Universidad de las Ciencias Informáticas

Facultad 1

"Aplicación para dispositivos móviles de apoyo a la migración a software libre"

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

Autora: Liset Beatríz Armas Aguila

Tutores: Ing. Nelio Véliz Pedraza

Ing. Ivelisse Montero Jimenez

La Habana, junio 2017

"Año 58 de la Revolución"

"El futuro de nuestra patria tiene que ser necesariamente un futuro de hombres de ciencia, tiene que ser un futuro de hombres de pensamiento, porque precisamente es lo que más estamos sembrando; o que más estamos sembrando son oportunidades a la inteligencia (…)"

Fidel Castro Ruz

Declaración de autoría

Declaro por este medio que yo Liset Beatríz Armas Aguila, con carné de identidad 94090131459 soy el autor principal del trabajo titulado “Aplicación para dispositivos móviles de apoyo a la migración” y autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Para que así conste firman la presente a los ____ días del mes de ___junio_____ del año _____2017__.

_____

Liset Beatríz Armas Aguila

Autor

_____

Ing. Nelio Véliz Pedraza

Tutor

_____

Ing. Ivelisse Montero Jimenez

Tutor

Agradecimientos:

A Dios por darme las fuerzas y hoy cumplir mis sueños, a mi familia, mamá, papá y hermana por todo el apoyo, cariño y comprensión. A mi abuela Juanita por sus sabios consejos y todo el amor que me brindó, eres un gran ejemplo en mi vida.

A mis tíos Carlitos y Yoyi por ser unos padres para mí, depositar tanta confianza y guiar mis pasos desde niña.

A mi novio y mis suegros por formar parte de mi vida y apoyarme en todo momento.

A mis tutores Nelio e Ivelisse por haberme guiado en todo este curso y por su dedicación.

A mis compañeros de aula por compartir momentos juntos y que recordaré con gran alegría.

A todos los profesores que contribuyeron a mi formación como estudiante en estos años de carrera.

A los compañeros de DroidLab, en especial a David y Leonardo por su paciencia, por brindarme todo su apoyo y conocimientos.

A Manuel por brindarme su amistad incondicional.

A la Revolución por esta hermosa obra que es la UCI.

A todas las personas que contribuyeron de una forma u otra que esto fuera posible

GRACIAS!!!

Dedicatoria:

A mis padres Betty y Raulito, por todo su amor y cariño.

A mi hermana (tata) Lisandra por ser mi mayor fuente de inspiración y ejemplo a seguir.

A mi abuela Juanita y a mis tíos Carlitos y Yoyi.

Resumen

La migración a software libre está guiada por tres etapas, las cuales deben ser cumplidas para garantizar un proceso exitoso. Durante la etapa de Ejecución, en la migración de las estaciones de trabajo, es necesario que los jefes de equipo de migración gestionen la información de la institución. La presente investigación tiene como propósito desarrollar una aplicación para dispositivos móviles con sistema operativo Android, para agilizar el registro y la consulta de información generada 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. Como resultado se obtuvo una aplicación de apoyo a la migración a software libre que permite importar un excel con las áreas de la institución a migrar. Además, registra la información durante la migración de las estaciones de trabajo y permite realizar un reporte general sobre el avance de la migración. Se realizaron las pruebas de software, donde se obtuvo la aceptación por parte del cliente.

Palabras clave:

Android, migración, software libre

Índice

Índice de figuras

Índice de tablas

Introducción

Las Tecnologías de la Información y las Comunicaciones (TIC) son el resultado de la interrelación de muchos componentes informáticos, uno de ellos es el software que es controlado por monopolios empresariales que obtienen todos los años millones de dólares por concepto de pagos de licencias de uso. 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 (Delgado, 2014).

Cuba promueve, como parte de su programa de informatización, de 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). Uno de los aspectos que se tuvo en cuenta en los lineamientos aprobados en el VI y VII Congreso del Partido Comunista de Cuba (PCC) resultó ser la soberanía tecnológica en ramas tan importantes como la informática, la electrónica y las comunicaciones (Lineamientos, 2011).

Una de las formas de alcanzar la soberanía tecnológica es mediante la migración a software libre, la cual es una vía factible para Cuba por ser un país bloqueado y del tercer mundo, lo que constituye una alternativa para superarse económicamente. El Centro de Software Libre (CESOL) perteneciente a la Universidad de las Ciencias Informáticas (UCI) ha conducido procesos de migración a software libre, principalmente en los Organismos de la Administración Central del Estado (OACE), aplicando buenas prácticas que garanticen un desarrollo exitoso de este proceso atendiendo a los posibles escenarios tecnológicos. Además, dicho centro está integrado por los departamentos Sistemas Operativos, Servicios Integrales en Migración Asesoría y Soporte (SYMAYS) y el laboratorio DroidLab donde se desarrollan aplicaciones para dispositivos móviles con sistema operativo Android.

El libro “Buenas prácticas para la migración a Código Abierto”, que es una continuación de la “Guía Cubana de Migración a Software Libre”, constituye un referente para llevar a cabo el proceso de migración. En el mismo se detalla la etapa de Ejecución que comprende las actividades necesarias para la migración de las estaciones de trabajo y tecnologías de una institución a un sistema operativo GNU/Linux (Villazón, 2014).

Actualmente durante la realización de esta etapa en una institución, el jefe de equipo de migración realiza el control de la migración mediante el levantamiento y la consulta de información de las estaciones de trabajo y se detectan las siguientes deficiencias:

- La información que recibe el jefe de equipo de migración sobre la institución es en formato duro y al actualizarla provoca la duplicación de documentos.
- El jefe de equipo de migración levanta información manualmente sobre el estado de cada estación de trabajo por lo que invierte mucho tiempo en tareas de gestión y registro de información.
- El jefe de equipo de migración incurre en errores humanos al digitalizar la información que se genera durante la migración de las estaciones de trabajo.
- El reporte general sobre el estado de la migración, de las estaciones de trabajo, no es posible actualizarlo automáticamente para la consulta de los jefes de equipo de migración.

Por todo lo expuesto anteriormente se plantea como **problema de investigación**: ¿Cómo agilizar el levantamiento y la consulta de información durante el proceso de migración de las estaciones de trabajo en la etapa de Ejecución?

El **objeto de estudio** lo constituye: las herramientas de apoyo al proceso de migración, el **campo de acción** se define en: las herramientas de apoyo al proceso de migración de las estaciones de trabajo en la etapa de Ejecución para dispositivos móviles con sistema operativo Android.

Para dar cumplimiento al problema planteado se establece como **objetivo general**: Desarrollar una aplicación para dispositivos móviles con sistema operativo Android que permita agilizar el levantamiento y la consulta de información durante el proceso de migración de las estaciones de trabajo en la etapa de Ejecución.

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

- Elaborar el marco teórico de la investigación mediante el estudio de los principales referentes teóricos del proceso de migración a software libre en la etapa de Ejecución.
- Diseñar una aplicación para agilizar el levantamiento y la consulta de información durante el proceso de migración de las estaciones de trabajo en la etapa de Ejecución.

- Implementar la aplicación para agilizar el levantamiento y la consulta de información durante el proceso de migración de las estaciones de trabajo en la etapa de Ejecución.
- Probar la aplicación mediante el desarrollo de pruebas funcionales.

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

- ¿Cómo se realiza el proceso de migración a software libre de las estaciones de trabajo en la etapa de Ejecución?
- ¿Cómo gestiona la información de la migración de las estaciones de trabajo el jefe de equipo de migración durante la etapa de Ejecución?
- ¿Cuáles son las causas que llevan a incurrir en errores durante el levantamiento y la consulta de información de la migración de las estaciones de trabajo?

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 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 conforman la aplicación.

Los **métodos empíricos** utilizados son la **Entrevista** permitió recopilar información a través de los jefes de equipo de migración (ver anexo 5) para definir 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.

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. En este capítulo se evidencia el estudio de las principales definiciones a tener en cuenta en la investigación, tales como el software libre, la migración a software libre y las etapas por las que atraviesa este proceso, específicamente la etapa de Ejecución. Además se

fundamenta el uso de los distintos temas referentes a la metodología, herramientas y tecnologías a utilizar para el desarrollo de la aplicación.

Capítulo 2. Análisis y Diseño. En este capítulo se describe la propuesta de solución. Se especifican los requisitos funcionales y no funcionales, así como las historias de usuario y las descripciones de las mismas. Se modelan los diagramas que representan las funcionalidades de la aplicación a partir de los patrones de diseño identificados.

Capítulo 3. Implementación y Prueba. 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 de la aplicación en los diferentes escenarios.

Capítulo 1: Fundamentación Teórica sobre el proceso de migración a software libre

Introducción

En el presente capítulo se precisan los elementos teóricos que sustentan la investigación y desarrollo de la aplicación para dispositivos móviles de apoyo a la migración a software libre. Entre los principales temas que se abordan están el software libre, la migración a software libre y la etapa de Ejecución en la migración de las estaciones de trabajo. Se describe la metodología de desarrollo a utilizar en la propuesta de solución. También se realiza un estudio de las tecnologías y herramientas que son necesarias para el desarrollo de la aplicación.

1.1 Definiciones

1.1.1 Software Libre

Existen dos tipos predominantes de software: libre y privativo. Richard Stallman (2004) plantea que el software libre es la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, modificar y mejorar el software. Especialmente a cuatro clases de libertad para los usuarios de software:

- Libertad 0: para ejecutar el programa sea cual sea nuestro propósito.
- Libertad 1: para estudiar el funcionamiento del programa y adaptarlo a tus necesidades — el acceso al código fuente es condición indispensable para esto.
- Libertad 2: para redistribuir copias y ayudar así a tu vecino.
- Libertad 3: para mejorar el programa y luego publicarlo para el bien de toda la comunidad —el acceso al código fuente es condición indispensable para esto.

1.1.2 Migración a Software Libre

La migración a software libre 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, 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, 2016).

La migración a software libre posee dos componentes fundamentales: migración social y migración técnica (Villazón, 2015).

- Migración social: el componente social de la migración está asociado al grado de aceptación y compromiso que logren alcanzar los usuarios con el proceso de cambio. Si los usuarios no están convencidos de la importancia del proceso y apoyan el mismo, será muy probable que este falle. Para lograr que los usuarios acepten el cambio deberá formularse una acertada estrategia de sensibilización.
- Migración técnica: componente relacionado con el cambio de la tecnología en la institución, dicha migración requiere como paso imprescindible para su cumplimiento el cambio de las aplicaciones privativas de la entidad por herramientas libres en su mayor porciento posible.

1.2 Migración a software libre en Cuba

En abril de 2004 el Consejo de Ministros adoptó el acuerdo 084/2004 donde indica al Ministerio de la Informática y las Comunicaciones (MIC) ordenar el proceso paulatino de migración de Cuba a software libre (Montano, 2015).

Para lograr el éxito en la ejecución de la migración se hace necesario pasar a través de varias etapas, descritas en la metodología cubana de migración a plataformas de código abierto, donde se plantean sus principales procesos. Dicha metodología se encuentra contenida en el libro “Buenas Prácticas para la migración a Código Abierto”, el cual se basa en las experiencias adquiridas y resultados satisfactorios durante la migración de varias organizaciones, tanto en Cuba como en el extranjero (Villazón,2014).

Las etapas del proceso de migración a software libre son las siguientes:

- Preparación: donde se ejecutarán todas las tareas de diagnóstico de los procesos, personas y tecnología de la entidad, se realizarán tareas de análisis de factibilidad de la migración y se emitirá 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 los escritorios de los usuarios.
- Consolidación: etapa que comprende las tareas destinadas a garantizar el soporte técnico a los usuarios e infraestructura.

