerse en cuenta en el proceso de desarrollo.

La **Observación** para realizar el estudio de las características y comportamientos de las herramientas que presenten soluciones similares, en el proceso de migración a software libre.

El presente trabajo consta de una introducción, tres capítulos, conclusiones generales, recomendaciones, referencias bibliográficas, bibliografía consultada y los anexos. A continuación se describe brevemente lo que aborda cada capítulo.

Capítulo 1. Fundamentación teórica sobre el proceso de diagnóstico para la migración a software libre de los servicios telemáticos de los OACE. En este capítulo se evidencia el estudio de los principales conceptos asociados al objeto de estudio de la investigación, tales como la migración a software libre y las etapas por las que atraviesa este proceso, específicamente la etapa de Diagnóstico, se presentan los conceptos de software libre, servidores, servicios telemáticos y topología de red, así como el análisis de la información a recolectar de los mismos. Se fundamenta el uso de los distintos temas referentes a la metodología, herramientas y tecnologías que se utilizan para el desarrollo de la propuesta de solución.

Capítulo 2. Análisis y Diseño de una aplicación Android para la gestión de información en el diagnóstico para la migración a software libre de los servicios telemáticos de los OACE. En este capítulo se describe la propuesta de solución. Se especifican los requisitos funcionales y no funcionales, así como las historias de usuario. Se modelan los diagramas del diseño de la aplicación teniendo en cuenta los patrones de diseño identificados.

Capítulo 3. Implementación y Prueba de una aplicación Android para la gestión de información en el diagnóstico para la migración a software libre de los servicios telemáticos de los OACE. En este capítulo se describen los artefactos relacionados con la implementación de la aplicación. Se especifica el estándar de codificación a utilizar para lograr una mayor legibilidad del código. Se diseñan y ejecutan las pruebas a realizar, con el objetivo de comprobar las funcionalidades y el código de la aplicación en los diferentes escenarios.

Capítulo 1. Fundamentación teórica sobre el proceso de diagnóstico para la migración a software libre de servicios telemáticos de los OACE

Introducción

En el presente capítulo se fundamenta teóricamente la investigación y se dan a conocer los principales conceptos asociados a esta. Los temas fundamentales que se abordan son: la migración a software libre, el software libre y la etapa de diagnóstico en la migración a software libre en Cuba, específicamente en el proceso de diagnóstico de los servicios telemáticos. Se describe la metodología de desarrollo que se utiliza en la propuesta de solución. También se proponen los lenguajes y tecnologías necesarias para el desarrollo de la aplicación.

1.1 Definición de conceptos asociados a la investigación

El **software libre** es aquel sobre el cual el usuario tiene amplios derechos de uso, difusión y modificación. Este ofrece al usuario 4 libertades esenciales como son la libertad de ejecutar el programa como se desee, con cualquier propósito, la libertad de estudiar cómo funciona el programa, y cambiarlo para lo que necesita acceso al código fuente, la libertad de redistribuir copias para ayudar a otros y la libertad de distribuir copias de sus versiones modificadas a terceros (Feal Delgado, y otros, 2014).

El **software privativo** es aquel que restringe los derechos del usuario al uso de sus funcionalidades bajo condiciones determinadas como el derecho a usarlo pero sin conocer el código fuente del mismo (Feal Delgado, y otros, 2014).

Servidor: En informática, se conoce como servidor (del inglés server) a un computador que forma parte de una red informática y provee determinados servicios al resto de los computadores de la misma, llamados a su vez estaciones o clientes. Dicho computador debe contar con una aplicación específica capaz de atender las peticiones de los distintos clientes y brindarles respuesta oportuna, por lo que en realidad dentro de una misma computadora física (hardware) pueden funcionar varios servidores simultáneos (software), siempre y cuando cuenten con los recursos logísticos necesarios (Raffino, 2019).

Servicios Telemáticos: son aquellos servicios que se brindan desde una computadora (servidor), a una o más computadoras, para el transporte, almacenamiento y procesamiento de cualquier tipo de información (datos, voz, vídeo, entre otros), como son Correo electrónico, proxy, DNS, Chat, etcétera (Aguilera, 2015).

La topología de red: es la disposición física en la que se conecta una red de ordenadores. La misma está compuesta por nodos y conexiones entre estos, el término nodo se refiere a un punto de intersección en el que confluyen dos o más elementos de una red de comunicaciones, los mismos pueden ser desde una computadora, hasta un servidor. La topología de red, se clasifica según el tipo de conexión y puede ser Bus, Estrella, Mixta, Anillo, Anillo Doble, Árbol, Maya y Totalmente Conexa (Tanenbaum, y otros, 2012).

1.2 Proceso de migración a software libre

Un **proyecto de migración a tecnologías de software libre y código abierto** es el proceso por el cual un entorno informático basado en sistemas y software privativo se adapta y traslada al uso de Software Libre. En el mismo, se deben buscar alternativas a las herramientas privativas existentes, asegurar una adecuada transición entre las mismas, y garantizar en todo momento la continuidad de la información y el trabajo realizado con anterioridad por los usuarios. Para poder realizar este cambio con seguridad, se deben seguir una serie de etapas que garanticen la correcta gestión del cambio de modelo desde el entorno privativo al nuevo paradigma libre (Soluciones T.I, 2019).

En el desarrollo de la investigación, una migración a tecnologías de software libre y código abierto puede definirse como: un proceso ordenado en el cual se sustituye, parcial o totalmente, el software privativo existente en la organización por alternativas liberadas bajo licencias libres o de código abierto.

El proceso de migración a software libre se divide en dos componentes fundamentales, la migración social y la migración técnica ya que este proceso afecta todas las estructuras y personas dentro de la institución en la cual se ejecuta, provocando desconfianza y resistencia al cambio en las personas involucradas en el mismo (Villazón, y otros, 2009).

- Migración social: es cuando en el proceso, las personas son sometidas a un cambio en su forma de pensar, rompiendo la cultura de fidelización a los programas privativos provocada por la costumbre y adaptándose a la filosofía asociada al movimiento de software libre y código abierto.

- Migración técnica: es el componente relacionado con el cambio de la tecnología. Su objetivo no es desterrar el software privativo de la entidad, sino sustituir aquellos programas que realmente puedan ser cambiados sin afectar sensiblemente el funcionamiento de la institución. No es una meta en sí misma, sino un camino para mejorar la productividad, reducir los costos, crear conocimiento y/o modelos nuevos de negocios.

1.3 Proceso de migración a software libre en Cuba

Desde la década de los 80 del pasado siglo, época de la adopción del Sistema Operativo Microsoft, el uso de la informática en Cuba estuvo soportado sobre plataforma privativa. En la actualidad, es una necesidad la migración a software libre, principalmente en los OACE y en el sistema de educación, en busca de la soberanía tecnológica y desarrollo y control de la economía del país. En abril de 2004 a partir del acuerdo 084/2004 del consejo de ministros, el país decide emprender la migración a plataformas de código abierto de forma paulatina por parte del entonces Ministerio de la Informática y las Comunicaciones (MIC) (Montano, 2015).

Para realizar el proceso de migración a software libre, el mismo se divide en tres etapas, bien definidas, en la Metodología Cubana para la Migración a Plataformas de Código Abierto, la cual plantea sus principales procesos. Las etapas por las que atraviesa el proceso de migración son (Villazón, y otros, 2009):

- Preparación o Diagnóstico: donde se ejecutan todas las tareas de diagnóstico de los procesos, personas y tecnología de la entidad, se realizan tareas de análisis de la información recuperada y se emite el plan de migración institucional
- Ejecución: comprende las actividades necesarias para la migración definitiva de usuarios y tecnologías de la institución. Incluye la migración de los servicios telemáticos y las computadoras de escritorio
- Consolidación: etapa que comprende tareas destinadas a garantizar el soporte técnico a los usuarios e infraestructura.

1.3.1 Etapa de Preparación o Diagnóstico

La etapa de Preparación o Diagnóstico de la entidad consiste en realizar el levantamiento informático de la entidad de hardware y software, además de los recursos humanos presente en la misma. El levantamiento informático se realiza sobre los servidores, las estaciones de trabajo y los dispositivos externos existentes. En el caso de las estaciones de trabajo y servidores es evaluado además de sus características de hardware, el software interno que los mismos poseen. Otro aspecto importante en la etapa es la topología de red con que cuenta la entidad. El proceso de migración como resultado final debe de desencadenar la menor cantidad de cambios posibles en aras de disminuir la resistencia al cambio, por esto es necesario que esta etapa se realice con la mayor calidad posible (Villazon, y otros, 2018).

La migración de servidores y sus servicios es el primer paso en el proceso de ejecución de la migración de la tecnología. Por esta razón debe realizarse un diagnóstico que permita conocer las características de la organización y su infraestructura tecnológica, para proponer las soluciones más acertadas que garanticen interoperabilidad tanto con sistemas privativos como libres. Según experiencias de los especialistas del centro CESOL, es muy común encontrarse en las empresas un servidor de directorio activo sobre Microsoft Windows Server 2003 y sobre el mismo autentican las computadoras de los usuarios, el sistema de gestión financiera y la intranet institucional. En tal caso existe una estrecha relación entre los sistemas y si se migra el servidor de directorio puede traer consigo que los usuarios no puedan acceder a sus estaciones de trabajo con el usuario del dominio, o que dejen de ser accesibles la intranet y el sistema de gestión financiera. Es por esto que obtener el tejido de relaciones y dependencias entre sistemas y servicios es tan importante para tomar decisiones futuras en la planificación de la migración sobre cómo serán migrados los sistemas y en qué orden (Villazon, y otros, 2018).

Por la importancia de los servicios telemáticos, su alcance y repercusión en el negocio de la entidad, debe realizarse un profundo proceso de recopilación de información. Los elementos necesarios diagnosticar son:

- Acerca del hardware
    - Nombre identificativo: debe ser un nombre que identifique al servidor únicamente.
    - Procesador: características del procesador.
    - Memoria RAM: características de la memoria RAM (*Random Access Memory*, Memoria de

Acceso Aleatorio) instalada.

- Disco duro: características del o los discos duros del servidor.
- Tarjeta de red: características de la tarjeta de red.
- Observaciones de hardware: en este aspecto se informa del hardware específico instalado en cada uno de los servidores y que pudieran necesitar de controladores en los sistemas operativos GNU/Linux. Tal es el caso de una UPS (*Uninterruptible Power Supply*, Sistemas de Alimentación Ininterrumpida), a la que esté conectada el servidor, también tarjetas HBAs (Hot Bus Adapter) para el almacenamiento en dispositivos con este fin desde el servidor; así como demás dispositivos que se entienda son necesarios.

➢ Acerca del software:

- Sistema operativo y versión: la variante del sistema operativo desde la cual se desea realizar la migración.
- Arquitectura del sistema operativo: la arquitectura del sistema operativo que se encuentra instalado en el servidor.
- Software usado para brindar servicio: debe colocarse todo el software que se encuentra brindando servicios en el servidor, además en los casos que se pueda se especifica la versión de los mismos.
- Relación entre servicios: debe especificarse la relación entre los servicios. Debido a que en varios casos, los servicios utilizan para dar acceso de autenticación a usuarios del Directorio Activo; los servidores de bases de datos, dan acceso a sus bases de datos a las aplicaciones desde determinados servidores; los servidores DHCP (*Dynamic Host Configuration Protocol*, Protocolo de Configuración Dinámica de Host) en ocasiones adicionan registros en las bases de datos de los servidores DNS (*Domain Name System*, Sistema de Nombres de Dominio) al asignar una dirección IP (*Internet Protocol*, Protocolo de Internet). Toda esta red de comunicación debe quedar plenamente diagnosticada.
- Cantidad de usuarios concurrentes: se estima la máxima cantidad de usuarios o peticiones que en un momento dado usan concurrentemente el servidor.
- Configuraciones específicas: se detallan las configuraciones específicas del servidor, tanto de autenticación, como de seguridad o rendimiento. Estas deben especificarse por servicios y de la forma más clara posible.

