UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 4

Aplicación móvil para el Sistema de Gestión del Agente Transitario de Cargas TRANSCARGO

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

Autor:
Ernesto Ruíz Rodríguez

Tutores:
Ing. Leannys Rodríguez Moreno

Ing. Orlando Grabiel Toledano López

La Habana, junio de 2017
"Año 59 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaro ser el autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____

Ernesto Ruíz Rodríguez

Autor

_____ _____

Ing. Leannys Rodríguez Moreno **Ing. Orlando Grabiel Toledano López**

Tutor Tutor

A mamá, papá y tatis: esas estrellas que llenan todos mis rincones sombríos con una luz cálida e imperecedera.
Gracias por traerme la dicha en los momentos de angustia, convertir mis lágrimas en sonrisas y guiarme hacia mis sueños.

RESUMEN

El desarrollo vertiginoso de las Tecnologías de la Información y las Comunicaciones ha abierto un mundo de posibilidades para cualquier empresa que persiga el cumplimiento de todos sus objetivos. Dentro de estas tecnologías, los Sistemas de Gestión Integral se han convertido en la columna vertebral de muchas organizaciones, permitiendo integrar y automatizar la mayoría de sus procesos. Una de las principales tendencias en los últimos años ha sido la adaptación de estos sistemas al entorno móvil, lo cual lleva a las empresas a sus máximas capacidades y ganancias. Actualmente el Centro de Tecnologías para la Formación, perteneciente a la Universidad de las Ciencias Informáticas se encuentra desarrollando un Sistema de Gestión Integral para el agente transitario de cargas con nombre comercial TRANSCARGO. El sistema, actualmente en fase de desarrollo, no ha sido adaptado para su uso en terminales móviles, lo cual provoca falta de movilidad en la empresa y la imposibilidad de acceder a este si no existe conexión de red. El presente trabajo describe el desarrollo de una aplicación móvil nativa que proveerá una solución a los anteriores problemas. La aplicación fue desarrollada mediante la metodología Mobile-D, utilizando el entorno de desarrollo Android Studio, el lenguaje de programación Java y el framework Odoo Mobile. Con el software desarrollado se permite el acceso a las principales funcionalidades del Sistema de Gestión Integral de TRANSCARGO desde terminales móviles, brindando la posibilidad de trabajar sin conexión.

Palabras claves: Android, Sistema de Gestión Integral, tecnologías móviles.

ÍNDICE

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Con el surgimiento de Internet y el desarrollo vertiginoso de las Tecnologías de la Información y las Comunicaciones (TIC) se ha abierto un mundo de posibilidades para la sociedad. El acceso a cualquier información es posible en cuestión de segundos, tareas y procesos de diversa índole han sido informatizados, los nuevos paradigmas han cambiado la manera en que individuos y organizaciones realizan los procesos e interactúan entre sí.

La posibilidad de conexión que brinda Internet ha permitido el desarrollo de un nuevo espacio en el cual se llevan a cabo un sin número de transacciones, ellas comprenden desde el simple intercambio de información, hasta la realización de actividades comerciales. El Internet ha revolucionado la vida social, cultural y económica a nivel global; transformando las modalidades de comunicación entre personas, la manera de hacer los negocios entre las empresas y la forma de trabajar (Sánchez-Torres, González-Zabala, y Muñoz 2013).

Quizás las empresas hayan sido las más beneficiadas dentro de la llamada "*Era de la Información"*; el uso cada vez más extendido de las redes de comunicación, programas de diseño, almacenes de datos y programas de gestión guían a las entidades hacia el cumplimiento de sus objetivos. Las TIC mejoran y optimizan los procesos de toda organización, agilizan sus actividades y operaciones, procesan los datos de manera eficiente para convertirlos en información útil y ayudar en la toma de decisiones. Todo lo anterior deviene en un importante ahorro de recursos, aumento de la productividad y la competitividad y el acercamiento a nuevos clientes. Actualmente, la mayoría de los procesos de las empresas están soportados por estas tecnologías (Costa 2012):

- El proceso de logística y distribución (con las herramientas llamadas "*Supply Chain Managemen*t" o Administración de redes de suministro[1]).
- El proceso productivo (con los conocidos y famosos ERP[2] ("*Enterprise Resource Planning* o Sistemas de Gestión Integral).
- Las actividades comerciales y de marketing con las herramientas conocidos como CRM[3] (*Custormer Relationship Management o* Administración de la relación con los clientes).
- Los procesos de desarrollo de nuevos productos, con herramientas de CAD[4] (*Computer-aided design* o *Diseño asistido por computadoras*).

[1] Es el proceso de planificación, puesta en ejecución y control de las operaciones de la red de suministro.
[2] Son sistemas informáticos destinados a la administración de recursos en una organización.
[3] Son sistemas informáticos de apoyo a la gestión de las relaciones con los clientes, a la venta y al marketing.
[4] Es el uso de un amplio rango de herramientas computacionales que asisten a ingenieros, arquitectos y diseñadores.

Entre las tecnologías mencionadas sin duda, las que más protagonismo han ganado actualmente son los ERP, plataformas que permiten integrar y automatizar la mayoría de los procesos llevados a cabo por las empresas: contabilidad, recursos humanos, compras, finanzas y manufacturación. Con ellos todo se encuentra integrado en el mismo software para que pueda ser gestionado de forma sencilla y apoyar en la toma de decisiones. Estos sistemas son considerados como la columna vertebral de muchas organizaciones y su uso conlleva al ahorro y aumento de la productividad (Cailean y Sharifi 2014).

Actualmente la sociedad se encuentra inmersa en lo que se denomina "*revolución tecnológica de las comunicaciones inalámbricas"*. Una de las principales ventajas de esta tecnología es la movilidad, no depender del cable, esto favorece su expansión, la cual es más rápida que cualquier otro tipo de tecnología (Blázquez et al. 2011). La movilidad puede ser considerada como un fenómeno general, que afectan la vida personal y el comportamiento de las organizaciones de manera decisiva. Las peticiones y expectativas que estas organizaciones necesitan satisfacer en el contexto del negocio, son afectadas por la movilidad; obviamente el acceso oportuno a la información afecta al negocio y lo hace más efectivo (Kurbel, Dabkowski, y Jankowska 2003).

Las tecnologías móviles se han integrado al proceso productivo y de gestión de las organizaciones. Según un estudio realizado por la consultora *IDG Connect*[5]**:** las empresas que utilizan las tecnologías y aplicaciones móviles son más productivas y generan más ingresos, son cada vez más conscientes del beneficio del uso de la tecnología y los dispositivos móviles, así como de las aplicaciones móviles **"**a la medida**"** (desarrolladas para necesidades particulares de productividad). Estas empresas consiguen resultados más rentables, según el estudio, a diferencia de lo que sucede con las que utilizan estrategias móviles limitadas o poco sistemáticas (Kelevra 2014).

Una de las principales tendencias en los últimos años ha sido la adaptación de los Sistemas de Gestión Integral al entorno móvil. Los ERP móviles son considerados la segunda etapa para la implementación de los sistemas ERP, llevando a las empresas a sus máximas capacidades y ganancias. Extender el uso de los ERP más allá de la oficina es la nueva línea a seguir en el mundo empresarial, entre sus amplias ventajas se pueden destacar (Cailean y Sharifi 2014):

- **Acceso a la información:** permiten a empleados y directivos acceder a la información de la empresa a través de su smartphone o tablet en cualquier momento y desde cualquier lugar.

[5] http://www.idc.com

- **Reducción de tiempos administrativos:** no es necesario trasladarse grandes distancias para administrar la empresa, estas funciones se realizan sobre la marcha.
- **Mejora en la toma de decisiones:** estas podrán ser tomadas en los momentos precisos reduciendo costos, incrementando la productividad y eficiencia y por ende los beneficios.
- **Respuestas inmediatas:** al aumentar la inmediatez en las respuestas a los clientes se acelera también la satisfacción de sus necesidades.
- **Incremento de la coordinación**: es posible para los directivos de la empresa dirigir los procesos sin estar conectados a la computadora de su oficina.

La gestión móvil de la empresa es una realidad que viene ganando espacios en el mercado de aplicaciones. Un estudio realizado por la firma de investigación *Marketsandmarkets*[6] reveló que el mercado de aplicaciones móviles empresariales crecerá una media de 15.24% anual en los próximos cinco años. El estudio concluyó que el mayor uso de smartphones, el incremento de conectividad y productividad tanto de empleados como organizaciones y la creciente demanda de información en tiempo real, es lo que está llevando a que las organizaciones hagan que sus procesos de negocio estén preparados para ser llevados a entornos móviles (MarketsandMarkets 2017).

Una de las empresas que ha apostado por el uso de las TIC, en específico por el uso de los Sistemas de Gestión Integral es la empresa transitaria de cargas con nombre comercial TRANSCARGO. Según su sitio[7] web oficial, TRANSCARGO tiene como misión prestar servicios transitarios y de operador logístico internacional para organizar eficientemente la cadena de servicios competitivos, desde el origen hasta la entrega en destino de las mercancías, personalizando los intereses de los clientes (TRANSCARGO 2017).

Actualmente el Centro de Tecnologías para la Formación (FORTES), perteneciente a la Universidad de las Ciencias Informáticas (UCI) se ha dado a la tarea desarrollar un nuevo Sistema de Gestión Integral adaptado a las necesidades de la empresa transitaría. El sistema, aún en fase de desarrollo bajo el nombre “Sistema de Gestión para Actividades Transitarias, Aduanales y Logísticas de Cargas” (SIGAX), cuenta con una limitante: no ha sido adaptado para su uso en dispositivos móviles. Si se intenta acceder al sistema desde plataformas móviles no se obtiene una correcta visualización de los contenidos, por lo que el despliegue de información es limitado y el acceso a algunas de las

[6] www.marketsandmarkets.com
[7] www.transcargo.transnet.cu

funcionalidades del sistema prácticamente imposible. No realizar la adaptación del sistema al entorno móvil trae consigo dos limitaciones fundamentales:

- **Falta de movilidad en la empresa:** la movilidad es un indicador clave para incrementar la productividad de los empleados, pues estos incrementan sus competencias en los procesos de las empresas y mejora su experiencia de forma drástica al interactuar con la información que se manejan en los procesos de forma rápida desde sus terminales móviles sin tener que depender de equipos de escritorio (IBM 2014).
- **Imposibilidad de acceder al sistema si no existe conexión de red:** este problema puede tener impacto en la efectividad de las gestiones que se llevan a cabo, pues cualquier trabajador que necesite tomar una determinada decisión, se ve imposibilitado al no tener toda la información que necesita.

Al haber detectado las limitaciones anteriores se formula el siguiente **problema a resolver:** ¿Cómo permitir el acceso a los datos del sistema SIGAX desde dispositivos móviles?

Se plantea como **objeto de estudio:** Aplicaciones móviles para los Sistemas de Gestión Integral, definiéndose como **campo de acción:** Aplicaciones móviles para el Sistema de Gestión Integral SIGAX.

Se tiene como **objetivo general**: Desarrollar una aplicación que permita el acceso a los datos del sistema SIGAX desde dispositivos móviles.

A partir del objetivo general se definen los **objetivos específicos**:

- Obtener la fundamentación teórica que permita sustentar una posible propuesta de solución.
- Realizar el análisis y diseño de la propuesta de solución.
- Implementar la solución diseñada.
- Probar la solución propuesta.

Para dar cumplimiento a los objetivos se proponen las siguientes **tareas**:

- Analizar las aplicaciones móviles desarrolladas para Sistemas de Gestión Integral.
- Identificar las tendencias, herramientas y tecnologías actuales en el desarrollo de aplicaciones móviles.
- Definir las metodologías de desarrollo, las herramientas y tecnologías a utilizar.
- Definir los requisitos del software.

- Realizar el diseño de la solución.
- Implementar las funcionalidades.
- Probar el funcionamiento de la solución.
- Analizar los resultados de las pruebas y corregir defectos en la solución.

Como **resultado** se espera obtener: Aplicación para dispositivos móviles que permita consultar datos en el sistema SIGAX.

Para el desarrollo de la aplicación fueron utilizados los siguientes métodos de investigación:

Métodos teóricos

Inductivo-Deductivo: permitió extraer las ideas fundamentales para la construcción y fundamentación teórica del trabajo de diploma. Este método fue utilizado para realizar razonamientos sobre los temas investigados y así obtener conocimientos generales y particulares del objeto de la investigación.

Analítico-Sintético: fue utilizado en la fundamentación teórica. Permitió conocer más el objeto de estudio al realizar un análisis de la teoría y la documentación, identificar conceptos, definiciones y la extracción de los elementos fundamentales relacionados con las aplicaciones para dispositivos móviles. Se utilizó, además, durante el desarrollo del sistema para el desglose de los requisitos funcionales, también para la identificación de los patrones de diseño y el análisis de los elementos del software que mejor se adapte a un patrón determinado.

Modelación: permitió reflejar la estructura, características y relaciones de la solución haciendo uso de diagramas que facilitaron los procesos de análisis y diseño.

Métodos Empíricos:

Observación: se utilizó para obtener información mediante la percepción directa del objeto de la investigación. Fue usado, además, en el diagnóstico de la problemática y para identificar posibles funcionalidades de la aplicación. Es uno de los métodos cruciales para el diseño de toda investigación.

INTRODUCCIÓN

El documento está estructurado en tres capítulos, siendo estos:

Capítulo 1: Fundamentación teórica de la investigación. Incluye un análisis y revisión del estado del arte, de aplicaciones móviles para la gestión empresarial. Se hace una descripción de las tendencias, tecnologías y herramientas utilizadas para el desarrollo de la aplicación.

Capítulo 2: Análisis y diseño. En el capítulo se describe la propuesta de solución y se explican los artefactos generados durante los procesos de análisis y diseño, de acuerdo con la metodología empleada.

Capítulo 3: Implementación y prueba*. S*e muestran los artefactos generados durante la implementación de la solución y los resultados de las pruebas realizadas.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

1.1. Introducción

En este capítulo se abordan los conceptos relacionados con la investigación, los cuales constituyen el conocimiento sobre los procesos y tecnologías asociadas al desarrollo de la solución requerida. Se exponen y explican los lenguajes de programación, metodologías, herramientas y tecnologías a utilizar en el desarrollo de la aplicación, justificando el motivo de su selección.

1.2. Conceptos asociados al dominio del problema

1.2.1. Comunicaciones inalámbricas

En un sentido amplio y general las comunicaciones inalámbricas son aquellas comunicaciones entre dispositivos (móviles o no) o que intercambian información utilizando el espectro electromagnético (Blázquez et al. 2011). Según el diccionario de la *Real Academia de Ingeniería*[8] se entiende por comunicación inalámbrica: "*la comunicación que se realiza entre dos dispositivos que no están conectados por un cable físico sino que utilizan el espectro electromagnético*" (Real Academia de Ingeniería 2017). Este término es utilizado en contraposición al término de "*comunicaciones fijas*", las cuales utilizan un medio físico, generalmente cable o fibra óptica. Las comunicaciones inalámbricas han cobrado vital importancia con el surgimiento de las WiFi[9], estándar más utilizado en la actualidad. Una vez abordado el primer concepto para la investigación es necesario analizar el concepto asociado a los dispositivos móviles, el cual podrá ser encontrado en el siguiente sub-acápite.

1.2.2. Dispositivos móviles

La compañía IBM define en su glosario a un dispositivo móvil como *"un dispositivo de cómputo portable capaz de operar sobre una red inalámbrica*" (IBM 2017). Aunque en la bibliografía fueron encontrados varios conceptos similares al anterior, el autor de la investigación considera que el concepto brindado en el libro *"Tecnología y desarrollo en dispositivos móviles"* y que es mostrado a continuación, es el que más aporta a la investigación, pues define algunas de las características fundamentales de estos dispositivos. *"Los dispositivos móviles son dispositivos de dimensiones reducidas con capacidad de procesamiento y medios para almacenar información internamente. Se asocian al uso individual de una persona, soportan comunicaciones inalámbricas y permiten la interacción mediante una pantalla y teclado"* (Blázquez et al. 2011). A partir del análisis de los conceptos anteriores el autor puede definir a

[8] http://diccionario.raing.es
[9] Es un mecanismo de conexión de dispositivos electrónicos de forma inalámbrica.

un dispositivo móvil como: un dispositivo portable con capacidad de procesamiento y almacenamiento de datos internos y que es capaz de conectarse a una red inalámbrica.

1.2.3. Aplicación móvil

El concepto de aplicación móvil puede ser bastante intuitivo. La compañía telefónica AT&T[10] la define en su glosario como "*un programa que se ejecuta en un dispositivo móvil*" (AT&T 2017). IBM por su parte ofrece un concepto similar en su glosario de términos: *"es una aplicación que ha sido diseñada para una plataforma móvil y ofrece funciones más allá que solo mostrar información estática*" (IBM 2017). En el libro *"Mobile Learning: Nuevas realidades en el aula"* de Raúl Santiago se define como: *"toda aplicación informática diseñada para ser ejecutada en teléfonos inteligentes, tablets y otros dispositivos móviles. Por lo general se encuentran disponibles a través de plataformas de distribución"* (Santiago et al. 2015). El autor de la presente investigación considera esta última definición como la más acertada. A partir de la recopilación de estos conceptos se puede definir una aplicación móvil como: un programa informático diseñado para ser ejecutado en dispositivos móviles como tablets y celulares.

1.2.4. ERP

En la bibliografía consultada fue posible encontrar varios conceptos para los sistemas ERP. Para Ronald E. McGaughey y Angappa Gunasekaran en su artículo *"Enterprise resource planning (ERP): past, present and future":* un ERP es "*un sistema de información que integra procesos de negocio, con el objetivo de crear valor y reducir los costos al hacer disponible la información conecta a las personas adecuadas en el momento adecuado, para ayudarles a tomar buenas decisiones en la gestión de recursos de manera productiva y proactiva"* (McGaughey y Gunasekaran 2009).

