**Universidad de las Ciencias Informáticas**

**Facultad 6**


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

**Título:**

# Cliente para dispositivos móviles de la Tienda de Aplicaciones Android del Caribe

**Autor:** Leonardo Quesada Cruz

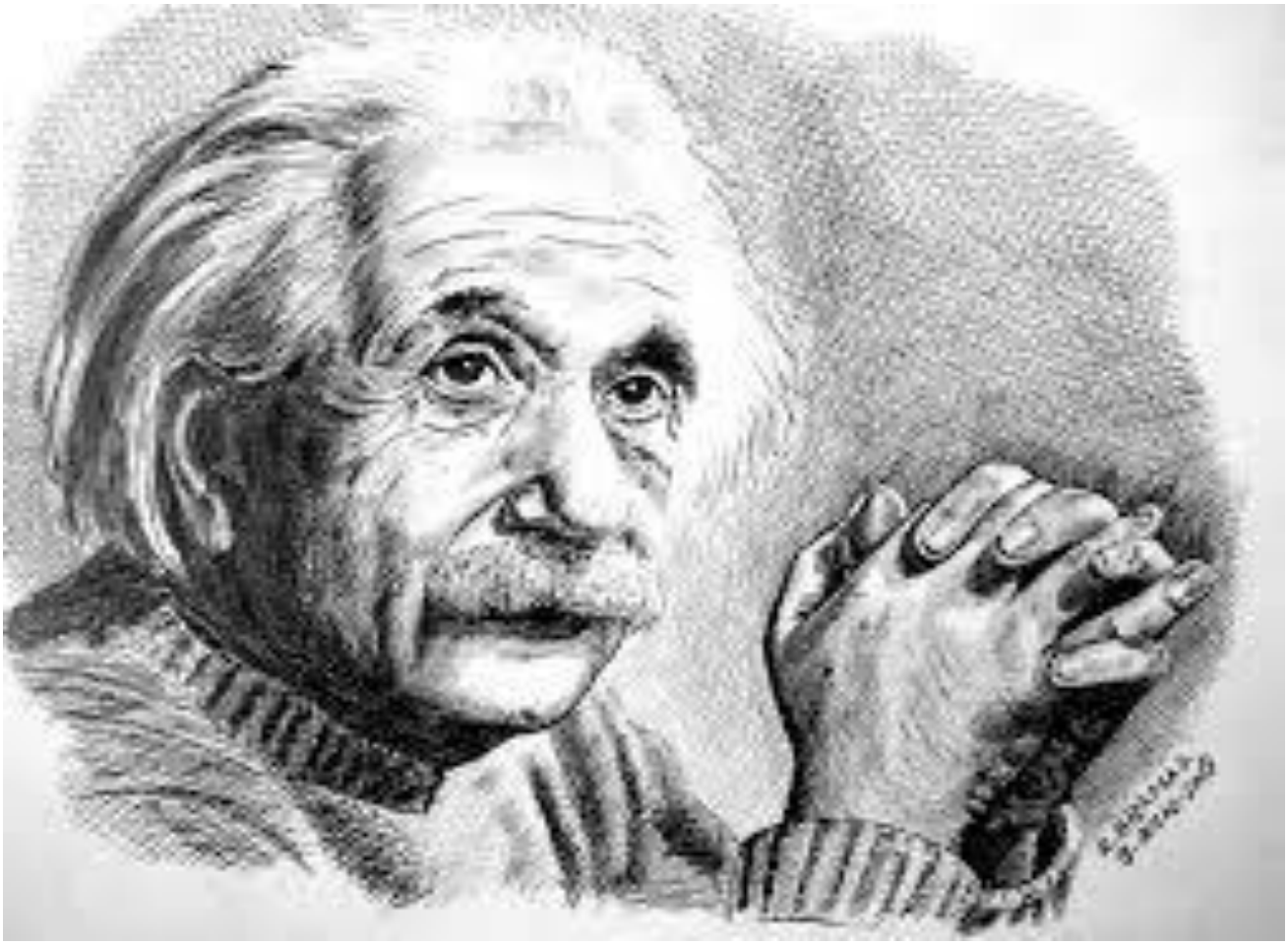
**Tutores:** Ing. José Gustavo Suárez Matilla

Ing. Miriela Velázquez Arias

**Co-tutora:** Ing. Celia Indira Hidalgo Tagle

La Habana, julio de 2016

"Año 58 de la Revolución"



*"Hay una fuerza motriz más poderosa que el vapor, la electricidad y la energía atómica: la voluntad".*

*Albert Einstein*

## Declaración de autoría

Declaro ser el autor del trabajo de diploma titulado “Cliente para dispositivos móviles de la Tienda de Aplicaciones Android del Caribe” y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales del mismo, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año 2016.

| | |
|---|---|
| _____ | _____ |
| Autor | Tutor |
| Leonardo Quesada Cruz | Ing. José Gustavo Suárez Matilla |
| _____ | _____ |
| Tutora | Co-Tutora |
| Ing. Miriela Velázquez Arias | Ing. Celia Indira Hidalgo Tagle |

## Datos de contacto

**Tutor: Ing. José Gustavo Suárez Matilla**

Graduado de Ingeniería en Ciencias Informáticas en el 2012 en la Universidad de las Ciencias Informáticas. Ha participado directamente en la migración y soporte en varias empresas e instituciones de Cuba: Prensa Latina y la Unión de Jóvenes Comunistas a nivel Nacional. Ha tutorado tesis de pregrado relacionadas con la administración y migración de servidores: Módulo para la configuración de servidores MySql en HMAST. Actualmente se desempeña como Especialista en desarrollo en aplicaciones móviles Android y jefe del laboratorio de desarrollo de aplicaciones Android DroidLab, en el Centro de Software Libre.

Correo electrónico: jgsuarez@uci.cu

**Tutora: Ing. Miriela Velázquez Arias**

Graduada de Ingeniería en Ciencias Informáticas en el año 2014 en la Universidad de las Ciencias Informáticas. Forma parte del departamento Servicios Integrales en Migración Asesoría y Sistemas del Centro de Software Libre donde se desempeña como analista y administrador de la calidad del proyecto HMAST. Cuenta con publicaciones en el área de la biometría, específicamente asociadas a las huellas dactilares.

Correo electrónico: mirielav@uci.cu

**Co-Tutora: Ing. Celia Indira Hidalgo Tagle**

Graduada de Ingeniería en Ciencias Informáticas en el año 2015 en la Universidad de las Ciencias Informáticas. Forma parte del departamento de Componentes del Centro de Tecnologías para la Formación (FORTES), donde se desempeña como analista y desarrolladora del proyecto Plataforma de Recursos de la Editorial Félix Varela.

Correo electrónico: cihidalgo@uci.cu

## Dedicatoria

A todos mis viejucos, espero puedan estar orgullosos.

## Agradecimientos

A mis tutores por el apoyo que me brindaron en todo este curso.

Al tribunal y al oponente por sus oportunas críticas y sugerencias.

A Celia y a David, sin ustedes no sé si estaría aquí hoy, graduándome.

A mis padres, nunca podré pagarles todo lo que han hecho por mí, pero seguiré intentando que se sientan orgullosos de mí, tal como yo lo estoy de ustedes, de ser su hijo, los quiero.