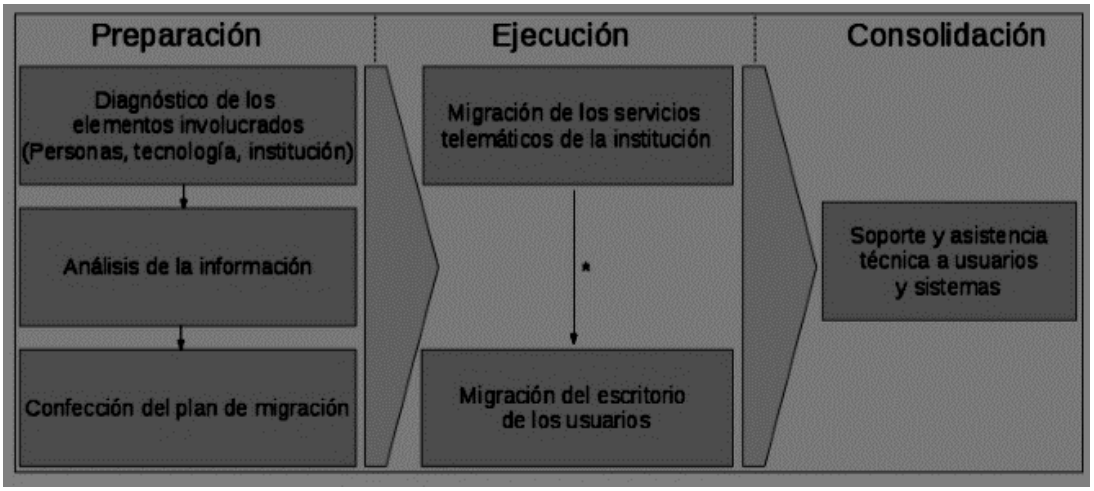


Figura 1. Actividades generales del proceso de migración (Villazón, 2015)

1.2.1 Etapa de Ejecución

La etapa de Ejecución se realiza una vez definido el plan de migración, la ejecución del proceso debe estar guiada totalmente por el mismo. Las actividades de ejecución de la migración deberán estar agrupadas según el elemento tecnológico a migrar:

- Los servidores.
- Las estaciones de trabajo.

Para llevar a cabo la migración de las estaciones de trabajo se realizan varias actividades durante el proceso:

Informar a los usuarios

Primeramente se informa a los usuarios el plan de migración que debe ser de conocimiento público, así como deben circularse por las diferentes vías y espacios (correo electrónico, reuniones, charlas, entre otras) las actividades que se irán realizando en cada momento (Villazón, 2014).

Instalar aplicaciones libres sobre el entorno privativo

Es fundamental la instalación de aplicaciones libres sobre el entorno privativo, ello permitirá a los usuarios conocer en el sistema operativo que ya dominan las herramientas que en el futuro próximo usarán sobre el sistema operativo libre. Dicho proceso de instalación hasta el momento no supone la desinstalación de la herramienta privativa, ambas deben convivir en el sistema. A medida que los usuarios se capaciten deberán ir usando las aplicaciones libres (Villazón, 2014).

Capacitar a los usuarios

El entrenamiento de los usuarios es de vital importancia en el manejo de las aplicaciones libres sobre Microsoft Windows. Al culminar esta actividad debe existir plena seguridad de que los usuarios dominan las herramientas en las que fueron capacitados y que fueron solucionadas las deficiencias detectadas en el proceso de diagnóstico (Villazón, 2014).

Migrar los archivos existentes

Las ventajas de hacer uso de los formatos de archivos en el sector público y la amplia compatibilidad de las aplicaciones libres con los mismos, conlleva la realización de una migración de todos los archivos de documentos de la institución hacia el estándar ODF[1]. En dicho cambio de formato pueden estar involucrados los formatos de archivos de documentos, hojas de cálculos, presentaciones digitales y bases de datos personales. Se debe prestar especial atención a los archivos de este tipo que incorporan macros. En las bases de datos solo es posible migrar de manera automatizada las tablas y sus datos (Villazón, 2014).

Desinstalar aplicaciones privativas

Los usuarios deberán trabajar sobre el entorno privativo con herramientas libres por un período de tiempo. No existe una cantidad de días definidos para ello, es decisión de los especialistas que estén ejecutando la actividad decidir qué plazo de tiempo se usará software libre sobre Microsoft Windows o en el sistema operativo privativo usado (Villazón, 2014).

Migrar el sistema operativo base

La migración del sistema operativo es la actividad que marcará la sustitución de Microsoft Windows y permitirá instalar un sistema operativo GNU/Linux u otro libre. El proceso de cambiar el sistema requiere:

1. Realizar un diagnóstico para detectar los archivos importantes existentes en cada ordenador de los usuarios que deban ser resguardados.
2. Realizar una salva de respaldo de dichos archivos en otra computadora o en un servidor centralizado.
3. Formatear e instalar el nuevo sistema.

[1] ODF: Open Document Format, ISO/IEC 26300:2006

4. Configurar el nuevo sistema, manteniendo la configuración de la red (número de IP, recursos compartidos).
5. Restaurar los archivos guardados en el paso 2.
6. Realizar las configuraciones y ajustes necesarios para el correcto funcionamiento del sistema, por ejemplo: montar particiones de forma permanente, configurar el cliente de navegación web, cliente de correo y mensajería instantánea, entre otras (Villazón, 2014).

La migración debe contar con personas encargadas de guiar cada paso y actividad que se desarrolle, según sus habilidades y conocimientos hasta lograr el éxito del proceso (Villazón, 2014). Los roles involucrados en la migración de las estaciones de trabajo en la etapa de Ejecución son:

- Los usuarios los cuales deben conocer las actividades que se realicen en cada momento y su capacitación es de vital importancia para el manejo del nuevo sistema operativo libre.
- El jefe del equipo de migración que es el responsable de dirigir el proyecto de migración y debe velar por su correcta planificación, ejecución y cierre.
- El especialista en migración cuya responsabilidad principal es el diseño, desarrollo y despliegue de la solución a utilizar en la migración de las estaciones de trabajo de la institución. Además, una de sus actividades fundamentales es la atención individualizada a los usuarios durante la migración de las estaciones de trabajo.

1.2.2 Avanzada para la migración a software libre en Cuba

El uso del software libre es de gran importancia para Cuba, por lo que se realiza una estrategia para alcanzar la independencia en el terreno del software garantizando la seguridad informática y afianzando el uso de los principios del software libre (Montano, 2015).

La avanzada para la migración a software libre del país está centrada en organismos tales como:

- Aduana General de la República.
- Universidad de las Ciencias Informáticas.
- Ministerio de la Informática y las Comunicaciones.
- Ministerio de Educación Superior.
- Empresa de Telecomunicaciones de Cuba.

- Ministerio de Cultura.
- Oficina Nacional de Estadísticas e Información.
- Unión de Jóvenes Comunistas Nacional.
- Fiscalías General de la República.

1.3 Herramientas de apoyo a la migración a software libre

La migración de un sistema informático con tecnologías privativas hacia software libre resulta ser una de las tareas más complejas para las personas que dirigen dicho proceso. Actualmente existen instituciones que poseen un gran número de estaciones de trabajo distribuidas físicamente en distintos locales, por lo que el uso de herramientas para informatizar varias de las tareas aporta un considerable ahorro de tiempo en el proceso de migración.

La Plataforma Cubana de Migración a Código Abierto (PCMCA) 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 (Villazón & otros, 2012).

A pesar de los avances mencionados, la plataforma alcanza su papel fundamental en la etapa de preparación (Villazón, 2013). La vía de acceso a la plataforma es a través de un navegador web y actualmente debido a la dependencia existente de una conexión directa y un navegador a los jefes equipo de migración se les dificulta transitar de un área a otra en una institución e ir registrando información al mismo tiempo haciendo uso de la plataforma.

Teniendo en cuenta que resulta de gran importancia que el jefe de equipo de migración transite por varias áreas de una institución, se propone realizar una aplicación para dispositivos móviles con sistema operativo Android. Actualmente los dispositivos móviles han alcanzado gran demanda en el mercado pues brindan como su nombre lo indica movilidad, tal es el caso de muchos empleados que realizan actividades fuera de su oficina y necesitan acceder a bases de datos, compartir archivos y guardar información de una manera rápida y sencilla, siendo vital para su productividad laboral. El auge de teléfonos móviles en el mundo a finales de 2016 ascendió al 97%, solo cuatro regiones presentan un índice de uso de la tecnología móvil

menor del 100% (PuroMarketing, 2016). Se puede afirmar que los últimos avances en los dispositivos móviles hacen que se conviertan en herramientas imprescindibles en la vida moderna (Shin, 2014).

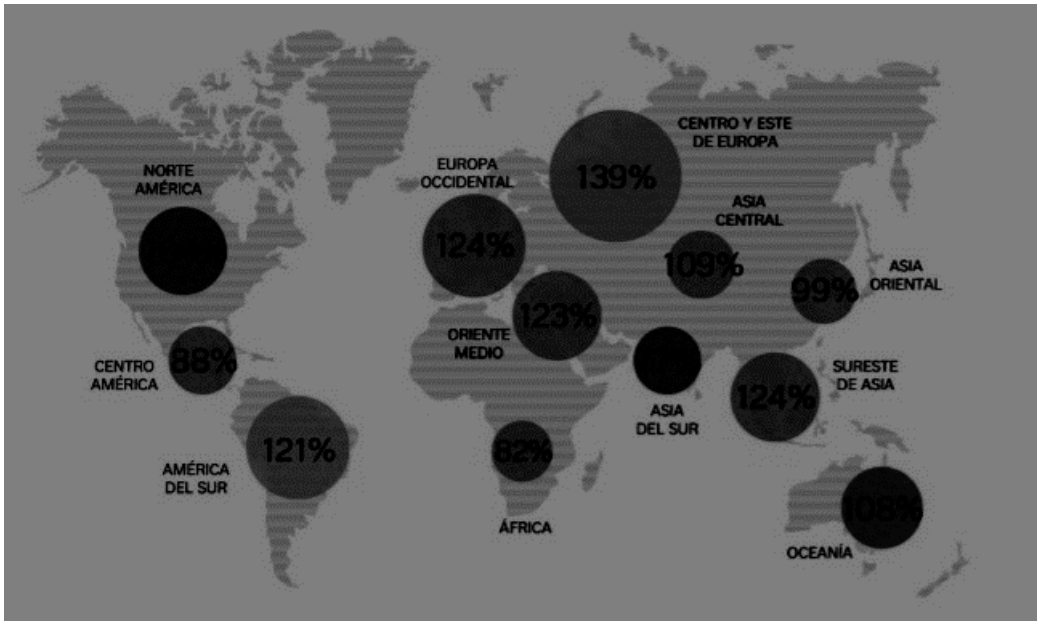


Figura 2. Uso del dispositivo móvil en el mundo (PuroMarketing, 2016)

1.4 Android

Android es un sistema operativo basado en Linux, que está enfocado para ser utilizado en dispositivos móviles como teléfonos inteligentes, tabletas, Google TV y otros dispositivos. Es desarrollado por la *Open Handset Alliance* que es un conglomerado de fabricantes y desarrolladores de hardware, software y operadores de servicio, la cual es liderada por Google. Fue desarrollado inicialmente por Android Inc., una firma comprada por Google en 2005. Google liberó la mayoría del código de Android bajo la licencia Apache, una licencia libre y de código abierto (Vilchez, 2009).

1.4.1 Arquitectura de Android

Android depende de Linux para los servicios base del sistema como: seguridad, gestión de memoria y de procesos, pila de red y modelo de controladores.

- Aplicaciones: las aplicaciones base incluyen un cliente de correo electrónico, programa de mensajería, calendario, mapas, navegador, contactos y otros.