En el artículo libro *"ERP - Guía práctica para la selección e implantación"* del autor Luis Muñiz se define un sistema ERP como: *"un paquete de software comercial que promete la integración de toda la información que fluye a través de la compañía"* (Muñiz 2000) . Para K. Nikolopoulos y otros autores en el artículo *"Integrating industrial maintenance strategy into ERP",* estos sistemas han sido diseñados para simplificar e integrar los procesos de operación y los flujos de información dentro de una empresa (Nikolopoulos et al. 2003).

Otro concepto a tener en cuenta es el brindado en el libro *"A study of open source ERP systems"* de Vittorio Fougatsaro: *"es una aplicación de varios módulos integrados diseñados para servir y apoyar*

[10] www.att.com

múltiples funciones de negocio y tratar esas funciones como un todo, permitiendo compartir datos entre diferentes departamentos" (Fougatsaro 2009).

Un concepto más actual se puede encontrar en la Tesis de Maestría *"Estado del arte de sistemas ERP"* del autor Pablo Hernán Masoero: "los ERP son sistemas integrales, modulares y adaptables que permiten controlar los diferentes procesos de una organización" (Masoero 2014).

Al realizar la recopilación de las anteriores ideas el autor de la presente investigación puede definir a un ERP como: un software capaz de integrar los diferentes procesos de una organización y favorecer los flujos de información. El concepto antes mencionado constituye el punto de partida para comprender la función del sistema SIGAX dentro de la empresa TRANSCARGO, el cual se encuentra siendo desarrollado sobre Odoo, sistema integrado ERP de código abierto.

1.3. Estado del arte

En este acápite se hace un análisis de las principales aproximaciones existentes para la gestión de los procesos en una empresa mediante el uso de tecnologías móviles. La investigación está centrada en dos aspectos fundamentales: la integración de sistemas ERP con aplicaciones móviles y el análisis de las aplicaciones móviles desarrolladas específicamente para la integración con Odoo.

1.3.1. Aplicaciones ERP móviles

En el presente sub-acápite se abordan las características de las principales soluciones ERP móviles existentes en la actualidad. El análisis de estas soluciones constituye el punto de partida para determinar si alguna de ellas puede ser utilizada para dar solución a la problemática de la presente investigación.

Aplicación móvil para SAP Busines One

El software de gestión empresarial SAP Business One ofrece un modo asequible de gestionar toda la empresa, desde las ventas y las relaciones con clientes hasta las finanzas y las operaciones. Las empresas que utilicen SAP Business One pueden obtener acceso en tiempo real a la información sobre clientes, ventas, inventario y procesos empresariales clave. El sistema puede ser accedido desde tablets y smarthphones con el uso de una aplicación móvil desarrollada específicamente para el mismo. La aplicación se encuentra disponible para iPhone, iPad y dispositivos con sistema operativo Android, siendo gratuita solo para estos últimos. Algunas de sus principales características son (Inforges 2016):

- **Alertas y aprobaciones**: recibe alertas sobre eventos específicos, tales como desviaciones respecto de los descuentos, precios o límites de crédito aprobados.
- **Procesa solicitudes de aprobación**: genera acciones remotas y realiza análisis de datos y métricas en profundidad para ayudar a asegurar una toma de decisiones rápida y eficiente.
- **Reportes en tiempo real**: accede a informes integrados habilitados por software, que presentan información empresarial clave. Agrega reportes personalizados y comparte fácilmente los resultados por correo electrónico desde el dispositivo móvil.
- **Datos de clientes y proveedores**: consulta o localiza registros de clientes y empresas asociadas, crea nuevas entradas y asienta nuevas actividades. Todos los cambios se actualizan automáticamente en la aplicación interna de *SAP Business One.*
- **Monitorización de inventario**: comprueba los niveles de inventario y accede a información detallada sobre productos actuales, tales como procesos de compra y precios de venta, cantidades disponibles, especificaciones de productos e imágenes de artículos.

ERP móvil de Acumatica

Acumatica es el proveedor líder de sistemas ERP basados en la nube. El Sistema ERP de Acumatica permite el acceso a aplicaciones totalmente integradas y desarrolladas sobre una plataforma flexible y robusta. El acceso al sistema no solo es posible desde los navegadores web, también se han desarrollado aplicaciones para dispositivos móviles que permiten el acceso a las principales características del ERP. Estas aplicaciones, disponibles de manera gratuita brindan las siguientes funcionalidades (Acumatica 2017):

- **Acceso en tiempo real:** conexión con Acumatica en tiempo real para el acceso a la información más actualizada.
- **Sincronización automática**: los cambios hechos en la aplicación móvil son actualizados y sincronizados en Acumatica automáticamente.
- **Integración con el dispositivo móvil:** permite el acceso a los recursos del dispositivo como la cámara y el lector de huellas dactilares.

Aplicación móvil para el GID

El sistema de gestión para el sector de distribución (GID) es un Software ERP y programa de gestión de alta integración, enfocado a los sectores de ferretería, suministro industrial, centros de bricolaje, almacenes de material eléctrico, saneamientos, materiales de construcción y otros similares. Existe

actualmente una aplicación móvil desarrollada específicamente para el sistema de gestión. La aplicación GID Móvil está dirigida al ámbito comercial de la empresa a través de dispositivos móviles tablets. Dispone de todas las funcionalidades necesarias para la correcta gestión comercial de la empresa como son la realización de pedidos, consultas de precios, consultas de stocks, pendientes de recibir y servir, estadísticas de ventas, situación de riesgos, gestión de visitas y otras opciones, todas ellas realizadas en tiempo real.

1.3.2. Aplicaciones móviles para Odoo

Fue mencionado anteriormente que el sistema SIGAX se encuentra siendo desarrollado utilizando la plataforma Odoo, lo cual hace necesario el análisis de aplicaciones móviles que permitan la conexión con esta plataforma. Odoo es uno de los softwares de gestión empresarial más populares del mundo, sus funcionalidades se organizan en módulos independientes los cuales no son más que carpetas con una estructura predefinida mediante archivos XML y código en lenguaje *Phyton*. Odoo cubre las necesidades de las áreas de: contabilidad y finanzas, ventas, RRHH, compras, proyectos, almacenes, CRM y fabricación entre otras (OpenERP Spain 2010). Durante el desarrollo de la investigación fue posible encontrar diversas aplicaciones desarrolladas específicamente para la integración con Odoo las cuales se presentan a continuación.

Modoo

Modoo (Mobile Odoo) es un conjunto de aplicaciones empresariales de código abierto que brindan la posibilidad de acceder a módulos de Odoo desde dispositivos móviles. Algunos de los módulos brindados por la aplicación son (JPC Technologies 2017):

- Ventas
- Recursos humanos
- CRM
- Lista de productos
- Gestión de proyectos
- Calendario
- Tareas pendientes

Todos los datos introducidos en la aplicación se sincronizan automáticamente con Odoo haciéndolos disponibles para todos los usuarios. Brinda soporte para todas las versiones de servidores de Odoo y permite la sincronización de eventos con el calendario del dispositivo móvil.

MyOdoo

La aplicación MyOdoo es un ERP móvil disponible para Android e iOS que permite el acceso a los módulos predefinidos de Odoo. Es una aplicación modular que brinda soporte para varias cuentas, widgets[11] para gastos, sincronización con la agenda y el calendario, además de presentar un modo fuera de línea. Algunos de las características de MyOdoo son (MyOdoo 2017):

- **Mensajes:** permite acceder a los mensajes de Odoo, crear listas de tares pendientes, reenviar archivos y enviar nuevos mensajes.
- **Libreta de direcciones:** almacena la información de los clientes y permite realizar llamadas y enviar correos electrónicos.
- **Manejo de inventario:** permite visualizar artículos, escanear el código de barra con la cámara y realizar trasferencias y movimientos de inventario.
- **Notas:** permite escribir una nota desde el teléfono o tablet y acceder a la misma más tarde desde Odoo.
- **Ventas:** convierte la administración de ventas en una tarea relativamente fácil. Permite manejar presupuestos, facturas y flujos comerciales.
- **Gastos:** brinda la posibilidad de administrar gastos empresariales.

Odoo Mobile

Odoo Mobile es una aplicación móvil nativa disponible para los sistemas operativos iOS y Android. Con cerca de cincuenta mil usuarios es la aplicación más usada para Odoo. La aplicación permite el acceso a módulos desarrollados en Odoo desde cualquier dispositivo móvil. Estos módulos son accesibles desde la aplicación permitiendo consultar registros, reportes, ventas y otras tareas sobre la marcha. Brinda notificaciones que mantienen informados sobre cualquier tarea o acción que el usuario siga. Se encuentra disponible para las versiones 9.0 y 10.0 de Odoo. La idea de esta aplicación no es llevar todo el contenido de los módulos al dispositivo, sino definir algunas características específicas y obtener el mayor provecho de su uso en dispositivos móviles (Odoo 2017).

1.3.3. Resultados obtenidos sobre el análisis de las soluciones similares

Al realizar el análisis de las soluciones similares existentes para la gestión empresarial se ha podido constatar que ninguna es aplicable al sistema de gestión de la empresa TRANSCARGO. Las

[11] Pequeña aplicación que tiene como principal cometido mostrar y dar fácil acceso a algunas de las funciones frecuentemente usadas por un terminal.

aplicaciones ERP móviles analizadas se encuentran desarrolladas para lograr la integración con sistemas ERP específicos, por lo que su uso no es extensible al sistema SIGAX. En el caso de las aplicaciones móviles para Odoo, su funcionamiento está enfocado en el acceso a los módulos predefinidos del ERP (ventas, inventario, CRM, entre otros) por lo que no es posible el acceso desde estas aplicaciones a los módulos definidos en el sistema SIGAX. Todo lo anterior provoca que sea necesario la construcción de una aplicación personalizada. A pesar de no existir ninguna aplicación que dé solución al problema de la investigación fueron identificadas algunas características comunes entre las aplicaciones analizadas. Estas características serán el punto de partida para el desarrollo de la aplicación y se mencionan a continuación:

- Al igual que las aplicaciones similares analizadas, la propuesta de solución no perseguirá como objetivo llevar todo el sistema de gestión al entorno móvil, solamente sus características principales.
- La aplicación estará más centrada en la visualización de información crítica para el negocio que en realizar cambios en el sistema.
- La aplicación brindará la posibilidad de trabajar sin conexión.

1.4. Tipos de aplicaciones móviles

Para la creación de la propuesta de solución se procede a realizar un estudio de las diferentes aplicaciones móviles con el objetivo de definir las características de la aplicación que se va a desarrollar. Dentro de las plataformas de distribución de las aplicaciones móviles, se pueden encontrar tres tipos: nativas, web e híbridas.

1.4.1. Aplicaciones Nativas

Las aplicaciones nativas son aquellas desarrolladas bajo un lenguaje y entorno de desarrollo específico, lo cual permite, que su funcionamiento sea muy fluido y estable para el sistema operativo que fue creada (Pimienta 2014) .

Ventajas:

- Utilización de los recursos tantos del sistema como del hardware.
- Permite ser publicada en tiendas para su distribución.
- En su mayoría, no necesitan estar conectadas a Internet para su funcionamiento.

Desventajas:

- Solo pueden ser utilizadas por un dispositivo que cuente con el sistema para el cual fue desarrollada.
- Requiere de un costo para distribuirla en una tienda, y dependiendo el sistema, para el uso del entorno de desarrollo.
- Necesitan aprobación para ser publicadas en la plataforma.

1.4.2. Aplicaciones Web

Son aquellas desarrolladas usando lenguajes para el desarrollo web como son: HTML, CSS y JavaScript y un marco de trabajo para el desarrollo de aplicaciones web, como por ejemplo Jquery mobile, Sencha, Kendo UI, entre otros. Este tipo de aplicaciones es muy usado para brindar accesibilidad a la información desde cualquier dispositivo, sin importar el sistema operativo, puesto que solo se necesita contar con un navegador para acceder a esta (Pimienta 2014).

Ventajas:

- Pueden ser utilizadas desde cualquier dispositivo sin importar el sistema operativo.
- Puede que requiera un coste para su desarrollo, pero este puede ser mínimo en comparación con las nativas.
- No requieren de ninguna aprobación para su publicación.

Desventajas:

- No pueden ser publicadas en plataformas para su distribución.
- No utilizan los recursos del sistema ni del dispositivo de manera óptima.

1.4.3. Aplicaciones Híbridas

Como su nombre lo indica tienen un poco de cada tipo de las aplicaciones ya nombradas. Este tipo de aplicaciones se desarrolla utilizando lenguajes de desarrollo web y un marco de trabajo dedicado a la creación de aplicaciones híbridas. (Pimienta 2014)

Ventajas:

- Uso de los recursos del dispositivo y del sistema operativo.
- El costo de desarrollo puede ser menor que el de una nativa.
- Son multiplataforma.
- Permite distribución a través de las tiendas de su respectiva plataforma.

Desventaja:

- La documentación puede ser un poco escasa y desordenada.

Selección del tipo de aplicación a utilizar

Se ha decidido implementar una aplicación móvil nativa debido a las numerosas ventajas que esta aporta. Generalmente las aplicaciones nativas suelen tener una interfaz que se ajusta mejor a las dimensiones y características del dispositivo, razón por la cual la interacción suele ser más fluida y dinámica, el rendimiento de los componentes gráficos es mejor. Están diseñadas para sacar el mayor provecho a los recursos del sistema, los tiempos de respuesta suelen ser menores y al no tener que usar un navegador web los datos del usuario están mejor protegidos. Otro motivo de peso para desarrollar una aplicación nativa, constituye el hecho que estas no necesitan estar conectadas a la red en todo momento, es posible guardar los datos en el dispositivo y consultarlos cuando falle la conexión.

1.5. Análisis de los sistemas operativos móviles

Una aplicación móvil nativa puede estar desarrollada para distintos tipos de sistemas operativos, dependiendo del dispositivo móvil que se tenga disponible, de los requerimientos de la aplicación, de la preferencia del equipo de desarrollo, de los potenciales clientes o simplemente del más usado en el mercado nacional e internacional. Entender las características y arquitectura de estos sistemas, así como su lugar en el mercado de los dispositivos móviles, es un punto a tener en cuenta para la elección del sistema a utilizar para propuesta de solución.

1.5.1. Sistemas operativos móviles más usados

Hace algunos años era habitual que la compra de un dispositivo dependiera fuertemente de las características de su hardware, esa tendencia ha cambiado drásticamente en los tiempos recientes hasta relegar estas características a un segundo plano. En la actualidad juega un papel primordial las aplicaciones y los servicios que el dispositivo pueda proveer, lo que está condicionado en gran parte

por su sistema operativo. Existe gran variedad de sistemas operativos, pero se profundizará en los más utilizados actualmente. Un estudio realizado por la firma *Net Aplications*[12] muestra que el sistema Android se encuentra instalado en el 65.6% de todos los dispositivos móviles, iOS en el 27.2% y Windows Phone en el 3.3%. Esto que hace que sea necesario el análisis de las principales características de estos tres sistemas.

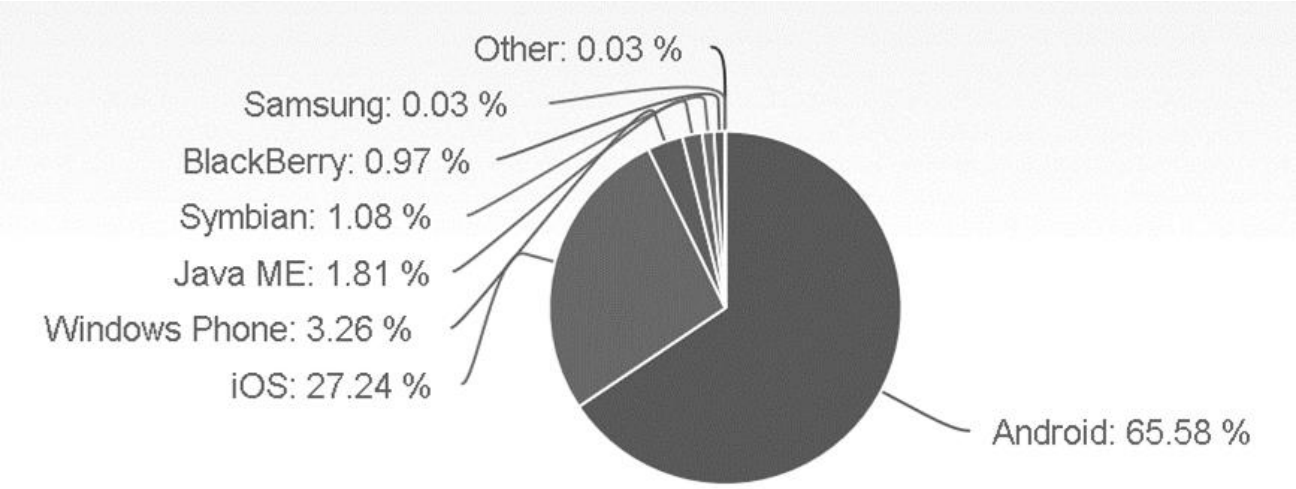


Figura 1: *Cuota de mercado de los Sistemas operativos para dispositivos móviles en 2017.*

Fuente: *www.netmarketshare.com*

Sistema operativo Android

Es un sistema operativo móvil basado en Linux, que junto con aplicaciones middleware está enfocado para ser utilizado en dispositivos móviles como teléfonos inteligentes, tablets, Google TV y otros dispositivos. Es desarrollado por la *Open Handset Alliance*[13], un consorcio de 78 compañías de hardware, software y telecomunicaciones dedicadas al desarrollo de estándares abiertos para dispositivos móviles, la cual es liderada por Google. Tiene una gran comunidad de desarrolladores escribiendo aplicaciones para extender la funcionalidad de los dispositivos. Los programas están escritos en el lenguaje de programación Java. No obstante, no es un sistema operativo libre de malware[14], aunque la mayoría de ellos son descargados de sitios de terceros (Pedrozo Petrazzini 2012).

[12] http://www.netmarketshare.com
[13] www.openhandsetalliance.com
[14] Es un tipo de software que tiene como objetivo infiltrarse o dañar una computadora o sistema de información sin el consentimiento de su propietario.