A continuación se presentan los principales servicios que se brindan en los servidores:

- DHCP: se especifica si el servicio DHCP escribe en la base de datos del DNS, base de datos que sirve para resolver nombres en las redes, si asigna direcciones IP en dos interfaces de red hacia dos redes en la organización. Las principales reglas de acceso en su firewall, así como las políticas por defecto, también deben ser especificadas.
- DNS: Se especifica si usa base de datos directa e inversa, registros que utiliza: MX[1], CNAME[2], PTR[3], y los nombres, direcciones IP, así como una pequeña descripción que indique para qué se usa cada uno. También debe especificarse si está configurado como esclavo o maestro, las principales reglas de acceso en el firewall, así como las políticas por defecto.
- FTP (*File Transfer Protocol*, Protocolo de Transferencia de Archivos): se especifica si está configurado para acceder anónimamente, si autentica con un Directorio Activo y si se usa para establecer un sistema de salvas de cada usuario de la institución. Las principales reglas de acceso en el firewall, así como las políticas por defecto, también deben ser especificadas.
- LDAP (*Lightweight Directory Access Protocol*, Protocolo Ligero de Acceso a Directorio): se especifica si el directorio activo está configurado con un conjunto de políticas y se mencionan las mismas, si está siendo usado como Controlador de Dominios de salva, especificando de quién o hacia quien está replicando. Las principales reglas de acceso en el cortafuego, así como las políticas por defecto, también deben ser especificadas.
- Jabber: se especifica si el servicio de mensajería instantánea autentica con usuarios locales en el sistema operativo, o autentica con un directorio activo, si por defecto los usuarios aparecen a cada contacto organizados por grupos almacenados en el Directorio Activo, mencionando la dirección IP del mismo, así como si permite la comunicación con Yahoo, Gmail, etcétera. Las principales reglas de acceso en su cortafuego, así como las políticas por defecto, también deben ser especificadas.
- Bases de datos: por cada servicio de bases de datos, ya sea MySQL, PostgreSQL, SQL Server,

[1] MX: (del inglés Mail eXchange record, en español "registro de intercambio de correo") es un tipo de registro, un recurso DNS que especifica cómo debe ser encaminado un correo electrónico en internet.

[2]CNAME: registro de Canonical Name es un tipo de registro que puede encontrarse en un DNS y que permite al usuario especificar el alias de un nombre de dominio.

[3] PTR: registros inversos de direcciones IP, se encargan de asociar un único IP a un nombre.

Oracle o cualquier otro, se puede especificar: qué servicios lo utilizan y desde qué direcciones IP, cuántas bases de datos almacena, si se replican datos hacia otro servidor o viceversa, si se realizan salvas de las bases de datos debe detallarse la periodicidad y el lugar hacia donde se guardan. Las principales reglas de acceso en el cortafuego, así como las políticas por defecto, también deben ser especificadas.

- Servicio web: Debe mencionarse la cantidad de sitios con que cuenta el servicio y los módulos en el servidor utilizados para servirlos, por ejemplo: si las aplicaciones hosteadas están desarrolladas en PHP, ASP, Java, Ruby, Python u otro, especificando por cada uno de estos sitios el servidor de bases de datos. De igual forma mencionar si cuenta con puertos virtuales configurados. Las principales reglas de acceso en el cortafuego, así como las políticas por defecto, también deben ser especificadas.
- Proxy: Debe especificarse la autenticación del mismo, ya sea contra un Directorio Activo o con usuarios locales, las diferentes reglas de control de acceso al servicio y restricciones de navegación, como pueden ser accesos al servicio por IP, usuarios, dirección lógica de tarjetas de red, si es un requisito estar unido al dominio para acceder al mismo, la denegación de direcciones de Internet, así como el acceso a solo determinados dominios.
- Correo: Debe identificarse el modo de autenticación del servicio: si autentica con usuarios locales, en una base de datos relacional como MySQL o PostgreSQL o si lo hace con un Directorio Activo. También si los usuarios para descargar sus correos utilizan POP3, IMAP o algún otro protocolo. Pueden especificarse los mecanismos de autenticación del servicio, ya sea SSL (Secure Sockets Lyer), TLS (Transport Layer Security) o algún otro que se crea necesario en dependencia de las configuraciones, si se gestionan varios dominios locales en la organización; calendarios y tareas, servicio de antivirus y antispam, alias y listas de correo.

  Debe obtenerse si existen restricciones en el tamaño de los mensajes y de los buzones, y en el envío de correo a determinados dominios, o si se filtra el cuerpo y asunto de los correos enviados y de acuerdo a determinadas palabras, que pueden ser ofensivas o constituir un peligro para la empresa, se le envíe una copia a determinado usuario, así mismo si se envía una copia del flujo de todos los correos a un usuario creado para su posible auditoría en el momento que le sea necesario, debe reflejarse en el documento.

El servicio de correo también puede estar cumpliendo la función de relay[4] o entregando los correos a través de un relay, así mismo, obteniendo los correos que le son enviados desde el exterior a través de un buzón remoto, o intercambiando información con otros servidores de correo que no cuentan con registros MX en servidores DNS, por lo que puede necesitarse colocar las direcciones y dominios que utiliza en los transportes configurados en el mismo. Se debe informar además, los lugares hacia donde se realiza salva de todos los buzones de correo, así como las principales reglas de acceso en el cortafuego y las políticas por defecto que utiliza el servicio.

En esta etapa también se realiza el levantamiento de información referente a la topología de red existente en la entidad, en función de conocer la disposición física de los dispositivos de cómputo, el tipo de conexión y los mecanismos de seguridad que implementan.

Los especialistas de migración deben recopilar la siguiente información de la misma:

- Tipo de topología de red: Especificar la clasificación de la topología de red.
- Existencia de Zona Desmilitarizada (DMZ): Existencia o no de DMZ en la entidad.
- Especificar la existencia de conexión a Internet: Verificar si a través de un proxy, portal cautivo o directo a Internet
- Existencia de wifi institucional: Especificar si existe una wifi institucional.

1.4 Herramientas de apoyo a la migración a software libre

La migración a software libre de una entidad es un proceso complejo debido las actividades que se realizan en el mismo y el tiempo que requieren cada una de ellas. El uso de herramientas de apoyo a dicho proceso agiliza y facilita el actuar de los especialistas encargados del mismo, por tal motivo, es necesario el desarrollo de herramientas que informaticen la mayor cantidad de tareas a realizar en el proceso de migración de la entidad. Actualmente CESOL cuenta con dos herramientas fundamentales para realizar la gestión de la información recolectada en el proceso de migración, siendo estas las únicas de las que se tenga conocimiento en Cuba como es el caso de:

[4] Relay: se refiere al proceso de la transferencia de correos electrónicos en Internet.

La Plataforma Cubana de Migración a Código Abierto (PCMCA), este es un sistema informático que tiene entre sus principales objetivos el apoyo al proceso de migración. La plataforma permite a partir de la información recopilada de una institución proponer alternativas libres a software privativo, homologar y certificar el hardware, procesar encuestas y diseñar cursos de formación. Con este sistema se logra una mayor eficiencia con un menor costo de implementación, con la diferencia que se realiza desde un entorno automatizado y con un control centralizado de la seguridad (Guevara, y otros, 2014).

A pesar de contar con esta plataforma, la misma necesita de la conexión de un ordenador de forma directa a la red para poder utilizarla, lo que dificulta un poco el proceso, debido a que para realizar las tareas de recolección de información, es necesario que el especialista transite por varias áreas de la entidad, lo que imposibilita el uso constante de la misma. Para dar solución a este problema CESOL decidió implementar una aplicación para dispositivos móviles con sistema operativo Android, que le permitiera a los especialistas hacer uso de la principal ventaja que esto provee que es la movilidad.

La aplicación móvil de apoyo al proceso de migración a software libre, actúa en el marco de la etapa de ejecución de la migración a las estaciones de trabajo de manera eficiente. La misma utiliza la información recogida previamente en un Excel y la procesa internamente para su uso, además permite realizar cambios del estado de migración de las estaciones de trabajo y demás datos de interés de la misma, por último genera un reporte general del estado de migración y porciento de migración por áreas.

A pesar de existir estas herramientas anteriormente descritas, no están destinadas para las actividades de la etapa de Preparación o Diagnóstico ni para el proceso de gestión de la información de los servicios telemáticos. Hasta el momento esas tareas se realizan de forma manual. Teniendo en cuenta la importancia de este proceso para realizar la migración a software libre de los servicios telemáticos de las entidades, se propone realizar una aplicación Android, que sea capaz de mejorar el proceso de gestión de la información en el diagnóstico de los servicios telemáticos de los OACE.

1.4.1 Android

Android es un sistema operativo móvil desarrollado por Google; es uno de los más conocidos junto con iOS de Apple. Se encuentra basado en Linux, que está enfocado para ser utilizado en dispositivos móviles como teléfonos inteligentes, tabletas, Google TV, entre otros dispositivos (Basterra, y otros, 2015). En los últimos

años Android se ha convertido en el sistema operativo más usado en el mundo llegando a tener casi el 90 % de las ventas de teléfonos inteligentes con este sistema operativo.

Android presenta las siguientes características:

- Es un sistema de código abierto
- Utiliza SQLite para almacenar los datos
- Se adapta a varias pantallas y resoluciones
- Ofrece diversas formas de mensajería
- Soporta HTML, HTML5, Adobe Flash Player, etcétera.
- Incluye un emulador de dispositivos, herramientas para depuración de memoria y análisis del rendimiento del software

Para guiar el proceso de desarrollo de la aplicación Android propuesta se emplea la metodología de desarrollo de software descrita en el siguiente epígrafe.

1.5 Metodología de desarrollo de software

Existen diferentes metodologías de desarrollo de software, estas son un modo sistemático de realizar, gestionar y administrar un proyecto para llevarlo a cabo con altas posibilidades de éxito. Una metodología para el desarrollo de software comprende actividades a seguir para idear, implementar y mantener un producto de software desde que surge la necesidad del producto hasta que se cumple el objetivo por el cual fue creado (Enríquez Ruiz, y otros, 2017).

Metodología de desarrollo de software AUP versión para la UCI

En el desarrollo de la aplicación se utiliza la metodología de desarrollo de software Variación de AUP para la UCI, una variante realizada por la Universidad de las Ciencias Informáticas a la metodología ágil AUP (*Agile Unified Process*, Proceso Ágil Unificado). La misma está formada por tres fases: Inicio, Ejecución y Cierre para el ciclo de vida de los proyectos de la universidad (Sánchez, 2015).

Las características de las fases son:

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

La investigación se centra en la fase de Ejecución, pasando por las disciplinas de análisis y diseño, implementación y pruebas, haciendo uso del escenario 4 para la encapsulación de requisitos mediante las historias de usuario.

1.6 Lenguajes y herramientas para el modelado de la solución

Una vez definida la metodología de desarrollo de software a utilizar se seleccionaron los lenguajes y herramientas para la construcción del mismo, explicados a continuación.

1.6.1 Lenguaje de modelado

“El modelado constituye una simplificación de la realidad donde se define lo esencial para la construcción del software con los objetivos de comunicar la estructura de un sistema complejo, especificar el comportamiento deseado del sistema, comprender mejor lo que se está desarrollando y descubrir oportunidades de simplificación y reutilización” (James Rumbaugh, y otros, 2007).

En la elaboración del diseño de la solución se empleó el Lenguaje Unificado de Modelado, descrito a continuación.

Lenguaje Unificado de Modelado (UML)

UML (Lenguaje Unificado de Modelado) en español, es el lenguaje estándar para visualizar, especificar, construir y documentar los artefactos de un sistema, utilizándose para el modelado del negocio y sistemas de software. También ofrece un estándar para describir los modelos, incluyendo aspectos conceptuales como procesos de negocio, funciones del sistema, expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables (James Rumbaugh, y otros, 2007).

1.6.2 Herramienta de modelado

Como herramienta de modelado, se emplea **Visual Paradigm** en su versión 8.0 para modelar los diagramas que se generen en cada una de las etapas del proceso de desarrollo de software. La misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación (Pressman, 2002).

La herramienta de modelado Visual Paradigm para UML en su versión 8.0 soporta el lenguaje de modelado UML, facilita el trabajo del equipo de desarrollo durante el ciclo de vida del software y permite la estandarización de la documentación que se va generando, además permite exportar en varios formatos, los productos que generan.

1.6.3 Lenguaje de programación

Se selecciona ***Java 8*** como lenguaje de programación porque es el nativo para el desarrollo de aplicaciones para el Sistema Operativo Android, además permite optimizar el tiempo y el ciclo de desarrollo (compilación y ejecución). La principal característica de Java es la de ser un lenguaje compilado e interpretado, además de ser de código abierto. Este posee características como ser orientado a objetos, independiente de la plataforma, dinámico, interpretado y robusto, además de ser fácil de aprender y ser uno de los lenguajes de programación más usados en el mundo.