- Marco de trabajo de aplicaciones: ofrece acceso completo a los mismos *APIs*[2] del *framework*[3] usados por las aplicaciones base. La arquitectura está diseñada para simplificar la reutilización de componentes.
- Librerías: incluye un conjunto de bibliotecas de C/C++ usadas por varios componentes del sistema. Estas características se exponen a los desarrolladores a través del marco de trabajo de aplicaciones; algunas son: C estándar, de medios, de gráficos, SQLite[4], entre otras.
- Android Runtime: incluye un conjunto de bibliotecas base que proporcionan la mayor parte de las funciones disponibles en las bibliotecas base del lenguaje Java. Cada aplicación Android corre su propio proceso, con su propia instancia de la máquina virtual Dalvik.
- Kernel de Linux: es el encargado de acoplar y hacer que todos los componentes de la terminal funcionen correctamente en el sistema operativo.

[2]APIs: Por sus siglas del inglés Aplication Programming Interface, traducido al español Interfaz de Programación de Aplicaciones.
[3]Framework: Infraestructura, marco de trabajo.
[4]SQLite: base de datos para almacenamiento estructurado que se integra directamente con las aplicaciones.

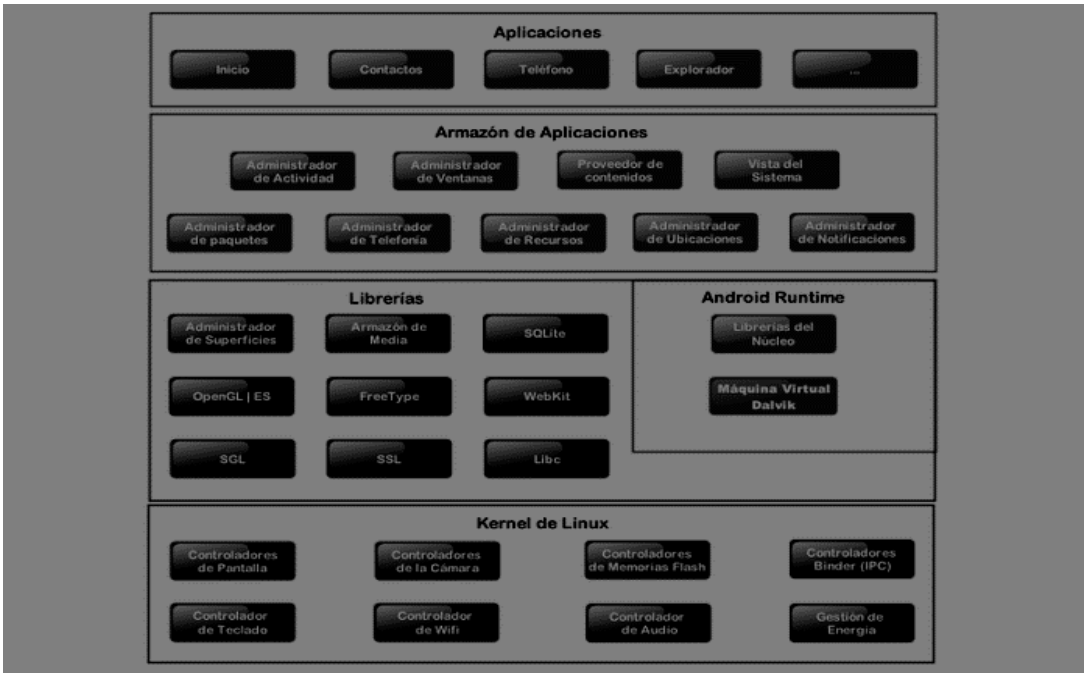


Figura 3. Arquitectura de Android (Vilchez, 2009)

1.5 Metodología, lenguaje, herramientas y entorno de desarrollo

A continuación se describe la metodología, el lenguaje de programación, herramientas y el entorno de desarrollo utilizados en la investigación.

1.5.1 Metodología de desarrollo de software

Existen diferentes modelos y metodologías que han sido en los últimos años herramientas de apoyo para el desarrollo del software. Sommerville (2011) define que un método de ingeniería de software es un enfoque estructurado para el desarrollo de software cuyo propósito es facilitar la producción de software de alta calidad de una forma costeable. Para usar este enfoque se debe manejar conceptos fundamentales como: procesos, métodos, tareas, procedimientos, técnicas, herramientas, productos, entre otros.

Metodología de desarrollo de software AUP

El Proceso Unificado Ágil de Scott Ambler o *Agile Unified Process* (AUP) es una versión simplificada del Proceso Unificado de Rational (RUP). El cual describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. El AUP aplica técnicas ágiles incluyendo:

- Desarrollo dirigido por pruebas.

- Modelado ágil.
- Gestión de cambios ágil.
- Refactorización de base de datos para mejorar la productividad.

Metodología de desarrollo de software AUP versión UCI

En la presente investigación se selecciona como metodología de desarrollo de software AUP, una versión simplificada de RUP adaptada a la UCI. Dicha metodología describe de una manera simple y fácil de entender, la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y manteniendo como principios básicos la simplicidad, agilidad, centrarse en actividades de alto valor y libertades en el momento de elegir las herramientas más adecuadas para el trabajo y adaptarla a las necesidades. De las 4 fases que propone AUP (Inicio, Elaboración, Construcción, Transición) se decide para el ciclo de vida de los proyectos de la UCI mantener la fase de Inicio, pero modificando el objetivo de la misma, se unifican las restantes 3 fases de AUP en una sola, llamada Ejecución y se agrega la fase de Cierre (Sánchez, 2015).

A continuación la tabla muestra las fases de la variante AUP para la UCI:

Tabla 1. AUP versión UCI (Sánchez, 2015)

Fases AUP	Fases Variación AUP-UCI	Objetivos de las fases (Variación AUP-UCI)
Inicio	Inicio	Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo, como decidir si se ejecuta o no el proyecto.
Elaboración	Ejecución	En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluye
Construcción		

Transición		el ajuste de los planes del proyecto considera los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, se obtienen los requisitos, se establecen la arquitectura y el diseño, se implementa y se libera el producto.
	Cierre	En esta fase se analizan los resultados del proyecto y se realizan las actividades formales de cierre del proyecto.

1.5.2 Lenguaje de modelado

El lenguaje de modelado permite dar apoyo a la mayoría de los procesos de desarrollo de software, estos son desarrollados con el esfuerzo de simplificar y consolidar el gran número de métodos de desarrollo que han surgido. Poseen como objetivo principal visualizar de manera gráfica el sistema que se desarrollará, como ayuda técnica a los ingenieros implicados (González, 2008).

Lenguaje de modelado UML v2.0

UML es un lenguaje de modelado de sistemas de software. Abarca la estructura de las aplicaciones, su comportamiento y arquitectura, procesos de negocio y datos. Incluye todo el proceso de desarrollo de software. Estandariza la forma de crear diagramas, el significado preciso de los mismos y las relaciones existentes entre ellos. Utilizado para especificar, construir, visualizar y documentar los artefactos de un sistema de software orientado a objeto (*Unified Modeling Language*, 2016).

1.5.3 Herramienta CASE para la modelación del sistema

La Ingeniería de Software Asistida por Computadoras (CASE, por sus siglas en inglés, *Computer Aided Software Engineering*) es un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores durante todo el ciclo de vida de desarrollo de un software. Este puede ser generalmente aplicado a cualquier sistema o colección de herramientas que ayudan a automatizar el proceso de diseño y desarrollo de software (Ambler, 2016).

Las herramientas CASE se pueden clasificar teniendo en cuenta los siguientes parámetros:

- Las plataformas que soportan.

- Las fases del ciclo de vida del desarrollo de sistemas que cubren.
- La arquitectura de las aplicaciones que producen.

Visual Paradigm v8.1

Visual Paradigm es una herramienta CASE profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Soporta, entre otros, el lenguaje de modelado UML. El software de modelado UML mejora la rapidez en la construcción de aplicaciones de calidad y a un menor costo. Está disponible para Windows y GNU/Linux. Permite dibujar diagramas de clases, código inverso, generar código desde diagramas y generar documentación, así como un diseño centrado en casos de uso y enfocado al negocio que genera un software de mayor calidad (Visual Paradigm, 2016).

1.5.4 Herramienta para el diseño de prototipo de interfaz

Las herramientas de prototipo de interfaz de usuario permiten el diseño de computadoras, aplicaciones, máquinas, dispositivos de comunicación móvil, aplicaciones de software y sitios web enfocado en la experiencia de usuario y la interacción. Normalmente es una actividad multidisciplinaria que involucra a varias ramas del diseño y el conocimiento como el diseño gráfico, industrial y de software. Está implicado en un amplio rango de proyectos, desde sistemas para computadoras, vehículos hasta aviones comerciales (Granollers, 2014).

Balsamiq Mockups v3.5

Balsamiq Mockups es una herramienta que reproduce la experiencia de realizar bocetos en una pizarra a través de un ordenador, por lo que es posible instalarlo en Windows, Mac y GNU/Linux. Permite crear de un modo dinámico, sencillo y práctico los bocetos de interfaces de usuario, no solo para páginas web, sino también para aplicaciones de escritorio o dispositivos móviles. Posee gran cantidad de componentes visuales para la construcción de interfaces de usuario, íconos y elementos reutilizables tales como plantillas. (Balsamiq, 2016).

1.5.5 Lenguaje de Programación

Un lenguaje de programación es un lenguaje artificial que puede utilizarse para definir una secuencia de instrucciones para su procesamiento por un ordenador o computadora. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión o como modo de comunicación humana (Alonso, 2014).

Java v1.8

Java es un lenguaje originalmente desarrollado por un grupo de ingenieros de Sun[5], utilizado por Netscape[6] posteriormente como base para JavaScript. Si bien su uso se destaca en la web, permite crear todo tipo de aplicaciones locales, intranet o internet (Oracle Corporation, 2016).

Algunas características notables son:

- Orientado a objetos.
- Independiente de la plataforma.
- Dinámico.
- Interpretado.
- Robusto.

1.5.6 Entorno de Desarrollo Integrado

El Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés, *Integrated Development Environment*) es un programa informático compuesto por herramientas de programación que pueden ser partes de aplicaciones existentes. Estos sistemas pueden manejar uno o varios lenguajes de programación ejemplo: Java, C# y C++. Han sido empaquetados como programas de aplicaciones, editores de código, depuradores, compiladores o constructores de interfaces gráficas (Suárez, 2015).

Android Studio v2.3

Android Studio es un IDE compatible para el desarrollo de aplicaciones en Android y con su uso recibe un nuevo sistema de construcción unificado, que está totalmente integrado para permitir la máxima flexibilidad en su proceso de desarrollo (Grant, 2014). Entre sus características principales se encuentra la renderización en tiempo real; la incorporación de Gradle como compilador, ofreciendo múltiples variantes para la construcción y generación de APKs[7]; la existencia de plantillas predeterminadas para crear diseños comunes de Android (Android Developers, 2016).

[5]Sun: Empresa informática comprada por Oracle Corporation, dedicada a fabricar semiconductores y software.

[6]Netscape: Navegador web, el primer producto comercial de la compañía *Netscape Communications*.

[7]APKs: Del inglés *Android Package*. Aplicación instalable en dispositivos móviles con sistema operativo Android.

El SDK (por sus siglas en inglés, *Software Development Kit*) 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 (Grant, 2014).

El SDK de Android incluye:

- Librerías necesarias.
- Entorno de depuración.
- Emulador de dispositivos móviles.
- Documentación relevante para las *APIs* de Android.
- Código fuente de ejemplo.
- Tutoriales para el sistema operativo Android.

Gradle v3.4

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++, iOS y Android. Provee plugins[8] y otras herramientas para la integración con IDEs como Eclipse, Android Studio e IntelliJ.

Genymotion

Es un emulador para Android que incluye un conjunto de sensores y características en aras de facilitar la interacción con un entorno virtual de este sistema operativo (Genymotion, 2016). Utilizado en el proceso de desarrollo, pruebas y demostración de aplicaciones Android. Disponible para GNU/Linux, Windows y Mac OS.