Sistema operativo iOS

iOS (anteriormente denominado iPhone OS) es un sistema operativo móvil de Apple. Originalmente desarrollado para el iPhone, siendo después usado en dispositivos como el iPod Touch, iPad y el Apple TV. La interfaz de usuario de iOS está basada en el concepto de manipulación directa, usando gestos multitáctiles. Los elementos de control contienen deslizadores, interruptores y botones. La respuesta a las órdenes del usuario es inmediata y provee una interfaz fluida. La interacción con el sistema operativo incluye gestos como deslices, toques, pellizcos, los cuales tienen definiciones diferentes dependiendo del contexto de la interfaz. Se utilizan acelerómetros internos para hacer que algunas aplicaciones respondan a sacudir el dispositivo (por ejemplo, para el comando deshacer) o rotarlo en tres dimensiones (un resultado común es cambiar de modo vertical al apaisado u horizontal). iOS se deriva de Mac OS X, que a su vez está basado en Darwin BSD y por lo tanto es un sistema operativo Unix (Pedrozo Petrazzini 2012).

Sistema operativo Windows Phone

Windows Phone es un sistema operativo móvil desarrollado por Microsoft, como sucesor de la plataforma Windows Mobile. A diferencia de su predecesor, está enfocado en el mercado de consumo generalista en lugar del mercado empresarial (Olivera, 2013). Este sistema está diseñado para ser estéticamente similar a las versiones de escritorio de Windows, brindando la posibilidad de utilizar las herramientas de Office Mobile, Outlook Mobile e Internet Explorer. Como principal desventaja de su uso está la poca disponibilidad de aplicaciones.

Selección del sistema operativo a utilizar

Para la realización de la propuesta de solución se ha decidido utilizar el sistema operativo Android. Esta decisión responde a la política de utilización de software libre en la que se encuentra enmarcada Cuba y la UCI, en específico. Es un sistema de código abierto, robusto y con miles de desarrolladores y aplicaciones en el mundo. Android domina la cuota de mercado de los smartphones con un 86.8%. Samsung su contribuyente número uno, permanece en el lugar más alto en ventas (IDC 2017). Todo lo anteriormente mencionado coloca a Android como la mejor opción para el desarrollo del proyecto.

1.6. Análisis del sistema operativo Android

Una vez seleccionado el sistema operativo Android como plataforma en la que se ejecutará la aplicación, es importante que la presente investigación aborde temas relacionados con sus principales

características y arquitectura, conocimiento que sentará las bases para el posterior desarrollo de la propuesta de solución.

Características

Según el portal Academia Android las principales características de Android son (Academia Android 2017):

- **Marco de trabajo de aplicaciones**: permite el reemplazo y la reutilización de los componentes.
- **Navegador integrado**: basado en los motores *Open Source Webkit*.
- **SQLite**: base de datos para almacenamiento estructurado que se integra directamente con las aplicaciones.
- **Multimedia**: soporte para medios con formatos comunes de audio, video e imágenes planas (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF).
- **Máquina virtual Dalvik**: base de llamadas de instancias muy similar a Java.

Arquitectura

A continuación, se detallarán los diferentes componentes principales de la arquitectura de Android:

Aplicaciones: las aplicaciones base incluirán un cliente de email, programa de SMS, calendario, navegador y otros. Todas las aplicaciones escritas en Java.

Marco de trabajo: los desarrolladores tienen acceso completo a las APIs[15] del marco de trabajo usado por las aplicaciones base. La arquitectura está diseñada para simplificar la reutilización de componentes.

Librerías: incluye un conjunto de librerías desarrolladas en C y C++ algunas de las cuales no son desarrolladas por Google o por el proyecto Android.

Runtime de Android: cada aplicación Android ejecuta su propio proceso, con su propia instancia de la máquina virtual Dalvik que ejecuta archivos en el formato Ejecutables Dalvik (*Dalvik Executable dex*), optimizado para memoria mínima.

[15] Son un conjunto de funciones y procedimientos que cumplen una o muchas funciones con el fin de ser utilizadas por otro software.

Núcleo - Linux: depende en la última de sus versiones, del núcleo 3.0.31 de Linux para los servicios base del sistema como seguridad, gestión de memoria, gestión de procesos, pila de red y modelo de drivers. El núcleo también actúa como una capa de abstracción entre el hardware y el resto de la pila.

1.7. Metodología de desarrollo

Una metodología es una colección de procedimientos, técnicas, herramientas y documentos auxiliares que ayudan a los desarrolladores de software en sus esfuerzos por implementar nuevos sistemas de información. Una metodología está formada por fases, cada una de las cuales se puede dividir en sub-fases, que guiarán a los desarrolladores de sistemas a elegir las técnicas más apropiadas en cada momento del proyecto y también a planificarlo, gestionarlo, controlarlo y evaluarlo (Avison y Fitzgerald 2003).

La selección y uso de la metodología adecuada es uno de los puntos vitales para el éxito de un proyecto. Esta dicta qué hacer y cómo hacerlo, es el cimiento que sostendrá todo el proceso de desarrollo posterior, por lo que resulta crítico tener buen juicio al seleccionar la que mejor se adapte a las características del proyecto en cuestión. Las metodologías de desarrollo de software suelen clasificarse según sus características fundamentales y el enfoque utilizado. Hasta el momento han sido identificados dos grandes grupos: metodologías tradicionales o prescriptivas y metodologías ágiles o ligeras.

1.7.1. Metodologías tradicionales

Las metodologías tradicionales buscan imponer disciplina, estructura, orden y consistencia en el proyecto de desarrollo de software. Se basan fundamentalmente en la planificación total, extensa documentación, y una definición rigurosa de actividades, artefactos y roles. Prescriben un conjunto de elementos del proceso como tareas, acciones y mecanismos de control, definiendo cómo estos han de relacionarse. Presentan dificultades para adaptarse al cambio por lo que no deben utilizarse en proyectos dónde los requisitos cambien o sean difíciles de predecir (Boehm 2002).

1.7.2. Metodologías ágiles

El enfoque ágil se tiene como objetivo reducir el tiempo de desarrollo al centrarse en la comunicación entre los miembros del equipo y buscando siempre la colaboración con el cliente más que la negociación de contratos. Estas metodologías buscan manejar los cambios y realizar entregas rápidas de software funcional dejando la documentación, formalidad y planeación exhaustiva del proceso en

segundo plano (Boehm 2002). Las diferencias fundamentales entre las metodologías ágiles y las tradicionales aparecen reflejadas en la tabla que se muestra a continuación.

Tabla 1: *Bases de las metodologías ágiles y las metodologías tradicionales.*

Fuente: *Adaptado de (Boehm 2002).*

Área	Metodología ágil	Metodología tradicional
Desarrolladores	Colaborativos, unidos, ágiles y entendidos.	Orientados al plan, habilidades adecuadas acceso a conocimiento externo.
Clientes	Son representativos y se les entrega poder.	Mezcla de niveles de aptitud.
Requerimientos	En gran parte emergentes y con rápidos cambios.	Conocibles tempranamente y bastante estables.
Arquitectura	Diseñada para los requerimientos actuales.	Diseñada para los requerimientos actuales y los del futuro próximo.
Refactorización	Económica.	Costosa.
Tamaño	Productos y equipos pequeños.	Productos y equipos grandes.
Objetivo primario	Valor rápido.	Alta seguridad.

Metodologías ágiles en el desarrollo móvil

Las metodologías ágiles presentan ciertas características que las hacen aplicables al desarrollo de aplicaciones móviles. En el artículo científico *"Keynote: Mobile software development–the business opportunity of today*" del autor Pekka Abrahamsson, se realiza un estudio de las características de las metodologías ágiles y sus puntos en común con el software móvil. Este estudio se centra en varios aspectos: identificación del cliente, tamaño del equipo de desarrollo, documentación, planificación y orientación a objetos. Como conclusión se plantea que *"la gran mayoría de las características de las metodologías ágiles resultan adecuadas para el desarrollo aplicaciones móviles"* (Abrahamsson 2005). La tabla mostrada a continuación realiza un mapeo de las características ágiles ideales, su razonamiento y respectiva inclusión en el desarrollo de software para móviles.

Tabla 2: *Comparación entre las características ágiles y los rasgos observados en el desarrollo de software móvil.*

Fuente: *Adaptado de (Abrahamsson 2005).*

Características ágiles ideales	Razonamiento	Software móvil
Alta volatilidad del entorno	Debido a la alta frecuencia en el cambio de los requerimientos, se tiene menos necesidad de diseño y planificación inicial y mayor necesidad de desarrollos incrementales e iterativos.	Alta incertidumbre, entornos dinámicos, cientos de nuevos terminales cada año.
Equipos de desarrollo pequeños	Equipos pequeños son capaces de reaccionar más rápido y compartir información, se necesita menos documentación.	La mayoría de los proyectos del software móvil se lleva a cabo en microempresas, PyME[16] o equipos de desarrollo.
Cliente identificable	Para evitar malentendidos.	Potencialmente, hay un número ilimitado de usuarios finales, pero los clientes son fáciles de identificar.
Entornos de desarrollo orientados a objetos	La mayoría de las herramientas de desarrollo ágil existen bajo plataformas orientadas a objetos.	Algunos problemas en herramientas como refactorizaciones o primeros test.
Software crítico no asegurado	Los fallos no causan pérdida de vidas. Se puede buscar mayor agilidad en el desarrollo.	La mayoría del software es para entretenimiento. Las terminales no son fiables.
Sistemas pequeños	Menos necesidad de diseño inicial.	Las aplicaciones, aunque variables en tamaño, no suelen superar las 10.000 líneas de código.
Ciclos de desarrollo cortos	Propósito de retroalimentación rápida.	Periodos de desarrollo de 1 a 6 meses generalmente.

[16] Pequeñas y medianas empresas.

1.7.3. Metodologías específicas para el desarrollo móvil

Durante el desarrollo de la investigación fueron identificadas tres metodologías enfocadas específicamente al desarrollo de aplicaciones móviles. Es objetivo del presente sub-acápite definir sus características fundamentales y determinar si alguna de ellas es adecuada para el desarrollo de la propuesta de solución.

Mobile-D

Según Pekka Abrahamsson y otros autores en el artículo *"Mobile-D: an agile approach for mobile application development"*, Mobile-D está basada en las metodologías *eXtreme Programming* (XP), *Crystal methodologies* y *Rational Unified Process* (RUP) (Abrahamsson et al. 2004). Los creadores de Mobile-D plantean que su metodología está pensada para grupos de no más de diez desarrolladores trabajando en el mismo espacio físico. Estos recalcan la necesidad de realizar un ciclo de desarrollo rápido en equipos muy pequeños, lanzando productos completamente funcionales en un plazo de diez semanas. La siguiente figura muestra el ciclo de desarrollo propuesto.

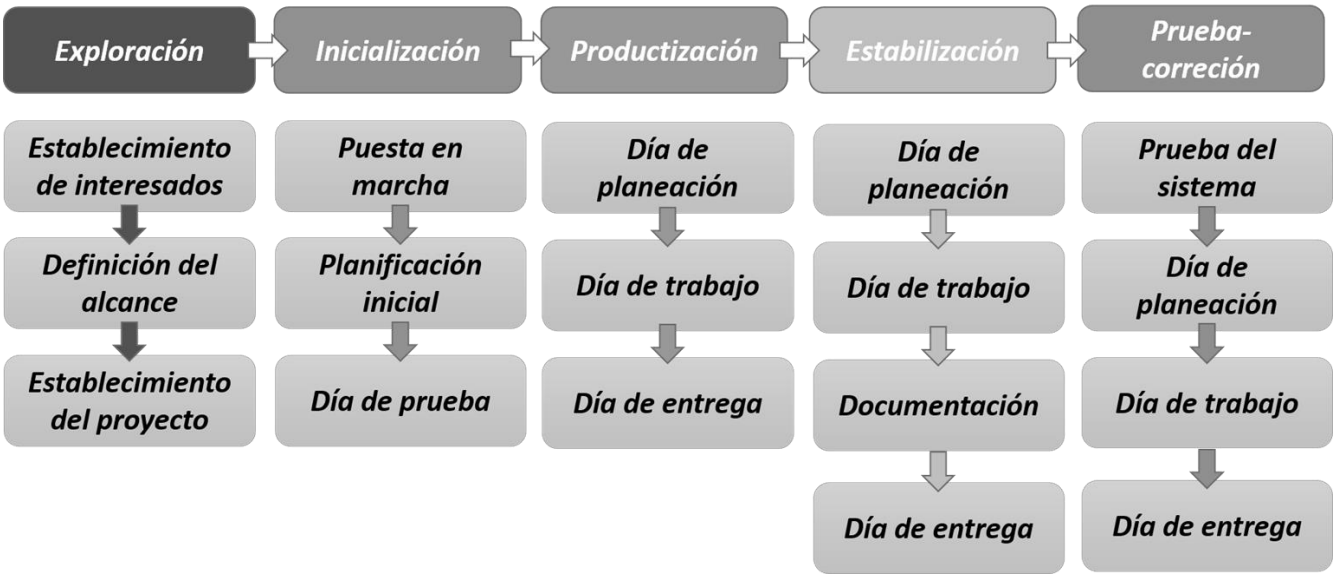


Figura 2: *Ciclo de desarrollo Mobile-D.*

Fuente: *Sitio[17] oficial de Mobile-D.*

Esta metodología ha sido utilizada en numerosos proyectos con clientes reales durante años, lo cual constituye una ventaja significativa frente a otras metodologías. Según Henrik Hedberg y Juha Iisakka

[17] http://agile.vtt.fi

en el artículo *"Technical Reviews in Agile Development: Case Mobile-D",* los ciclos de desarrollo se han actualizado y mejorado a partir de la experiencia obtenida (Hedberg y Iisakka 2006).

HMD

Otra metodología muy importante en la actualidad es HMD (*Hybrid Methodology Design o Diseño de Metodología Híbrida),* esta es una combinación del desarrollo adaptativo de software (*Adaptive Software Development*, ASD) y el diseño de nuevos productos (*New Product Development*, NPD). Parte de un ciclo de vida tradicional (análisis, diseño, implementación, pruebas y transición), incluyendo al final una fase de comercialización (Balaguera 2015).

En la primera iteración se divide la fase de análisis con la intención de mitigar riesgos de desarrollo; de la misma forma, el diseño también se segmenta para introducir algo de diseño basado en arquitectura. La implementación y las pruebas sin embargo se fusionan introduciendo conceptos de desarrollo orientado a pruebas (*Test-Driven Development, TDD*) (Blanco et al. 2009). Según Yohn Daniel Balaguera en el artículo *"Metodologías ágiles en el desarrollo de aplicaciones para dispositivos móviles. Estado actual",* esta metodología utiliza el modelo iterativo incremental para el proceso de desarrollo y así lograr la rápida entrega de software y mejorar las capacidades de gestión de riesgos (Balaguera 2015).

Proceso de desarrollo móvil en espiral

Es la más reciente propuesta de metodologías diseñadas específicamente para aplicaciones móviles y que aún se encuentra en etapa experimental. Esta propuesta metodológica utiliza el modelo de desarrollo en espiral como base e incorpora procesos de evaluación de la usabilidad. Prioriza la participación del usuario en todos los procesos del ciclo de vida de diseño, con el fin de garantizar un diseño centrado en el usuario, aun cuando se trata de un modelo de proceso orientado a proyectos grandes y costosos, ya que está destinado a ser un modelo de reducción de riesgos. (Nosseir et al. 2012)

El proceso permite a los desarrolladores de aplicaciones móviles, detallar los criterios de usabilidad de la aplicación, el primer paso es identificar a los usuarios, las tareas y los contextos en los que se utilizará la aplicación móvil. El siguiente paso es dar prioridad a los atributos de usabilidad, identificar qué atributos son los más importantes para la aplicación, y para cada uno definir un conjunto de métricas para verificar el grado en que se cumplen en la aplicación final. El modelo en espiral tiene varias iteraciones, cada una producirá prototipos que ayudarán la evaluación del usuario (Nosseir et al. 2012).

Selección de la metodología a utilizar

Después de un análisis de los temas relacionados al uso de las metodologías de desarrollo para aplicaciones móviles se ha seleccionado a Mobile-D para la realización de la propuesta de solución. Esta selección se ha basado en los siguientes criterios:

- Toma las mejores prácticas de las metodologías XP, Crystal y RUP adaptándolas al desarrollo de aplicaciones móviles.
- Resulta ideal para proyectos con ciclos de desarrollo cortos y equipos pequeños.
- Se establece una comunicación más fluida con el cliente el cual participa activamente en el proceso de desarrollo de software.
- Requiere una documentación menos rigurosa, propiciando mayor dinamismo.
- Ha sido utilizada en numerosos proyectos con clientes reales durante años, obteniendo buenos resultados.
- Las tareas a realizar en cada fase están bien definidas.

1.8. Herramientas y tecnologías a utilizar

A fin de garantizar a lo largo de todo el proceso de desarrollo, que el sistema cumpla con las características que fueron planteados a desarrollar, se utilizaron un conjunto de técnicas y herramientas, la cuales se describen a continuación.

1.8.1. Lenguaje de modelado UML

El Lenguaje Unificado de Modelado (UML) es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas. Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios. UML pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar. UML incluye conceptos semánticos, notación, y principios generales. Tiene partes estáticas, dinámicas, de entorno y organizativas. Está diseñado para ser utilizado en herramientas interactivas de modelado visual que tengan generadores de código, así como generadores de informes. Pretende dar apoyo a la mayoría de los procesos de desarrollo orientados a objetos. (Rumbaugh et al. 2000)

1.8.2. Herramienta de modelado

Visual Paradigm: es una herramienta CASE[18] que soporta el ciclo de vida completo del desarrollo del software: análisis, diseño orientado a objetos, construcción, pruebas y despliegue. Esta herramienta provee el modelado de procesos de negocios, además de un generador de mapeo de objetos-relacionales para los lenguajes de programación Java, .NET y PHP. Permite dibujar todos los tipos de diagramas de clases, generar código desde diagramas y generar documentación. Es una herramienta multiplataforma que contribuye a una rápida construcción de aplicaciones de calidad y cuenta con la posibilidad de realizar un control de versiones durante el ciclo de trabajo. (Rondón, Domínguez, y Berenguer 2011)

1.8.3. Lenguaje de Programación

Un lenguaje de programación es un lenguaje formal diseñado para expresar procesos que pueden ser llevados a cabo por computadoras. Son utilizados para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. Está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones (Sebesta y Mukherjee 2002).