A Mónica, mi hermanita, no importa cuánto nos fajemos, sigo agradecido de tenerte en mi vida.

A Mami Martha, un beso, no puedo decirte cuán importante eres para mí.

A mis tíos Omar, Héctor, Martha y Teresa, aquí estoy y mucho se los debo a ustedes. A mis primos que sé que están disfrutando este momento tanto como yo.

A todos mis viejucos y viejucas, ojalá pudieran estar aquí en este día.

A Xavier y Yuri, los dos hermanos que me dio la vida, espero sigan conmigo por más tiempo.

A Jorgito el Niche, por aguantarme estos cinco años, por estar siempre cuando lo he necesitado.

A Annareya, por tu sazón, por escucharme, por estar.

A Mayi, por seguir ahí para mí, sin importar nada, me alegro de tenerte como amiga.

A los personajes con los que he pasado esta universidad, gracias por hacerla más interesante y divertida: Carlos, Eduardo, Marlon, Leosdany, Lokiño, Michel, Alejandro, Fortún, Lijandy, y por último, pero no menos importante, a Migue.

A los socios de la facultad 1: El Chino, Adrián y Justiz, por compartir tantos buenos momentos.

A la gente del lab, en especial a Adrián, Ivelisse, Collazo, Bia, Stephany, Abel, David el Friki, Ernesto y Ronny.

A mis compañeros de aula y del año, con los cuales he compartido estos cinco años. A Yanitza, Yanara, Claudia (Pini), Elián, Víctor, Henry, Ronald y los Zecas: Edel, César y Rafa.

A Jose y a Ilsen, por seguir presentes aún después de diez años, en la secundaria, el pre y ahora en la universidad.

A todos los profes que tuve durante la carrera, sobre todo a Yanelis, Liudmila, Quintero y el Jeem.

A Rachel, Jessica, Irismay y Magdiel, quienes en poco tiempo de conocernos ya se merecen una parte de estos agradecimientos, no cambien.

Siempre se quedan nombres, pero no los olvido a ninguno, así que discúlpenme si no los mencioné.

En resumen, a todos aquellos que han hecho posible que haya llegado hasta aquí, GRACIAS.

## Resumen

El auge de las tecnologías móviles en la actualidad trae consigo que gran parte de los desarrolladores opten por esta rama (Statista, 2015). Dentro de los sistemas operativos asociados a esta tecnología resalta Android por sus características. Teniendo en cuenta este panorama, en la Universidad de las Ciencias Informáticas surgió el Laboratorio de Android (DroidLab) para el desarrollo de aplicaciones para este sistema operativo. Este laboratorio está optando por insertarse como un mHub en el Proyecto de Innovación Móvil para el Caribe (CMIP por sus siglas en inglés) que llevan a cabo el Banco Mundial y el gobierno de Canadá, el cual cuenta con varios laboratorios de desarrollo en el Caribe, y contará con una tienda virtual donde serán publicadas las aplicaciones. Por tal motivo se decidió desarrollar una aplicación Android para dispositivos móviles que funcione como cliente de dicha tienda virtual.

Para regir el proceso de desarrollo de la aplicación se utilizó como metodología de desarrollo de software AUP-UCI, como lenguaje de modelado UML, como herramienta de modelado Visual Paradigm, como lenguaje de programación Java y como entorno de desarrollo Android Studio. Como resultado se obtuvo la aplicación Cliente para dispositivos móviles de la Tienda de Aplicaciones Android del Caribe, la cual permite descargar, instalar, desinstalar y actualizar aplicaciones desde la Tienda de Aplicaciones Android del Caribe desarrollada por DroidLab. Fueron ejecutadas pruebas a la solución, proceso que culminó con la aceptación por parte del cliente.

**Palabras clave**

Android, DroidLab, tecnología móvil, tienda virtual

## Abstract

*The rise of mobile technologies currently brings with it that much of developers choose this branch (Statista 2015). Among the operating systems associated to this technology Android highlights due to its features. Given this scenario, in the University of Informatic Sciences the Android Laboratory (DroidLab) emerged for developing applications for this operating system. This laboratory aims to be inserted as a mHub in the Caribbean Mobile Innovation Project (CMIP) that carry out the World Bank and the government of Canada, which has several development laboratories in the Caribbean and will feature a virtual store where applications will be published. For this reason it was decided to develop an Android application for mobile devices that function as a client of this store.*

*To govern the process of application development was used the AUP-UCI software development methodology, UML as modeling language, Visual Paradigm as modeling tool, Java as programming language and Android Studio as the development environment.*

*As a result the client application named CaribbeanStore was obtained, which allows to download, install, update and uninstall applications from the Caribbean Android Applications Store developed by DroidLab. Testing were applied to the solution, process that culminated with the acceptance by the customer.*



**Keywords**

*Android, DroidLab, mobile technology, virtual store*

# Índice de Contenido

## Índice de Figuras

## Índice de Tablas

## Introducción

El avance de las Tecnologías de la Información y la Comunicación (TIC) se produce a un ritmo acelerado. La tecnología móvil no queda exenta de dicho avance y, debido a los disímiles beneficios que ofrece, ha alcanzado un elevado porcentaje de popularidad a nivel mundial. Los más recientes dispositivos móviles están dirigidos a satisfacer diversas necesidades y gustos, por lo que el mercado asociado al desarrollo de aplicaciones para estos se encuentra en constante auge.

Los dispositivos móviles permiten, entre otras funciones, reproducir música y videos, tomar fotos y películas y consultar Internet; así como enviar y recibir correos electrónicos, mensajes de texto y gráficos. Ofrecen facilidades entre las que se incluyen: agenda personal, calculadora, juegos electrónicos e incluso servicios GPS. Un estudio realizado en el año 2015, reveló que el 85% de los usuarios considera a los dispositivos móviles como herramientas esenciales en su cotidianidad; un porcentaje que se eleva al 90% en el caso de los jóvenes de entre 18 y 24 años (PuroMarketing, 2015). Se puede afirmar que los últimos avances en los dispositivos móviles hacen que se conviertan en herramientas imprescindibles en la vida moderna (Shin, 2014), llegando incluso en algunos casos a la dependencia.

Entre los sistemas operativos asociados a la tecnología móvil destaca Android. En un estudio realizado recientemente por *International Data Corporation* (IDC) se concluyó que un 82,8% de los usuarios hace uso de este sistema (IDC, 2015). Android está basado en una versión modificada del núcleo Linux. Inicialmente fue desarrollado por Android Inc., una pequeña empresa que posteriormente fue comprada por Google. En la actualidad lo desarrollan los miembros de la *Open Handset Alliance* (Núñez, 2013).

La Universidad de las Ciencias Informáticas (UCI) no está aislada con respecto al desarrollo de aplicaciones en esta esfera, por lo que surge el Laboratorio de Android (DroidLab) para el desarrollo de aplicaciones para este sistema operativo. Este laboratorio está optando por insertarse como un mHub[1], en este caso en Android, en el CMIP[2] llevado a cabo por el Banco Mundial y el gobierno de Canadá.