1.6.4 Entorno de desarrollo

Se utiliza **Android Studio** en su versión 3.5.3.0 pues es el Entorno de Desarrollo Integrado (IDE) oficial para el desarrollo de aplicaciones Android, basado en IntelliJ IDEA. Además del potente editor de códigos

y las herramientas para desarrolladores de IntelliJ, Android Studio ofrece incluso más funciones que aumentan la productividad cuando desarrollas apps[5] para Android (Developers, 2019), como las siguientes:

- Un sistema de compilación flexible basado en Gradle (descrito en el epígrafe 1.5.6)
- Un emulador rápido y cargado de funciones
- Un entorno unificado donde puedes desarrollar para todos los dispositivos Android
- Aplicación de cambios para insertar modificaciones al código y los recursos al proyecto en ejecución sin reiniciar la emulación del mismo.

Además de todas estas ventajas se selecciona Android Studio por que el autor posee experiencia en el uso de este IDE y en la comunidad de Android de la UCI se puede contar con apoyo tanto para temas de Android como de Android Studio, lo que facilita el desarrollo de la propuesta de solución.

1.6.5 Android SDK 27

El SDK (*Software Development Kit*, Kit de Desarrollo de Software), es un conjunto de herramientas y programas de desarrollo que permite al programador crear aplicaciones para un determinado paquete de software, estructura de software, plataforma de hardware, sistema de computadora, consulta de videojuego, sistema operativo o similar (Alegsa, 2018).

El SDK de Android contiene las herramientas para desarrollar aplicaciones desde la línea de comandos, así como otras herramientas que ayudaran a encontrar y diagnosticar problemas, y optimizar las aplicaciones. El SDK de Android incluye:

- Librerías necesarias.
- Entorno de depuración.
- Emulador de dispositivos móviles.
- Documentación relevante para las APIs[6] de Android.

[5] Apps: es una abreviatura de la palabra en inglés application

[6] APIs: Aplication Programming Interface, traducido al español Interfaz de Programación de Aplicaciones.

- Código fuente de ejemplo.
- Tutoriales para el sistema operativo Android.

1.6.6 Gradle v5.1.1

Gradle es una herramienta para automatizar la construcción de proyectos, tareas de compilación, pruebas, empaquetado y el despliegue de los mismos (Gradle Inc., 2016). Utilizado como compilador de disímiles lenguajes de programación como Java, Python, C/C++, etcétera. Provee plugins[7] y otras herramientas para la integración con IDEs como Eclipse, Android Studio e IntelliJ (Gradle Inc, 2016).

1.6.7 Sistema gestor de base de datos

Un Sistema de Gestión de Bases de Datos (SGBD) es una capa de software necesaria para crear, manipular y recuperar datos desde una base de datos. Esta es una herramienta de propósito general, útil para estructurar, almacenar y controlar los datos ofreciendo interfaces de acceso a la base de datos. Las tareas fundamentales que desempeñan estos sistemas hacen referencia a la seguridad de acceso a los datos, al mantenimiento de la integridad de los datos, a mecanismos de recuperación debidos a fallos físicos y lógicos, al control de concurrencia en el momento de acceder a los datos y a la eficiencia del sistema evaluado, generalmente, en términos del tiempo de respuesta a las consultas de los usuarios (Millán, 2017).

Para el desarrollo de la solución se emplea el siguiente sistema gestor de bases de datos:

SQLite

Es una librería de software que implementa un gestor de bases de datos relacionales adaptada para dispositivos de bajo consumo o móviles. Está escrito en C y rodeado de un envoltorio Java proporcionado por el SDK de Android. Es de código abierto. La base de datos se guarda como un único fichero en el dispositivo. Está integrado o auto contenido en el programa que lo usa, no existe gestor de base de datos externo. Todas las operaciones de base de datos se manejan dentro de la aplicación mediante llamadas y funciones contenidas en la librería SQLite. Base de Datos de hasta 2 Terabytes de tamaño. Permite campos

[7]Plugins: programa informático que se relaciona con otro para agregarle una función nueva y generalmente muy específica.

de tipo BLOB (Binary Large Object) para almacenar archivos binarios grandes como puede ser una imagen. Es posible trabajar con Bases de Datos virtuales en memoria, sin archivo físico (Sqlite, 2019).

El autor decide utilizar SQLite debido a que posee experiencia en su uso, además de existir una vasta documentación que apoyan el desarrollo de aplicaciones Android utilizando esta librería.

1.6.8 Herramienta de pruebas de software

Para controlar la calidad del software, es necesario utilizar herramientas que sean capaces de ejecutar pruebas de forma autónoma y masiva, de forma tal que permitan la validación desde el punto de vista estático y de caja blanca del software.

JUnit v5

Es una herramienta para Java, creada por Erich Gamma y Kent Beck apoyada y adoptada por grupos partidarios de la programación extrema. Esta y todas las herramientas descendientes de JUnit consisten en una serie de clases que se auxilian en la preparación y codificación de casos de pruebas y algunos mecanismos auxiliares, que en conjunto, permiten ejecutar y verificar el cumplimiento de los casos de prueba. Incluido a través de plugins en los principales IDEs como son NetBeans, Eclipse y Android Studio. Además provee una interfaz que permite amenizar la ejecución de grupos de casos de pruebas (JUnit, 2016).

1.7 Conclusiones parciales del capítulo

En este capítulo se fundamentó teóricamente la investigación, para el desarrollo de la propuesta de solución. El estudio de las herramientas de apoyo a la migración existentes arrojó como resultado que no solucionan de forma completa el problema a resolver, debido a que el proceso se realiza de forma manual, para lo que se decidió desarrollar una aplicación móvil para dispositivos con sistema operativo Android. Se eligió como metodología para guiar el proceso de desarrollo de software AUP variación para la UCI, debido a que esta se ajusta a las necesidades de los proyectos de desarrollo de software realizados en la institución y como herramienta de modelado Visual Paradigm versión 8.0, usando el lenguaje UML, que permite la creación de los diagramas necesarios en la investigación. Para la implementación de la solución se estableció el uso de Android Studio en su versión 3.3.2 como IDE de desarrollo, haciendo uso de Java 8 como lenguaje de

programación y SQLite como sistema gestor de bases de datos. Para la etapa de pruebas se estableció como herramienta a JUnit 5 integrado en Android Studio que permite la realización de las pruebas unitarias desde un mismo IDE.

Capítulo 2. Análisis y Diseño de la aplicación Android para la gestión de información del diagnóstico para la migración a software libre de los servicios telemáticos en los OACE

Introducción

En el presente capítulo se describen las principales características de la propuesta de solución. Se realiza la especificación de los requisitos funcionales y no funcionales, se conforman las historias de usuario correspondientes a los requisitos y los prototipos de diseño de interfaz de usuario. Se describe la arquitectura de la aplicación, así como los patrones de diseño utilizados, y se representa el diagrama de clases del diseño, el diagrama de paquetes y el modelo de datos de la solución.

2.1 Propuesta de solución

El presente trabajo propone una aplicación Android para mejorar la gestión de la información en el diagnóstico para la migración a software libre de los servicios telemáticos de los OACE. El especialista debe añadir la entidad a migrar poniendo su nombre y descripción, los cuales serán almacenados en una base de datos para mantener la persistencia de los datos. Luego, en la empresa añadida podrá adicionar los servidores que posea la entidad y a estos se le agregará la información necesaria, llenando un formulario y además de esta misma forma podrá almacenar los datos de los servicios telemáticos que brinda. Además podrá añadir la topología de red que posee la empresa, mediante otro formulario y podrá modelar gráficamente dicha topología. La aplicación permitirá generar un informe en formato xls (Excel), con toda la información recolectada de la entidad.

2.2 Requisitos

La obtención de requisitos es uno de los pasos fundamentales en el desarrollo de software. Esta tarea está encaminada a identificar los requerimientos del cliente, analizar las necesidades y especificar los requisitos de la propuesta de solución. En la presente investigación se identificaron los requisitos funcionales y no funcionales de la solución a partir de entrevistas con el cliente. A continuación se muestran los requisitos funcionales obtenidos.

2.2.1 Requisitos funcionales

Los requisitos funcionales definen las acciones que debe realizar el sistema. 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 (Sommerville, 2011).

Para concretar las funcionalidades a desarrollar se obtienen los siguientes requisitos funcionales:

Tabla 1. Requisitos funcionales.

No	**Nombre**	**Descripción**	**Prioridad**	**Complejidad**
RF. 1	RF1. Listar Empresas	La aplicación debe permitir mostrar un listado con las empresas a las que se les ha realizado o se le realiza actualmente el diagnóstico para la migración a software libre de los servicios telemáticos.	Media	Alta
RF. 2	RF2. Adicionar Empresa	La aplicación debe permitir adicionar una empresa al sistema para su posterior consulta o modificación.	Alta	Alta
RF. 3	RF3. Modificar Empresa	La aplicación debe permitir modificar los datos de las empresas añadidas al sistema.	Media	Media
RF. 4	RF4. Eliminar Empresa	La aplicación debe permitir eliminar las empresas añadidas al sistema.	Media	Media

RF. 5	RF5. Listar Servidores.	La aplicación debe mostrar la lista de servidores pertenecientes a la empresa seleccionada.	Media	Alta
RF. 6	RF6. Adicionar Servidores.	La aplicación debe permitir adicionar servidores a las empresas.	Alta	Alta
RF. 7	RF7. Modificar Servidores.	La aplicación debe permitir modificar los datos de los servidores	Media	Media
RF. 8	RF8. Eliminar Servidores.	La aplicación debe permitir eliminar los servidores de la empresa.	Media	Media
RF. 9	RF9. Listar Servicios Telemáticos	La aplicación debe mostrar la lista de servicios pertenecientes a un servidor seleccionado en la empresa.	Media	Alta
RF.10	RF10. Adicionar Servicios Telemáticos.	La aplicación debe permitir adicionar servicios telemáticos a los servidores de las empresas.	Alta	Alta
RF.11	RF11. Modificar Servicios Telemáticos.	La aplicación debe permitir modificar los servicios telemáticos añadidos.	Media	Media
RF.12	RF12. Eliminar Servicios Telemáticos.	La aplicación debe permitir eliminar los servicios telemáticos del servidor.	Media	Media

RF.13	RF13. Adicionar Topología de red.	La aplicación debe permitir adicionar la topología de red de la empresa.	Alta	Media
RF.14	RF14. Modificar Topología de red.	La aplicación debe permitir modificar los datos de la topología de red de la empresa.	Media	Media
RF.15	RF15. Eliminar topología de red.	La aplicación debe permitir eliminar la topología de red de la empresa.	Media	Media
RF. 16	RF16. Modelar topología de red.	La aplicación debe permitir modelar gráficamente la topología de red de la empresa.	Alta	Alta
RF.17	RF17. Generar informe	La aplicación debe permitir generar un informe en pdf, con la información de la empresa.	Alta	Alta

2.2.2 Requisitos no funcionales

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener; son las características que lo hacen atractivo, usable, rápido y confiable (Sommerville, 2011). Para lograr la satisfacción del cliente, y una buena calidad en el sistema, se definieron los siguientes requisitos no funcionales, basados en lo establecido por las normas ISO 25000 Calidad del Producto de Software, específicamente la ISO/IEC 25010, la cual define las características de calidad que se tienen en cuenta al evaluar las propiedades de un producto de software, además de las pruebas realizadas con los dispositivos móviles emulados en el IDE de desarrollo:

Portabilidad

RNF1. Sistema operativo Android con versión desde 4.0 hasta 7.0.

RNF2. Memoria RAM del dispositivo con capacidad mínima de 256 MB.

RNF3. Almacenamiento interno con capacidad mínima de 100 MB.

Usabilidad

RNF4. Interfaz de usuario basada en Material Design[8].

La metodología de desarrollo de software empleada define distintas formas para encapsular los requisitos, para el desarrollo de la solución se emplea el escenario que usa las historias de usuario, como se muestra en el epígrafe siguiente.

2.3 Historias de usuario

Las historias de usuario (HU) constituyen una forma de administración de requisitos sin tener que elaborar gran cantidad de documentos formales y sin requerir de mucho tiempo para administrarlos. Las mismas son escritas utilizando el lenguaje común del usuario. Son empleadas en las metodologías de desarrollo ágiles para la especificación de requisitos.