1.5.7 Gestor de Base de Datos

Un Sistema Gestor de Base de Datos (SGBD) (DBMS por sus siglas en inglés *DataBase Management System*) es una colección de programas cuyo principal objetivo es servir de interfaz entre las bases de datos, el usuario y las aplicaciones. Es un software de propósito general que facilita el proceso de definir, construir y manipular la base de datos para diversas aplicaciones (Cavsi, 2014).

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

Principales Características:

- Independencia de los datos.
- Integración y sincronización de las bases de datos.
- Seguridad y protección de los datos.

SQLite v3.6

Es un sistema gestor de bases de datos relacional compatible con Atomicidad, Consistencia, Aislamiento y Durabilidad (ACID, por sus siglas en inglés *Atomicity, Consistency, Isolation and Durability*), comprendida en una biblioteca relativamente pequeña. Es una librería escrita en C que implementa un motor de bases de datos para SQL92 (Sqlite, 2016).

Algunas de sus ventajas son:

- No requiere un proceso o sistema de servidor independiente para operar.
- No se necesita configuración ni administración.
- Posee una pequeña memoria y una única biblioteca necesaria para acceder a bases de datos.
- Compatible con la mayoría de las características de lenguaje de consulta que se encuentran en el estándar SQL92 (SQL2).
- Está disponible en UNIX (Linux, Mac OS-X, Android, iOS) y Windows (Win32, WinCE, WinRT).

1.5.8 Herramientas de pruebas

El control de la calidad de software lleva consigo herramientas que permiten realizar pruebas autónomas y masivas permitiendo así la verificación desde el punto de vista estático y de caja blanca. Son pruebas donde se analiza el software sin ejecutarlo mediante el código fuente del mismo (Barrientos, 2014).

JUnit v4.12

Es un marco de trabajo creado por Erich Gamma y Kent Becket para realizar pruebas unitarias a aplicaciones que empleen el lenguaje de programación Java. Incluye formas de ver los resultados de las pruebas ya sea en forma de texto, gráfico o como tarea. Incluido a través de plugins en los principales IDEs como son NetBeans, Eclipse y Android Studio (JUnit, 2016).

Conclusiones Parciales

Como resultado de la investigación y el análisis bibliográfico de los principales conceptos asociados al problema planteado, se sentaron las bases para el desarrollo de la propuesta de solución. El estudio de la herramienta existente arrojó como resultado que no brinda una solución completa al problema a resolver, debido a que resulta imprescindible la movilidad del jefe de equipo de migración por varias áreas se decide desarrollar una aplicación para dispositivos móviles con sistema operativo Android. Para guiar el proceso de desarrollo del software se seleccionó la metodología AUP en su variante para la UCI y como herramienta para el modelado se utilizó Visual Paradigm versión 8.1, mediante el lenguaje de modelado UML versión 2.0. Durante la implementación de la solución se hará uso del IDE Android Studio versión 2.3, utilizando como lenguaje de programación Java versión 1.8, además para la gestión de los elementos locales se utilizará SQLite versión 3.6.

Capítulo 2: Análisis y Diseño

En el presente capítulo se describen las principales características de la propuesta de solución. También 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, y se representa el diagrama de clases del diseño.

2.1 Propuesta de solución

El presente trabajo propone una aplicación para dispositivos móviles con sistema operativo Android de apoyo a la migración a software libre. El jefe de equipo de migración debe importar hacia la aplicación un excel con una estructura definida (ver anexo 1), y que contiene la información de la institución a migrar. Dicha aplicación utiliza una base de datos local sqlite para almacenar de manera persistente los datos necesarios. Además, registra durante la migración de las estaciones de trabajo el estado de cada una en específico, en caso de que se haya migrado, el responsable de la estación de trabajo, la fecha de migración, el especialista y las observaciones. También, permite generar un reporte sobre el avance de la migración y conocer el porciento de migración por cada una de las áreas.

2.2 Captura de requisitos

La captura 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.

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 el correcto funcionamiento de la propuesta de solución se identificaron los siguientes requisitos funcionales:

Tabla 2. Requisitos funcionales (Elaboración propia)

No	Requisito Funcional (RF)	Descripción	Complejidad	Prioridad para el cliente
1	Importar excel	El usuario copia en la tarjeta interna o externa del dispositivo móvil un archivo excel con una estructura definida (ver anexo 1). La aplicación debe permitir seleccionar la acción “Importar excel” del menú lateral. Se muestra una ventana con varias rutas de acceso al dispositivo móvil. Los formatos requeridos para importar el excel son: .xls y .xlsx. Una vez seleccionado el excel, se lista en la aplicación la siguiente información: • Nombre del lugar o lugares a migrar. • Ubicaciones. • Áreas. • Subáreas.	Alta	Alta
2	Listar lugar	La aplicación debe mostrar al usuario el lugar o listado de lugares cuando es importado el excel. Se muestran los siguientes datos: • Ícono del lugar. • Nombre del lugar o lugares.	Media	Alta
3	Eliminar lugar	La aplicación debe permitir al usuario eliminar un lugar de la lista, y por	Media	Media

		consiguiente las ubicaciones, áreas y subáreas.		
4	Listar ubicación	La aplicación debe mostrar al usuario las ubicaciones asociadas a un lugar o lugares cuando es importado el excel. Se muestran los siguientes datos: • Ícono de las ubicaciones. • Nombre de las ubicaciones.	Media	Alta
5	Listar área	La aplicación debe mostrar al usuario las áreas asociadas a un lugar o lugares cuando es importado el excel. Se muestran los siguientes datos: • Ícono de las áreas. • Nombre de las áreas.	Media	Alta
6	Listar subárea	La aplicación debe mostrar al usuario las subáreas asociadas a un lugar o lugares cuando es importado el excel. Se muestran los siguientes datos: • Ícono de las subáreas. • Nombre de las subáreas.	Media	Alta
7	Listar estaciones de trabajo	La aplicación debe mostrar al usuario el listado de las estaciones de trabajo asociadas a una subárea. Se muestran los siguientes datos: • Ícono de las estaciones de trabajo. • Estado de la estación de trabajo. Por defecto es: “Pendiente”.	Media	Alta

		• Nombre del responsable de la estación de trabajo, inicialmente el campo está vacío. • Nombre del especialista de migración, inicialmente el campo está vacío. La cantidad de estaciones de trabajo debe ser especificada en el excel (ver anexo 1).		
8	Adicionar estación de trabajo	La aplicación debe permitir al usuario añadir una estación de trabajo al listado. Se muestran los siguientes datos: • Ícono de la estación de trabajo. • Estado de la estación de trabajo. Por defecto es: “Pendiente”. • Nombre del responsable de la estación de trabajo, inicialmente el campo está vacío. • Nombre del especialista de migración, inicialmente el campo está vacío.	Media	Alta
9	Eliminar estación de trabajo	La aplicación debe permitir al usuario eliminar una estación de trabajo de la lista.	Media	Baja
10	Eliminar todas las estaciones de trabajo	La aplicación debe permitir al usuario eliminar todas las estaciones de trabajo de la lista.	Media	Baja
11	Insertar datos en estaciones de trabajo	La aplicación debe permitir al usuario registrar de cada estación de trabajo: • Estado.	Media	Alta

		• Responsable de la estación de trabajo. • Fecha de migración. • Especialista de migración. • Observaciones.		
12	Editar estaciones de trabajo	La aplicación debe permitir al usuario modificar: • Estado • Fecha de migración • Responsable de la estación de trabajo. • Especialista de migración • Observaciones. Los cambios son actualizado en el listado de las estaciones de trabajo que muestra la siguiente información: • Estado. • Responsable de la estación de trabajo. • Especialista de migración.		Media
13	Filtrar listado de estaciones de trabajo por estado	La aplicación debe permitir al usuario filtrar el listado de estaciones de trabajo a partir de los siguientes estados: • Pendientes. • Migradas. • No migradas. • Total de estaciones de trabajo.	Alta	Alta

14	Generar reporte	La aplicación debe permitir al usuario generar un reporte general que muestra el avance de la migración y el porciento de migración por cada área. El reporte se almacena en la tarjeta interna del dispositivo, en la carpeta: "Reportes".	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:

Portabilidad

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

RNF2. Memoria RAM[9] 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[10].

2.3 Historias de usuario

La metodología empleada define distintas formas para encapsular los requisitos, surgiendo a su vez escenarios para la modelación de una solución informática (Sánchez, 2015). En el caso de la propuesta de solución el cliente está acompañando al equipo de desarrollo para acordar los detalles de los requisitos y

[9]RAM: Del inglés Random Access Memory.Traducido al español Memoria de Acceso Aleatorio.

[10]Material Design: normativa de diseño enfocado en la visualización del sistema operativo Android, además en la web y en cualquier plataforma.

así poder implementarlos, probarlos y validarlos. Teniendo en cuenta estas características se tomó la decisión de adoptar el escenario que emplea las historias de usuario para encapsular los requisitos funcionales.

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 (Cohn, 2008).

Para la descripción de los requisitos funcionales de la propuesta de solución se define una historia de usuario para cada uno de ellos, para un total de 14 historias de usuario. A continuación se muestran las HU "Importar excel, "Insertar datos en estaciones de trabajo" y "Generar reporte" por ser las que describen las funcionalidades que tienen mayor prioridad para el cliente. El resto de las historias de usuario se encuentran definidas en el anexo 6.

Tabla 3. Historia de Usuario Importar excel

Historia de Usuario	
Número: 1	**Nombre del Requisito:** Importar excel
Programador: Liset Beatríz Armas Aguila	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 1 semana
Riesgo en Desarrollo:	**Tiempo Real:** 5 días
Descripción: La aplicación debe permitir al usuario seleccionar la opción del menú lateral "Importar excel". Se muestra una ventana de acceso a varias rutas del dispositivo móvil entre ellas la tarjeta externa o interna.	

Una vez seleccionado el excel, la aplicación muestra el listado con el nombre del lugar, las ubicaciones, áreas y subáreas de la institución a migrar.
Observaciones: El excel puede contener más de un lugar. Está formado por dos hojas. La primera hoja contiene cinco columnas: • Columna 1: nombre del lugar. • Columna 2: nombre de las ubicaciones. • Columna 3: nombre de las áreas. • Columna 4: nombre de las subáreas. • Columna 5: cantidad de estaciones de trabajo. Y la segunda hoja contiene una columna con el nombre de los especialistas de migración. • Columna 1: nombre de los especialistas de migración. Los formatos requeridos para importar el excel a la aplicación son: .xls y .xlsx
Prototipo de interfaz: Ver Anexo 1

Tabla 4. Historia de Usuario Insertar datos en estaciones de trabajo

Historia de Usuario	
Número: 11	**Nombre del Requisito:** Insertar datos en estaciones de trabajo
Programador: Liset Beatríz Armas Aguila	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 1 semana
Riesgo en Desarrollo:	**Tiempo Real:** 3 días

Descripción: La aplicación debe permitir al usuario registrar el estado de las estaciones de trabajo, en caso de que la estación de trabajo se haya migrado se registra además, el nombre del responsable de la estación de trabajo, la fecha de migración, el nombre del especialista asigando a la estación de trabajo y las observaciones.	
Observaciones: Si la estación de trabajo no ha sido migrada el estado por defecto es: “Pendiente”.	
Prototipo de interfaz: Ver Anexo 3	

Tabla 5. Historia de usuario Generar reporte

Historia de Usuario	
Número: 14	**Nombre del Requisito:** Generar reporte
Programador: Liset Beatríz Armas Aguila	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 1 semana
Riesgo en Desarrollo:	**Tiempo Real:** 3 días
Descripción: La aplicación debe permitir al usuario seleccionar la acción “Reporte” del menú lateral para generar un reporte sobre el avance de la migración. El reporte se almacena en la tarjeta interna del dispositivo móvil en la carpeta: “Reportes”. El reporte es almacenado con el nombre del lugar y contiene la siguiente información: • Nombre de las áreas y subáreas asociadas a un lugar. • Total de estaciones de trabajo por cada área.	