CMIP pretende crear un ecosistema de desarrollo de aplicaciones móviles en el Caribe. Cuenta con varios laboratorios de desarrollo en el área, pero no con una plataforma propia donde los desarrolladores y los usuarios puedan publicar sus aplicaciones, por lo que se publican en la tienda de aplicaciones *Google*

[1] mHub: Laboratorio para desarrollo de aplicaciones móviles y capacitación para profesionales.

[2] CMIP: Del inglés *Caribbean Mobile Innovation Project*. Traducido al español Proyecto de Innovación Móvil para el Caribe.

*Play Store*[3], donde tienen que pagar por registrarse y pierden parte de las ganancias de las ventas de las aplicaciones. Por tal razón, DroidLab decidió iniciar un proyecto encaminado al desarrollo de una plataforma web que funcione en el área del Caribe como una tienda virtual de aplicaciones móviles.

La vía de comunicación concebida entre los usuarios y la tienda es a través de un navegador web. Esto implica que las aplicaciones no puedan ser gestionadas desde un dispositivo móvil, ya que con un navegador web como intermediario solo es posible descargar las aplicaciones desde la tienda. Esto conlleva a que los procesos de instalación, actualización y desinstalación resulten un tanto engorrosos.

Puede afirmarse entonces que resulta estratégico contar con un mecanismo que favorezca la interacción de los dispositivos móviles con sistema operativo Android con la tienda virtual de aplicaciones que se desarrolla como iniciativa de DroidLab. Dicho mecanismo debe proveer las mismas funcionalidades de la tienda, pero eliminando el uso de un navegador web como intermediario y permitiendo gestionar las aplicaciones desde el propio dispositivo móvil del usuario.

Partiendo de la **situación problemática** planteada surge el siguiente **problema de la investigación**: ¿Cómo gestionar las aplicaciones de la Tienda de Aplicaciones Android del Caribe desde un dispositivo móvil con sistema operativo Android?

Se establece como **objeto de estudio** las aplicaciones clientes para dispositivos móviles, enmarcando en el **campo de acción** las aplicaciones clientes de las tiendas de aplicaciones para dispositivos móviles.

Para dar solución al problema de investigación planteado se define como **objetivo general** desarrollar una aplicación cliente para dispositivos móviles con sistema operativo Android que permita descargar, instalar, desinstalar y actualizar aplicaciones desde la Tienda de Aplicaciones Android del Caribe desarrollada por DroidLab.

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

- Definir los elementos teóricos relacionados con las aplicaciones clientes de una tienda de aplicaciones para dispositivos móviles.
- Desarrollar la aplicación Cliente para dispositivos móviles de la Tienda de Aplicaciones Android del Caribe.
- Validar el funcionamiento de la aplicación Cliente para dispositivos móviles de la Tienda de Aplicaciones Android del Caribe.

[3] *Google Play Store*: Tienda de Aplicaciones Android de Google.

Para el cumplimiento del objetivo general planteado se utilizarán los siguientes métodos de investigación:

**Teóricos**

- **Analítico-Sintético**: se utiliza para realizar un estudio los elementos más significativos relacionados con las aplicaciones clientes de una tienda de aplicaciones para dispositivos móviles.
- **Análisis Histórico-Lógico**: se utiliza para el estudio del estado del arte de las aplicaciones clientes existentes que permiten la interacción con tiendas de aplicaciones, destacando las ventajas y desventajas de las mismas.
- **Modelación**: se utiliza en la realización de los diagramas correspondientes a los modelos de análisis, diseño e implementación del Cliente para dispositivos móviles de la Tienda de Aplicaciones Android del Caribe.

**Empíricos**

- **Observación**: se utiliza para analizar el comportamiento de las aplicaciones estudiadas.

El presente trabajo de diploma consta de tres capítulos estructurados de la siguiente forma:

- **Capítulo 1. Fundamentación teórica**: se realiza el estudio del estado del arte de las aplicaciones clientes de tiendas de aplicaciones para dispositivos móviles, se abordan elementos teóricos vinculados a la investigación, y se analiza la metodología de desarrollo a emplear, así como las tecnologías y herramientas a utilizar en el desarrollo de la aplicación.
- **Capítulo 2. Análisis y diseño de la propuesta de solución**: se describe la solución propuesta y se ofrecen detalles de los principales aspectos relacionados con su diseño. Se explica la dinámica de la aplicación a través de las historias de usuario y otros modelos auxiliares.
- **Capítulo 3. Implementación y prueba**: en este capítulo se describen los artefactos relacionados con la implementación y las pruebas realizadas a la aplicación, con el objetivo de validar su correcto funcionamiento y su correspondencia con los requerimientos especificados previamente.

# Capítulo 1: Fundamentación teórica

## 1.1 Introducción

En el presente capítulo se presenta la definición del marco teórico de la investigación. Se abordan los conceptos relacionados a las tiendas de aplicaciones y a los clientes móviles utilizados por las mismas. Se analizan las soluciones existentes a problemas similares. Además, se precisan sus principales características y las posibles limitantes que pudieran imposibilitar su utilización como solución al problema a resolver de la investigación. Se hace referencia a la metodología de desarrollo escogida para el desarrollo de la solución. Por último, se realiza un estudio de las tecnologías y herramientas que serán necesarias para dicho proceso.

## 1.2 Conceptos asociados al dominio del problema.

**Tienda Virtual o En Línea:** se puede describir como a una plataforma de comercio convencional que se vale de un sitio web para realizar sus ventas y transacciones. Por lo general, las compras en una tienda virtual se pagan con tarjeta de crédito en el mismo sitio web y luego los productos son enviados por correo. Sin embargo, se pueden utilizar otros medios de pago como transferencias bancarias, cupones de pago, entre otros. En la mayoría de los casos la tienda virtual suele requerir que los usuarios se registren (ingresando sus datos) antes de poder realizar una compra (Headways Media, 2015).

**Tienda de aplicaciones para dispositivos móviles:** es una plataforma de distribución digital de aplicaciones móviles, así como una tienda en línea. Esta plataforma permite a los usuarios descargar aplicaciones desde su interfaz. Las aplicaciones se encuentran disponibles tanto de forma gratuita como con costo (Google Support, 2014).

**Repositorio de aplicaciones:** Conjunto de aplicaciones alojadas en uno o varios servidores que, haciendo uso de un programa, pueden ser descargadas e instaladas (Ortiz, 2014).

**Aplicación móvil:** Aplicación informática diseñada para ser ejecutada en teléfonos inteligentes, tabletas y otros dispositivos móviles. Por lo general se encuentran disponibles a través de plataformas de distribución operadas por las compañías propietarias de los sistemas operativos móviles como Android, iOS, BlackBerry OS, Windows Phone, entre otros (Cuello y Vittone, 2015).

**Aplicación cliente para tienda de aplicaciones móviles:** Aplicación móvil que interactúa con una tienda de aplicaciones móviles. Elimina al navegador como intermediario entre dicha tienda y el dispositivo móvil. Consume los servicios que provee la tienda, los cuales posibilitan listar, filtrar y ordenar aplicaciones, y a

su vez ofrece funcionalidades de descarga, instalación, actualización y desinstalación de aplicaciones en el dispositivo en que se encuentre instalada.

## 1.3 Estudio de aplicaciones similares

Para contribuir a la elaboración del marco teórico de la investigación y la posterior identificación de los requisitos funcionales, se analizó un grupo de soluciones existentes a problemas similares. De las mismas se tomaron en cuenta sus principales características, ventajas y desventajas, las cuales se mencionan a continuación.