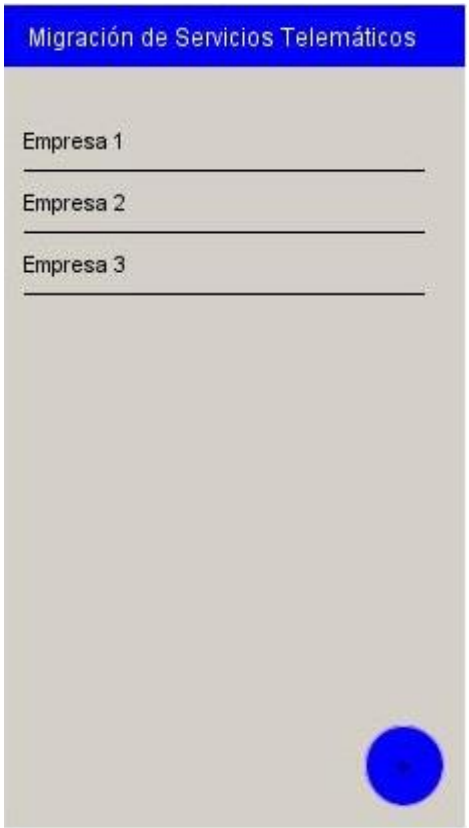
Para la descripción de requisitos funcionales de la propuesta de solución se define una historia de usuario para cada uno de ellos, para un total de 17 historias de usuario. A continuación se muestra la HU “Listar Empresas”, como ejemplo. El resto de las HU se encuentran en el anexo 3.

Tabla 2. Historia de usuario: Listar Empresas.

Número: HU-1	Nombre del requisito: Listar Empresas
Programador: Alejandro Puerta Díaz	Iteración Asignada: 1
Prioridad: Media	Tiempo Estimado: 7 días

[8] Material Design: normativa de diseño enfocado en la visualización del sistema operativo Android, además en la web y en cualquier plataforma, este debe brindar una interfaz amigable y comprensible para el usuario.

Riesgo en Desarrollo: Alto	Tiempo Real: 4 días
Descripción: Debe mostrar un listado de empresas anteriormente añadidas a la aplicación.	
Observaciones: Cada ítem de la lista tiene el nombre de la empresa como se muestra en el prototipo de interfaz (1), en el caso de que la lista de empresas esté vacía la apariencia de la pantalla será la del prototipo de interfaz (2), mostrando una imagen con el mensaje “No hay archivos”, una vez añadida una empresa esto se intercambia por la lista de empresas.	
Prototipo de interfaz:	



Prototipo 1

Prototipo 2

2.4 Análisis y diseño

En esta disciplina los requisitos pueden ser refinados y estructurados con el objetivo de lograr una comprensión más precisa de estos, y una descripción que sea fácil de mantener y ayude a la estructuración del sistema (incluyendo su arquitectura). Se modela el sistema, su forma y su arquitectura para que soporte todos los requisitos, incluyendo los requisitos no funcionales (Rodríguez, 2015).

En los epígrafes siguientes se muestran las actividades desarrolladas en esta disciplina, así como los productos de trabajo elaborados en el proceso de desarrollo de la investigación.

2.4.1 Descripción de la arquitectura

Según (Pressman, 2010) 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.

La arquitectura empleada para el desarrollo del sistema es Modelo-Vista-Controlador.

Esta arquitectura separa presentación e interacción de los datos del sistema. El sistema se estructura en tres componentes lógicos que interactúan entre sí:

- Modelo: el componente maneja lo referente a la persistencia de datos de la aplicación, las clases entidades, el acceso a la red y los elementos necesarios para manejar dichos elementos (paquete Model).
- Vista: el componente representa la interfaz gráfica para la interacción con el usuario. Dentro se ubican todos los componentes que intervienen en la visualización del resultado de la comunicación con el Controlador. (paquete View).
- Controlador: componente que contiene las clases que interactúan con la Vista recibiendo las solicitudes de eventos de los usuarios y con el Modelo registrando los cambios realizados por el mismo (paquete Controller).

A continuación se representa el diagrama de paquetes:

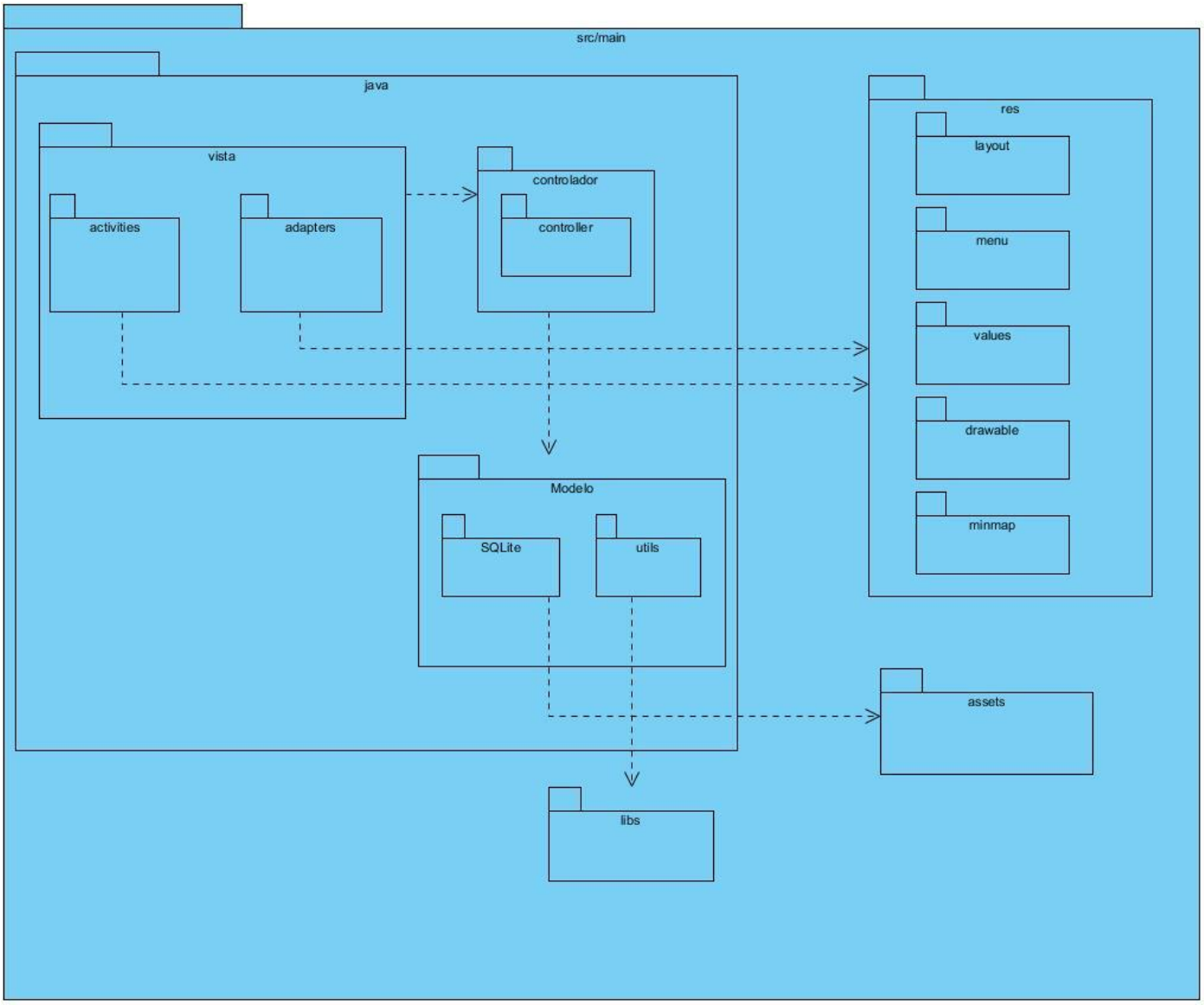


Ilustración 1. Diagrama de paquetes.

Los proyectos Android eventualmente construyen en los archivos .apk el código fuente, los recursos y elementos necesarios para el funcionamiento del software. Estos archivos .apk son los ejecutables de

instalación de las aplicaciones. La estructura de paquetes y archivos de un proyecto de Android es la siguiente:

src/main: contiene los archivos de clases y actividades.

bin/: directorio de salida de la construcción del archivo .apk y otros recursos compilados.

gen/: contiene los archivos de Java, como el archivo R. Java para el control de los identificadores de todos los elementos implicados en la herramienta.

res/: contiene los recursos de la aplicación, como archivos drawable, layout y los valores string (archivo XML que contiene los textos visualizados en la herramienta).

drawable/: para archivos de imágenes.png, .jpeg o .gif; archivos XML que describen las formas drawable u objetos drawable que contengan múltiples estados.

layout/: archivos XML que son compilados en los layouts de pantalla.

values/: para archivos XML que son compilados en diversos tipos de recursos. A diferencia de otros recursos del directorio res/, los recursos que son escritos a archivos XML en esta carpeta no son referenciados por su nombre, sino que el tipo de elemento XML contenido es controlado a través de la clase R.

libs/: contiene las librerías privadas que son utilizadas internamente en el proyecto.

assets/: contiene los demás ficheros auxiliares necesarios para la aplicación (y que se incluirán en su propio paquete), como por ejemplo ficheros de configuración y de datos.

2.5 Patrones de diseño

Un patrón de diseño provee un esquema para refinar componentes de un sistema de software y la forma en que se relacionan entre sí. Describe una estructura generalmente recurrente de comunicación de

componentes que resuelve un problema de diseño general dentro de un contexto particular (Almeira, y otros, 2016).Los patrones de diseño se caracterizan por:

- Representar soluciones técnicas a problemas concretos.
- Propiciar la reutilización.
- Representar problemas frecuentes.

Patrones GRASP del inglés General Responsibility Assignment Software Patterns, estos patrones brindan principios generales para asignar responsabilidades y se utilizan sobre todo en la realización de diagramas de interacción (Larman, 2004).

Principales patrones GRASP:

- Experto: se usa más que cualquier otro al asignar responsabilidades, es un principio básico que suele utilizarse en el diseño orientado a objeto. Consiste en la asignación de una responsabilidad a la clase que cuenta con la información necesaria para llevarla a cabo. El uso de este patrón da pie a un bajo acoplamiento y una alta cohesión, lo que favorece tener sistemas más robustos y de fácil mantenimiento. El cumplimiento de una responsabilidad requiere a menudo información distribuida en varias clases de objetos (Larman, 2004). En la aplicación este patrón se evidencia en las clases ModelEmpresas.java, ModelServidores.java, ModelServicios.java y ModelTopologia.java las cuales se encargan de manejar la información perteneciente a la entidad donde se está migrando.

- Alta Cohesión: en la perspectiva del diseño orientado a objetos, la cohesión o cohesión funcional es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme, clases con responsabilidades moderadas en un área funcional que colaboran con las otras para llevar a cabo las tareas (Larman, 2004). La clase DBHelper.java usa constantes para generalizar los elementos descritos con el fin de mejorar la reusabilidad, además administra la conexión de la base de datos y su estructuración.

- Bajo Acoplamiento: el acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Una clase con bajo o débil acoplamiento no depende de muchas otras. El bajo acoplamiento soporta el diseño de clases más independientes y reutilizables, lo cual reduce el impacto de los cambios y acrecienta la oportunidad de una mayor productividad (Larman, 2004). Este patrón se manifiesta en la clase ExpotExcel.java que permite exportar la información recolectada en un documento en formato Excel en el dispositivo y posee una cantidad mínima de relaciones entre clases.

- Controlador: un controlador es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. La clase Controladora.java es la encargada de interactuar con la vista y el acceso a datos.

Patrones GOF Los patrones de diseño GOF fueron publicados por Gamma, Helm, Johnson y Vlossodes en 1995. Conocidos como patrones de la pandilla de los cuatro (gang of four). Se clasifican en creacionales, estructurales y de comportamiento (Gamma, y otros, 1994).

- Singleton: 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 solo tenga una instancia y proporcionar un punto de acceso global a ella (Gamma, y otros, 1994). En las clases DBHelper.java y Controladora.java se implementa un patrón Singleton, lo cual significa definir un miembro estático de la clase y generar un método estático que permita la obtención del único miembro.

- Fachada: permite proveer una interfaz unificada sencilla que haga de intermediaria entre un cliente y una interfaz o grupo de interfaces más complejas. Se utiliza ampliamente para hacer más fácil y entendible el trabajo con bibliotecas de software (Gamma, y otros, 1994). Las clases modelos utilizadas permiten representar los registros de la base de datos como entidades dentro de la aplicación Android que ayudan a procesar eventos desde la vista y la clase Controladora actúa como intermediario al modelo para el acceso a los datos.

2.5.1 Diagrama de clases del diseño

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

- Clases, relaciones y atributos.
- Métodos.
- Información acerca del tipo de los atributos.
- Dependencias.