• Cantidad de estaciones de trabajo migradas por cada área. • Porciento de migración por cada área.
Observaciones: El reporte es exportado en excel con formato .xls.
Prototipo de interfaz: Ver Anexo 4

2.4 Modelo del diseño

El diseño crea una representación o modelo de software, pero a diferencia del modelo de análisis (que se enfoca en la descripción de los datos, las funciones y el comportamiento requeridos), el modelo de diseño proporciona detalles acerca de las estructuras de los datos, las arquitecturas, las interfaces y los componentes del software que son necesarios para implementar el sistema (Pressman, 2010).

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.

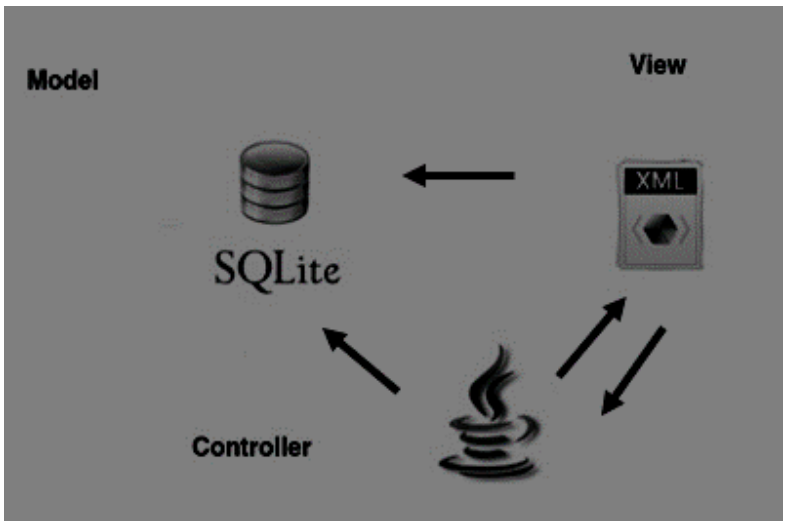


Figura 4. Arquitectura MVC (Android Development, 2015)

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:

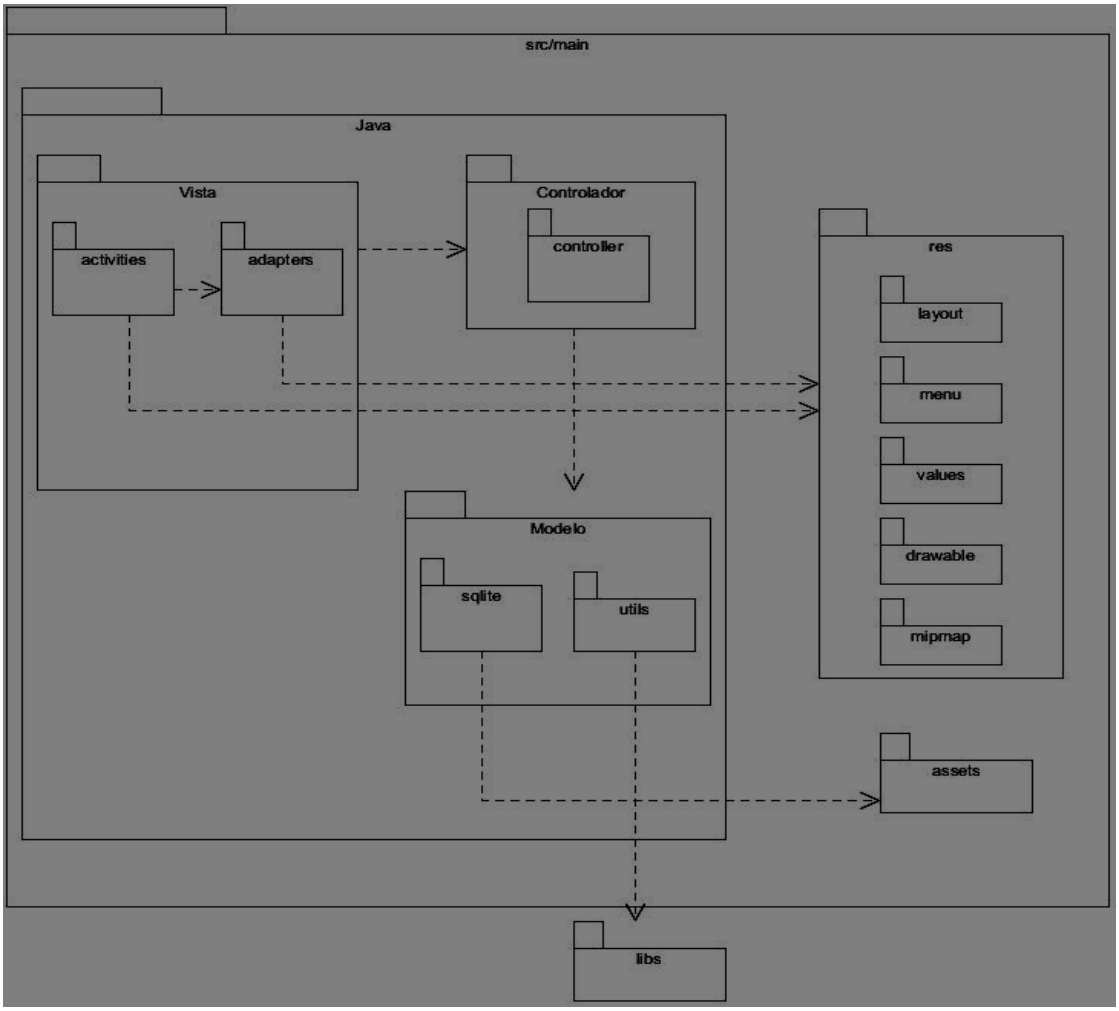


Figura 5. Diagrama de paquetes (Elaboración propia)

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/: 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.

2.4.2 Patrones de diseño

Un patrón de diseño es una descripción de la comunicación entre objetos y clases, personalizada para resolver un problema de diseño general en un contexto particular. Identifica clases, instancias, roles, colaboraciones y la distribución de responsabilidades (Larman, 2003).

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* (GRASP), 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 ModelLugar.java, ModelUbicación.java, ModelÁrea.java, ModelSubárea.java y ModelPC.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 MigracionDBHelper.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 las clases ReporteActivity.java cuya responsabilidad es la de generar los reportes y exportarlos al dispositivo móvil, no desempeña responsabilidades externas al mismo y ExcelManager.java que permite leer o escribir información del excel.

- 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, 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, 1994). En las clases MigracioDBHelper.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, 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.

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

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

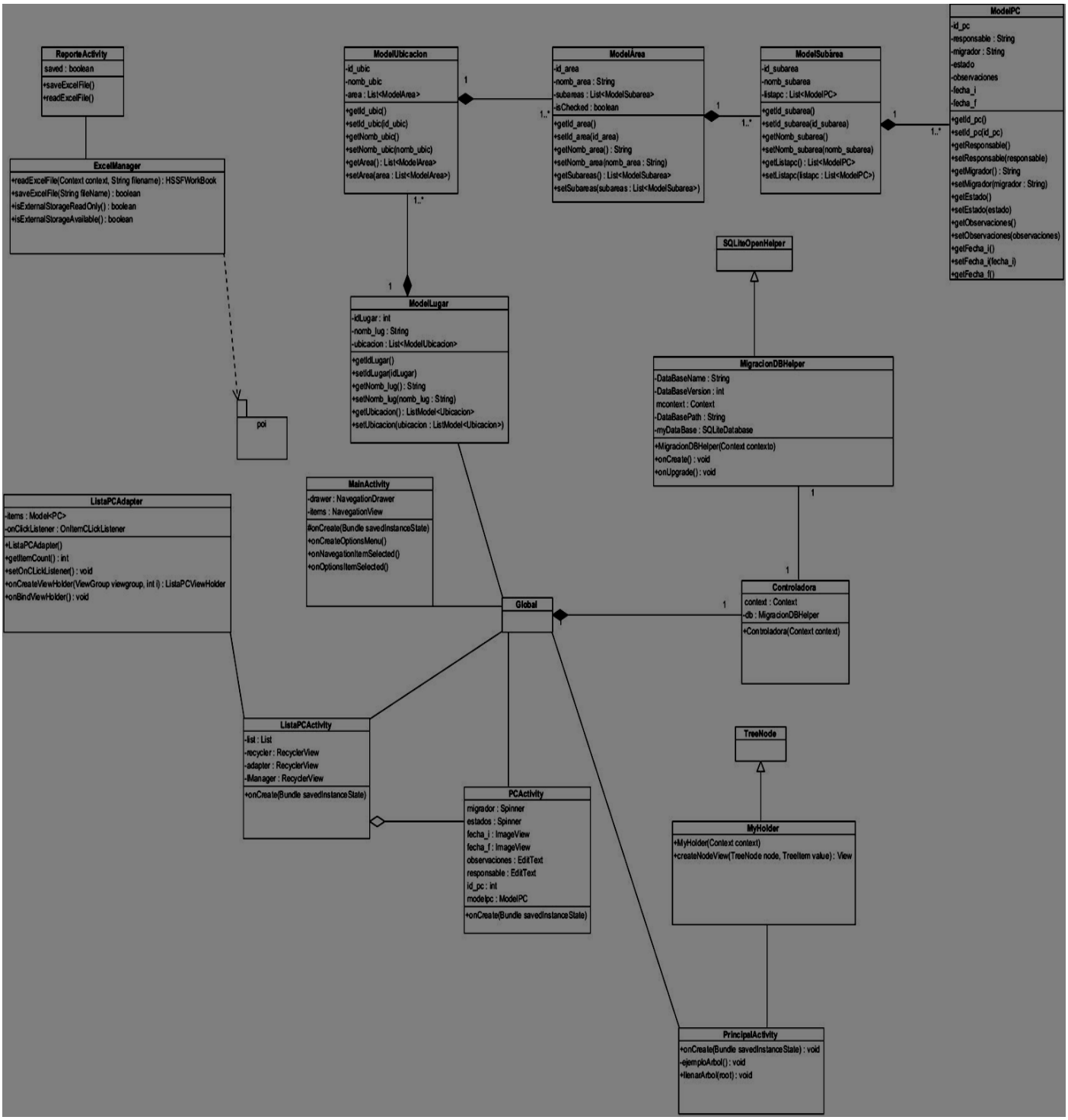


Figura 6. Diagrama de clases del diseño (Elaboración propia)

2.4.4 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.