Selección del lenguaje de programación a utilizar

Para el desarrollo de aplicaciones Android se pueden emplear diversos lenguajes de programación entre ellos Java, C, C++, Visual Basic y .NET. Para la implementación de la propuesta de solución se ha decidido utilizar el lenguaje Java 8 debido a las grandes potencialidades que posee:

- Es un lenguaje orientado a objetos, interpretado, multiplataforma, multihilo, de gran rendimiento y seguridad.
- Realiza numerosas comprobaciones en compilación y en tiempo de ejecución. Libera a los programadores de la responsabilidad de realizar la liberación de memoria explícitamente.
- Posee una extensa documentación física y digital, además de varias comunidades de desarrolladores.
- Cuenta con numerosas librerías nativas y desarrolladas por terceros que amplían sus funcionalidades.

[18] *Computer Aided Software Engineering* o Ingeniería de Software Asistida por Computadora. Son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software.

- Posee una curva de aprendizaje muy rápida.
- Es el lenguaje nativo y más utilizado para el desarrollo Android.

1.8.4. Entorno de desarrollo integrado

Un entorno de desarrollo integrado, llamado también IDE (sigla en inglés de *Integrated Development Environment*), es un programa informático compuesto por un conjunto de herramientas de programación. Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación; es decir, que consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes (Carreño 2017).

El Android SDK brinda las herramientas necesarias para construir, probar y depurar aplicaciones para Android. Mediante el mismo, Google ofrece un paquete completo que incluye emulador, depurador, documentación y el código de muchas aplicaciones de ejemplo; dotando al desarrollador de todo lo necesario para la creación de aplicaciones. Este paquete de desarrollo no funciona por sí solo, ya que es necesario ser importado por un IDE.

Eclipse

Eclipse es un IDE, de código abierto y gratuito, cuyo diseño sigue un patrón de actualización basado en plugins[19], su objetivo es convertirse en una plataforma de integración de herramientas de desarrollo. Utiliza el plugin *Android Developer Tools* (ADT) con el objetivo de integrar el Android SDK para utilizar sus herramientas y librerías adicionales.

Se puede destacar de Eclipse los siguientes aspectos (Academia Android 2014):

- **Gestión de proyectos:** El desarrollo en este IDE se basa en proyectos, que son un conjunto de recursos relacionados entre sí, como puede ser el código fuente, documentación, ficheros, entre otros.
- **Depurador de código**: Tiene un depurador de código, fácil e intuitivo proporcionando de forma gráfica una opción de mejora en los proyectos. Dispone de una perspectiva dedicada a la depuración donde se realiza y supervisa dicha tarea.
- **Perspectivas, editores y vistas:** El concepto de trabajo se basa en las perspectivas, que es una pre-configuración de ventanas y editores que permiten trabajar en un determinado entorno

[19] Es una aplicación (o programa informático) que se relaciona con otra para agregarle una función nueva y generalmente muy específica.

de trabajo de forma óptima proporcionando puntos de vista y acciones visibles, así como la entrega de los mecanismos para la interacción con los recursos, multitarea y filtrado de información.

- **Colección de plugins:** Están disponibles una gran cantidad de plugins, tanto desarrollados por Eclipse como por terceros. Actualmente el número de ellos es alto, sobrepasando los 1.000, aumentando las funcionalidades del IDE.

Android Studio

Android Studio está basado en IntelliJ IDEA de la compañía JetBrains, que proporciona varias mejoras con respecto al plugin ADT para Eclipse. Fue presentado por Google el 16 de mayo del 2013, tiene como objetivo crear un entorno dedicado exclusivamente a la programación de aplicaciones para dispositivos con sistema operativo Android, proporcionando un mayor control sobre el proceso de producción. Se trata de una alternativa real a Eclipse, el IDE recomendado por Google hasta la fecha, pero que presentaba problemas debido a su lentitud en el desarrollo de versiones que solucionaran las carencias actuales. Está programado en Java, es multiplataforma y es publicado de forma gratuita a través de la licencia Apache 2.0, por lo que es posible usarlo tanto para proyectos personales como comerciales.

Principales características de Android Studio (Academia Android 2014):

- Nueva interfaz específica para el desarrollo en Android.
- Editor de diseño que muestra una vista previa de los cambios realizados directamente en el archivo xml.
- Herramientas Lint (detecta código no compatible entre arquitecturas diferentes o código confuso que no es capaz de controlar el compilador) para detectar problemas de rendimiento, usabilidad y compatibilidad de versiones.
- Utiliza *ProGuard* para optimizar y reducir el código del proyecto al exportar a APK (útil para dispositivos de gama baja con limitaciones de memoria interna).
- Integración de la herramienta Gradle encargada de gestionar y automatizar la construcción de proyectos, como pueden ser las tareas de prueba, compilación o empaquetado.
- Nuevo diseño del editor con soporte para la edición de temas.
- Posibilita el control de versiones accediendo a un repositorio como Mercurial, Git, Github o *Subversion*.

- Alertas en tiempo real de errores sintácticos, compatibilidad o rendimiento antes de compilar la aplicación.
- Vista previa en diferentes dispositivos y resoluciones.

Tabla 3: *Comparación entre Android Studio y ADT Eclipse.*

Fuente: *(Academia Android 2017).*

Características	Android Studio	ADT Eclipse
Licencia	Apache 2.0	Licencia Pública Eclipse *(*EPL)
Sistema de construcción	Gradle	ANT
Construcción y gestión de proyectos basado en *Maven*[20]	Si	No (es necesario instalar un plugin auxiliar)
Soporte para *NDK* [21]	Si	Si
Refactorización y completado avanzado de código Android	Si	No
Diseño del editor gráfico	Si	No
Firma APK y gestión de almacén de claves	Si	No
Vista en tiempo real de renderizado de layouts	Si	No
Nuevos módulos en proyecto	Si	No
Editor de navegación	Si	No
Datos de ejemplo en diseño de *layout* (sin renderizar en tiempo de ejecución)	Si	No
Visualización de recursos desde editor de código	Si	No

[20] Herramienta de *software* para la gestión y construcción de proyectos Java, similar a Apache ANT, pero su modelo es más simple ya que está basado en XML.
[21] *Native Development Kit*: herramientas para implementar código nativo escrito en C y C++.

Selección del Entorno de Desarrollo Integrado a utilizar

Se ha decidido utilizar el IDE Android Studio ya que es el entorno oficial para desarrollar aplicaciones Android. Es creado por Google, quien le brinda soporte y actualiza constantemente. La anterior tabla, adaptada de (Academia Android 2014) permite observar que este IDE posee varias características que lo hacen más amigable para el programador:

- Código más ordenado y estructurado. Brinda opciones de completamiento de código y sugerencias.
- Facilidad para crear interfaces.
- Intuitivo y fácil de usar.
- Plantillas para empezar proyectos.
- Proyecto menos engorroso.

1.8.5. Framework de desarrollo

Un framework de desarrollo de software es una arquitectura para aplicaciones que provee una estructura estándar y funcionalidades generales extensibles. Los framewoks habilitan y simplifican la implementación de tecnologías complejas para el desarrollo de la aplicación, también brindan un conjunto de librerías y modelos que definen las relaciones entre clases (IBM 2017).

Para el desarrollo de la propuesta de solución se utilizará Odoo Mobile como framework de desarrollo. Odoo Mobile es un framework de desarrollo de aplicaciones móviles de código libre que permite la integración con Odoo. Con su ayuda se puede desarrollar casi cualquier aplicación soportada por Odoo en el menor tiempo posible. Contiene un ORM (Object-Relational Mapping o Mapeo Objeto-Relacional) para manejar su propia base de datos móvil local, no necesario preocuparse por los datos provenientes de Odoo. Tiene servicios pre-definidos para hacer que los datos de la aplicación se sincronicen con el servidor de Odoo.

El framework está compuesto básicamente por cuatro capas, la conexión con el servidor de Odoo se realiza a través de una librería interna definida para este propósito. A continuación, se dará una breve descripción de cada capa y sus funciones.

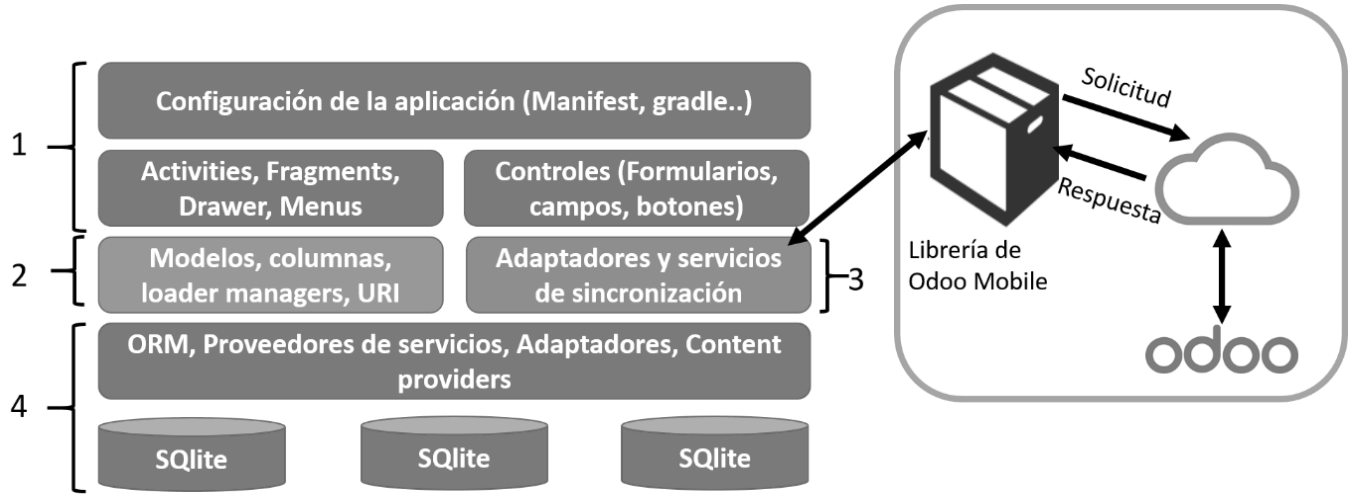


***Figura 3**: Arquitectura del framework Odoo Mobile.*

***Fuente:** (Odoo Mobile 2017).*

Primera capa

Es la parte completamente personalizable. Incluye la configuración de la aplicación, *Android Manifest*, configuración del *Gradle*, registro de activities, fragments, navigation drawer y otros componentes visuales.

Una activitiy o actividad es un componente de la aplicación que provee una vista con la cual los usuarios pueden interactuar para realizar alguna acción. A cada actividad se le asigna una ventana en la que dibujar la interfaz de usuario. Esta actividad usualmente ocupa toda la pantalla.

Un fragment o fragmento representa el comportamiento de una porción de la interfaz de usuario en la actividad. Se pueden combinar varios fragmentos en una sola actividad para construir una interfaz de usuario con varios paneles. También se pueden reutilizar estos fragmentos en varias actividades.

El navigation drawer o menú lateral es un panel que transita desde el borde izquierdo de la pantalla y muestra las principales opciones de la aplicación. El framework provee fácil manejo de este menú lateral.

Segunda capa

Contiene todos los modelos (tablas de la base de datos) para la arquitectura de los addons, lógica de negocio y código para las activities y los fragments. Contiene, además, adaptadores para los listviews.

Tercera capa

Es la responsable de obtener datos desde el servidor Odoo y proveerla para el adaptador de la sincronización con el framework. Odoo Mobile se hará cargo de los datos offline, comparará los últimos datos, realizará la sincronización y actualizará los datos locales y en el servidor. El framework trata de brindar cien por ciento de soporte sin conexión, pero en algunos casos es necesario crear un mecanismo personalizado de sincronización para la aplicación.

Cuarta capa

Contiene todos los componentes principales usados por el framework: ORM, adaptadores para la sincronización, proveedores de servicios y proveedores del contenido de los modelos, además de otras clases útiles para la aplicación. Esta capa también es responsable de manejar cuentas de usuarios, opciones de sincronización. Es en parte dependiente de la tercera capa.

1.9. Conclusiones parciales

Luego de ser consultada la bibliografía especializada y obtener un marco teórico y estado del arte para la investigación, se concluye lo siguiente:

- El análisis del estado del arte concluyó que no existe ninguna solución aplicable al problema de la investigación, sin embargo, las aplicaciones ERP móviles tratadas poseen funcionalidades comunes que serán tomadas en cuenta para la construcción de la aplicación.
- El estudio de los diferentes tipos de aplicaciones móviles permitió identificar al desarrollo de una aplicación nativa, como la mejor opción para lograr un buen diseño de la propuesta de solución.
- El estudio de las metodologías de desarrollo de software demostró que el uso de la metodología Mobile-D resulta ideal para guiar el proceso de desarrollo de la aplicación.
- El análisis de las herramientas y lenguajes utilizados para el desarrollo de aplicaciones móviles evidenció la factibilidad del IDE Android Studio y el lenguaje Java para el desarrollo de la propuesta de solución.
- El framework de desarrollo Odoo Mobile demostró ser la mejor opción para la implementación de la aplicación móvil, por lo que será utilizado para lograr una correcta sincronización entre el sistema SIGAX y la propuesta de solución.

CAPÍTULO 2. ANÁLISIS Y DISEÑO

2.1. Introducción

En este capítulo se describe la propuesta de solución y las actividades desarrolladas durante sus procesos de análisis y diseño. Para lograr ese objetivo se describen los requisitos funcionales y no funcionales identificados, la arquitectura del sistema y los patrones de diseño utilizados, además, se generarán los artefactos definidos por la metodología de desarrollo escogida.

2.2. Descripción del sistema

La propuesta de solución tiene como objetivo acceder a los datos del sistema SIGAX a través de los servicios web suministrados por Odoo. La aplicación permitirá el acceso a las principales funcionalidades del sistema gestor desde teléfonos inteligentes o tablets con sistema operativo Android 4.0 o superior. La aplicación podrá realizar la sincronización con el sistema SIGAX cuando el dispositivo esté conectado a una red inalámbrica, guardando los datos del usuario y las entidades de la base de datos dentro del propio dispositivo. El usuario podrá definir las entidades a sincronizar, la cantidad de datos y el tiempo para la sincronización automática. Permitirá, además, la utilización de múltiples cuentas de usuario.

2.3. Fase de exploración

La fase de exploración es la primera fase definida por la metodología Mobile-D, su objetivo es la planeación y el establecimiento del incipiente proyecto. Es importante para establecer las bases de la implementación controlada del software, resolver problemas relacionados con la arquitectura del producto, el proceso de desarrollo y la selección del entorno. Esto se realiza en tres etapas: establecimiento de los interesados, definición del alcance y el establecimiento del proyecto (AGILE 2017).

- **Establecimiento de los interesados:** el propósito de esta etapa es identificar y establecer el grupo de interesados que se necesita para desarrollar las tareas de la fase de exploración, así como el resto del proceso de desarrollo de software. En esta etapa fueron definidos los involucrados del proyecto y se identificó sus tareas, roles y responsabilidades. Se identificó como cliente al equipo de desarrollo del sistema SIGAX y como usuarios a todos los usuarios registrados en el sistema gestor. En un encuentro con el cliente se definió la propuesta del producto: una aplicación móvil para el sistema operativo Android.

- **Definición del alcance:** el propósito de esta etapa es la definición de las metas para el incipiente proyecto. Para ese propósito durante la etapa de definición del alcance fueron identificados los requisitos funcionales y no funcionales de la aplicación, realizando un análisis preliminar de los mismos y asignando niveles de importancia.
- **Establecimiento de proyecto:** el propósito de esta etapa es definir y asignar los recursos (tanto técnicos como humanos) que se necesitan para que el proyecto de desarrollo de software inicie. También se establece la línea de base para el proyecto, las necesidades de formación para el equipo y se define la documentación que guiará el proceso. Esta etapa asegura que el proyecto pueda iniciar sin demoras causadas por falta de herramientas o entrenamiento. Para cumplir con estas actividades se definió el entorno técnico y físico del proyecto, se identificaron las necesidades de formación (principalmente concernientes al uso de Mobile-D y el desarrollo de aplicaciones Android) y se definió la línea de arquitectura para la aplicación.

2.3.1. Requisitos funcionales

Un requisito funcional es la declaración de una funcionalidad o servicio que un sistema debe brindar, estableciendo su comportamiento ante determinadas situaciones o entradas suministradas. A continuación, se listan los requisitos funcionales identificados para el desarrollo de la propuesta de solución, estos fueron obtenidos en una reunión con el cliente. A cada requisito se le asignó un nivel de importancia o prioridad entre 1 (poco importante) y 5 (muy importante).

Tabla 4: *Requisitos funcionales de la aplicación.*

Funcionalidad		Importancia
1	**Autenticar usuario**	**5**
2	**Listar contratos**	**5**
3	**Ver detalles de un contrato**	**5**
4	**Realizar búsqueda rápida de contratos**	**2**
5	**Filtrar contratos**	**3**
6	**Listar solicitudes de embarque**	**5**
7	**Ver detalles de una solicitud de embarque**	**5**
8	**Realizar búsqueda rápida de solicitudes de embarque**	**2**
9	**Filtrar solicitudes de embarque**	**3**
10	**Listar expedientes**	**5**
11	**Ver detalles de un expediente**	**5**
12	**Realizar búsqueda rápida de expedientes**	**2**
13	**Filtrar expedientes**	**3**
14	**Listar autorizos sin corresponsal**	**5**
15	**Ver detalles de un autorizo sin corresponsal**	**5**
16	**Realizar búsqueda rápida de autorizos sin corresponsal**	**2**
17	**Filtrar autorizos sin corresponsal**	**3**

18	**Listar avisos de confirmación**	**5**
19	**Ver detalles de un aviso de confirmación**	**5**
20	**Realizar búsqueda rápida de avisos de confirmación**	**2**
21	**Filtrar avisos de confirmación**	**3**

2.3.2. Requisitos no funcionales

Según Ian Sommerville en el libro *"Ingeniería de Software"* los requisitos no funcionales son restricciones en los servicios o funciones ofrecidas por el sistema. Estos incluyen requisitos de tiempo, restricciones en el proceso de desarrollo y estándares. A menudo se aplican al sistema como un todo (Sommerville 2010). A continuación, se presentan los requerimientos no funcionales identificados para la propuesta de solución.