A continuación, se muestra el diagrama de clases del diseño de la aplicación:

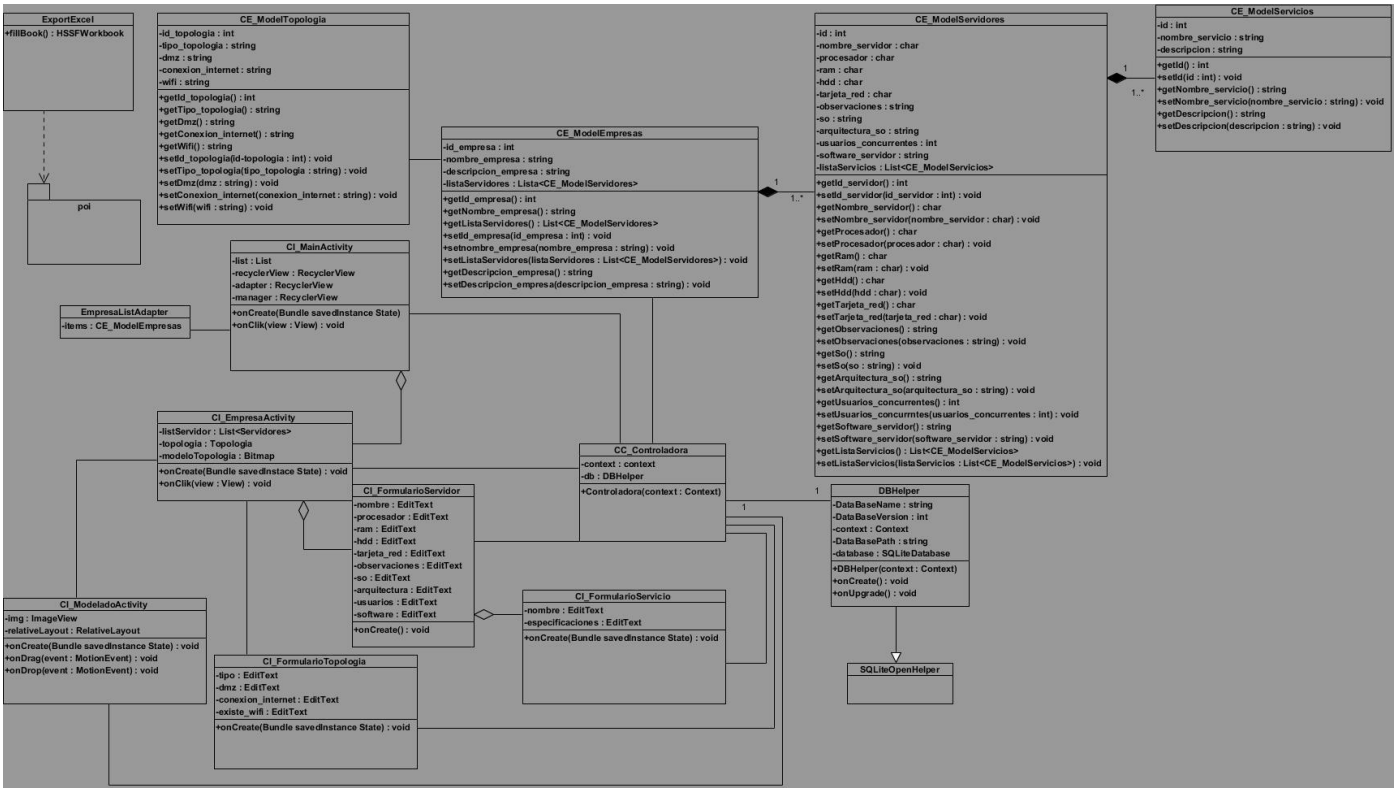


Ilustración 2. Diagrama de clases de diseño.

2.5.2 Modelo de datos

Los modelos de datos aportan la base conceptual para diseñar aplicaciones que hacen un uso intensivo de datos. Un modelo de datos describe las representaciones lógicas y físicas de datos persistentes utilizados por la aplicación. Es usado para describir la representación lógica y física de la información persistente manejada por el sistema. La propuesta de solución presenta las clases entidad Empresa, Servidor, Servicio y Topología, de las cuales sus atributos son los datos que deben persistir en la base de datos y mediante los cuales se relacionan como es el caso de los identificadores.

A continuación, se representa el diseño de la base de datos:

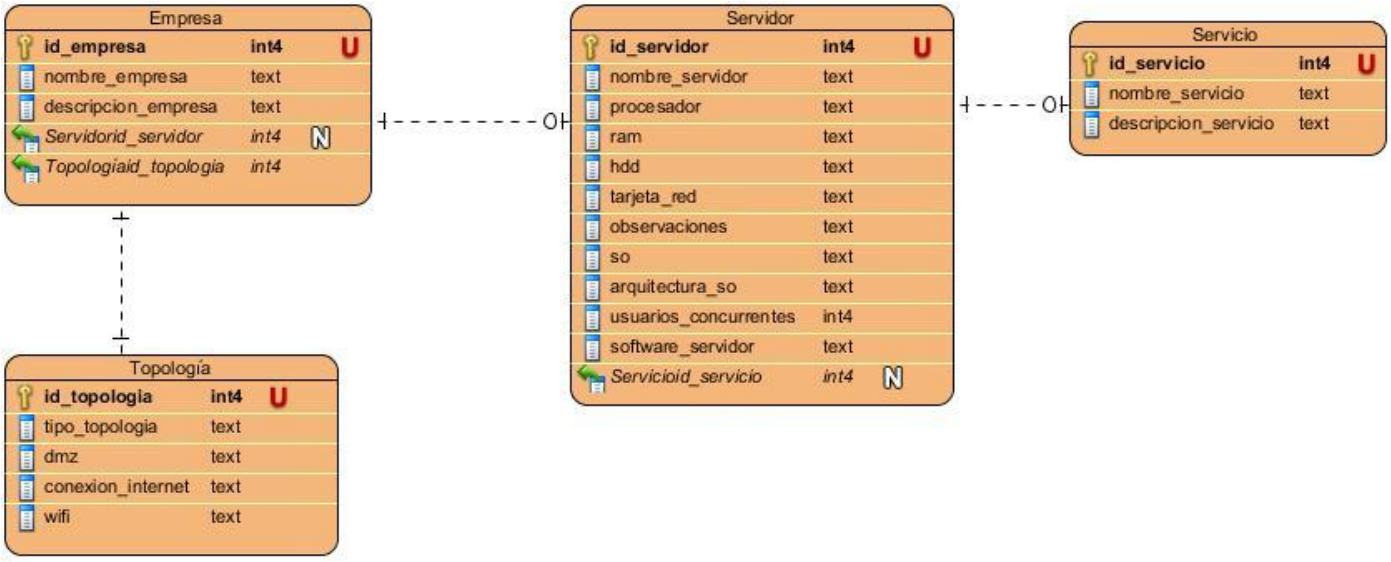


Ilustración 3. Modelo físico de datos.

2.7 Conclusiones parciales

A partir del desarrollo del presente capítulo se describen textualmente los requisitos funcionales y no funcionales, lo cual permitió una mayor comprensión de las funcionalidades de la aplicación a desarrollar. El empleo de la arquitectura MVC y los patrones de diseño GRASP y GOF contribuyen al diseño de la aplicación, proporcionan una estructura más sólida y permiten el empleo de buenas

prácticas de programación, la reutilización de código y a evitar futuras anomalías de mantenimiento. Con la realización del diagrama de clases del diseño se obtuvo una visión más exacta de la aplicación en términos de implementación. El modelo de datos relacional especifica las clases persistentes de la base de datos. Como resultado principal de este capítulo se propuso el diseño de una aplicación Android para mejorar la gestión de la información en el diagnóstico de los servicios telemáticos en la migración a software libre.

Capítulo 3. Implementación y Prueba de una aplicación Android para la gestión de información en el diagnóstico para la migración a software libre de los servicios telemáticos de los OACE

Introducción

En el presente capítulo se exponen las especificaciones asociadas a la implementación de la aplicación. Se describen las pautas de codificación utilizadas, además la aplicación es sometida a un proceso de pruebas con el objetivo de verificar el cumplimiento de los requerimientos especificados anteriormente.

3.1 Implementación

La codificación de la propuesta de solución se realiza una vez que se definen las historias de usuario y se concluye 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.

3.1.1 Estándares de codificación

Los estándares de codificación se emplean para una mejor comprensión del código por parte del equipo de desarrollo y así mantener la coordinación del mismo (Aguila, 2017). Estos son parte de las llamadas buenas prácticas de programación y son un conjunto no formal de reglas, que han ido surgiendo en las distintas comunidades de desarrolladores con el paso del tiempo y las cuales, bien aplicadas pueden incrementar la calidad del código.

A continuación se presentan los estándares de codificación usados en la solución:

- Se empleará el idioma inglés para la codificación.
- El código tabulado y espaciado a través del formato que aplica la combinación de teclas ALT+Shift+F definida en la configuración del IDE Android Studio, en la sección referente a los atajos de teclado (File/Settings /Keymap).

- Los comentarios que abarcan un bloque de instrucciones se escriben comenzando con los caracteres “/*” y terminando con “*/”.
- Los comentarios para una línea comienzan con los caracteres “//”.
- No se usan nombres de variables que coincidan con palabras reservadas.
- No se emplean caracteres especiales (@, #, $, %, ^, &, * u otros) para la nomenclatura.
- Se debe declarar cada variable en una línea distinta, de esta forma cada variable se puede comentar por separado.
- Los nombres de variables globales deben escribirse en mayúsculas con las palabras separadas por un guion bajo ("_"). Deben ser declaradas como public static.

3.1.2 Diagrama de Despliegue

El diagrama de despliegue modela la topología del hardware sobre el que se ejecuta un sistema, el cual muestra la configuración de los nodos que participan en la ejecución de los componentes que residen en ellos. Además representa el despliegue físico de un componente (Aguila, 2017). La aplicación implementada se despliega en un dispositivo con sistema operativo Android y usa una base de datos local para almacenar los datos de forma persistente. A continuación se presenta el diagrama de despliegue de la solución:

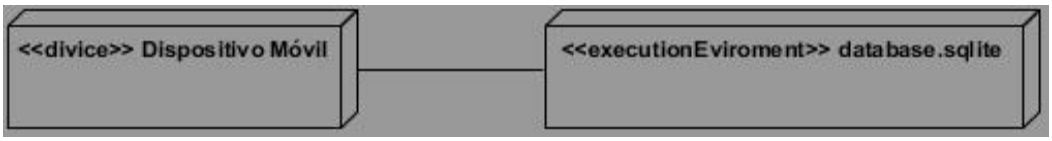


Ilustración 4: Diagrama de despliegue.

3.2 Interfaces del sistema

A continuación se muestra una de las interfaces del sistema, el resto se encuentra en el anexo 5:

Ilustración 5: Interfaz principal

3.3 Pruebas del 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 (Pressman, 2010).

3.3.1 Estrategia de prueba

Una estrategia de pruebas de software proporciona una guía que describe los pasos que deben realizarse como parte de la prueba. Por tanto, una estrategia de prueba debe incorporar la planificación de la prueba, el diseño de casos de prueba, la ejecución de la prueba y la recolección y evaluación de los resultados arrojados. Las pruebas se dividen en los siguientes niveles principales: pruebas de unidad, de integración, de validación, de sistema y de aceptación (Pressman, 2010).

Para la propuesta de solución se utilizó una estrategia de pruebas basada en la ejecución de las mismas tomando como guía tres de los niveles propuestos por Pressman: unidad, validación y aceptación.

Pruebas de unidad

Las pruebas de unidad o unitarias son el proceso de probar componentes del programa tales como métodos o clases de objetos. Las funciones o los métodos individuales son el tipo más simple de componente. Las pruebas deben llamarse para dichas rutinas con diferentes parámetros de entrada (Sommerville, 2011).

El método aplicado para esta prueba fue el de caja blanca, donde las pruebas se enfocan en la estructura de control del programa. Los casos de prueba se derivan para asegurar que todos los enunciados en el programa se ejecutaron al menos una vez durante las pruebas y que todas las condiciones lógicas se revisaron (Pressman, 2010).

Para automatizar este tipo de pruebas sobre la propuesta de solución se decidió emplear la herramienta JUnit, la cual está integrada con el IDE Android Studio utilizado para el desarrollo de la aplicación. Se implementaron los casos de prueba a través de la clase ControladoraTest, la cual comprueba el correcto funcionamiento de las principales funcionalidades definidas en la clase Controladora, responsable de la lógica principal de la aplicación.