A continuación, se presentan las entidades y campos fundamentales del modelo de datos propuesto:

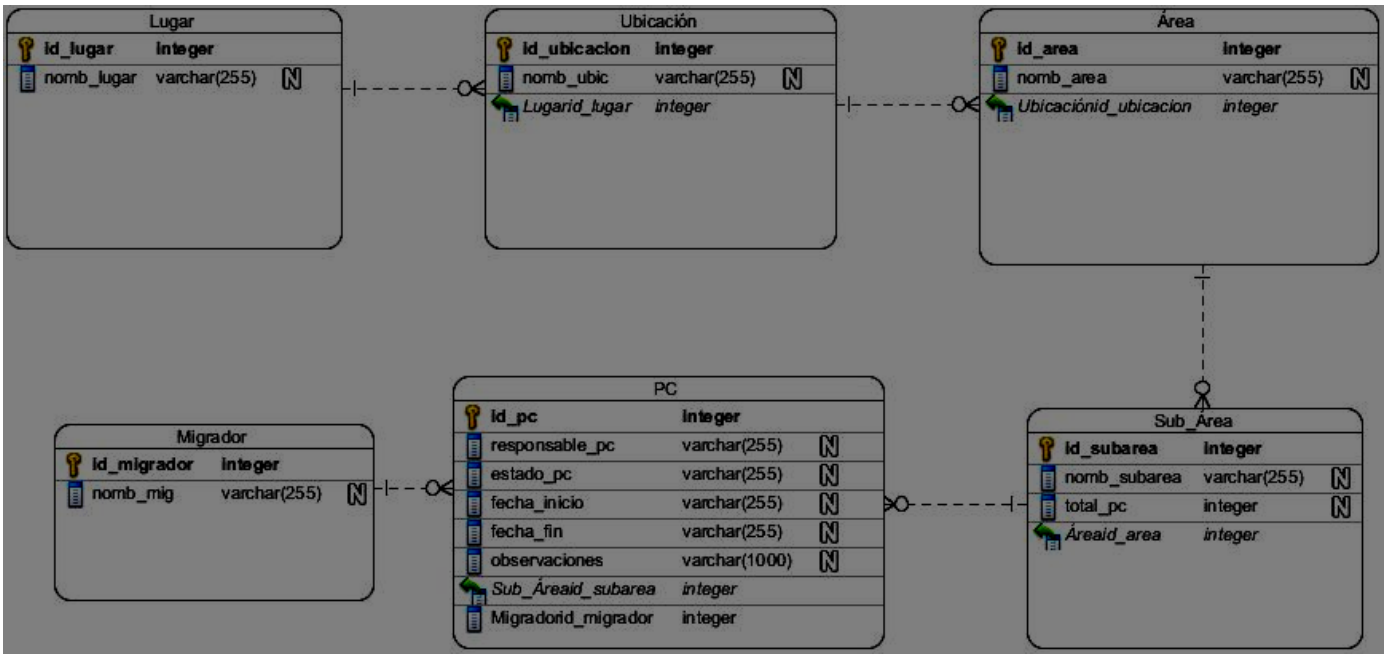


Figura 7. Modelo de datos (Elaboración propia)

Conclusiones Parciales

A partir del desarrollo del presente capítulo se describieron 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 y la reutilización de código. 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 especificó las clases persistentes de la base de datos. Como resultado principal de este capítulo se propuso una aplicación para dispositivos móviles de apoyo a la migración a software libre.

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

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.

3.1.2 Estándares de codificación

Los estándares de codificación permiten un mejor entendimiento del código por parte de los miembros del equipo de desarrollo y en consecuencia hacen que el código sea fácil de mantener. Al comenzar un software, se establece un estándar de codificación para asegurarse de que los programadores del proyecto trabajen de forma coordinada.

El uso de estándares de codificación facilita el mantenimiento de una aplicación y mejora la legibilidad del código, al mismo tiempo permite su rápida compresión (Microsoft, 2016).

Para el desarrollo de la propuesta de solución se tuvo en cuenta los siguientes estándares de codificación:

- 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 “//”.
- Para la nomenclatura de las clases, interfaces y métodos se utiliza el estilo de capitalización Pascal (Cunningham Inc., 2014), con el cual se capitaliza la primera letra de cada palabra. Ejemplo: ExcelManager.
- 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*.

3.2 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 (Sarmiento, 2013). La aplicación es desplegada en un dispositivo móvil con sistema operativo Android que contiene una base de datos local sqlite para almacenar de manera persistente los datos necesarios.

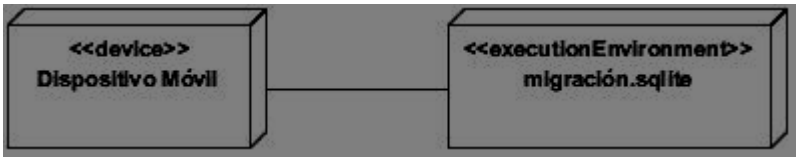


Figura 8. Diagrama de despliegue (Elaboración propia)

3.3 Pruebas de software

La prueba del software es un elemento crítico para la garantía de calidad del software y representa una revisión de las especificaciones, del diseño y de la codificación. Una vez generado el código fuente es necesario probar el software para descubrir y corregir la mayor cantidad de errores posibles antes de ser entregado. Su objetivo es diseñar una serie de casos de prueba que tengan una alta probabilidad de encontrar errores (Pressman, 2010).

3.3.1 Estrategias de Prueba

Una estrategia de pruebas de software proporciona una guía que describe los pasos que deben realizarse como parte de la prueba, cuándo se planean y se llevan a cabo dichos pasos, y cuánto esfuerzo, tiempo y recursos requerirán. Por tanto, cualquier 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. Según Pressman (2010), las pruebas se dividen en los siguientes niveles principales: pruebas de unidad, de integración, de validación, de sistema y de aceptación.

Para la propuesta de solución se utilizó una estrategia de pruebas basada en la ejecución de las mismas tomando como guía tres de estos niveles: 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 muestran los casos de prueba correspondientes a las historias de usuario “Importar excel”, “Insertar datos en estaciones de trabajo” y “Generar reporte”. Los casos de prueba asociados al resto de las historias de usuario son definidos en el anexo 7.

Tabla 6. Caso de prueba de validación Importar excel

Caso de Prueba de Validación	
Código: CP1-HU1	HU 1: Importar excel
Responsable: Liset Beatríz Armas Aguila	
Descripción: El caso de prueba se inicia al seleccionar la acción “Importar excel” del menú lateral. Se presenta al usuario una ventana de búsqueda para acceder a la tarjeta interna o externa del dispositivo móvil. El caso de prueba termina cuando el excel es importado a la aplicación y se muestra un mensaje de éxito.	
Condiciones de ejecución: El excel debe cumplir con la siguiente estructura para ser importado a la aplicación:	

<table>
<tr><td colspan="4">

- Columna 1: nombre del lugar.
- Columna 2: nombre de las ubicaciones.
- Columna 3: nombre de las áreas.
- Columna 4: nombre de las subáreas.
- Columna 5: cantidad de estaciones de trabajo.

Los formatos requeridos para importar el excel a la aplicación son: .xls y .xlsx
</td></tr>
<tr><th>Escenario</th><th>Descripción</th><th>Respuesta del sistema</th><th>Flujo central</th></tr>
<tr><td>EC 1.1 Mostrar la ventana de acceso a la tarjeta externa o interna del dispositivo y seleccionar el excel.</td><td>El usuario ejecuta la aplicación y al seleccionar la acción “Importar excel” del menú lateral, se muestra la ventana de acceso al dispositivo.</td><td>La aplicación muestra al usuario la ventana de acceso a la tarjeta externa o interna del dispositivo para seleccionar el excel. Una vez importado el excel con la estructura requerida, se muestran los siguientes datos: Nombre del lugar o lugares a realizar la migración, Acciones (Eliminar el lugar o listado de lugares). Nombre de las ubicaciones, áreas y subáreas.</td><td>El usuario selecciona la acción “Importar Excel” del menú lateral.</td></tr>
</table>

EC 1.2 Error al importar el excel a la aplicación.	El usuario selecciona la acción “Importar excel” del menú lateral y el contenido no es visualizado en pantalla.	La aplicación muestra al usuario un mensaje de error indicando que ha fallado la acción de importar los datos del excel por tener una estructura incorrecta.	El usuario selecciona la acción “Importar excel” del menú lateral.

Tabla 7. Caso de prueba de validación Insertar datos en estaciones de trabajo

Caso de Prueba de Validación	
Código: CP6_HU6	HU 6: Insertar datos en estaciones de trabajo
Responsable: Liset Beatríz Armas Aguila	
Descripción: El caso de prueba se inicia cuando es importado el excel a la aplicación. El usuario selecciona la estación de trabajo a migrar y registra los siguientes datos: • Estado (Pendiente: valor por defecto, Migrada, Microsoft Windows, Doble buteo, Rota, el usuario no migró, otra distribución de GNU/Linux) • Fecha de migración. • Especialista de migración. • Responsable de la estación de trabajo. • Observaciones. El caso de prueba termina cuando el usuario selecciona la acción “Guardar” y se actualizan los cambios en el listado de las estaciones de trabajo.	
Condiciones de ejecución: Debe ser importado el excel a la aplicación. Los formatos requeridos para importar el excel a la aplicación son: .xls y .xlsx.	

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 6.1 Mostrar el listado de las estaciones de trabajo.	El usuario una vez que selecciona la acción “Importar excel” se muestra el lugar o listado de lugares. Cuando selecciona el ícono de una subárea se listan las estaciones de trabajo.	La aplicación muestra al usuario un ícono, el nombre del especialista y el responsable de la estación de trabajo inicialmente con los campos vacíos y el estado de la estación de trabajo por defecto: “Pendiente”. Cuando selecciona el ícono de la estación de trabajo se muestran los siguientes datos: estado, fecha de migración, responsable, especialista y observaciones. Acciones (Eliminar una estación de trabajo, editar la información de la estación de trabajo y guardar los cambios realizados).	El usuario selecciona una estación de trabajo de la lista.
EC 6.2 No se actualizan los campos del listado de las estaciones de trabajo.	El usuario selecciona una subárea y se muestra el listado de estaciones de trabajo con los campos	La aplicación muestra al usuario el listado de las estaciones de trabajo con los campos Responsables y Especialista vacíos y el	El usuario selecciona una estación de trabajo de la lista.

	Responsable, Especialista y Estado.	estado por defecto: Pendiente.	

Tabla 8. Caso de prueba de validación Generar reporte

Caso de Prueba de Validación			
Código: CP11-HU11		HU 11: Generar reporte	
Responsable: Liset Beatríz Armas Aguila			
Descripción: El caso de prueba se inicia cuando se registran los datos en las estaciones de trabajo. El usuario selecciona la acción “Generar reporte” del menú lateral. La aplicación muestra un listado con el nombre del lugar o lugares. El usuario selecciona un lugar y la aplicación muestra un mensaje de alerta para verificar si desea crear el reporte. El caso de prueba termina cuando el usuario selecciona la acción “Aceptar” y el reporte es exportado en la tarjeta interna del dispositivo móvil en la carpeta “Reportes”.			
Condiciones de ejecución: Debe ser importado el excel a la aplicación. Los formatos requeridos para importar el excel a la aplicación son: .xls y .xlsx.			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 11.1 Mostrar el lugar o listado de lugares.	El usuario selecciona un lugar de la lista. Cuando es presionado, se muestra un mensaje para verificar si desea o no crear el reporte.	La aplicación muestra al usuario la lista de lugares o lugar. Acciones (generar un reporte en excel). El reporte es exportado en la tarjeta interna del dispositivo móvil cuando se selecciona el ícono del lugar y se	El usuario selecciona la acción “Generar reporte” del menú lateral.

		almacena en la carpeta: “Reportes”).	
EC 11.2 No se especifica la ruta de acceso al reporte.	El usuario selecciona la acción “Generar reporte” del menú lateral. Cuando presiona el ícono del lugar se muestra un mensaje de confirmación para crear el reporte.	El usuario selecciona un lugar de la lista y cuando selecciona la acción “Aceptar” no especifica la ruta de acceso al dispositivo.	El usuario selecciona la acción “Generar reporte” del menú lateral.

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 resultado de las pruebas de software

Para la ejecución de las pruebas de validación y de aceptación 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
- Para ilustrar el desarrollo del proceso de pruebas unitarias se muestra una imagen de los resultados arrojados por JUnit para los casos de pruebas implementados.