Usabilidad

- **RnF1.** La aplicación será distribuida en idioma español.

Hardware:

- **RnF4.** Será necesario disponer de un dispositivo móvil, ya sea tablet o celular, con al menos 4.0 pulgadas de pantalla.
- **RnF5.** El dispositivo móvil que se disponga debe tener 50 megabytes de espacio disponible para la instalación de la aplicación y 100 megabytes para datos.

Software:

- **RnF6.** El dispositivo móvil que se disponga debe tener el sistema operativo Android versión 4.0 o superior.
- **RnF7.** La aplicación tendrá una orientación completamente vertical.

2.3.3. Modelo de dominio

El modelo de dominio es una representación visual de los conceptos u objetos del mundo real significativos para un problema o área de interés. Representa clases conceptuales del dominio del problema, conceptos del mundo real, no de componentes de software (Larman 1999).

El modelo del dominio captura, comprende y describe los objetos más importantes dentro del contexto de un sistema. Los objetos del dominio o clases pueden obtenerse a partir de una especificación de

requisitos o mediante la entrevista con los expertos del tema. Las clases que lo componen suelen aparecer en tres formas típicas (Jacobson et al. 2000):

- Objetos del negocio que representan los elementos que se manipulan en el negocio.
- Objetos del mundo real y conceptos de los que el sistema debe hacer un seguimiento.
- Sucesos que ocurrirán o que han ocurrido.

A continuación, se presenta el modelo de dominio de la aplicación, el cual constituye una representación gráfica de los conceptos fundamentales a tener en cuenta para su desarrollo.

Definición de las clases

Sistema SIGAX: Sistema de Gestión Integral desarrollado por el centro FORTES para el Agente Transitario de Cargas TRANSCARGO.

Usuario: persona registrada en el sistema SIGAX.

Servicios web: hace referencia a los servicios que se utilizarán para lograr la conexión.

Aplicación móvil: aplicación desarrollada por el autor de este trabajo para el sistema de gestión.

Servicios web: hace referencia a los servicios web suministrados por el sistema SIGAX.

Datos: hace referencia al conjunto de datos manejados por el sistema gestor.

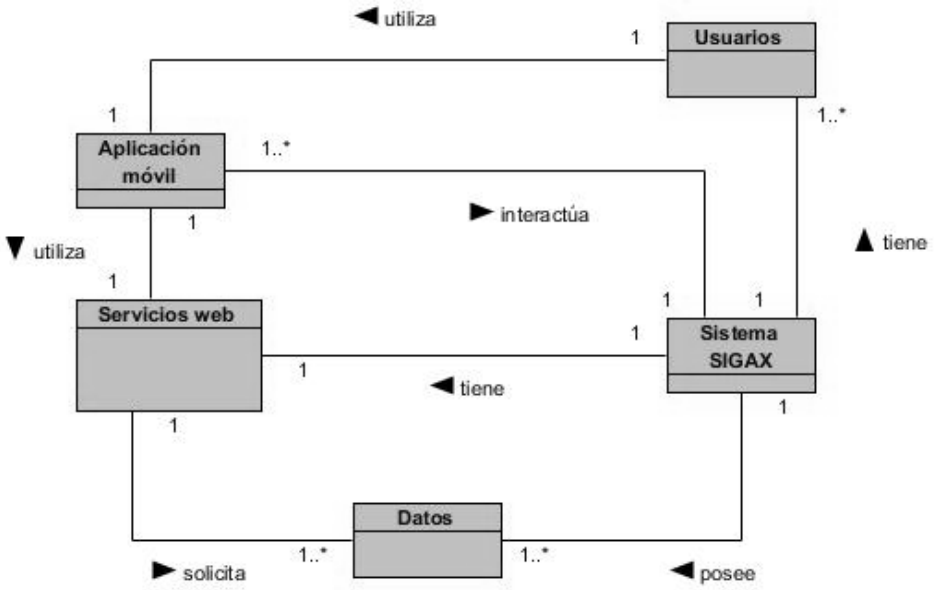


Figura 4: *Modelo de dominio de la propuesta de solución.*

Fuente: *Elaboración propia.*

2.4. Fase de inicialización

En esta fase los desarrolladores preparan e identifican todos los recursos necesarios. Se preparan los planes para las siguientes fases y se establece el entorno técnico como los recursos físicos, tecnológicos y de comunicaciones, incluyendo el entrenamiento del equipo de desarrollo. Esta fase se divide en tres etapas: la puesta en marcha del proyecto, la planificación inicial y el día de prueba (AGILE 2017).

- **Puesta en marcha del proyecto:** el propósito de esta etapa es la configuración de los recursos físicos y técnicos para el proyecto, así como el medio para su seguimiento, se capacita al equipo de desarrollo de ser necesario, y se establecen las formas específicas de comunicación con el cliente. Para el cumplimiento de estas tareas se procedió a configurar el IDE de desarrollo Android Studio y el sistema SIGAX en el servidor, se realizó una capacitación sobre tecnologías de desarrollo móvil con Android Studio y el uso de la metodología Mobile-D y se solicitó la lista de correos y teléfonos del equipo de desarrollo de SIGAX para asegurar así una buena comunicación.
- **Planificación inicial:** el propósito de la etapa de planificación inicial es obtener una buena comprensión general del producto a desarrollar, para preparar y perfeccionar los planes para las próximas fases del proyecto y preparar planes de comprobación y resolución de todas las cuestiones fundamentales del desarrollo final de la fase de Inicialización. Durante esta etapa se analizaron y priorizaron los requisitos iniciales del software lo que llevó a la redacción de las Historias de usuario y la construcción de los primeros prototipos de interfaz.
- **Día de prueba:** el propósito de esta etapa es asegurarse que todo esté listo para comenzar la implementación del software. Además de implementar algunas funciones al núcleo del sistema (por ejemplo, la comunicación cliente-servidor) o resolver algún problema crítico de desarrollo sin producir ningún código de trabajo. Durante el día de prueba se desarrollaron las vistas e interfaces más genéricas de la aplicación, incluyendo la pantalla principal y los menús. Se desarrollaron las clases y funciones que permiten la sincronización entre la aplicación móvil y el sistema SIGAX.

2.4.1. Historias de usuario

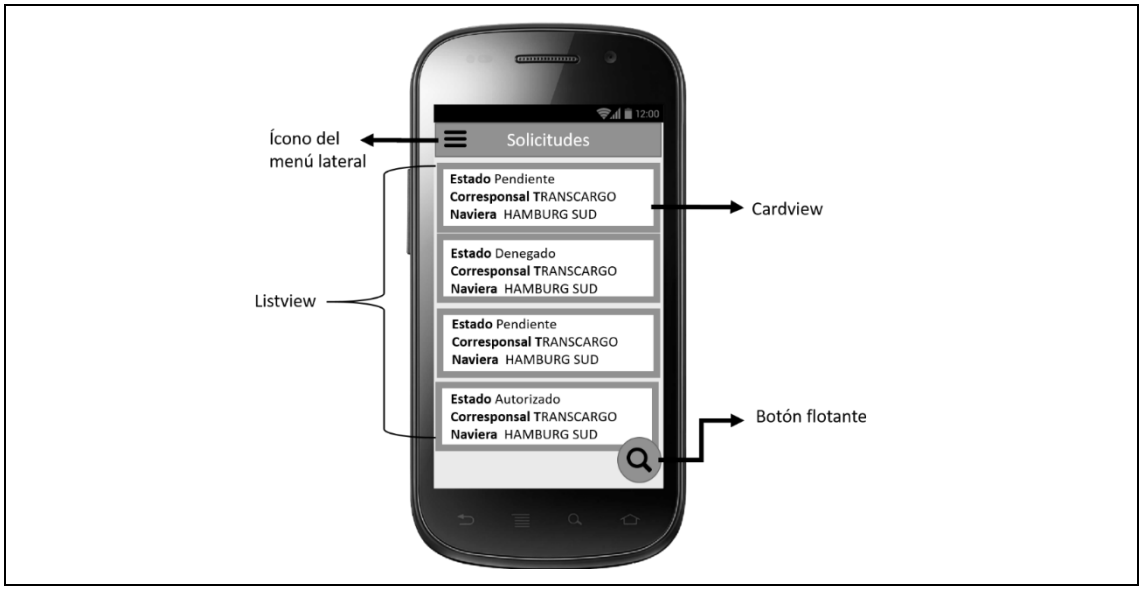
Uno de los artefactos generados por la metodología Mobile-D son las historias de usuario, utilizadas como herramientas para dar a conocer los requerimientos del sistema al equipo de desarrollo.

Las historias de usuario son utilizadas en las metodologías de desarrollo ágil para la especificación de requisitos. Estas responden la pregunta ¿Qué se debe hacer? Cada tarjeta describe una característica deseada para el proyecto en forma de historia, permitiendo administrar estos requisitos sin elaborar gran cantidad de documentación. Cada historia debe representar una sola característica (Chromatic 2003). Al realizar la discusión y el análisis de un requisito determinado se procede a realizar un prototipo de la interfaz de usuario a manera de apoyo gráfico, su objetivo será asegurar el total entendimiento por parte de los desarrolladores del requisito en cuestión.

A continuación, se muestra la historia de usuario para el requisito “Listar solicitudes de embarque” el resto de historias se reflejarán en el Anexo I.

Tabla 5: *Historia de usuario para el requisito "Listar solicitudes de embarque".*

Historia de Usuario						
Número	Tipo	Dificultad		Esfuerzo		Prioridad
		Antes	Después	Estimado	Gastado	
6	Nuevo	4	4	1	1	5
Nombre						
Listar solicitudes de embarque.						
Notas						
Descripción						
El objetivo de esta actividad es listar las solicitudes de embarque. Para ello el usuario selecciona la opción “Solicitudes de embarque” del menú y aparecerá un listado con todas las solicitudes de embarque en el sistema. Sólo se mostrarán los datos más descriptivos de una solicitud.						
Prototipo de Interfaz						



2.5. Diagrama de despliegue

Un diagrama de despliegue modela la arquitectura en tiempo de ejecución de un sistema. Esto muestra la configuración de los elementos de hardware (nodos) y muestra cómo los elementos y artefactos del software se trazan en esos nodos. (SparxSystems, 2007)

El diagrama de despliegue de la aplicación móvil que se muestra en la figura, está estructurado de la siguiente forma: el dispositivo externo se conecta al servidor web de TRANSCARGO mediante el protocolo de comunicación HTTP(s) y el estándar 802.11 para las comunicaciones inalámbricas a través de un router.

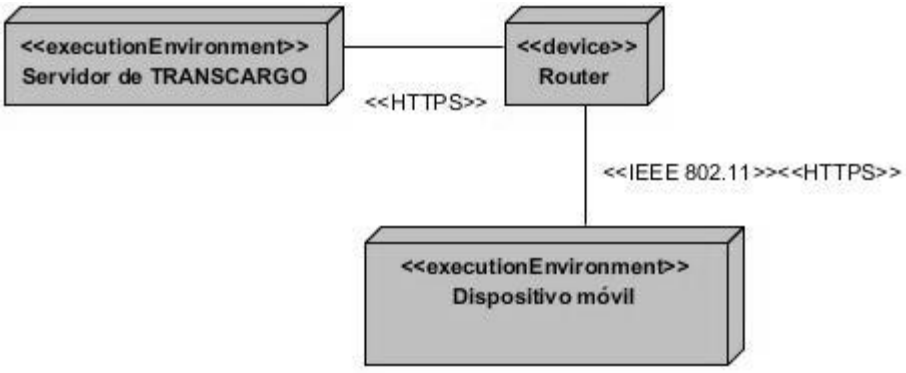


Figura 5: *Diagrama de despliegue de la aplicación.*

Fuente: *Elaboración propia.*

2.6. Modelo de datos

El modelo de datos constituye la representación de un fenómeno de la realidad objetiva a través de los objetos, sus propiedades y las relaciones que se establecen entre ellos. El mismo está compuesto por tres piezas fundamentales: el objeto de datos, los atributos y las relaciones entre las que se conectan. Permite almacenar, organizar, manipular cantidades de datos con cierta facilidad y describir los elementos de la realidad que intervienen en el problema dado y la forma en que se relacionan estos elementos entre sí. Cuando se utiliza una base de datos para gestionar información, se está plasmando una parte del mundo real en una serie de tablas, registros y campos ubicados en un ordenador (Mato García 2005).

El sistema que se propone hace uso de la base de datos del sistema SIGAX pues la información consultada se encuentra en esta. A continuación, se muestra el modelado de la base de datos de SIGAX, pero solamente las tablas que son relevantes para el desarrollo de la aplicación móvil.