***Google Play Store***

Es el cliente móvil de la tienda de aplicaciones para Android de Google. Generalmente se instala por defecto con el sistema operativo, pero es posible que en algunos dispositivos con versiones modificadas de Android se haya omitido su instalación, por lo que se hace necesario instalarla manualmente. Recomienda aplicaciones en base a las aplicaciones que más se han descargado. Almacena aplicaciones gratuitas y de pago (Android Developers, 2014). Ofrece más de 30 categorías, y dentro de cada una las aplicaciones se clasifican según una combinación de calificaciones, reseñas, descargas, país y otros factores. La búsqueda permite a los usuarios encontrar una aplicación rápidamente. Ofrece vínculos directos a aplicaciones en forma de sugerencias; en los resultados los usuarios encuentran en primer lugar las aplicaciones más relevantes y populares (Google, 2015). Permite acceder únicamente a la tienda de aplicaciones Android de Google, o sea, que no puede ser adoptada para dar solución al problema de la investigación, pues no se dispone de su código fuente.

**AppStore**

Es el cliente móvil de la tienda de aplicaciones móviles para dispositivos con el sistema operativo iOS, desarrollada y mantenida por Apple Inc. Permite a los usuarios descargar aplicaciones en dispositivos tales como iPhone, iPod Touch e iPad (Apple Inc., 2015). Las aplicaciones están divididas en diferentes categorías que, a su vez, están integradas por subcategorías, lo cual facilita la búsqueda. Provee una opción de búsqueda que incluye en sus resultados las tendencias más populares. Presenta aplicaciones destacadas, las cuales contienen una etiqueta de selección del editor, lo que certifica la calidad de las funciones que ofrece y garantiza una buena relación entre valor y precio. Los dispositivos con control parental impedirán que un usuario sin autorización, como un niño, compre aplicaciones o haga transacciones dentro de las aplicaciones sin permiso, pero se enviará una notificación a los padres con la

petición de compra (Qore, 2014). Utiliza un sistema operativo no compatible con Android, por lo que no puede ser adoptada para dar solución al problema de la investigación.

**Windows Phone Store**

Es un cliente móvil que se conecta a *Windows Phone Store* (anteriormente *Windows Phone Marketplace*), tienda desarrollada por Microsoft para su sistema operativo móvil Windows Phone (Microsoft, 2015). Permite a los usuarios descargar aplicaciones que han sido desarrolladas por terceros. Emplea una interfaz de usuario que se presenta en una vista panorámica donde el usuario puede filtrar por categorías y títulos, ver los elementos destacados y obtener detalles con calificaciones, comentarios, capturas de pantalla e información de precios. Tiene soporte para las compras con tarjeta de crédito, la facturación del operador y el contenido con publicidad. Cuenta con una opción de "probar-antes-de-comprar", donde el usuario tiene la opción de descargar una prueba o demostración de una aplicación comercial. Tiene 61 categorías divididas en 16 categorías principales y 25 subcategorías. Las aplicaciones solo se pueden colocar en una categoría (Windows Central, 2015). Utiliza un sistema operativo no compatible con Android, por lo que no puede ser adoptada para dar solución al problema de la investigación.

***1Mobile Market***

Aplicación cliente para dispositivos móviles con sistema operativo Android para descargar aplicaciones. Ofrece recomendaciones diarias de aplicaciones al igual que *Google Play Store*. Facilita la búsqueda de aplicaciones a través de un filtro por categorías. Entre sus principales características están la reanudación de la descarga si esta es interrumpida y la actualización de las notificaciones. Permite realizar copias de seguridad a las aplicaciones instaladas en el dispositivo móvil en cuestión, moverlas desde el almacenamiento interno de dicho dispositivo hacia una tarjeta SD, y mostrar la capacidad de almacenamiento ocupada y disponible. Permite acceder únicamente a su página oficial, o sea, que no puede ser adaptada para dar solución al problema de la investigación, pues no se dispone de su código fuente (1 Mobile Market, 2015).

**Aptoide**

Aplicación cliente para dispositivos móviles con sistema operativo Android que sirve de conexión entre el dispositivo instalado y el servidor de los repositorios de los usuarios de Aptoide en la Web. La función principal de esta aplicación es ofrecer aplicaciones Android gratuitas. Permite a los usuarios registrados agregar su propio repositorio de aplicaciones y a la vez acceder a los ya existentes (Aptoide, 2015b). Incluye la interacción con redes sociales como Facebook y Twitter. Presenta listados de actualizaciones

para las aplicaciones instaladas en el dispositivo móvil y para mostrar las aplicaciones más votadas. Es una alternativa que libera su código fuente (Aptoide, 2015a).

***F-Droid***

Es un cliente móvil que se conecta al repositorio de aplicaciones móviles para Android del mismo nombre. Funciona de manera similar a la tienda *Google Play Store*, pero solo ofrece aplicaciones libres de pago y de código abierto (F-Droid, 2015a). Actualiza de forma automática las aplicaciones descargadas con una versión superior a otras previamente instaladas. No utiliza sistema de autenticación. Es una alternativa que libera su código fuente. La interfaz principal muestra tres pestañas. La primera presenta la lista con todas las aplicaciones disponibles en F-Droid. La segunda aborda las aplicaciones instaladas en el dispositivo móvil. Por último, la tercera pestaña lista las actualizaciones que haya disponibles para dichas aplicaciones instaladas (F-Droid, 2015b).

**AndroidUCLV**

Aplicación cliente para dispositivos móviles con sistema operativo Android desarrollada en la Universidad Central de Las Villas (Collazo Linares, 2014b). Se basa en el proyecto libre y de código abierto F-Droid, por lo que presenta todas las ventajas de este. Permite instalar una aplicación en el dispositivo móvil y compartirla por las vías disponibles en el mismo. Muestra notificaciones de las actualizaciones existentes. Permite ejecutar desde su interfaz una aplicación instalada. (Collazo Linares, 2014a).

Se hace necesario aclarar que la concepción de la Tienda de Aplicaciones Android del Caribe abarca, para su desarrollo, tres elementos fundamentales: un servidor de aplicaciones, el cual funcionará como repositorio para las mismas; una capa de servicios intermedia que proveerá los servicios web necesarios para la comunicación con dicho servidor; y un componente referente a los clientes que utilizarán esa capa de servicios para mostrar los resultados a los usuarios, ya sea a través de la web o empleando un cliente para dispositivos móviles, como es el caso.

En el caso de estas últimas tres soluciones, contienen embebida la capa de servicios en sus respectivos servidores y utilizan para el manejo de la información ficheros en formato XML[4], lo que trae consigo que se haga más engorrosa la transformación, lectura y escritura de los datos a medida que aumenta el tamaño de estos ficheros, ralentizando el funcionamiento.

A partir de lo antes expuesto se decide que, a pesar de disponer de su código fuente, no pueden adoptarse estas soluciones, tomando en consideración que requeriría más esfuerzo modificar alguna de

[4] XML: Del inglés eXtensible Markup Language. Traducido al español Lenguaje de Marcas Extensible.

ellas para adaptarla como solución al problema de la investigación que desarrollar una nueva solución que se adecúe a la estructura concebida.