Pruebas de validación

La prueba de validación proporciona un aseguramiento final de que el software cumple con todos los requisitos funcionales, de comportamiento y desempeño. La prueba se concentra en las acciones visibles

para el usuario y en la salida que este puede reconocer. La validación se alcanza cuando el software funciona de tal manera que satisface las expectativas razonables (especificación de requisitos de software) del cliente. Se logra mediante una serie de pruebas que demuestran que se cumple con los requisitos (Pressman, 2010).

Para realizar estas pruebas se hace necesario emplear pruebas funcionales, ya que aseguran el apropiado trabajo de los requisitos funcionales, incluyendo la navegación, entrada de datos, procesamiento y obtención de resultados. Las metas de estas pruebas son verificar la apropiada aceptación de datos y verificar el procesamiento, recuperación e implementación adecuada de las reglas del negocio (Pressman, 2010).

Para llevarlas a cabo, el método a emplear fue el de caja negra, el cual se centra en los requisitos funcionales del software. La prueba de caja negra permite obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. Estas pruebas pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto.

Como técnica se utiliza la partición de equivalencia, o sea, dividir el dominio de entrada de un programa en clases de datos a partir de las cuales pueden derivarse casos de prueba. Es una de las más efectivas pues permite examinar los valores válidos e inválidos de las entradas existentes en la herramienta, descubre de forma inmediata una clase de errores que, de otro modo, requieren la ejecución de muchos casos antes de detectar el error genérico. El diseño de casos de prueba para la partición de equivalencia se basa en una evaluación de las clases de equivalencia para una condición de entrada (Pressman, 2010).

Diseño de casos de pruebas

Los casos de prueba se diseñan según las funcionalidades descritas en las historias de usuario. La intención que se persigue con estos artefactos es lograr una compresión específica de las condiciones que la solución debe cumplir. Cada caso de prueba muestra la especificación de una historia de usuario, dividida en secciones y escenarios, se detallan las funcionalidades descritas en ella y se describe cada variable.

A continuación se muestra el caso de prueba correspondiente a la historia de usuario “Listar empresa”. Los casos de prueba asociados al resto de las historias de usuario son definidos en el anexo 3.

Tabla 3: Caso de prueba de validación Listar empresa.

<table>
<tr><td colspan="4">Caso de Prueba de Validación</td></tr>
<tr><td colspan="2">Código: CP1-HU1</td><td colspan="2">HU 11: Listar empresa</td></tr>
<tr><td colspan="4">Responsable: Alejandro Puerta Díaz</td></tr>
<tr><td colspan="4">Descripción: El caso de prueba inicia al abrir la aplicación por primera vez, esta muestra una imagen con un letrero que dice “No hay archivos”, indicando que la lista está vacía. Al pulsar el botón de adicionar, presenta un formulario para añadir la información de la empresa. El caso de prueba culmina cuando muestra el nombre de la empresa como ítem de la lista.</td></tr>
<tr><td colspan="4">Condiciones de ejecución: Para añadir empresas a la lista se debe cumplir con las siguientes condiciones:

• Otorgar los permisos de escritura y lectura del almacenamiento del dispositivo.
• Los campos del formulario no deben estar vacíos.</td></tr>
<tr><td>Escenario</td><td>Descripción</td><td>Respuesta del sistema</td><td>Flujo central</td></tr>
<tr><td>EC 1.1 Añadir una empresa y se muestra en la lista</td><td>El usuario ejecuta la aplicación, otorga los permisos una primera vez y añade la empresa correctamente y se muestra en la lista.</td><td>La aplicación muestra al usuario el formulario de la empresa, este pide el nombre y la descripción de la empresa. Una vez escrita la información, se guarda y cambia la imagen de vacío por el nombre de la empresa como ítem de la lista.</td><td>El usuario pulsa el botón de adición, llena el formulario y pulsa guardar.</td></tr>
<tr><td>EC 1.2 Error al listar la empresa.</td><td>El usuario añade la empresa y no se muestra en la lista.</td><td>La aplicación no muestra el nombre de la empresa añadida y</td><td>El usuario pulsa el botón de adición, llena el formulario y pulsa guardar.</td></tr>
</table>

		mantiene la imagen de vacío.	

Pruebas de aceptación

Las pruebas de aceptación constituyen la etapa final en el proceso de pruebas, antes de que el sistema se acepte para uso operacional. El sistema se pone a prueba con datos suministrados por el cliente del sistema, en vez de datos de prueba simulados. Estas pruebas revelan los errores y las omisiones en la definición de requerimientos del sistema, ya que los datos reales ejercitan el sistema en diferentes formas a partir de los datos de prueba. Además de problemas de requerimientos, donde las instalaciones del sistema en realidad no cumplan las necesidades del usuario o cuando sea inaceptable el rendimiento del sistema (Sommerville, 2011).

Para llevar a cabo las pruebas de aceptación se hizo necesario a través del tipo de prueba alfa, que consiste en incorporar al cliente o usuario final directamente al proceso de prueba de la aplicación. Entiéndase por prueba alfa cuando esta se lleva a cabo en el sitio del desarrollador por un grupo representativo de usuarios finales, es decir, en un ambiente controlado, propiciando que el desarrollador pueda registrar errores y problemas de uso (Pressman, 2010).

3.4 Ejecución y resultados de las pruebas de software

Para la ejecución de las pruebas de validación de tipo alfa se utilizaron los siguientes dispositivos:

Teléfono Celular:

- Modelo: Lenovo A560
- Versión de sistema operativo: Android 4.3 (Jelly Bean)
- API level 18

- RAM 418 MB
- Almacenamiento interno 1,23 Gb
- Almacenamiento externo 16 Gb

Tablet:

- Modelo: Acer B1-770
- Versión de sistema operativo Android: 5.0.1 (Lollipop)
- API level
- RAM 1 GB
- CPU Quad core
- Almacenamiento interno 16 GB

Emulador Genymotion:

- Modelo Genymotion _vbox86p_4.4_150216_21300
- Versión de SO Android 4.4.2 (Kitkat)
- SDK 19
- RAM 512 MB
- Almacenamiento interno 2024 MB
- Almacenamiento externo 8189 MB

Los problemas detectados en el período de pruebas de validación y aceptación se clasificaron en: no conformidades significativas (NCS) y en no conformidades no significativas (NCNS).

A continuación, se describen los aspectos que se tuvieron en cuenta en cada clasificación:

- NCS: son las no conformidades referentes a las funcionalidades de la aplicación como son las validaciones incorrectas o respuestas de la aplicación diferentes a lo descrito previamente en las historias de usuario.

- NCNS: son las no conformidades en cuanto al diseño de la propuesta de solución.

Fueron realizadas 3 iteraciones de pruebas, ejecutándose al término de cada una de ellas las pruebas de regresión, con el objetivo de asegurar que al resolverse las no conformidades detectadas, estas no introdujeran nuevos errores en la solución.

En la primera iteración se detectaron 6 NCS y 4 NCNS. Las cuales fueron resueltas satisfactoriamente en la iteración.

No conformidades significativas:

1. La aplicación no actualiza el listado de empresas existentes en el dispositivo móvil cuando se elimina un lugar de la lista.

2. La aplicación no actualiza los campos del servidor cuando se guarda el servicio y los muestra vacíos.

3. La aplicación no actualiza el nombre en la lista de empresas cuando este es modificado.

4. La aplicación no guarda la imagen con el gráfico de la topología de red de la empresa.

5. La aplicación no muestra el listado de servicios.

6. La aplicación no elimina de la base de datos los servicios asociados a un servidor una vez eliminado el servidor.

No conformidades no significativas:

1. La aplicación muestra el botón de adicionar empresa encima del botón guardar.
2. La aplicación no muestra completamente el texto de la descripción de las empresas.
3. La aplicación muestra el mensaje de notificación “Se guardó la imagen con éxito” con errores ortográficos.
4. La aplicación no muestra el ícono del botón “Adicionar Topología” al mismo nivel del enunciado.

En la segunda iteración se detectaron 2 NCS y 2 NCNS, siendo solucionadas en la misma iteración.

No conformidades significativas:

1. La aplicación no muestra un mensaje de error cuando se dejan campos vacíos en el formulario de los servidores.
2. La aplicación no muestra los datos de un servicio al consultar su estado.

No conformidades no significativas:

1. La aplicación muestra intercambiados los campos nombre y descripción en la interfaz de un servicio.
2. La aplicación muestra deformación visual en la interfaz de la topología de red.

En la tercera iteración no se detectaron no conformidades, por lo que se demostró que la aplicación cumple con los requisitos funcionales establecidos y fue considerada concluida.

A continuación, se muestra un gráfico con un resumen de los resultados obtenidos tras la realización de las pruebas:

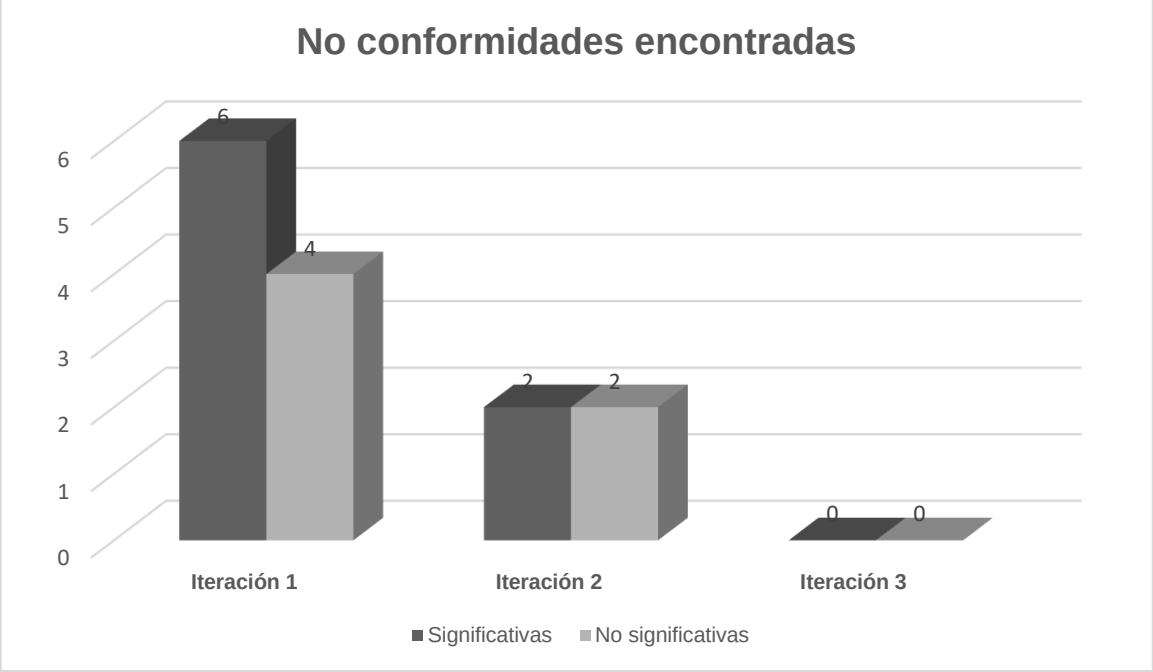


Ilustración 6: NC encontradas por iteraciones en el proceso de prueba.

3.5 Evaluación del cumplimiento del objetivo de la investigación

Para la evaluación del cumplimiento del objetivo de la investigación el autor realizó una encuesta a 10 especialistas de migración de servicios telemáticos pertenecientes al Centro de Software Libre de la UCI (ver en Anexo 4), utilizando la técnica de Iadov, para determinar su grado de satisfacción con la herramienta desarrollada. Esta técnica se basa en el análisis de un cuestionario que tiene una estructura interna determinada, la cual sigue las relaciones que se establecen entre tres preguntas cerradas (cuya relación el sujeto desconoce) y el análisis posterior de cinco preguntas abiertas. La relación entre las preguntas cerradas se establece a través del denominado “Cuadro lógico de Iadov” (ver Tabla 4), indicando la posición de cada de los especialistas en la escala de satisfacción.