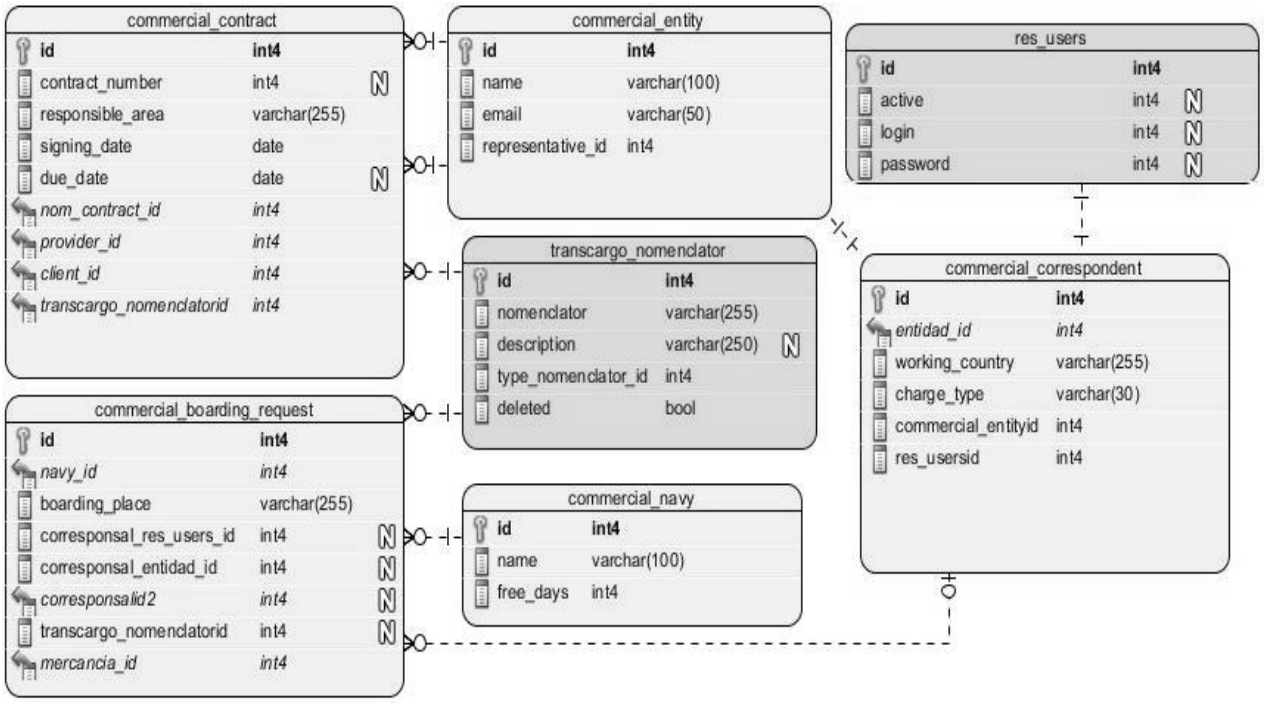


Figura 6: *Modelo de datos (entidades contrato y solicitud de embarque).*

Fuente: *Elaboración propia.*

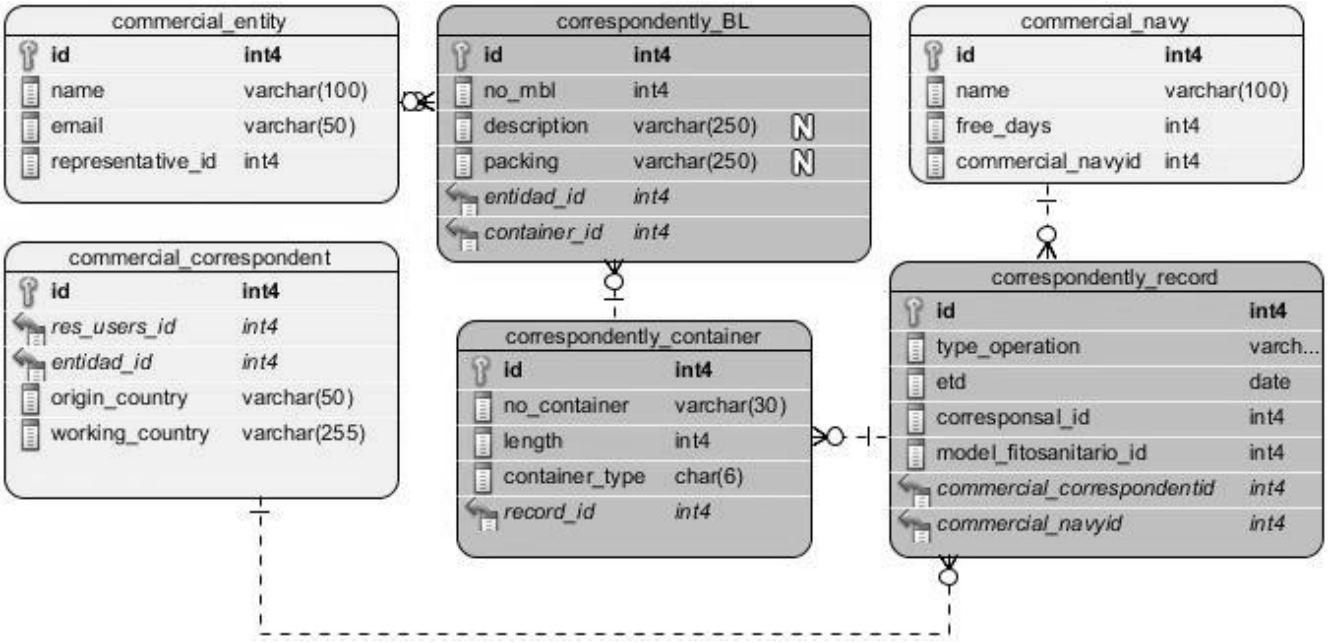


Figura 7: *Modelo de datos (entidad expediente).*

Fuente: *Elaboración propia.*

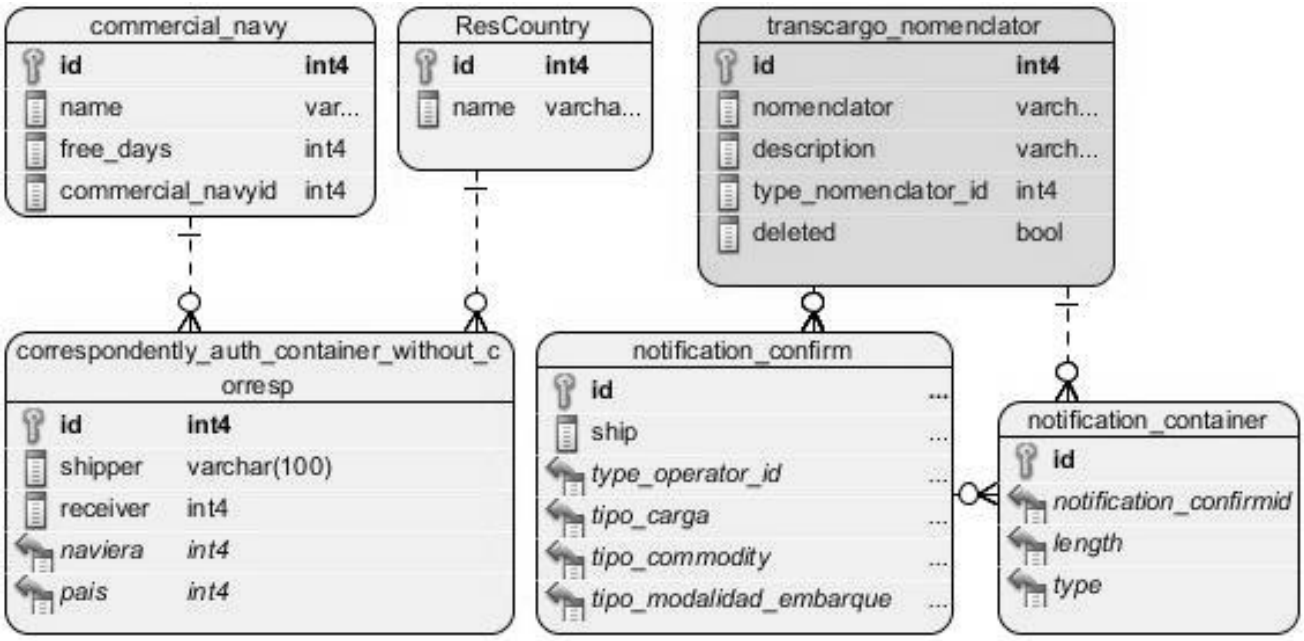


Figura 8: *Modelo de datos (entidades aviso de confirmación y autorizo sin corresponsal).*

Fuente: *Elaboración propia.*

2.7. Patrón Arquitectónico

Para el desarrollo de la propuesta de solución se prevé utilizar el patrón arquitectónico Modelo-Vista-Controlador. Como su nombre indica este patrón divide un sistema en tres componentes fundamentales: el modelo (representado por la lógica del negocio), la vista (que hace referencia a la interfaz de usuario) y el controlador (que actuará de mediador entre los componentes anteriores ejecutando las funcionalidades de la aplicación). Este patrón es basado en los principios de reutilización de código y separación de conceptos, ideas que persiguen la meta de facilitar el desarrollo de software y su posterior mantenimiento. A continuación, se presenta una representación gráfica de la relación entre estos componentes, las líneas sólidas indican una asociación directa, y las punteadas una indirecta.

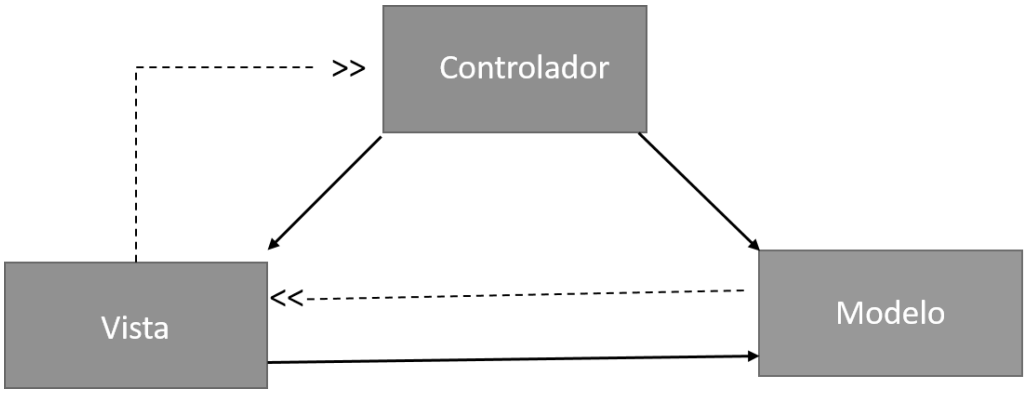


***Figura 9**: Diagrama que muestra la relación entre los componentes del MVC.*

***Fuente**: Elaboración propia.*

Modelo: es la representación de los datos del programa, se encarga de manejarlos y controlar sus transformaciones. Puede almacenar estos datos en clases de tipos de datos complejos o primitivos, incluso puede no almacenarlos en sí mismo sino devolver estos datos remotamente. El modelo envía a las vistas la información solicitada en cada momento para que sea mostrada (usualmente al usuario). Estas peticiones de acceso llegan mediante el controlador.

Vista: la Vista representa la interfaz gráfica vista por los usuarios, se encarga de manejar la presentación de los datos representados en el modelo, por tanto, requiere del modelo la información que debe brindar como salida. Interactúa preferentemente con el controlador, pero es posible que trate directamente con el modelo a través de una referencia.

Controlador: el controlador es el encargado de dar significado a las órdenes del usuario haciendo de intermediario entre la vista y el modelo. Responde a los eventos e invoca peticiones al modelo cuando se requiere alguna información almacenada en él, estos datos son enviados a las vistas. También

puede enviar comandos a su vista si se solicita algún cambio en la forma en que se representa el modelo.

MVC en Android

Según Tian Lou en su Tesis de Maestría *"A Comparison of Android Native App Architecture–MVC, MVP and MVVM":* la arquitectura MVC es la arquitectura por defecto para las aplicaciones Android nativas. El modelo se refiere a las representaciones que se construyen basadas en la información con la que operará en la aplicación, usualmente es un objeto de una clase Java (Lou 2016).

La vista es la combinación de un *layout* XML y una *activity/fragment*. No se puede considerar solamente como el recurso XML ya que este recurso no tiene el control total sobre la interfaz de usuario. Los componentes en los ficheros del layout tienen que ser inflados sobre las activities/fragments antes de que puedan ser mostrados en pantalla. Adicionalmente el fichero del layout no puede cambiar la visibilidad de los componentes dinámicamente (Lou 2016).

El controlador es la activity/fragment y se encarga de capturar los eventos de las vistas y enviar las respuestas de vuelta, o enviar una solicitud al modelo para obtener y actualizar los datos. Se puede decir entonces que en Android la Vista y el Controlador se han solapado en la activity y el fragment. La siguiente figura es una representación del patrón MVC en Android y muestra como el límite entre Vista y Controlador no está bien definido (Lou 2016).

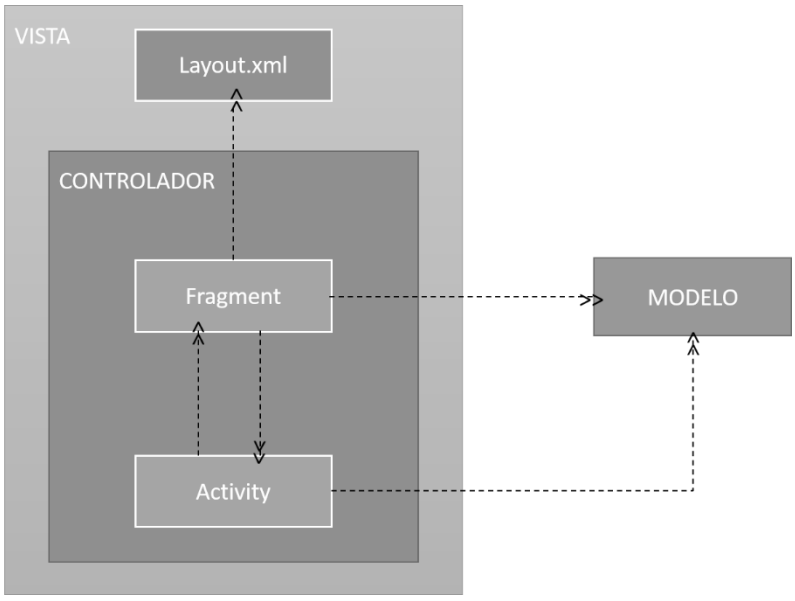


Figura 10: *Implementación en Android del patrón MVC.*

Fuente: *Adaptado de (Lou 2016)*

2.8. Patrones de diseño

Un patrón de diseño consiste en un diagrama de objetos que forma una solución a un problema conocido y frecuente. El diagrama de objetos está constituido por un conjunto de objetos descritos por clases y las relaciones que enlazan los objetos. Los patrones responden a problemas de diseño de aplicaciones en el marco de la programación orientada a objetos. Se trata de soluciones conocidas y probadas cuyo diseño proviene de la experiencia de los programadores. No existe un aspecto teórico en los patrones, en particular no existe una formalización (a diferencia de los algoritmos). (Debrauwer 2012). A continuación, se presentan los patrones de diseño identificados durante el desarrollo de la propuesta de solución clasificados en dos grandes grupos: patrones GRASP y patrones GoF.

2.8.1. Patrones GRASP

Los patrones GRASP (del inglés *General Responsibility Assignment Software Patterns* o Patrones para la asignación de responsabilidades) como su nombre lo indican son patrones que establecen cual es la manera de asignar responsabilidades a objetos software. En la programación Orientada a objetos resulta esencial elegir las clases adecuadas y decidir cómo estas clases deben interactuar. En el libro *"UML y patrones"* de Craig Larman se describen estos patrones y se presentan como una buena práctica a asumir para la creación de software altamente confiable (Larman 1999). A continuación, se abordarán los patrones GRASP identificados en el desarrollo de la propuesta de solución.

Patrón Alta cohesión

El patrón alta cohesión fue utilizada para asignar responsabilidades a las clases que trabajen sobre una misma área de aplicación. Para el desarrollo de la propuesta de solución fueron creados varios fragments encargados de trabajar con una entidad específica del sistema; de esta manera se evita que una clase sea la única responsable de muchas tareas en áreas funcionales muy heterogéneas. La clase "ContractList" es ejemplo de la utilización de este patrón. Es la encargada de listar los elementos de la entidad Contrato (Contract), permitiendo la sincronización en tiempo real con la base de datos y la realización de búsquedas sobre el listado.

Patrón Bajo acoplamiento

Este patrón fue utilizado con el objetivo de evitar una gran dependencia entre las clases y permitir una alta reutilización en el código. El diseño propuesto respeta este patrón al evidenciar clases con el menor número de clases relacionadas posible, ejemplo de ello son los servicios de sincronización. La clase "CorrespondentlyRecordSyncService" se encarga, solamente, de realizar la sincronización con la base

de datos de la entidad Expediente (Record). La creación de estas clases independientes provoca que los cambios realizados en otros servicios de sincronización no influyan en el funcionamiento de un servicio específico.

Patrón Experto

El patrón Experto en Información fue utilizado para el desarrollo de la aplicación con el objetivo de definir las iteraciones básicas entre los objetos, asignando responsabilidades a las clases para que el sistema sea fácil de entender. Este patrón se ve reflejado en las clases que integran el modelo, cada clase maneja solamente la información de una entidad en específico; ejemplo de ello es la clase “BoardingRequest” la cual maneja la información referente a una solicitud de embarque. Con el uso de este patrón cada clase es experta en información que maneja y la utiliza para llevar a cabo sus tareas, esto permite la reutilización de componentes para futuras aplicaciones (Martínez, 2011).

Patrón Creador

El patrón Creador fue utilizado con el propósito fundamental de asignar responsabilidades relacionadas con la creación de objetos, lo cual facilitó un Bajo Acoplamiento y mayores oportunidades de reutilización. Este patrón se ve reflejado en la clase “AuthorizeList”, la cual tiene la responsabilidad de crear una instancia de la clase “Authorize” con el objetivo de acceder a sus atributos y mostrarlos en una lista.

2.8.2. Patrones GoF

El avance reciente más importante en el diseño orientado a objetos es probablemente el movimiento de los patrones de diseño, inicialmente narrado en el libro *“Design Patters Elements of Reusable Object-Oriented Software*”, que suele llamarse el libro de la "Banda de los Cuatro" (en inglés, GoF: Gang of Four). Los patrones de diseño GoF se clasifican en tres grandes categorías basadas en su propósito: creacionales, estructurales y de comportamiento desglosados en 23 patrones. La propuesta de solución hace uso de algunos de estos patrones, los cuales serán mostrados a continuación.

Singleton

El patrón Singleton fue utilizado con el objetivo de asegurar que una clase determinada solo pueda poseer una instancia, proporcionando un método de clase único que la devuelva. El patrón se ve reflejado en la clase “OdooAccountManager” la cual maneja la información relacionada con el usuario

activo en la aplicación. Con este patrón la clase será la responsable de crear su propia instancia única la cual es devuelta con el método “getActiveUser”.

Adapter

El uso del patrón Adapter (Adaptador) es ampliamente utilizado en la aplicación para poblar elementos gráficos como listas con un conjunto de datos brindado por otra clase. La utilización de este patrón se ve reflejada específicamente en la clase “OCursorListAdapter” la cual recibe los datos de una entidad del modelo, proporciona acceso a cada ítem del conjunto de datos y crear una vista para cada uno.

Template Method

El patrón Template Method se ve reflejado en la clase “Query” la cual posee el método “generateQuery”; este método se encarga de recibir una lista de condiciones y generar una consulta SQL para realizar la búsqueda filtrada de elementos. La implementación del método “generateQuery” varía en cada una de las subclases de “Query” (“ContractQuery”, “BoardingQuery”, “RecordQuery” entre otras) por lo que este patrón tiene como objetivo permitir que las subclases redefinan ciertos pasos del método sin cambiar su estructura.; lo anterior evita que exista código duplicado en la aplicación.

2.9. Conclusiones parciales

Luego de transitar por las etapas de exploración e inicialización del proyecto descritas en la metodología Mobile-D se concluye lo siguiente:

- La representación del modelo de dominio sirvió de base para la comprensión de las diferentes clases que integran el sistema.
- La definición de las historias de usuario junto a los prototipos de interfaz, permitió la comprensión de los requisitos funcionales.
- La definición de la arquitectura basada en Modelo-Vista-Controlador junto al uso de patrones de diseño sentó las bases para la construcción de un software donde se apliquen buenas prácticas y las mejores soluciones probadas hasta el momento.
- El tránsito por las fases iniciales de la metodología Mobile-D y la generación de los artefactos correspondientes establecerá el punto de partida para la implementación de la propuesta de solución en las fases posteriores.

CAPÍTULO 3. IMPLEMENTACIÓN Y PRUEBA

3.1. Introducción

Mobile-D, como metodología ágil, establece que los procesos de implementación y prueba deben ser realizados con una documentación lo más ligera posible al generar pocos artefactos y centrarse en la comunicación entre el cliente y el equipo de desarrollo. Estos procesos están claramente definidos en las tres últimas fases de la metodología: fase de producto, fase de estabilización y fase de prueba-corrección.

Es objetivo del presente capítulo abordar las tareas desarrolladas durante estas fases, presentar los artefactos generados y hacer un análisis de los resultados de las pruebas realizadas a la aplicación, evidenciando así, el satisfactorio desenlace del proceso de desarrollo de la propuesta de solución.

3.2. Fase de producto

El propósito de la fase de producto es la implementación de las funcionalidades requeridas del software aplicando un ciclo de desarrollo iterativo e incremental. Parte del criterio de que los requisitos funcionales más importantes han sido identificados y persigue como meta enfocarse en las funcionalidades fundamentales al implementarlas en las primeras iteraciones. Esta fase se encuentra dividida en tres etapas: día de planeación, día de trabajo y día de entrega (AGILE 2017).

- **Día de planeación:** su objetivo es seleccionar y planear los contenidos de trabajo para la iteración, se ha de prever para ello que los requerimientos sean correctamente entendidos. Durante esta etapa se analizaron los requerimientos y se generaron las pruebas de aceptación para cada historia de usuario.
- **Día de trabajo:** el propósito de esta etapa es implementar las funcionalidades del sistema creadas durante el día de planeación. El equipo de desarrollo se enfoca en la funcionalidad de más prioridad definida por el cliente. Una iteración puede contener uno o varios días de trabajo.
- **Día de entrega**: tiene como propósito hacer la liberación completamente funcional del sistema bajo desarrollo. Durante esta etapa se ejecutaron, las pruebas de aceptación para verificar los requerimientos implementados.