### *1.3.1 Conclusiones del estudio de las aplicaciones existentes.*

Partiendo del análisis realizado sobre los clientes móviles para tiendas de aplicaciones existentes se concluye lo siguiente:

- No puede emplearse ninguna de estas aplicaciones como solución al problema de la investigación debido a las desventajas que presenta cada una de ellas.
- Pueden tomarse en cuenta elementos referentes al diseño de interfaz de usuario para el desarrollo de la solución.
- Se identificaron funcionalidades comunes en estas aplicaciones: descargar, instalar, actualizar y desinstalar una aplicación, realizar búsquedas, filtrar el listado de aplicaciones por categorías y ordenarlo en cuanto a varios criterios.
- Teniendo en cuenta lo antes referido se decide desarrollar una aplicación cliente de la Tienda de Aplicaciones Android del Caribe para dispositivos móviles con sistema operativo Android, utilizando para ello los servicios web que proveerá dicha tienda.

## 1.4 Metodología de desarrollo

Una metodología de desarrollo, en ingeniería de software, es un conjunto de herramientas, técnicas, procedimientos y soporte documental encaminados a estructurar, planificar y controlar el proceso de desarrollo de forma organizada y lógica, que tienen como objetivo apoyar a los desarrolladores en la creación de un nuevo software (Kaisler, 2005).

En la presente investigación no se hace necesario realizar un estudio a profundidad de las metodologías existentes, esto se debe fundamentalmente a que se utilizará la metodología definida por el centro DroidLab para el desarrollo de proyectos. La misma consiste en una versión desarrollada por la UCI basada en la metodología de desarrollo de software AUP[5]. A continuación se muestran las características fundamentales de la metodología AUP y de la variante de la misma para la UCI.

[5] AUP: Del inglés *Agil Unified Process*. Traducido al español Proceso Unificado Ágil.

### *1.4.1 Metodología de desarrollo de software AUP*

La metodología de desarrollo de software AUP es una versión simplificada de la metodología de desarrollo RUP[6]. Describe de una manera fácil y simple de entender la forma de desarrollar aplicaciones de software de negocio empleando técnicas ágiles y conceptos que se mantienen válidos en RUP (Ambler, 2014). Aplica técnicas ágiles que incluyen:

- Desarrollo dirigido por pruebas.
- Modelado Ágil.
- Gestión de cambios ágil.
- Refactorización de base de datos para mejorar la productividad.

AUP se preocupa especialmente por la gestión de riesgos. Propone que los elementos con alto riesgo obtengan prioridad en el desarrollo y sean abordados en etapas tempranas del mismo.

Establece cuatro fases que transcurren de manera consecutiva:

1. **Inicio:** El objetivo de esta fase es obtener una comprensión común cliente-equipo de desarrollo del alcance del nuevo sistema y definir una o varias arquitecturas candidatas para el mismo.
2. **Elaboración:** El objetivo es que el equipo de desarrollo profundice en la comprensión de los requisitos del sistema y en validar la arquitectura.
3. **Construcción:** Durante la fase de construcción el sistema es desarrollado y probado al completo en el ambiente de desarrollo.
4. **Transición:** El sistema se lleva a entornos donde se somete a pruebas de validación y aceptación; finalmente se despliega en los sistemas de producción.

### *1.4.2 Metodología de desarrollo de software AUP versión UCI*

La Universidad de las Ciencias Informáticas (UCI) desarrolló una versión de la metodología de desarrollo de software AUP, con el fin de crear una metodología que se adapte al ciclo de vida definido por la actividad productiva de la Universidad. Esta versión decide mantener para el ciclo de vida de los proyectos la fase de Inicio, pero modificando el objetivo de la misma, y se unifican las restantes fases de la metodología de desarrollo de software AUP en una sola, denominada Ejecución, y agregándose también una nueva fase denominada Cierre (Sánchez, 2015).

[6] RUP: Del inglés *Rational Unified Process*. Traducido al español Proceso Unificado Racional.

Tabla 1. Fases de la variante de AUP para la UCI

| Fases AUP | Fases Variación AUP-UCI | Objetivos de las fases(Variación AUP-UCI) |
|---|---|---|
| Inicio | Inicio | Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto. |
| Elaboración<br>Construcción<br>Transición | Ejecución | En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, se obtienen los requisitos, se establecen la arquitectura y el diseño, se implementa y se libera el producto. |
| | Cierre | En esta fase se analizan tanto los resultados del proyecto como la ejecución y se realizan las actividades formales de cierre del proyecto. |

## 1.5 Herramientas y tecnologías

El proceso de desarrollo de software se apoya en el uso de diferentes herramientas y tecnologías, las cuales, unidas a la metodología seleccionada, conforman el ambiente de desarrollo de un sistema.

### *1.5.1 Lenguaje de modelado*

**UML v2.0**

UML[7] es un lenguaje de modelado de sistemas de software. Abarca la estructura de las aplicaciones, su comportamiento y arquitectura, procesos de negocio y datos. Incluye todo el proceso de desarrollo de software (Unified Modeling Language, 2015). Utilizado para especificar, construir, visualizar y documentar los artefactos de un sistema de software orientado a objetos (Object Management Group, 2005).

[7] UML: Del inglés Unified Modeling Language. Traducido al español Lenguaje Unificado de Modelado.

### *1.5.2 Lenguaje de programación*

**Java v1.7.0**

Es un lenguaje de programación de propósito general. Fue creado basado en el paradigma de la programación orientada a objetos, diseñado específicamente para tener tan pocas dependencias de implementación como sea posible (Oracle Corporation, 2015). Es posible ejecutarlo en diferentes sistemas operativos y tiene como máxima permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo (conocido en inglés como WORA, o write once, run anywhere), lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra. Es fácil de usar y aprovecha ventajas de otros lenguajes orientados a objetos como C++ (Gosling, 2000).

### *1.5.3 Herramienta de modelado*

**Visual Paradigm v8.0**

Es una herramienta CASE[8]. La misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. Soporta, entre otros, el lenguaje de modelado UML. Permite el análisis, diseño, codificación, prueba y despliegue de aplicaciones. Está disponible para Windows y Linux (Visual Paradigm, 2015). Entre sus características se encuentra la capacidad de realizar ingeniería directa e inversa, generación de código, generación de bases de datos, la incorporación de varios idiomas y una licencia gratuita y comercial, así como un diseño centrado en casos de uso y enfocado al negocio que genera un software de mayor calidad.

### *1.5.4 Herramienta para el diseño de prototipos de interfaz*

**Balsamiq Mockups v2.1.13**

Es una herramienta que reproduce la experiencia de realizar bocetos en una pizarra, a través de un ordenador. Permite el diseño de aplicaciones de escritorio, para la web y orientadas a dispositivos móviles. Cuenta con gran cantidad de componentes visuales para la construcción de Interfaces de Usuario, íconos y elementos reutilizables tales como plantillas. Incluye la funcionalidad de exportar a los formatos PDF y PNG, y una interfaz sencilla que posibilita el diseño de prototipos simplemente arrastrando

[8] CASE: Del inglés Computer Aided Software Engineering. Traducido al español Ingeniería de Software Asistida por Computación.

componentes hacia una pantalla. Cuenta con soporte para los sistemas operativos Windows y MacOS, Linux no lo soporta pero el ejecutable de Windows (.exe) puede emularse a través de la aplicación Wine (Balsamiq, 2016).