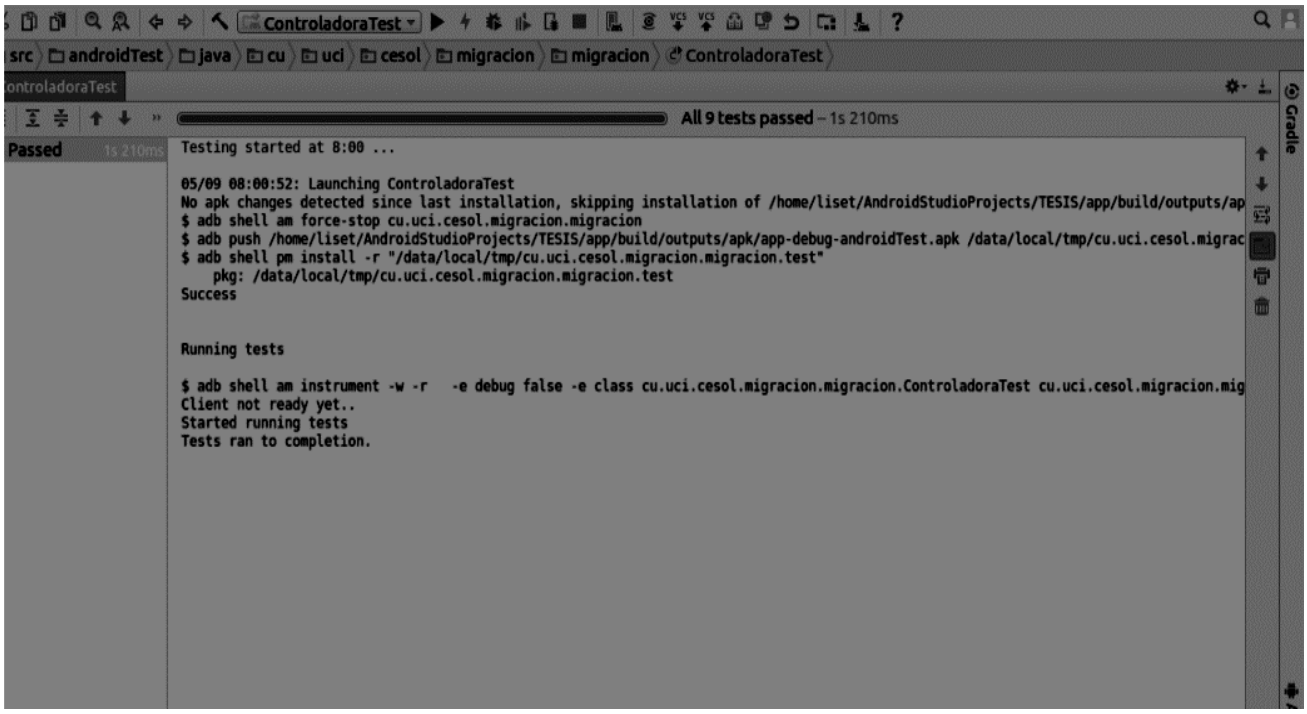


Figura 9. Resultados de JUnit (Android Studio v2.3)

A partir de la interpretación de dichos resultados se puede afirmar que los mismos garantizan el correcto funcionamiento de los métodos de la clase Controladora.

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 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 lugares existentes en el dispositivo móvil cuando se elimina un lugar de la lista.
2. La aplicación no actualiza el contenido del listado de estaciones de trabajo cuando se editan los campos estado, responsable y especialista.
3. La aplicación duplica el listado de los lugares cuando es importado nuevamente el excel.
4. La aplicación no actualiza el reporte con la información actualizada de las estaciones de trabajo de alguna de las áreas.
5. La aplicación no muestra el listado de lugares actualizados al seleccionar la acción “Generar reporte” del menú lateral.
6. La aplicación se detuvo al seleccionar la acción eliminar un lugar. El lugar se elimina de la base de datos pero no se actualiza el listado de lugares.

No conformidades no significativas:

1. La aplicación no muestra completamente en el listado de las estaciones de trabajo el texto con el nombre del especialista y el responsable de la estación de trabajo.
2. La aplicación no muestra completamente el texto de las acciones a realizar en el listado de las estaciones de trabajo.
3. La aplicación muestra la palabra “Responsable” y el nombre del responsable sin un espacio entre ambos textos.
4. La aplicación muestra el botón “Adicionar estación de trabajo”, en la interfaz de las estaciones de trabajo, de forma que este queda por encima del nombre del especialista, lo que impide la visualización por parte del usuario.

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 no pudo importarse el excel a la aplicación debido a la estructura.
2. La aplicación no muestra ningún resultado al seleccionar un lugar de la lista para generar el reporte.

No conformidades no significativas:

1. La aplicación muestra incorrectamente los textos presentes en el cuadro de diálogo “Selección” que se visualiza al seleccionar la opción Filtrar estaciones de trabajo por estado.
2. La aplicación muestra los nombres de los especialistas duplicados en el listado de la interfaz de estación de trabajo.

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:

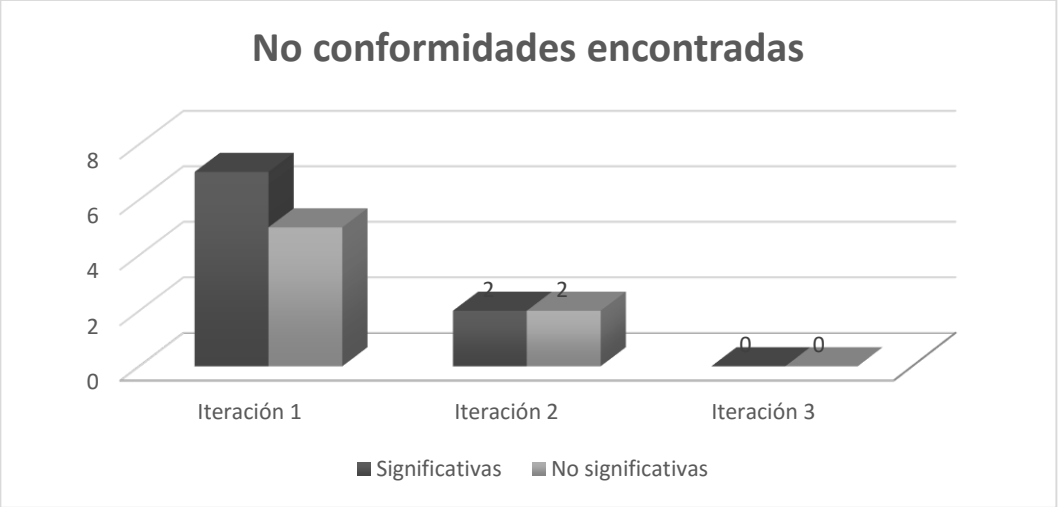


Figura 10. No conformidades encontradas por iteración en el proceso de pruebas (Elaboración propia)

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.

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 Ejecución.
- 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 a la aplicación de apoyo a la migración a software libre, obteniendo una herramienta capaz de generar reportes sobre el estado de la migración, facilitándole a los jefes de equipo de migración el registro y consulta de la información que se requiera con rapidez.
- La ejecución de pruebas a la aplicación de apoyo a la migración a software libre, permitió verificar el correcto funcionamiento de los requisitos identificados y la aceptación por parte del cliente.

Recomendaciones

Para posteriores versiones de la aplicación se recomienda:

- Agregar una funcionalidad que permita sincronizar varios dispositivos móviles para el intercambio de información entre los jefes de equipo de migración.
- Agregar una funcionalidad que permita registrar la información generada en la etapa de Consolidación de la migración a software libre.

Referencias Bibliográficas

Ambler S.W, 2016. Herramientas CASE. [Consulta: 27 septiembre 2016]. [En línea].Disponible en: http://www.agilemodeling.com/essays/simpleTools.htm#SelectingCASE

Android developers, 2016. Download Android Studio and SDK Tools | Android Studio. [En línea]. [Consulta: 8 diciembre 2016]. Disponible en: https://developer.android.com/studio/index.html?hl=es.

Barrientos, Pablo Andrés (25 de abril de 2014). Enfoque para pruebas de unidad basado en la generación aleatoria de objetos. [En línea]. [Consulta: 8 diciembre 2016]. Disponible en: http://sedici.unlp.edu.ar/handle/10915/34969

Cunningham & Cunningham Inc., 2014. Pascal Case. [En línea]. [Consulta: 6 de marzo 2017]. Disponible en: http://c2.com/cgi/wiki?PascalCase.

Feal Delgado W., Alvarez Acosta H., Miguel Canosa Reyes R., 2014. Estrategia de migración al software libre en la Universidad de Cienfuegos. Universidad y Sociedad. [En línea] [Consulta: 15 septiembre 2016]. Disponible en: http://rus.ucf.edu.cu

Grant Kevin, Haseman Chris, 2014. Beginning Android Programming. Develop and Design. Printed and bound in the United States of America. ISBN-13: 978-0-321-95656-9

Gamma, E., Helm, R., Johnson, R. y Vlissides, J., 1994. Design Patterns : Elements of Reusable Object-oriented Software. S.l.: Addison-Wesley. ISBN 0-201-63361-2.

Genymotion, 2016. Genymotion – Fast and Easy Android Emulation. [En línea]. [Consulta: 8 diciembre 2016]. Disponible en: https://www.genymotion.com/.

Gradle Inc., 2016. Getting Started with Gradle for Android Build l Gradle. [En línea]. [Consulta: 14 diciembre 2016]. Disponible en: http://gradle.org/getting-started-android-build/.

Granollers T., 2014. Herramientas de Prototipo de interfaz de usuario [Consulta: 27 septiembre 2016]. [En línea].Disponible en: http://www.grihotools.udl.cat/mpiua/herramientas-de-prototipado-de-interfaces-de-usuario

Herramientas Case, 2016. [En línea] [Consulta: 8 diciembre 2016.] Disponible en: http://fdsherramientascase.blogspot.com/

Historia — Android OS 0.1 documentation, 2012. [Consulta: 27 octubre 2016]. [En línea].Disponible en: http://androidos.readthedocs.io/en/latest/data/historia/

JUnit, 2016. JUnit - About. [En línea]. [Consulta: 27 octubre 2016]. Disponible en: http://junit.org/junit4/.

Larman, C., 2004. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, Third Edition [en línea]. S.l.: Addison Wesley Professional. [Consulta: 11 octubre 2016]. ISBN 0-13-148906-2. Disponible en: https://aanimesh.files.wordpress.com/2013/09/applying-umland-patterns-3rd.pdf.

Lineamientos de la Política Económica y Social del Partido y la Revolución, 2011.

Microsoft, 2016. Microsoft. MSDN. Revisiones de código y estándares de codificación. [En línea]. [Consulta: 28 febrero 2016]. Disponible en: http://msdn.microsoft.com/eses/library/aa291591(v=vs.71).aspx.

Mockups, Balsamiq, 2016. Balsamiq Mockups. [En línea] [Consulta: 25 octubre 2016.] Disponible en: http://www.balsamiq.com.

Montano José L., 2015. La migración hacia software libre en Cuba: complejo conjunto de factores sociales y tecnológicos en el camino de la soberanía nacional [En línea] [Consulta: 27 octubre 2016]. Disponible en: http://scielo.sld.cu/scielo.php?script=sci_abstract&pid=S2

Oracle Corporation, 2016. ¿Qué es Java? [En línea]. [Consulta: 15 de noviembre 2016]. Disponible en: https://www.java.com/es/about/whatis_java.jsp.

Pierra C Allan., 2011. Nova, distribución cubana de GNU/Linux: reestructuración estratégica de su proceso de desarrollo. [En línea] [Consulta: 17 octubre 2015]. Disponible en: http://catalogoenlinea.uci.cu/cgi-bin/koha/opac-detail.pl?biblionumber=11161

Pressman R., 2010. Software Engineering. A practitioner's approach. McGraw-Hill, 7ma edición.

Proyectos de migración a software libre, 2016. [Consulta: 8 diciembre 2016]. [En línea]. http://www.solucionesit.com.ve/servicios/servicios-consultoria/proyectos-de-migracion-a-sl.html

PuroMarketing, 2016. PuroMarketing. Nuevos datos demuestran la dependencia a los dispositivos móviles. [En línea]. [Consulta: 8 diciembre 2016]. Disponible en: http://www.puromarketing.com/12/19353/datos-demuestran-dependencia-dispositivos-moviles.html.

¿Qué es un Sistema de Gestor de Bases de Datos o SGBD? CAVSI. [Citado el: 15 diciembre de 2016]. Disponible en: http://www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-debases-de-datoso-sgbd.