3.2.1. Tareas de ingeniería

Las tareas de ingeniería son la principal herramienta de planeación para los desarrolladores. Ellas responden a la pregunta ¿Cómo se debe hacer? Cada tarea representa los pasos necesarios para

implementar una historia de usuario. Son de naturaleza técnica e identifican los detalles de la implementación, comunicando ideas de diseño de alto nivel entre desarrolladores. (Chromatic 2003)

Durante el día de planeación se dividieron las historias de usuario en varias tareas de ingeniería con el objetivo de guiar la implementación al brindar una descripción detallada de los procedimientos a realizar. A continuación, se presentan las tareas de ingeniería correspondientes al requisito funcional "Listar solicitudes de embarque". El resto de las tareas se podrán encontrar en el Anexo II.

Tabla 6: *Tarea "Crear módulo solicitudes de embarque".*

Tarea de Ingeniería						
Número	Tipo	Dificultad		Esfuerzo		Historia
		Antes	Después	Estimado	Gastado	
6.1	Nuevo	1	1	0.1	0.1	6
Nombre						
Crear módulo solicitudes de embarque.						
Notas						
Descripción						
Se debe definir el módulo contratos en el archivo "Addons". Se crea la estructura de ficheros correspondiente y se define el "BaseFragment" para listar las solicitudes de embarque.						

Tabla 7: *Tarea "Definir interfaz de usuario para listar las solicitudes de embarque".*

Tarea de Ingeniería						
Número	Tipo	Dificultad		Esfuerzo		Historia
		Antes	Después	Estimado	Gastado	
6.2	Nuevo	1	1	0.2	0.2	6
Nombre						
Definir interfaz de usuario para listar las solicitudes de embarque.						
Notas						
Descripción						
Se debe definir el layout para el listview y se deben definir los cardview para mostrar los elementos. Los campos han de mostrarse como indica la documentación.						

Tabla 8: *Tarea "Crear modelo, proveedores y servicios de sincronización".*

Tarea de Ingeniería						
Número	Tipo	Dificultad		Esfuerzo		Historia
		Antes	Después	Estimado	Gastado	
6.3	Nuevo	2	2	0.5	0.5	6
Nombre						
Crear modelo, proveedores y servicios de sincronización para la entidad solicitud de embarque.						
Notas						
Descripción						
Se define el modelo para la entidad solicitud de embarque y todas las entidades relacionadas con él. Se define el servicio de sincronización, proveedor de sincronización con la autoridad (AUTHORITY) del modelo. Se define el archivo XML de metadatos para el adaptador de sincronización. Se declaran los servicios y proveedores en el Android Manifest.						

Tabla 9: *Tarea "Declarar e inicializar controles".*

Tarea de Ingeniería						
Número	Tipo	Dificultad		Esfuerzo		Historia
		Antes	Después	Estimado	Gastado	
6.4	Nuevo	1	1	0.1	0.1	6
Nombre						
Declarar e inicializar controles.						
Notas						
Descripción						
Se registra el loader manager definiendo los métodos "onCreateLoader", "onLoadFinished" y "onLoaderReset". Se inicializa el listview y se llenan los campos del cardview con los datos correspondientes.						

3.2.2. Estándares de codificación

Los estándares de codificación son un conjunto de directrices, normas y reglamentos enfocados a la especificación de cómo debe escribirse el código fuente de la aplicación. Estos incluyen pautas sobre la nomenclatura de las variables, clases y paquetes, la correcta indentación del código, cómo escribir estructuras de control, entre otros aspectos (Vermeulen 2000).

El uso de estándares de codificación requiere que todos los desarrolladores escriban y mantengan el código en un formato común y consistente (Beck 2000). Las reglas que gobiernen este formato facilitan

la comunicación efectiva entre desarrolladores al proveer una base común a través de la cual entender varias unidades de código (Maruping, Zhang, y Venkatesh 2009).

A continuación, se describen los estándares de codificación que regirán la implementación de las funcionalidades.

Paquetes: el nombre de los paquetes debe comenzar con letra minúscula. En caso de existir un cambio de palabra se utilizará el carácter guion bajo.

Clases: los nombres de las clases deben ser sustantivos, se deben evitar el uso de abreviaturas o acrónimos. Es obligatorio empezar el nombre con letra mayúscula, de estar compuesto por más de una palabra se alternarán mayúsculas y minúsculas.

Métodos: los métodos deben ser verbos. Es obligatorio empezar el nombre con letra minúscula, de estar compuesto por más de una palabra se alternarán mayúsculas y minúsculas.

Variables: todas las variables deben comenzar con letra minúscula, al cambiar la palabra se usará letra mayúscula. Los nombres de las variables no deben empezar por el carácter guion bajo o el signo de dólar, aunque ambos son permitidos en el lenguaje Java.

Sentencias: se proporcionará una sentencia por línea de código. Todo bloque de sentencia debe ser colocado entre llaves, aunque sea solamente una sentencia. Si una línea de código es demasiado larga por poseer expresiones complejas se debe dividir en varias líneas después de una coma u operador.

Comentarios: es obligatorio proporcionar comentarios por cada clase o método creado, este comentario debe consistir en la descripción de la clase o método y su responsabilidad. Se deben proporcionar datos adicionales como parámetros, tipos de retorno y el número de la tarea de ingeniería para la que fue creado.

3.3. Fase de estabilización

El propósito de la fase de estabilización es asegurar la calidad en la implementación del proyecto. Parte de la primicia de que las funcionalidades esenciales del producto han sido implementadas y mostradas al cliente. Sus objetivos fundamentales son: finalizar la implementación del producto, mejorar su calidad y finalizar su documentación. Esta fase se encuentra dividida en cuatro etapas: día de planeación, día de trabajo, etapa de documentación y día de entrega. Los días de planeación, trabajo y entrega no difieren mucho en cuanto a tareas con respecto a sus similares en la fase de producto, aunque con un ligero cambio en el enfoque y los objetivos primarios. Durante las etapas de la fase de producto se implementan las funcionalidades que más valor aporten al negocio, mientras que las etapas de la fase estabilización se centran en corregir deficiencias, mejorar la calidad y finalizar la documentación. Esta

última tarea es llevada a cabo durante la etapa de documentación o encapsulamiento de la documentación. Durante la misma se finalizan los documentos de diseño, arquitectura e interfaz de usuario de manera que resulten breves, útiles, completamente entendibles y que concuerden con el código (AGILE 2017).

Durante la fase de estabilización fueron corregidas algunas deficiencias encontradas en la propuesta de solución, fundamentalmente relacionadas con la interfaz de usuario. Las historias de usuario y demás artefactos de la documentación fueron actualizados para concordar con las nuevas especificaciones. Una vez implementadas las funcionalidades restantes se consideró que el sistema está listo para probarse, por lo que se decide pasar a la fase de prueba-corrección. En el siguiente acápite serán abordados los temas relacionados con las pruebas ejecutadas a la aplicación.

3.4. Fase de prueba-corrección

La fase de prueba-corrección tiene como propósitos verificar que el sistema producido implementa las funcionalidades definidas por el usuario correctamente, proveer al equipo de desarrollo de una retroalimentación sobre funcionalidades y arreglar los defectos encontrados. Esta fase se compone de las siguientes etapas:

- **Prueba del sistema:** el propósito de esta etapa es encontrar defectos en el software producido después de la fase de implementación del proyecto.
- **Corrección:** es una variación de una iteración normal, sin embargo, no se implementan nuevas funcionalidades y el tiempo de duración es significativamente menor. Al igual que la fase anterior también se compone de un día de planeación, día de trabajo, etapa de documentación y día de entrega.
    - **Día de planeación:** su objetivo es definir los contenidos para la iteración. Los defectos encontrados en la etapa de prueba son las entradas para esta fase.
    - **Día de trabajo:** su propósito es corregir los defectos encontrados en la etapa de prueba del sistema y finalizar la implementación del producto.
    - **Documentación:** persigue como objetivo finalizar los documentos de arquitectura de software, diseño e interfaz de usuario. La documentación se actualiza para que concuerde con los cambios hechos durante la iteración de corrección.
    - **Día de entrega:** tiene como propósito verificar y validar las funcionalidades implementadas, la calidad de todo el software y su documentación. Culmina con el lanzamiento final del producto.

Los siguientes sub-acápites estarán dedicados a describir las pruebas usadas en la propuesta de solución y analizar los resultados obtenidos en su ejecución.

3.4.1. Pruebas de software

Se entiende como prueba de software el proceso de analizar los elementos de un software para detectar las diferencias entre las condiciones existentes y las requeridas (IEEE 1990). Estas pruebas intentan mostrar lo que un programa hace, lo que pretende hacer y descubrir sus defectos antes de ponerlo a funcionar. Cuando se prueba el software, se ejecuta el programa usando datos artificiales y se analizan los resultados para encontrar errores o anomalías. Usualmente un software tiene que transitar por tres etapas de pruebas (Sommerville 2010):

1. **Pruebas de desarrollo:** donde el sistema es probado durante su desarrollo para descubrir defectos. Los diseñadores y programadores están vinculados en el proceso de prueba.
2. **Pruebas de liberación:** donde un equipo de prueba realiza las comprobaciones antes de entregarle el sistema a los usuarios.
3. **Pruebas de usuario:** donde usuarios potenciales del sistema lo prueban en su propio ambiente.

Las pruebas de software se ejecutan en diferentes niveles a través de los procesos de desarrollo y mantenimiento, estos niveles se distinguen basándose en el objeto de la prueba (también llamado blanco) o en el propósito (también llamado objetivo) (Bourque, Fairley, y otros 2014).

Atendiendo al objeto las pruebas pueden clasificarse en:

- **Pruebas unitarias:** verifican el funcionamiento aislado de elementos de software que puedan ser individualmente probados.
- **Pruebas de integración:** se centran en verificar las interacciones entre los componentes del software.
- **Prueba de sistema:** son responsables de probar el comportamiento del sistema como un todo.

El proceso de pruebas es conducido en vista de algunos objetivos específicos, los cuales son más o menos explícitos y tienen niveles de precisión variante. Las pruebas pueden apuntar a la verificación de varias propiedades. A continuación, se listan las pruebas mencionadas frecuentemente en la literatura (Bourque, Fairley, y otros 2014).

- **Pruebas de aceptación**: determinan si el sistema satisface o no los criterios de aceptación, usualmente al chequear los requerimientos del cliente.

- **Pruebas de instalación**: a menudo, después del completamiento de un sistema y las pruebas de aceptación, el software es verificado a través de su instalación en un ambiente designado.
- **Pruebas Alfa y Beta**: antes que el software sea liberado usualmente se le entrega a un grupo selecto de usuarios potenciales para que lo prueben (pruebas Alfa) o a un gran conjunto de usuarios representativos.
- **Pruebas de regresión**: se basa en encontrar defectos después de haber ocurrido un gran cambio en el código.
- **Pruebas de rendimiento**: verifica que el software cumple con los requerimientos de rendimiento especificados, por ejemplo, capacidad y tiempo de respuesta.
- **Pruebas de seguridad**: se enfocan en la verificación de que el software está protegido contra ataques externos.

Durante la etapa de pruebas del sistema fueron ejecutados dos tipos de pruebas: pruebas unitarias y pruebas de aceptación, a continuación, se ofrece una breve descripción de las mismas.

Pruebas unitarias

La prueba unitaria enfoca los esfuerzos de verificación en la unidad más pequeña del diseño (Pressman 2007). Es el proceso de probar los componentes del programa como métodos o clases de objetos. Las funciones individuales o métodos son los tipos de componentes más simples. Estas pruebas deberán llamar a las rutinas con diferentes parámetros de entrada (Sommerville 2010). Para la realización de las pruebas unitarias a la aplicación fue utilizado el framework JUnit en la versión 4.12 y el framework Mockito en su versión 2.8.9.

JUnit es una herramienta simple, de código abierto que se ha convertido en el estándar para probar clases Java de forma unitaria al ser fácil de usar y configurar. JUnit ejecuta casos de prueba contra un conjunto de objetos y provee formas de inicializar y configurar cada caso. Al suministrar datos de entradas y comparar con los resultados esperados JUnit devolverá si un método determinado pasó la prueba exitosamente o mostrará un mensaje de error (Rodríguez 2006).

Mockito es un framework de simulación de pruebas unitarias para Java. Permite simular la creación y verificación de objetos y dependencias externas. Utiliza para ello la técnica conocida como mocking. El mocking es una técnica utilizada en las pruebas de software donde los componentes reales son remplazados por objetos que tienen un comportamiento predefinido solamente para la prueba para la que han sido creados, de esta manera se configuran los objetos para que devuelvan un valor específico sin realizar ninguna acción (Grzejszczak 2014). Las pruebas unitarias fueron realizadas durante todo

el proceso de implementación de las funcionalidades. Los resultados de estas pruebas se mostrarán en el sub-acápite 3.4.2

Pruebas de aceptación

Los desarrolladores escriben las pruebas unitarias para determinar si el código está funcionando correctamente. Los clientes escriben las pruebas de aceptación para determinar si el sistema está cumpliendo con sus propósitos. Las pruebas de aceptación representan los intereses del cliente y le brindan confianza que la aplicación cuenta con las funcionalidades requeridas y se comportan correctamente. En teoría si todas las pruebas de aceptación pasan, entonces, el proyecto ha sido terminado (Miller y Collins 2001).

Las pruebas de aceptación cumplen tres funciones para el equipo de desarrollo de software:

1. Capturan los requerimientos del cliente de forma directa y verificable, además, miden que tan bien el sistema cumple con esos requerimientos.
2. Exponen problemas que las pruebas unitarias pasan por alto.
3. Proveen una definición confeccionada de cómo hacer las actividades.

Las pruebas de aceptación, consideradas como *pruebas de caja negra*[22] (Joskowicz 2008), en principio comparten la noción que los usuarios deben decidir si el sistema es aceptable o no. Estas pruebas se deberían ejecutar después de las pruebas de liberación, sin embargo, en metodologías ágiles las pruebas de aceptación tienen un significado diferente. En metodologías basadas en XP (como es el caso de Mobile-D) el usuario forma parte del equipo de desarrollo o está estrechamente vinculado a este, es responsable de proveer requerimientos en forma de historias de usuario y de definir las pruebas, las cuales deciden si el software desarrollado cumple con lo definido en la historia de usuario (Sommerville 2010).

Durante la etapa de prueba se definen dos tareas fundamentales: la recolección de las pruebas presentes en la documentación y su ejecución para obtener posibles errores. Cada caso de prueba fue representado en forma de tabla con los siguientes campos: identificador, número y nombre de la historia a que pertenece, fecha en que fue creado, fecha en que se ejecutó, si la prueba pasó, entradas y resultados esperados. A continuación, se presenta una de las pruebas de aceptación generadas para la historia de usuario “Listar solicitudes de embarque”. El resto de las pruebas podrán ser encontradas en el Anexo III.

[22] Es un enfoque de prueba donde los probadores no tienen acceso al código del sistema o sus componentes. Las pruebas se derivan de las especificaciones del sistema.

Tabla 10: *Prueba de aceptación para la historia "Listar solicitudes de embarque".*

CASO DE PRUEBA DE ACEPTACIÓN	
CÓDIGO: CPA_6.1	**HISTORIA DE USUARIO:** HU_6
DESCRIPCIÓN: prueba la funcionalidad de listar solicitudes de embarque.	
CONDICIONES DE EJECUCIÓN: - Estar autenticado en el sistema. - Debe existir en el sistema al menos una solicitud de embarque. - El usuario debe poseer los permisos de comercial.	
ENTRADA/PASOS DE EJECUCIÓN: Una vez autenticado el usuario se selecciona la opción “solicitudes de embarque”.	
RESULTADO ESPERADO: Aparece un listado con todas las solicitudes de embarque en el sistema. En el listado se deben mostrar los siguientes datos: - Estado actual - Corresponsal - Lugar de embarque - Naviera De no existir ninguna solicitud de embarque en el sistema se mostrará un mensaje.	
EVALUACIÓN DE LA PRUEBA: SATISFACTORIA	

3.4.2. Resultados de las pruebas

Una vez ejecutadas las pruebas definidas para la aplicación fueron obtenidas un conjunto de no conformidades (NC) para cada iteración. Estas NC fueron clasificadas de acuerdo con su prioridad en altas, medias y bajas.

En cuanto a las pruebas unitarias, se realizaron un total de 12 casos de prueba. Para la primera iteración se obtuvieron 9 NC (3 altas,2 medias y 4 bajas). Para la segunda iteración se detectaron 7 NC (1 alta, 3 medias y 3 bajas) y para la tercera iteración fueron detectadas 4 NC (2 medias y 2 bajas). En la cuarta iteración no se detectaron no conformidades.

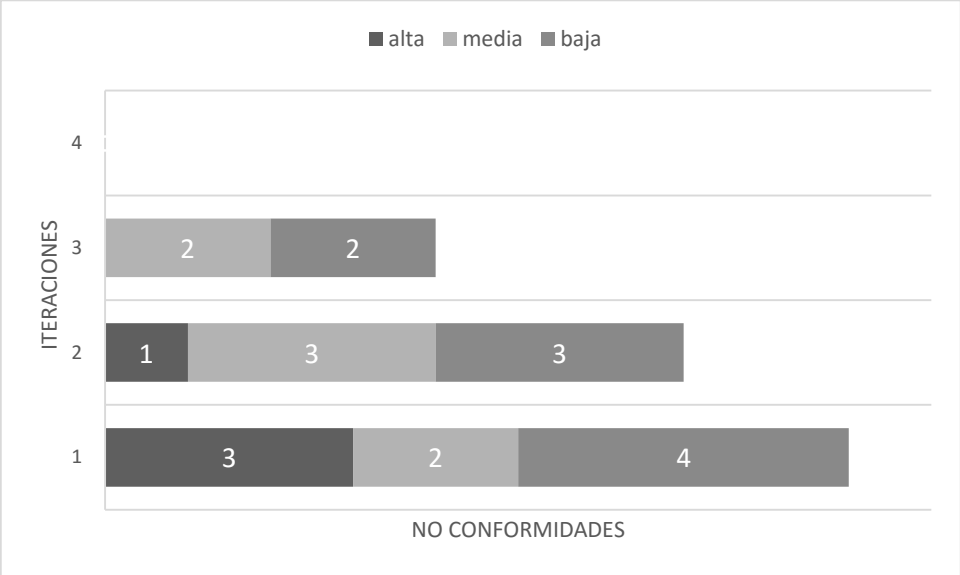


Figura 11: *Pruebas unitarias por iteración.*

Fuente: *Elaboración propia.*

En cuanto a las pruebas de aceptación fueron ejecutados un total de 33 casos de prueba. Para la primera iteración se detectaron 10 NC (2 altas, 3 medias y 5 bajas). Para la segunda iteración fueron encontradas 6 NC (1 alta, 2 medias y 3 bajas). En la tercera iteración se detectaron 2 NC (1 media y 1 baja). Para la cuarta iteración no fueron detectadas no conformidades. Los datos anteriormente expuestos aparecen reflejados en el gráfico siguiente.