### *1.5.5 Herramientas y tecnologías de desarrollo*

**Retrofit v1.9.0**

Es una biblioteca desarrollada para la implementación de aplicaciones Android que consuman servicios REST[9]. Provee un marco de trabajo para la autenticación e interacción con APIs[10] y el envío de peticiones de red que utilizan el protocolo HTTP[11]. Esta biblioteca realiza la descarga de datos en formato JSON[12] o XML desde la web y, una vez finalizada, los convierte a clases POJO[13], las cuales deben ser definidas previamente para cada recurso contenido en la respuesta obtenida a raíz de la petición realizada (GitHub, 2016).

**RxJava-Android v1.0.1**

Es una biblioteca para la implementación de programas asíncronos y basados en eventos mediante el uso de secuencias observables (ReactiveX, 2015). Está basada en el patrón Observer que soporta secuencias de datos y/o eventos, y operadores de adición que permiten construir estas secuencias, mientras que posibilita una abstracción de operaciones tales como la concurrencia a bajo nivel, la sincronización, estructuras de datos concurrentes y procesos de entrada/salida (Mttkay, 2013). El módulo rxjava-android contiene especificaciones de Android para RxJava. Añade un conjunto de clases para asistir en la creación de aplicaciones de este tipo para este sistema operativo (GitHub, 2015).

---

[9] REST: Del inglés *REpresentational State Transfer*. Traducido al español Transferencia de Estado Representacional.

[10] API: Del inglés *Application Programming Interface*. Traducido al español Interfaz de Programación de Aplicaciones.

[11] HTTP: Del inglés Hypertext Transfer Protocol. Traducido al español Protocolo de Transferencia de Hipertexto.

[12] JSON: Del inglés JavaScript Object Notation. Traducido al español Notación de Objetos JavaScript.

[13] POJO: Del inglés Plain Old Java Object. Traducido al español Objeto Java Plano Antiguo.

**ADA v2.4.4**

Es una biblioteca de tipo ORM[14] diseñada para simplificar el código fuente en aplicaciones Android. ADA[15] provee la generación automática de modelos de datos, a partir de clases entidades, separando el modelo del código. Soporta relaciones, herencia y dependencias entre ellas. Permite realizar copias de respaldo a las bases de datos creadas, así como el enlace de datos entre elementos de Interfaz de Usuario y las entidades. Puede utilizarse en aplicaciones Android a partir de la versión 2.2 (Mob&Me, 2013).

**Gradle v1.5.0**

Es una herramienta para automatizar la construcción de proyectos, díganse tareas de compilación, pruebas, empaquetado y el despliegue de los mismos (Gradle Inc., 2015b). Es muy flexible para la configuración. Utilizado como compilador de disímiles lenguajes de programación como Java, Scala, Python, C/C++, iOS y Android. Provee *plugins* y otras herramientas para la integración con IDEs[16] como Eclipse, Android Studio e IntelliJ. Ofrece además el manejo de dependencias, facilitando su resolución mediante repositorios como Ivy y Maven. Otras de sus ventajas son la compilación solo de los cambios y la evaluación del rendimiento de dicho proceso (Gradle Inc., 2015a).

**Maven v3.0.4**

Es una herramienta de código abierto y estandarizada para la gestión de proyectos que simplifica el proceso de construcción o compilación, pruebas, reporte y empaquetado de dichos proyectos (Varanasi y Belida, 2014). Se basa en un fichero central, pom.xml, donde se define todo lo que necesita un proyecto. Maneja las dependencias del proyecto que permite saber qué librerías y qué versiones son necesarias en el proyecto para ejecutarse. Compila, empaqueta y ejecuta las pruebas. Utiliza una matriz de repositorios remotos que le permite localizar y descargar todo lo necesario para generar un proyecto de forma transparente al desarrollador. Además de los repositorios remotos, también existe un repositorio local que es utilizado como una memoria de trabajo, evitando la descarga en las siguientes generaciones del proyecto y así reducir el tiempo que supondría volver a descargar todas las librerías (González, 2012).

---

[14] ORM: Del inglés *Object Relational Mapping*. Traducido al español Mapeo Objeto-Relacional.

[15] ADA: Del inglés Android Data Abstraction. Traducido al español Abstracción de Datos para Android.

[16] IDE: Del inglés *Integrated Development Environment*. Traducido al español Entorno de Desarrollo Integrado.

**Android Studio v1.5**

Es el IDE oficial para Android, basado en el software IntelliJ IDEA de JetBrains. Entre sus características principales se encuentra la renderización en tiempo real; la incorporación de Gradle como compilador, ofreciendo múltiples variantes para la construcción y generación de APKs[17]; la existencia de plantillas predeterminadas para crear diseños comunes de Android y otros componentes; herramientas Lint para detectar problemas de rendimiento, usabilidad y compatibilidad de versiones (Android Developers, 2015b). Ofrece además un editor visual y un editor de temas para las aplicaciones, así como refactorización específica de Android y arreglos rápidos, consola de desarrollador, consejos de optimización, ayuda para la traducción y estadísticas de uso. No menos importante, añade soporte para Android Wear y las ventajas de seguridad y optimización que brinda ProGuard, al cual incluye (Android Developers, 2015a).

**Genymotion v2.4.0**

Es un emulador para Android que incluye un conjunto de sensores y características en aras de facilitar la interacción con un entorno virtual de este sistema operativo (Genymotion, 2015a). Utilizado en el proceso de desarrollo, pruebas y demostración de aplicaciones Android. Es rápido, fácil de instalar y presenta una interfaz amigable para el usuario. Disponible para Linux, Windows y Mac OS. Es compatible con las herramientas del SDK de Android e incorpora *plugins* para su integración con Android Studio y Eclipse. Incluye además una herramienta para línea de comandos, API Java y un *plugin* para Gradle (Genymotion, 2015b).

### *1.5.6 Herramienta de pruebas*

**JUnit v4.12**

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

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

## 1.6 Conclusiones parciales

A partir del desarrollo del presente capítulo se arribó a las siguientes conclusiones:

1. El estudio de los conceptos básicos asociados al objeto de estudio posibilitó la adquisición de una mayor comprensión del problema a resolver.
2. El estudio de las soluciones existentes tanto a nivel internacional como nacional arrojó como resultado que las aplicaciones estudiadas no brindan una solución completa al problema a resolver, debido fundamentalmente a incompatibilidades con la Tienda de Aplicaciones Android del Caribe, por lo que se decidió desarrollar el Cliente para dispositivos móviles de la Tienda de Aplicaciones Android del Caribe.
3. El análisis realizado a las diferentes herramientas y tecnologías, así como de la metodología a utilizar, permitió profundizar los conocimientos necesarios para el desarrollo de la solución a la problemática plateada.

# Capítulo 2. Análisis y diseño de la propuesta de solución

## 2.1 Introducción

En el presente capítulo se exponen las principales características de la propuesta de solución informática a desarrollar. Se realiza la captura de los requisitos funcionales y no funcionales, se conforman las historias de usuario correspondientes a estos. Se describe la arquitectura de la aplicación, así como los patrones de diseño a aplicar, y se expone el diagrama de clases del diseño.