¿Le satisface la solución desarrollada para la gestión de información en el diagnóstico para la migración a software libre de los servicios telemáticos?	¿Considera usted que se deba seguir realizando el diagnóstico para la migración a software libre de los servicios telemáticos sin el uso de una herramienta que mejore la calidad del proceso?								
	No			No sé			Sí		
	¿Utilizaría la solución para realizar el diagnóstico para la migración a software libre de los servicios telemáticos y así mejorar la gestión de la información?								
	Sí	No sé	No	Sí	No sé	No	Sí	No sé	No
Me gusta mucho	1	2	6	2	2	6	6	6	6
No me gusta mucho	2	2	3	2	3	3	6	3	6
Me da lo mismo	3	3	3	3	3	3	3	3	3
Me disgusta más de lo que me gusta	6	3	6	3	4	4	3	4	4
No me gusta nada	6	6	6	6	4	4	6	4	5
No sé qué decir	2	3	6	3	3	3	6	3	4

Tabla 4: Cuadro lógico de Iadov

El número resultante de la interrelación de las tres preguntas indica la posición de cada sujeto en la escala de satisfacción. La escala de satisfacción es la siguiente: (1 = Clara satisfacción; 2 = Más satisfecho que insatisfecho; 3 = No definida; 4 = Más insatisfecho que satisfecho; 5 = Clara insatisfacción; 6 = Contradictoria). De los 10 especialistas encuestados, 7 respondieron clara satisfacción, 2 más satisfechos que insatisfechos y 1 indefinido. A continuación se calcula el índice de satisfacción grupal (ISG), donde A, B, C, D, E, representan el número de sujetos con índice individual 1; 2; 3 ó 6; 4; 5, respectivamente y donde N representa el número total de sujetos del grupo.

$$ISG = \frac{A(1)+B(0.5)+C(0)+D(-0.5)+E(-1)}{N}$$

$$ISG = \frac{7(1)+2(0.5)+1(0)+0(-0.5)+0(1)}{10}$$

$$ISG = \frac{7+1+0+0+0}{10} = 0.8$$

Posteriormente se trabaja con los diferentes niveles de satisfacción que se expresan en la escala numérica que oscila entre -1 y + 1, como se observa a continuación:

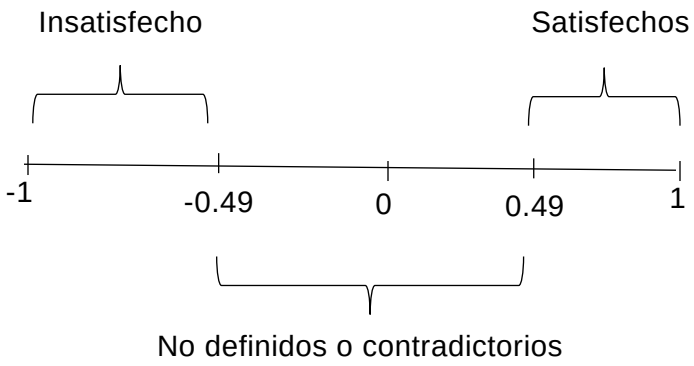


Ilustración 7: Escala numérica de Iadov

Como se puede observar el valor obtenido 0.8 se encuentra entre 0.5 y 1, lo que indica que los especialistas de migración de servicios telemáticos presentan satisfacción con la solución desarrollada, con lo que se demuestra el cumplimiento del objetivo de la investigación.

3.6 Conclusiones Parciales

Con el desarrollo del presente capítulo se obtuvo la descripción del proceso de implementación de la aplicación, a través de la definición de las convenciones utilizadas para la codificación, lo cual posibilitó una mayor legibilidad del código, haciéndolo más comprensible y estandarizado. Las pruebas de caja blanca,

caja negra y alfa, permitieron comprobar el correcto funcionamiento del código de la aplicación y verificar el funcionamiento de los requisitos para determinar la aceptación del cliente. Además se validó el cumplimiento del objetivo de la investigación mediante la técnica de Iadov, con lo que se comprobó la satisfacción del cliente con la solución.

Conclusiones generales

Con la realización de la investigación y el cumplimiento de los objetivos trazados se arriba a las siguientes conclusiones:

- El análisis de los elementos teóricos asociados al negocio y el estudio del estado del arte acerca de las herramientas de apoyo a la migración a software libre facilitó la definición de la propuesta de solución acorde a las necesidades existentes en la etapa de Diagnóstico.

- El análisis y diseño de la solución, guiado por la metodología de desarrollo de software AUP en su variante UCI, permitió una correcta estructura en todo el proceso de desarrollo del software, facilitando la selección de las herramientas necesarias y materializando así una solución acorde a los requerimientos del cliente.

- Se realizó la implementación de la aplicación Android para gestionar la información en el diagnóstico para la migración a software libre de los servicios telemáticos, obteniendo una herramienta capaz de generar un informe con la información recolectada, facilitándole a los especialistas de migración de servicios telemáticos el registro y consulta de la información que se requiera con rapidez.

- La ejecución de pruebas a la aplicación Android para gestionar la información en el diagnóstico para la migración a software libre de los servicios telemáticos, permitió verificar el correcto funcionamiento de los requisitos identificados y la aceptación por parte del cliente.

Recomendaciones

Referencias bibliográficas

CALABRESE , JULIETA y MUÑOZ , ROCÍO. 2018. *ASISTENTE PARA LA EVALUACIÓN DE CALIDAD DE PRODUCTO DE SOFTWARE SEGÚN LA FAMILIA DE NORMAS ISO/IEC 25000 UTILIZANDO EL ENFOQUE GQM.* 2018.

División de Sistemas. 2017. *METODOLOGÍA DE DESARROLLO DE SOFTWARE .* PERÚ : s.n., 2017.

Garay, Polanco y Wuilber , Leodanny . 2016. *GUÍA PARA EL DESARROLLO DE ALMACENES DE DATOS.* 2016.

HUARACA BUÑAY , RICHARD IVAN. 2018 . *ANÁLISIS DE LA IMPORTANCIA DE LAS HERRAMIENTAS CASE EN EL DESARROLLO DE SOFTWARE.* ECUADOR : s.n., 2018 .

Ricardo , Catherine M. 2009. *Bases de datos.* México : McGRAW-HILL INTERAMERICANA EDITORES, S.A. de C.V. , 2009. ISBN 13: 978-970-10-7275-2.

Vélez de Guevara , Luis. 2019. *Gestión de Bases de Datos.* 2019.

Aguila, Liset Beatríz Armas. 2017. *Aplicación para dispositivos móviles de apoyo a la migración a software libre.* La Habana : s.n., 2017.

Aguilera, Manuel Reina. 2015. *Sistema para la identificación de violaciones de Seguridad Informática en el uso de mensajería instantánea, correo electrónico y navegación por Internet en la UCI.* La Habana : s.n., 2015.

Alegsa, Leandro. 2018. Alegsa. *DICCIONARIO DE INFORMÁTICA Y TECNOLOGÍA.* [En línea] 2018. http://www.alegsa.com.ar/Dic/sdk.php.

Almeira, Adriana Sandra y Pérez, Vanina. 2016. *Arquitectura de software. Estilos y Patrones.* s.l. : Argentina, 2016.

Android Studio. *Android Studio. .* [En línea] https://developer.android.com/studio/intro?hl=es-419.

Baltarejo Sousa , Paulo . 2016. *SQLite Database and ListView Adapter.* Portugal : s.n., 2016.

Basterra, y otros. 2015. *Android OS Documentation.* 2015.

Desarrollo de una aplicación móvil para reconocimiento de personas después de una catástrofe natural utilizando la tecnología NFC. **Cáceres Álvarez, Luis y Ossandón Carpio1, Angela. 2018.** Número Especial, Chile : Revista chilena de ingeniería, 2018, Vol. 26 .

Developers, Android. 2019. Android Studio. *Android Developers.* [En línea] 2019. https://developer.android.com/studio/intro/index.html?hl=es-419..

Enríquez Ruiz, José Luis, y otros. 2017. *METODOLOGÍA DE DESARROLLO DE SOFTWARE.* CHIMBOTE–PERÚ : s.n., 2017.

Feal Delgado, William, Alvarez Acosta, Hugandy y Canosa Reyes, Rewer Miguel. 2014. *Estrategia de migración al software libre en la Universidad de Cienfuegos.* Cienfuegos : s.n., 2014.

Gamma, E., y otros. 1994. *Design Patterns : Elements of Reusable Objectoriented Software.* s.l. : Addison-Wesley, 1994. ISBN 0-201-63361-2.

Gradle Inc. 2016. Getting Started with Gradle for Android Build l Gradle. *Getting Started with Gradle for Android Build l Gradle.* [En línea] 2016. http://gradle.org/getting-started-android-build/.

Guevara, Diosbel Pérez, y otros. 2014. *Plataforma Cubana de Migración a Código Abierto .* La Habana : s.n., 2014.

James Rumbaugh, Booch, Grady y jacobson, ivar. 2007. *ELLENGUAJE UNIFICADO DE MODELADO MANUALDE REFERENCIA.* Madrid : s.n., 2007.

JUnit. 2016. JUnit 5. *JUnit 5.* [En línea] 2016. [Citado el: 11 de 11 de 2019.] https://junit.org/junit5/.

Larman, C. 2004. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development.* s.l. : ISBN 0-13-148906-2, 2004.

Medrano, Carlos Molina. 2014. *El Software Libre y sus perspectivas.* El Salvador : s.n., 2014.

Metodologías actuales de desarrollo de software. **RIVAS, Carlos Ignacio, y otros. 2015 .** No.5 980-986 , México : s.n., 2015 , Vol. Vol.2 .

Millán, Martha Elena. 2017. *Fundamentos de bases de datos .* Cali, Colombia : Programa Editorial Universidad del Valle, 2017. ISBN PDF: 978-958-765-487-5.

Montano, José Luis Montes de Oca. 2015. *LA MIGRACIÓN HACIA SOFTWARE LIBRE EN CUBA: COMPLEJO CONJUNTO DE FACTORES SOCIALES Y TECNOLÓGICOS EN EL CAMINO DE LA SOBERANÍA NACIONAL.* Cienfuegos : s.n., 2015.

Pérez Guevara , Diosbel, y otros. 2014. *Plataforma Cubana de Migración a Código Abierto .* La Habana : s.n., 2014.

Pressman, Roger S. 2010. *Software Engineering.* s.l. : McGraw-Hill, 2010.

Pressman, Roger S. 2002. *Ingeniería de Software, un enfoque práctico. Quinta edición. Adaptación.* España : Concepción Femández Madrid , 2002.

Raffino, María Estela. 2019. Concepto.de. *Concepto.de.* [En línea] 2019. [Citado el: 12 de 10 de 2019.] https://concepto.de/servidor/.

Rodríguez Montoya , Roberto , y otros. 2009. *Migración de Servicios de Servidores a Software Libre.* Venezuela : “Energy and Technology for the Americas: Education, Innovation, Technology and Practice” , 2009.

Rodríguez, Tamara. 2015. *Metodologia de desarrollo para la actividad productiva de la UCI.* La Habana : s.n., 2015.

Sánchez, Tamara Rodríguez. 2015. *Metodología de desarrollo para la Actividad productiva de la UCI.* La Habana : s.n., 2015.

Silberschatz , Abraham , Korth, Henry F. y Sudarshan, S. 2002. *FUNDAMENTOS DE BASES DE DATOS.* ESPAÑA : McGRAW-HILL/INTERAMERICANADE ESPAÑA, S. A. U. , 2002. ISBN: 0-07-228363-7.

Sistema para el an´alisis de acciones t´acticas significativas de los equipos de balonmano. **Milián Núñez, Vladimir , Olavarrieta Martínez, Dario Marcel y Martínez Casanova, Ernesto. 2018.** Especial UCIENCIA, Habana(Cuba) : Grupo Editorial “Ediciones Futuro” , 2018, Vol. Vol. 12. ISSN: 2227-1899.

Soluciones T.I. 2019. Soluciones T.I. *Soluciones T.I.* [En línea] 2019. [Citado el: 6 de 11 de 2019.] http://www.solucionesit.com.ve/servicios/servicios-consultoria/proyectos-de-migracion-a-sl.html.

Sommerville, Ian. 2011. *Ingeniería de Software 7ma edición.* Madrid : s.n., 2011.

Sqlite. 2019. SQLite. *SQLite.* [En línea] 2019. [Citado el: 8 de 11 de 2019.] https://sqlite.org/.

Studio, Android. Android Studio. [En línea] https://developer.android.com/studio/intro?hl=es-419.

Tanenbaum, Andrew S. y Wetherall, David J. 2012. *Redes de computadoras.* México : Pearson Educación de México, S.A. de C.V. , 2012. ISBN: 978-607-32-0817-8 .

Vázquez, Juan Ignacio Cerca, y otros. 2014. *Uso de herramientas CASE para la gestión de proyectos de software.* México : s.n., 2014.

Villazon, Yoandy Perez, Goñi Oramas, Ángel y García Vitier, Abel. 2018. *Buenas prácticas para la migración a código abierto.* La Habana : Ediciones Futuro, 2018.

Villazón, Yoandy Pérez, Paumier Samón, Ramón y Meneses Abad, Abel. 2009. *Guía Cubana de Migración a Software Libre.* La Habana : s.n., 2009.