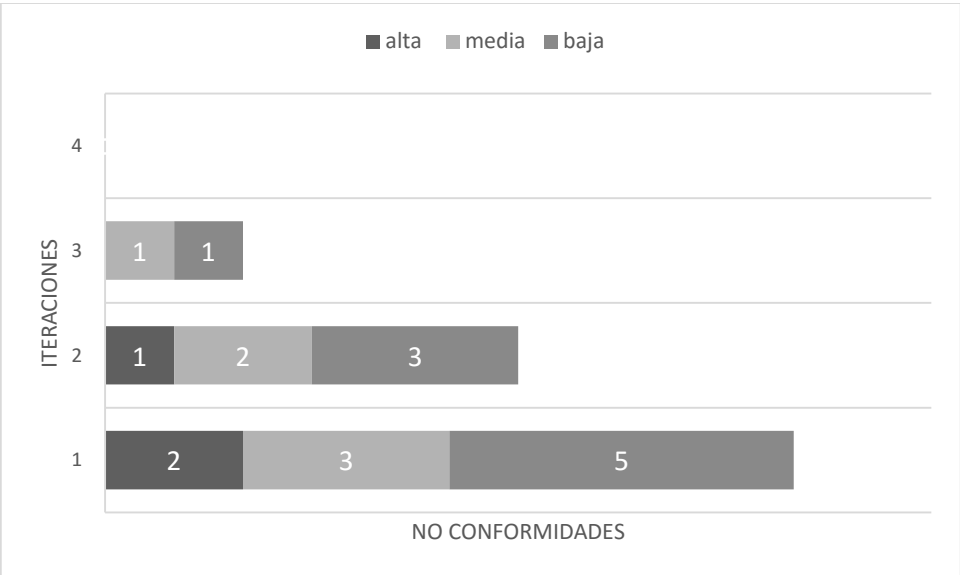


Figura 12: *Pruebas de aceptación por iteración.*

Fuente: *Elaboración propia.*

3.5. Conclusiones parciales

Luego de transitar por las fases de producto, estabilización y prueba-corrección definidas en la metodología Mobile-D se concluye lo siguiente:

- El desglose de las historias de usuario en tareas de ingeniería permitió describir la manera en que serán implementados los requisitos funcionales del sistema.
- El uso de estándares de codificación permitió guiar el proceso de implementación, obteniendo un código más estructurado y entendible.
- El uso en conjunto de las pruebas unitarias y de aceptación permitió la identificación de no conformidades en la propuesta de solución. En total se ejecutaron 33 pruebas de aceptación y 12 pruebas unitarias. Las deficiencias fueron corregidas satisfactoriamente.

CONCLUSIONES GENERALES

Atendiendo a los objetivos propuestos en la investigación se concluye lo siguiente:

- El estudio de las aplicaciones ERP móviles evidenció la ausencia de soluciones aplicables al problema de la investigación, sin embargo, este análisis sirvió de base para identificar las posibles funcionalidades a implementar en la propuesta de solución.
- El estudio de las herramientas y tecnologías permitió definir la propuesta de solución: una aplicación móvil nativa para dispositivos con sistema operativo Android que utilizara el framework Odoo Mobile.
- Se implementó la aplicación móvil que permite el acceso a las principales funcionalidades del sistema SIGAX desde las plataformas móviles, permitiendo la sincronización de los datos de forma personalizada y el trabajo sin conexión.
- Las pruebas realizadas al sistema permitieron detectar un total de 34 no conformidades, donde 16 fueron detectadas en pruebas unitarias y 18 en pruebas de aceptación. Las no conformidades fueron resueltas en cuatro iteraciones.

RECOMENDACIONES

Para perfeccionar la solución propuesta se realizan las siguientes recomendaciones:

- Actualizar las funcionalidades de la solución a medida que se realicen las actualizaciones en el sistema SIGAX.
- Implementar un módulo de mensajería para la aplicación.

REFERENCIAS BIBLIOGRÁFICAS

Abrahamsson, Pekka. 2005. «Keynote: Mobile software development–the business opportunity of today». En *proceedings of the International Conference on Software Development*, 20–23. Citeseer. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.100.7385&rep=rep1&type=pdf.

Abrahamsson, Pekka, Antti Hanhineva, Hanna Hulkko, Tuomas Ihme, Juho Jäälinoja, Mikko Korkala, Juha Koskela, Pekka Kyllönen, y Outi Salo. 2004. «Mobile-D: an agile approach for mobile application development». En *Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications*, 174–175. ACM. http://dl.acm.org/citation.cfm?id=1028736.

Academia Android. 2014. «Android Studio v1.0: características y comparativa con Eclipse – Academia Android». http://academiaandroid.com/android-studio-v1-caracteristicas-comparativa-eclipse/.

———. 2017. «Que es Android: Características y Aplicaciones.» *Academia Android*. enero 11. http://www.configurarequipos.com/doc1107.html.

Acumatica. 2017. «Mobile Applications. How Acumatica Mobile ERP App for iOS and Android can work for you».

AGILE. 2017. «MobileD process library». http://agile.vtt.fi/mobiled.html.

AT&T. 2017. «Terminology Glossary». http://www.corp.att.com/erate/resources/glossary/.

Avison, David, y Guy Fitzgerald. 2003. *Information systems development: methodologies, techniques and tools*. McGraw Hill. http://eprints.soton.ac.uk/35879/.

Balaguera, Yohn Daniel Amaya. 2015. «Metodologías ágiles en el desarrollo de aplicaciones para dispositivos móviles. Estado actual». *Revista de Tecnología* 12 (2): 111–124.

Beck, Kent. 2000. *Extreme programming explained: embrace change*. addison-wesley professional. https://books.google.com/books?hl=es&lr=&id=G8EL4H4vf7UC&oi=fnd&pg=PR13&dq=BECK+K+(2000)+Extreme+Programming+Explained.+Addison-Wesley,+Reading,MA.&ots=j9BHxplQwj&sig=vi-2YfzZaJJ18rigsGKyyX3KNKw.

Blanco, Paco, Julio Camarero, Antonio Fumero, Adam Werterski, y Pedro Rodríguez. 2009. «Metodología de desarrollo ágil para sistemas móviles. Introducción al desarrollo con Android y el iPhone». *Dr. en Ing. Sist. Telemáticos*, 1–30.

Blázquez, Josep Prieto, Roberto Ramírez Vique, Julián David Morillo Pozo, y Marc Domingo Prieto. 2011. *Tecnología y desarrollo en dispositivos móviles*. Universitat Oberta de Catalunya. https://www.exabyteinformatica.com/uoc/Informatica/Tecnologia_y_desarrollo_en_dispositivos_moviles/Tecnologia_y_desarrollo_en_dispositivos_moviles_(Intro).pdf.

Boehm, Barry. 2002. «Get ready for agile methods, with care». *Computer* 35 (1): 64–69.

REFERENCIAS BIBLIOGRÁFICAS

Bourque, Pierre, Richard E. Fairley, y otros. 2014. *Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0*. IEEE Computer Society Press. http://dl.acm.org/citation.cfm?id=2616205.

Cailean, Diana Andreea, y Kobra Sharifi. 2014. *Mobile ERP: a literature review on the concept of Mobile ERP systems*. http://www.diva-portal.org/smash/record.jsf?pid=diva2:698608.

Carreño, Lohanny. 2017. «Rapid ApplicationDevelopment (RAD), Entorno integrado de desarrollo (IDE) Ingeniería de Software Asistida por computador (CASE) | Tópicos generales de Ingeniería de Software». Accedido enero 12. https://ingsoftwarei2014.wordpress.com/category/rapid-applicationdevelopment-rad-entorno-integrado-de-desarrollo-ide-ingenieria-de-software-asistida-por-computador-case/.

Chromatic. 2003. *Extreme Programming Pocket Guide: Team-Based Software Development*. O'Reilly Media, Inc.

Costa, Ramón. 2012. «El uso de las TIC en las organizaciónes». *EADA View*. julio 5. http://blogs.eada.edu/2012/07/05/tecnologias-informacion-en-empresa/.

Debrauwer, Laurent. 2012. *Patrones de diseño en Java. Los 23 modelos de diseño: descripcion y soluciones ilustradas en UML 2 y Java*.

Fougatsaro, Vittorio. 2009. *A study of open source ERP systems*. http://www.diva-portal.org/smash/record.jsf?pid=diva2:832902.

Grzejszczak, Marcin. 2014. *Mockito Cookbook*. Packt Publishing Ltd. https://books.google.com/books?hl=es&lr=&id=rhjkAwAAQBAJ&oi=fnd&pg=PT15&dq=Geeks,+Java+Code.+Mockito+Programing+CookBook&ots=qUC7aH62tP&sig=orTnkDWGGdFcU62BsNbZ1Hboq_w.

Hedberg, Henrik, y Juha Iisakka. 2006. «Technical Reviews in Agile Development: Case Mobile-D^TM». En *Quality Software, 2006. QSIC 2006. Sixth International Conference on*, 347–353. IEEE. http://ieeexplore.ieee.org/abstract/document/4032304/.

IBM. 2014. «El reto de la movilidad en la empresa. Definiendo la agenda estratégica.» IBM Institute for Business Value. http://www-05.ibm.com/services/es/gbs/consulting/pdf/el-reto-de-la-movilidad-2014.pdf.

———. 2017. «IMB Terminology». http://www-01.ibm.com/software/globalization/terminology/.

IDC. 2017. «Smartphone OS Market Share, 2016 Q3». http://www.idc.com/promo/smartphone-market-share/os.

IEEE. 1990. *IEEE: Standard Glossary of Software Engineering Terminology*.

Inforges. 2016. «La aplicación móvil de SAP Busines One para iPhone, iPad y para dispositivos Android es gratuita». http://www.inforges.es.

REFERENCIAS BIBLIOGRÁFICAS

Jacobson, IvarBooch, Grady Rumbaugh, JamesIvar Jacobson, Grady Booch, y James Rumbaugh. 2000. *El proceso unificado de desarrollo de software/The unified software development process.* 004.41. Pearson Educación,. http://www.sidalc.net/cgi-bin/wxis.exe/?IsisScript=UCC.xis&method=post&formato=2&cantidad=1&expresion=mfn=049900.

Joskowicz, José. 2008. «Reglas y prácticas en eXtreme Programming». *Universidad de Vigo*, 22.

JPC Technologies. 2017. «Modoo». http://www.jpctechnologies.net/solutions/modoo.

Kelevra. 2014. «El impacto de la Tecnología Móvil en las empresa». *Kelevra - Consultoría móvil y expertos en desarrollo iOS / Android / Mac / PC / Web*. marzo 14. http://kelevra.es/el-impacto-de-la-tecnologia-movil-en-las-empresas/.

Kurbel, Karl, Andrzej Dabkowski, y Anna Maria Jankowska. 2003. «A multi-tier architecture for mobile enterprise resource planning». En *Wirtschaftsinformatik 2003/Band I*, 75–93. Springer. http://link.springer.com/chapter/10.1007/978-3-642-57444-3_5.

Larman, Craig. 1999. *UML y Patrones*. Pearson. http://www.academia.edu/download/32421917/PREVIEW-LIBRO-9788483229279.pdf.

Lou, Tian. 2016. «A Comparison of Android Native App Architecture–MVC, MVP and MVVM». https://pure.tue.nl/ws/files/48628529/Lou_2016.pdf.

MarketsandMarkets. 2017. «Mobile Enterprise Application Market by Software & System -2021 | MarketsandMarkets». enero 26. http://www.marketsandmarkets.com/Market-Reports/mobile-enterprise-application-market-150461084.html.

Maruping, Likoebe M., Xiaojun Zhang, y Viswanath Venkatesh. 2009. «Role of collective ownership and coding standards in coordinating expertise in software project teams». *European Journal of Information Systems* 18 (4): 355–371.

Masoero, Pablo Hernán. 2014. «Estado del arte de sistemas ERP». http://repositorio.udesa.edu.ar/jspui/handle/10908/2739.

Mato García, Rosa María. 2005. *Sistemas de Base de Datos*.

McGaughey, Ronald E., y Angappa Gunasekaran. 2009. «Enterprise resource planning (ERP): past, present and future». *Selected readings on strategic information systems*, 359–371.

Miller, Roy, y C. Collins. 2001. «Acceptance testing». *Proc. XPUniverse* 238. http://www.dsc.ufcg.edu.br/~jacques/cursos/map/recursos/Testing05.pdf.

Muñiz, Luis. 2000. «ERP, Guía práctica para la selección e implantación». *Edición Gestión*. http://korazzaejecutivos.com/images/erp.pdf.

MyOdoo. 2017. «Manage your business at your fingertips. The power of Odoo features in one mobile app. Access your company data everywhere.» https://www.myodoo.com.

REFERENCIAS BIBLIOGRÁFICAS

Nikolopoulos, K., K. Metaxiotis, N. Lekatis, y V. Assimakopoulos. 2003. «Integrating industrial maintenance strategy into ERP». *Industrial Management & Data Systems* 103 (3): 184-91. doi:10.1108/02635570310465661.

Nosseir, Ann, Derek Flood, Rachel Harrison, y Osman Ibrahim. 2012. «Mobile development process spiral». En *Computer Engineering & Systems (ICCES), 2012 Seventh International Conference on*, 281–286. IEEE. http://ieeexplore.ieee.org/abstract/document/6408529/.

Odoo. 2017. «Odoo Mobile - Mobile apps for Android and IOS». https://www.odoo.com/es_ES/event/odoo-experience-2015-edition-2015-06-03-2015-06-05-304/track/odoo-mobile-mobile-apps-for-android-and-ios-243.

Odoo Mobile. 2017. «Odoo Mobile Documentation (V2.3.0)». http://odoo-mobile-doc-v2.readthedocs.io/en/latest/.

OpenERP Spain. 2010. «¿Que es OpenERP? ERP 100% Libre». *Odoo - OpenERP - ERP, CRM, MRP, SGA 100% Libre - | Sin Licencias*. diciembre 3. http://openerpspain.com/openerp/que-es-openerp/.

Pedrozo Petrazzini, Gabriel Osmar. 2012. «Monografía: Sistemas Operativos en Dispositivos Móviles».

Pimienta, Pedro. 2014. «Tipos de aplicaciones móviles y sus características. – De Idea a App». https://deideaaapp.org/tipos-de-aplicaciones-moviles-y-sus-caracteristicas/.

Pressman, Roger S. 2007. *Software engineering: a practitioner's approach*. 7th Edition. McGraw-Hill. https://books.google.com/books?hl=es&lr=&id=bL7QZHtWvaUC&oi=fnd&pg=PR27&dq=software+engineering+a+practitioner%27s+approach+2007&ots=O6CbaSrQae&sig=6eu8by37DL9r66CWTMd67eP5Yzs.

Real Academia de Ingeniería. 2017. «DEI 1.0 Diccionario Español de Ingeniería». http://diccionario.raing.es/.

Rodríguez, Jorge. 2006. *Pruebas unitarias*. Marzo. http://blog.continuum.cl/wp-content/uploads/2008/08/pruebas-unitarias.pdf.

Rondón, Y., L. Domínguez, y A. Berenguer. 2011. «Diseño de la base de datos para sistemas de digitalización y gestión de medias». *Revista de Informática Educativa y Medios Audiovisuales* 8 (15): 17–25.

Rumbaugh, JamesBOOCH, GRADY JACOBSON, IvARJames Rumbaugh, Ivar Jacobson, y Grady Booch. 2000. *El lenguaje unificado de modelado: manual de referencia*. Addison Wesley,. http://www.sidalc.net/cgi-bin/wxis.exe/?IsisScript=UCC.xis&method=post&formato=2&cantidad=1&expresion=mfn=049894.

REFERENCIAS BIBLIOGRÁFICAS

Sánchez-Torres, Jenny Marcela, Mayda Patricia González-Zabala, y María Paloma Sánchez Muñoz. 2013. «La Sociedad de la Información: Génesis, Iniciativas, Concepto y su Relación con Las TIC». *Revista UIS Ingenierías* 11 (1). http://revistas.uis.edu.co/index.php/revistauisingenierias/article/view/3201.

Santiago, Raúl, Susana Trabaldo, Mercedes Kamijo, y Álvaro Fernández. 2015. *Mobile learning: nuevas realidades en el aula*. Editorial Oceano. https://books.google.com/books?hl=es&lr=&id=AULhBgAAQBAJ&oi=fnd&pg=PT113&dq=Raul+et+al.+(2015).+Mobile+learning:+nuevas+realidades+en+el+aula.&ots=k9nk78oF2M&sig=1u9b6KVLeibV9eEAsK7xCGtgCqA.

Sebesta, Robert W., y Soumen Mukherjee. 2002. *Concepts of programming languages*. Vol. 281. Addison-Wesley Reading. http://www.scis.nova.edu/~willsmit/MMIS%20610%20Summer%202005.pdf.

Sommerville, Ian. 2010. *Software engineering*. Pearson. http://cds.cern.ch/record/2131750.

TRANSCARGO. 2017. «Sitio oficial de TRANSCARGO». www.transcargo.transnet.cu.

Vermeulen, Al. 2000. *The Elements of Java (TM) Style*. Vol. 15. Cambridge University Press. https://books.google.com/books?hl=es&lr=&id=Z2CKvVZjPPEC&oi=fnd&pg=PR9&dq=Vermeulen.+The+Elements+of+Java(TM)+Style.+New+York+:+Cambridge+University+Press,+2001&ots=8Y-kVsDm8W&sig=v3scrtkLjm3E58sGLVFPkiUo-_I.