## 2.2 Propuesta de solución

El presente trabajo propone como solución una aplicación cliente para dispositivos móviles con sistema operativo Android. Dicha aplicación consumirá desde un servidor web, el cual proveerá los servicios necesarios que permitan listar la información de las aplicaciones almacenadas en él, realizar búsquedas, filtros y ordenamientos sobre este listado, así como descargar el fichero (.apk) correspondiente a estas aplicaciones hacia el dispositivo móvil en cuestión. De igual manera, se incluye la funcionalidad de instalar una aplicación descargada en el dispositivo móvil, así como listar las aplicaciones instaladas en este, con las opciones de desinstalarlas o actualizarlas en caso de existir versiones superiores de estas aplicaciones publicadas en el servidor mencionado anteriormente.

## 2.3 Captura de requisitos

La captura de requisitos es uno de los pasos cruciales en el desarrollo de cualquier software. Esta tarea está encaminada a identificar lo que el cliente quiere, analizar las necesidades y especificar los requerimientos de la solución sin ambigüedades.

En la presente investigación se identificaron los requisitos funcionales y no funcionales de la solución a partir de entrevistas con el cliente y tomando como base el estudio realizado acerca de las aplicaciones similares existentes.

### *2.3.1 Requisitos Funcionales*

Los requisitos funcionales definen las acciones que debe realizar el sistema. Son capacidades o condiciones que el sistema debe cumplir, cómo debe comportarse en situaciones específicas. En algunos casos también pueden plantear explícitamente qué no debe hacer el sistema (Sommerville, 2011).

En aras del correcto funcionamiento de la solución propuesta, se definieron los siguientes requisitos funcionales:

Tabla 2. Descripción de los requisitos funcionales

| No | Requisito funcional (RF) | Descripción | Complejidad | Prioridad para el cliente |
|---|---|---|---|---|
| **1** | Listar aplicaciones de la Tienda de Aplicaciones Android del Caribe. | La aplicación debe mostrar al usuario el listado de las aplicaciones disponibles en la Tienda de Aplicaciones Android del Caribe. Teniendo en cuenta los siguientes datos:<br>- Ícono<br>- Nombre<br>- Número de descargas<br>- Promedio de calificaciones<br>- Acciones:<br>• Compartir<br>• Descargar<br>• Instalar | Baja | Alta |
| **2** | Listar aplicaciones instaladas. | La aplicación debe mostrar al usuario el listado de las aplicaciones instaladas en su dispositivo móvil. Teniendo en cuenta los siguientes datos:<br>- Ícono<br>- Nombre<br>- Acciones:<br>• Desinstalar<br>• Actualizar: Se mostrará en caso de estar disponible en la tienda una actualización de la aplicación. | Baja | Alta |
| **3** | Descargar aplicación. | La aplicación debe permitir al usuario descargar una aplicación de la Tienda de Aplicaciones Android del Caribe hacia su dispositivo móvil. | Alta | Alta |

| | | | | |
|---|---|---|---|---|
| **4** | Instalar aplicación. | La aplicación debe permitir al usuario instalar una aplicación de la Tienda de Aplicaciones Android del Caribe en su dispositivo móvil. | Baja | Alta |
| **5** | Compartir aplicación. | La aplicación debe permitir compartir la URL de una aplicación seleccionada por el usuario a través de las vías disponibles en su dispositivo móvil. | Media | Media |
| **6** | Calificar aplicación. | La aplicación permite al usuario a partir de un calificador de cinco estrellas, calificar la aplicación. | Alta | Media |
| **7** | Desinstalar aplicación. | La aplicación debe permitir al usuario desinstalar una aplicación instalada previamente en su dispositivo móvil. | Baja | Alta |
| **8** | Actualizar aplicación. | La aplicación debe permitir al usuario actualizar una aplicación instalada en su dispositivo móvil desde la Tienda de Aplicaciones Android del Caribe. | Baja | Alta |
| **9** | Buscar aplicaciones. | La aplicación permite buscar aplicaciones a partir de un criterio de búsqueda introducido por el usuario. | Media | Alta |
| **10** | Mostrar resultado de búsqueda. | La aplicación debe mostrar las aplicaciones que coinciden con el criterio de búsqueda introducido por el usuario. | Baja | Alta |
| **11** | Filtrar listado de aplicaciones por categorías. | La aplicación debe permitir al usuario filtrar el listado de aplicaciones a partir de las siguientes categorías: UCI, Multimedia, Juego, Oficina, Internet, Sistema y Utilidad. | Media | Alta |
| **12** | Ordenar listado de aplicaciones. | La aplicación permite ordenar el listado de aplicaciones atendiendo los siguientes criterios:<br>- Espacio en disco<br>- Fecha de publicación<br>- Fecha de instalación<br>- Fecha de actualización | Media | Media |

| | | | | |
|---|---|---|---|---|
| | | - Número de descargas<br>- Promedio de calificaciones | | |
| **13** | Notificar sobre nuevas publicaciones. | La aplicación debe notificar al usuario sobre la existencia de nuevas aplicaciones en la Tienda de Aplicaciones Android del Caribe. | Media | Baja |
| **14** | Notificar sobre actualizaciones. | La aplicación debe notificar al usuario sobre la existencia de actualizaciones en la Tienda de Aplicaciones Android del Caribe. | Media | Baja |
| **15** | Mostrar información de una aplicación de la tienda. | La aplicación debe mostrar los datos referentes a una aplicación almacenada en la tienda seleccionada por el usuario. Los datos a mostrar son los siguientes:<br>• Ícono<br>• Descargas<br>• Votaciones<br>• Nombre<br>• Paquete<br>• Versión<br>• Permisos<br>• Categoría<br>• Espacio en disco<br>• Descripción<br>• Fecha de publicación<br>• Adjuntos | Media | Baja |
| **16** | Mostrar información de una aplicación instalada. | La aplicación debe mostrar los datos referentes a una aplicación instalada en el dispositivo móvil la cual es seleccionada por el usuario. Los datos a mostrar son los siguientes:<br>• Ícono<br>• Nombre<br>• Paquete<br>• Versión<br>• Permisos | Media | Baja |

| | | | | |
|---|---|---|---|---|
| | | • Versión de código<br>• Versión de SDK<br>• Características<br>• Dirección<br>• Fecha de instalación<br>• Fecha de última modificación | | |

### *2.3.2 Requisitos No Funcionales*

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

**Fiabilidad**

RNF1. Si se interrumpe la conexión de red con la tienda, la aplicación notificará al usuario y brindará la opción de intentar conectarse nuevamente.

RNF2. Si se produce una excepción, la aplicación la notificará al usuario, reanudará su funcionamiento e intentará nuevamente la operación que se ejecutaba al momento de producirse la excepción.

**Portabilidad**

RNF3. Sistema operativo Android con versión desde 4.0 hasta 6.0, la actual.

RNF4. Memoria RAM del dispositivo con capacidad mínima de 256 Mb.

RNF5. Almacenamiento interno o tarjeta micro SD con capacidad mínima de 100Mb.

RNF6. Soporte para conexiones WIFI.

**Usabilidad**

RNF7. Soporte para los idiomas: inglés y español.