Sánchez Alonso, 2014. Entorno de Desarrollo Integrado. [Consulta: 27 septiembre 2016]. [En línea]. Disponible en: https://prezi.com/7wmx8_d6ertl/entorno-desarrollo-integrado-ide

Sánchez T.R., 2015. Metodología UCI [En línea]. 8 abril 2015. S.l.: s.n. [Consulta: 15 noviembre 2016]. Disponible en: http://excriba.prod.uci.cu/proxy/alfresco/api/node/content/workspace/SpacesStore/a622adab-eac54fb3ba08-a266767fff5f/Metodologia%20UCI.pdf?a=true.

Shin, L.Y., 2014. Journal of European Psychology Students. A Comparative Study of Mobile Internet Usage between the U.S. and Korea. [En línea]. [Consulta: 5 diciembre 2016]. Disponible en: http://jeps.efpsa.org/articles/10.5334/jeps.cg/.

Sommerville I., 2011. Ingeniería de Software 7ma edición. . [Consulta: 17 noviembre 2016]. [En línea]. Disponible en: http://zeus.inf.ucv.cl/~bcrawford/Modelado%20UML/Ingenieria%20del%20Software%207ma.%20Ed.%20-%20Ian%20Sommerville.pdf

Soberanía tecnológica e infraestructura de Internet en Cuba, 2016. [Consulta: 11 septiembre 2016]. [En línea]. Disponible en: http://www.cubadebate.cu/noticias/2016/03/25/soberania-tecnologica-einfraestructura-de-internet-en-cuba/

Sqlite, 2016. ¿Qué es sqlite? [En línea]. [Consulta: 15 de noviembre 2016]. Disponible en: https://www.sqlite.org

Stallman Richard M., 2004. Software Libre para una sociedad libre. Editorial Traficantes de Sueños. España.

Suzarte Medina S., 2013. Frente al bloqueo de windows, NOVA sí va. Contra el Terrorismo Mediático. [En línea]. Recuperado de: http://www.cubadebate.cu/especiales/2010/01/13/cuba-frente-bloqueowindows-nova/

Unified Modeling Language, 2016. Unified Modeling Language (UML). [En línea]. [Consulta: 15 diciembre 2016]. Disponible en: http://www.uml.org/.

Villazón Y.P & otros, 2014. Buenas Prácticas para la Migración a Código Abierto. La Habana, Universidad de las Ciencias Informáticas.

Villazón, Y.P. 2013. Reestructuración del modo de ejecución de los procesos de migración a aplicaciones de Código Abierto.

Villazón, Y. P; Vitier A.G, García J.G. Plataforma Cubana de Migración a Código Abierto. . 2012. P. 3–4.

Vílchez Ángel, 2009. Configurarequipos. Qué es Android: Características y Aplicaciones. [En línea]. Disponible en: http://www.configurarequipos.com/doc1107.html.

Visual Paradigm, 2016. Visual Paradigm for UML 8.1 Community Edition. [Consulta: 27 septiembre 2016]. [En línea].Disponible en: http://www.software.com.ar/p/visual-paradigm-para-uml.

Sarmiento Johana, 2013. Diagrama de despliegue. [En línea]. [Consulta: 15 diciembre 2016]. Disponible en: http://umldiagramadespliegue.blogspot.com

Suárez Y., 2015. Entorno de Desarrollo Integrado. Qué es un IDE? [En línea] 2015. [Citado el: 8 de 12 de 2016.] http://deprogramacion.cubava.cu/2016/02/01/que-es-un-ide/.

Bibliografía Consultada

Android Developers, 2016. Android Developers. [En línea]. Disponible en: https://developer.android.com/index.html.

Android Development, 2015. [En línea]. Disponible en: piazza.com/gatech/spring2015/cicrnoc-lab-staff@lists.gatech.edu

Báez & otros, 2014. Introducción a Android ISBN: 978-84-96285-39-5

Carbonell F.L, 2016. Resolución la Implementación de los Lineamientos de la Política Económica y Social aprobados en el 6to. Congreso y su actualización para el período 2016. [En línea]. [Consulta: 15 septiembre 2016]. Disponible en: www.granma.cu/ séptimo-congreso-del-pcc

González, Y., 2016. Por los caminos de la soberanía tecnológica. [En línea]. [Consulta: 15 septiembre 2016]. Disponible en: www.granma.cu

Jose, 2015. Diseñando aplicaciones Android con Material Design. [En línea] [Consulta:]. Disponible en: http://android.uci.cu

ISO 25000, 2015. ISO 25010. [En línea]. [Consulta: 15 noviembre 2016]. Disponible en: http://iso25000.com/index.php/normas-iso-25000/iso-25010.

Larman, C., 1999. UML y patrones. Introducción al análisis y diseño orientado a objetos. Prentice-Hall. ISBN 0-13-148906-2.

Microsoft, 2014. Técnicas de codificación. [En línea]. Disponible en: http://msdn.microsoft.com/es-es/library/aa291593(v=vs.71).aspx.

Oracle Corporation, 2016. Oracle. Oracle Technology Network for Java Developers. [En línea]. [Consulta: 12 diciembre 2015]. Disponible en: http://www.oracle.com/technetwork/java/index.html.

Pruebas de software., 2016. Gestión de Calidad y Pruebas de Software. [En línea]. Disponible en: http://www.pruebasdesoftware.com/laspruebasdesoftware.htm.

Samón R.P, Villazón Y.P, Abad A.M. ,2009. Guía Cubana de Migración a Software Libre

Sampieri R.H., Collado, C.F. y Lucio, M. del P.B., 2010. Metodología de la investigación. 5ta. México D.F.: McGraw-Hill. ISBN 978-607-15-0291-9.

Sommerville I., 2011. Ingeniería de Software 7ma edición. . [Consulta: 17 noviembre 2016]. [En línea]. Disponible en: http://zeus.inf.ucv.cl/~bcrawford/Modelado%20UML/Ingenieria%20del%20Software%207ma.%20Ed.%20-%20Ian%20Sommerville.pdf

Plataforma Cubana de Migración a Código Abierto. Revista Cubana de Ciencias Informáticas Vol. 8, No. Especial UCIENCIA 2014, [En línea] [Consulta: 19 octubre 2016] Disponible en: http://rcci.uci.cu

Pressman, Roger "Ingeniería del Software. Un enfoque práctico".2010. McGraw-Hill/Interamericana de España.

Qué es un Sistema de Gestor de Bases de Datos o SGBD CAVSI. [En línea] [Consulta: 8 noviembre de 2016]. Disponible en: http://www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-de-bases-de-datos-osgbd.

Villazón, Y.P. ,2013 El proceso de migración a aplicaciones de código abierto en Cuba desde un enfoque metodológico. Revista Cubana de Ciencias Informáticas *versión On-line* ISSN 2227-1899. . [En línea] Disponible en: http://rcci.uci.cu/index.php/rcci/article/view/287

Anexos

Anexo 1: Estructura del excel

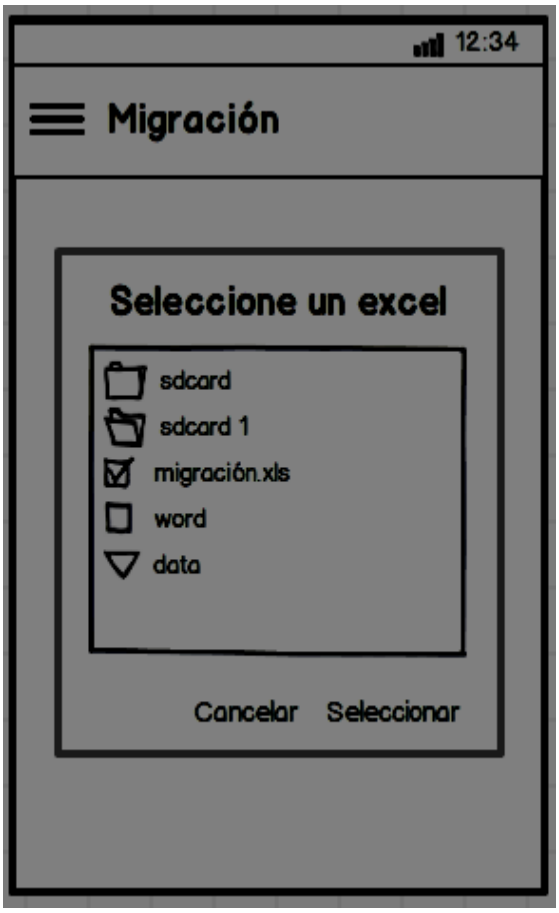
Hoja 1

	A	B	C	D	E
1	UCI				
2		Rectorado			
3			Rectorado pasillo exterior		
4				Local de protocolo	2
5				Dirección de mantenimiento	1
6				Departamento de Economía.	2
7				Asesoría Jurídica.	1
8				Finanzas.	2
9				Oficina de Ubicación Laboral.	2
10				Subdirección.	3
11				Oficina de especialistas.	5
12				Dirección de Inversiones y Mtto.	5
13			Casona		
14				Oficina VR Extensión	4
15				Dirección General	6
16		Fac 1			
17			CESOL		
18				Lab 101	15
19				Lab 102	25

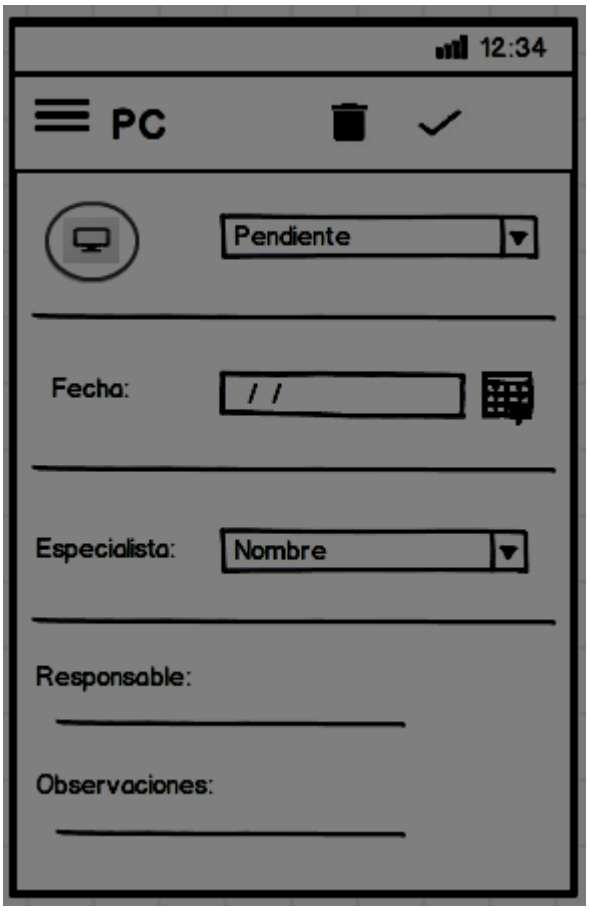
Hoja 2

	A
1	Luis
2	Alberto
3	Pedro
4	Luisa
5	Ana
6	Pepe

Anexo 2: Importar excel



Anexo 3: Insertar datos en estación de trabajo


12:34
PC
Pendiente
Fecha:
/ /
Especialista:
Nombre
Responsable:
Observaciones:

Anexo 4: Generar Reporte

Anexo 5: Entrevista realizada a los jefes de equipo de migración en el Centro de Software Libre

1. ¿Cómo realizas el control de la migración a software libre de las estaciones de trabajo en una institución?
2. ¿Qué información manejas durante el control de la migración a software libre de las estaciones de trabajo en una institución?
3. ¿Qué herramienta utilizas para el control de la migración a software libre de las estaciones de trabajo en una institución?
4. Teniendo en cuenta que existen varias áreas en una institución por las que el jefe de equipo de migración debe transitar, ¿cree que el uso de una aplicación para dispositivos móviles agilizaría el registro y la consulta de información para el control de la migración de las estaciones de trabajo? ¿Por qué?

Anexos