Anexo

Anexo 1: Entrevista realizada a los jefes de equipo de migración en el Centro de Software Libre

1. ¿Cómo realizas el levantamiento de la información de los servidores, los servicios y la tipología de red en una institución?

2. ¿Qué información necesitas obtener de los servidores, los servicios y la tipología de red en una institución?

3. ¿Qué deficiencias tiene el proceso de obtención de la información de los servidores, los servicios y la tipología de red en una institución?

4. Teniendo en cuenta la cantidad de información que se necesita recolectar, ¿cree que el uso de una aplicación para dispositivos móviles agilizaría el proceso de levantamiento de la información de los servidores, los servicios y la tipología de red en una institución? ¿Por qué?

Anexo 2: Historias de usuario

Tabla 5: Historia de usuario: Generar informe.

Número: HU 17	Nombre del requisito: Generar informe
Programador: Alejandro Puerta	Iteración Asignada: 1
Prioridad: Alta	Tiempo Estimado: 5 días
Riesgo en Desarrollo: Alto	Tiempo Real: 3 días
Descripción: Permite generar un informe en pdf o xls, con la información de la empresa.	
Observaciones: el informe será guardado en el almacenamiento del dispositivo móvil	

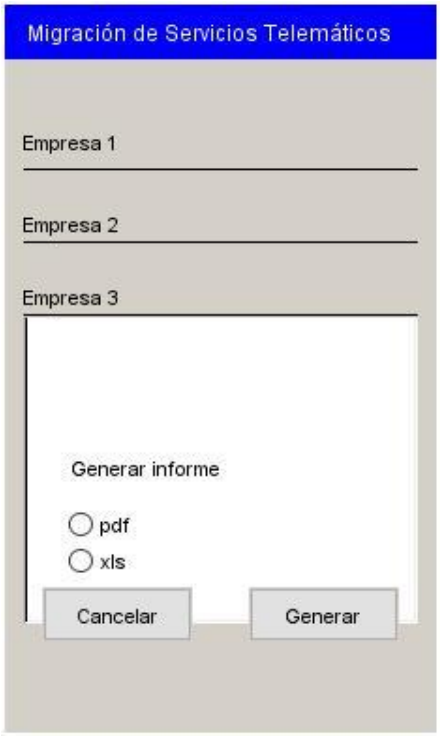
Prototipo de interfaz: 

Tabla 6. Historia de usuario: Modelar Topología de Red.

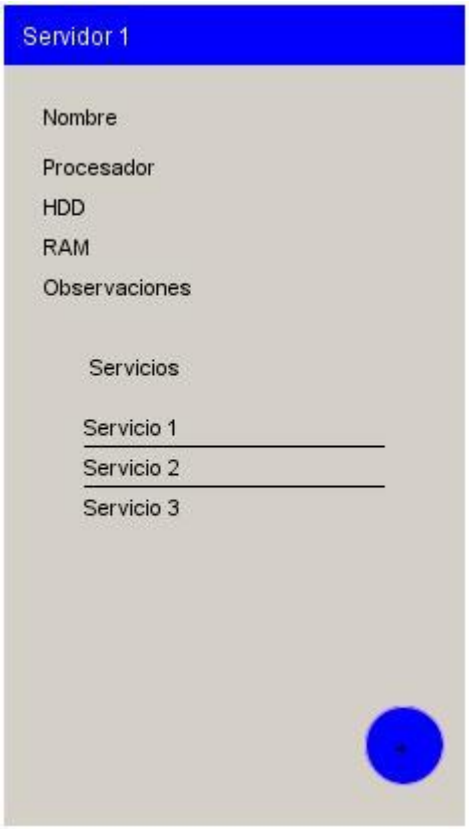
Número: HU 16	Nombre del requisito: Modelar Topología de Red
Programador: Alejandro Puerta	Iteración Asignada: 1
Prioridad: Alta	Tiempo Estimado: 15 días
Riesgo en Desarrollo: Alto	Tiempo Real: 10 días
Descripción: Permite modelar gráficamente la topología de red de la empresa.	

Observaciones:
Prototipo de interfaz: 

Tabla 7. Historia de usuario: Listar Servicios Telemáticos.

Número: HU-9	Nombre del requisito: Listar Servicios Telemáticos
Programador: Alejandro Puerta	Iteración Asignada: 1
Prioridad: Media	Tiempo Estimado: 7 días
Riesgo en Desarrollo: Alto	Tiempo Real: 4 días

Descripción: Debe mostrar una lista de servicios añadidos en un servidor de la empresa.
Observaciones: Cada ítem de la lista tiene el nombre del servicio añadido en el servidor, como se muestra en el prototipo de interfaz (1), en el caso de que la lista de servicios esté vacía la apariencia de la pantalla será la del prototipo de interfaz (2).
Prototipo de interfaz:

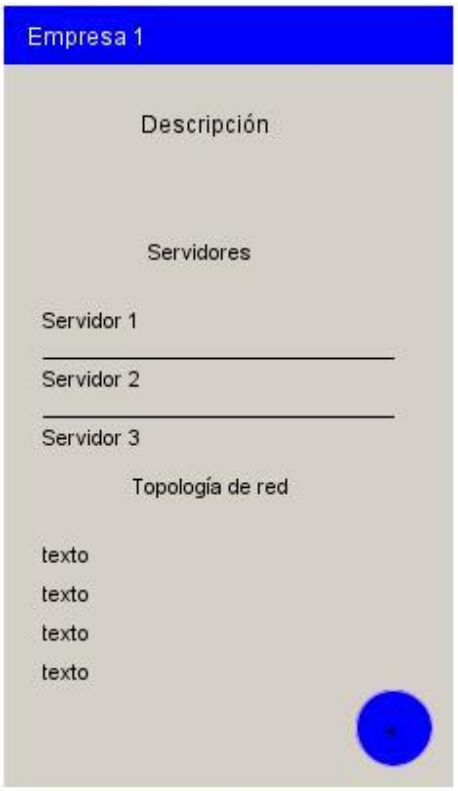


Prototipo 1

Prototipo 2

Tabla 8. Historia de usuario: Listar Servidores.

Número: HU-5	Nombre del requisito: Listar Servidores
Programador: Alejandro Puerta	Iteración Asignada: 1
Prioridad: Media	Tiempo Estimado: 7 días
Riesgo en Desarrollo: Alto	Tiempo Real: 4 días
Descripción: Debe mostrar un listado de servidores pertenecientes a la empresa añadida.	
Observaciones: Cada ítem de la lista tiene el nombre del servidor añadido en la empresa, como se muestra en el prototipo de interfaz (1), en el caso de que la lista de servidores esté vacía la apariencia de la pantalla será la del prototipo de interfaz (2).	
Prototipo de interfaz:	



Prototipo 1

Prototipo 2

Anexo 3: Casos de pruebas de validación

Tabla 9: Caso de prueba de validación Modificar empresa

Caso de Prueba de Validación	
Código: CP1-HU1	**HU 11:** Modificar empresa
Responsable: Alejandro Puerta Díaz	
Descripción: El caso de prueba inicia al mantener presionado el nombre de la empresa en la lista, la aplicación muestra una un cuadro de diálogo con las opciones “Modificar” y “Eliminar”, Al escoger	

modificar, presenta un formulario para cambiar la información de la empresa. El caso de prueba culmina al pulsar guardar.			
Condiciones de ejecución: Debe existir al menos una empresa a la cual realizarle los cambios.			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Modificar los datos de una empresa	El usuario mantiene presionado el nombre de la empresa, la modifica y se realiza el cambio correctamente.	La aplicación muestra al usuario el cuadro de diálogo con las opciones “Modificar” y “Eliminar”, al seleccionar modificar, muestra el formulario de la empresa, este pide el nombre y la descripción de la empresa. Una vez escrita la información, se guarda y cambia estos datos en la base de datos y en la lista.	Mantener pulsado el nombre de la empresa, seleccionar “Modificar” en el cuadro de Diálogo”, escribir le nombre y la descripción de la empresa y pulsar guardar.

Caso de Prueba de Validación	
Código: CP6-HU6	**HU 6:** Adicionar servidores
Responsable: Alejandro Puerta Díaz	
Descripción: El caso de prueba inicia al pulsar el botón de adición en la empresa, el cual despliega un menú con varios botones flotantes y se escoge el botón “Adicionar Servidor”. Una vez pulsado este botón la aplicación muestra un formulario para llenar la información del servidor. El caso de prueba culmina al pulsar guardar y la aplicación muestra en la interface de la empresa el servidor en una lista.	
Condiciones de ejecución: Debe haber una empresa añadida.	

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Añadir un servidor y se muestra en la lista.	El usuario añade un servidor a la empresa y este se muestra en la lista de servidores de la empresa.	Al pulsar el botón de adición en la empresa muestra un menú de varios botones flotantes, se presiona Adicionar Servidor y la aplicación muestra el formulario del servidor. Al llenar los datos se presiona guardar y el servidor se muestra en la lista de servidores de la empresa.	Presionar el botón adicionar, seleccionar el botón adicionar Servidor, llenar el formulario y presionar guardar.

Tabla 10: Caso de prueba de validación Añadir servidor

Anexo 4: Interfaces del sistema

Ilustración 8: Interfaz de Empresa.

Ilustración 9:Interfaz de Formulario de servidor.

Ilustración 10: Interfaz de Menú.

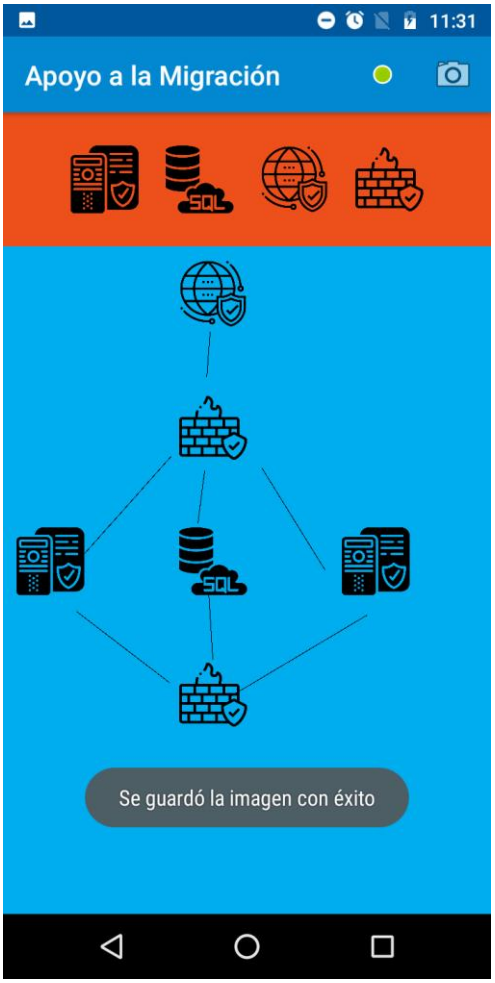


Ilustración 11: Interfaz de Modelo de topología.

Anexo 5: Encuesta para determinar nivel de satisfacción con la solución informática

Estimado especialista de migración a software libre de servicios telemáticos, la siguiente encuesta tiene el propósito de determinar su nivel de satisfacción en cuanto a la solución informática para la gestión de la información necesaria en el proceso.

1) ¿Considera usted que se deba seguir realizando el diagnóstico para la migración a software libre de los servicios telemáticos sin el uso de una herramienta que mejore la calidad del proceso?

_____ Sí _____ No _____ No sé

2) ¿Utilizaría la solución para realizar el diagnóstico para la migración a software libre de los servicios telemáticos y así mejorar la gestión de la información?

_____ Sí _____ No _____ No sé

3) ¿Observa diferencia entre el tiempo requerido para realizar el diagnóstico como se realiza hasta el momento y el tiempo que demora al realizarlo mediante la herramienta?

4) ¿Observa mayor control de la información recolectada debido al uso de la herramienta?

5) ¿Considera usted que la herramienta carece de alguna funcionalidad necesaria para realizar el diagnóstico?

6) ¿Qué opina acerca de la calidad del proceso al usar la herramienta?

7) ¿Qué opina acerca de la capacidad de almacenar la información en bases de datos?

8) ¿Le satisface la solución desarrollada para la gestión de información en el diagnóstico para la migración a software libre de los servicios telemáticos?

____ Me gusta mucho.

____ Me gusta más de lo que me disgusta.

____ Me da lo mismo.

____ Me disgusta más de lo que me gusta.

____ No me gusta nada

____ No sé qué decir