RNF8. Interfaz de usuario basada en Material Design.

## 2.4 Historias de usuario

La metodología empleada define distintas formas para encapsular los requisitos, surgiendo a su vez escenarios para la modelación de una solución informática (Sánchez, 2015). En el caso de la solución propuesta, se cuenta con un negocio muy bien definido y el cliente estará acompañando al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos. Teniendo en cuenta estas características se tomó la decisión de adoptar el escenario que emplea las historias de usuario para encapsular los requisitos funcionales.

Las historias de usuario (HU) constituyen una forma de administración de requisitos sin tener que elaborar gran cantidad de documentos formales y sin requerir de mucho tiempo para administrarlos. Las mismas son escritas utilizando el lenguaje común del usuario. Son empleadas en las metodologías de desarrollo ágiles para la especificación de requisitos (Cohn, 2008).

Para la descripción de los requisitos funcionales de la propuesta de solución se definió una historia de usuario para cada uno de ellos, a excepción de los requisitos Buscar aplicaciones y Mostrar resultado de búsqueda, los cuales se agruparon en una historia de usuario, para un total de 15 HU. A continuación se muestran las HU “Listar aplicaciones de la Tienda de Aplicaciones Android del Caribe” y “Descargar aplicación”, por ser las que describen las funcionalidades que tienen mayor prioridad para el cliente. El resto de las historias de usuario se encuentran definidas en el Expediente de Proyecto.

Tabla 3. Historia de Usuario Listar aplicaciones de la Tienda de Aplicaciones Android del Caribe.

| | |
|---|---|
| **Número:**1 | **Nombre del requisito:** Listar aplicaciones de la Tienda de Aplicaciones Android del Caribe. |
| **Programador:** Leonardo Quesada Cruz | **Iteración Asignada:** 1 |
| **Prioridad:** Alta | **Tiempo Estimado:** 7 días |
| **Riesgo en Desarrollo:**<br>-Fallo de conexión.<br>-Fallo del servidor.<br>-Interrupción del fluido eléctrico. | **Tiempo Real:** 1 semana |
| **Descripción:**<br>La aplicación debe mostrar al usuario el listado de las aplicaciones disponibles en la Tienda de Aplicaciones Android del Caribe. | |

La funcionalidad tiene lugar en la interfaz principal de la aplicación. La misma presenta dos listados de aplicaciones, de los cuales se visualizará el que se encuentra asociado a la primera pestaña.

Cada vez que se vuelva a seleccionar esta pestaña se visualizará este listado.

Se muestran al usuario los siguientes datos de las aplicaciones:

- Ícono: Muestra el ícono representativo de la aplicación.
- Nombre: Muestra el nombre de la aplicación, el cual incluye además la versión de dicha aplicación.
- Promedio de calificaciones: Muestra el promedio de calificaciones alcanzado por la aplicación basado en el criterio de los usuarios.
- Número de descargas: Muestra la cantidad de veces que ha sido descargada la aplicación desde la tienda.
- Acciones:
    - Compartir
    - Descargar
    - Instalar

**Observaciones:**

Inicialmente se visualizará este listado, pero en el caso de no existir conexión con la tienda, se mostrará al usuario un mensaje de error y se visualizará el listado de las aplicaciones instaladas en su dispositivo móvil.

**Prototipo de interfaz:**

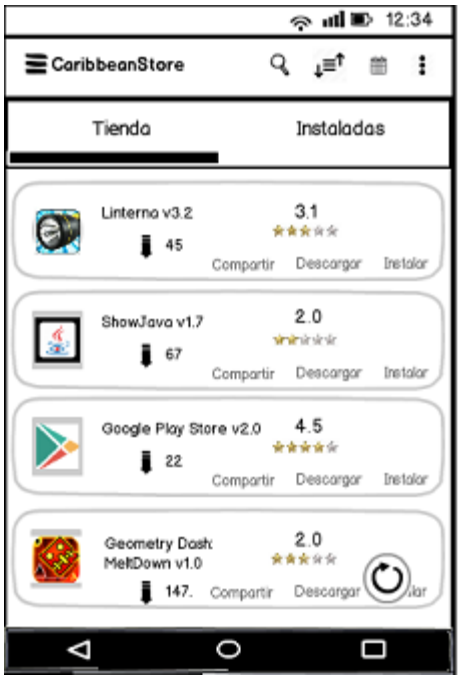



Fig. 1 Listar aplicaciones de la Tienda de Aplicaciones Android del Caribe.

Tabla 4. Historia de Usuario Descargar aplicación.

| | |
|---|---|
| **Número:** 3 | **Nombre del requisito:** Descargar aplicación. |
| **Programador:** Leonardo Quesada Cruz | **Iteración Asignada:** 1 |
| **Prioridad:** Alta | **Tiempo Estimado:** 10 días |
| **Riesgo en Desarrollo:**<br>-Fallo de conexión.<br>-Fallo del servidor.<br>-Interrupción del fluido eléctrico. | **Tiempo Real:** 1 semana |
| **Descripción:**<br>La aplicación debe permitir al usuario descargar una aplicación de la tienda de aplicaciones Android del Caribe hacia su dispositivo móvil.<br>Debe existir al menos una aplicación en la tienda.<br>La funcionalidad tiene lugar en la interfaz principal al ejecutarse la acción “Descargar” presente en cada elemento de la lista de aplicaciones de la Tienda de Aplicaciones Android del Caribe. | |

| **Observaciones:** |
| --- |
| **Prototipo de interfaz:**   Fig. 2 Descargar aplicación. |

## 2.5 Diseño

El papel del diseño en el ciclo de vida de un software es facilitar la comprensión de su funcionamiento y proveer una representación o modelo del mismo con el propósito de definirlo con los suficientes detalles como para permitir su realización física. El modelo de diseño provee una representación arquitectónica del software que sirva de punto de partida para las tareas de implementación, dando al traste con los requisitos del sistema.

### *2.5.1 Descripción de la arquitectura*

El estándar IEEE[18] define la arquitectura de software como la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución (Garlan y Shaw, 1993). Según Roger Pressman: “En su forma más simple, la arquitectura del software es la estructura u organización de los componentes del programa, la manera en que estos interactúan y la estructura de datos que utilizan” (Pressman, 2010).

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

Para el presente trabajo de diploma se propone una arquitectura Modelo-Vista-Controlador (MVC), ya que la aplicación estará basada en la interacción de un cliente con un servidor a través de la web y esta arquitectura es ampliamente usada para este propósito, además de que es empleada cada vez más en la implementación de aplicaciones Android.

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

Esta arquitectura separa presentación e interacción de los datos del sistema. El sistema se estructura en tres componentes lógicos que interactúan entre sí. El componente Modelo maneja los datos asociados al sistema y las operaciones asociadas a esos datos. El componente Vista define y gestiona como se presentan esos datos al usuario. El componente Controlador dirige la interacción del usuario y pasa estas interacciones a Vista y Modelo (Sommerville, 2011).

Permite que los datos cambien de manera independiente de su representación y viceversa. Soporta en diferentes formas la presentación de los mismos datos, y los cambios en una representación se muestran en todos ellos.

La solución propuesta constará de tres componentes fundamentales: Modelo, Vista y Controlador. A continuación se muestra una vista que lo ilustra:

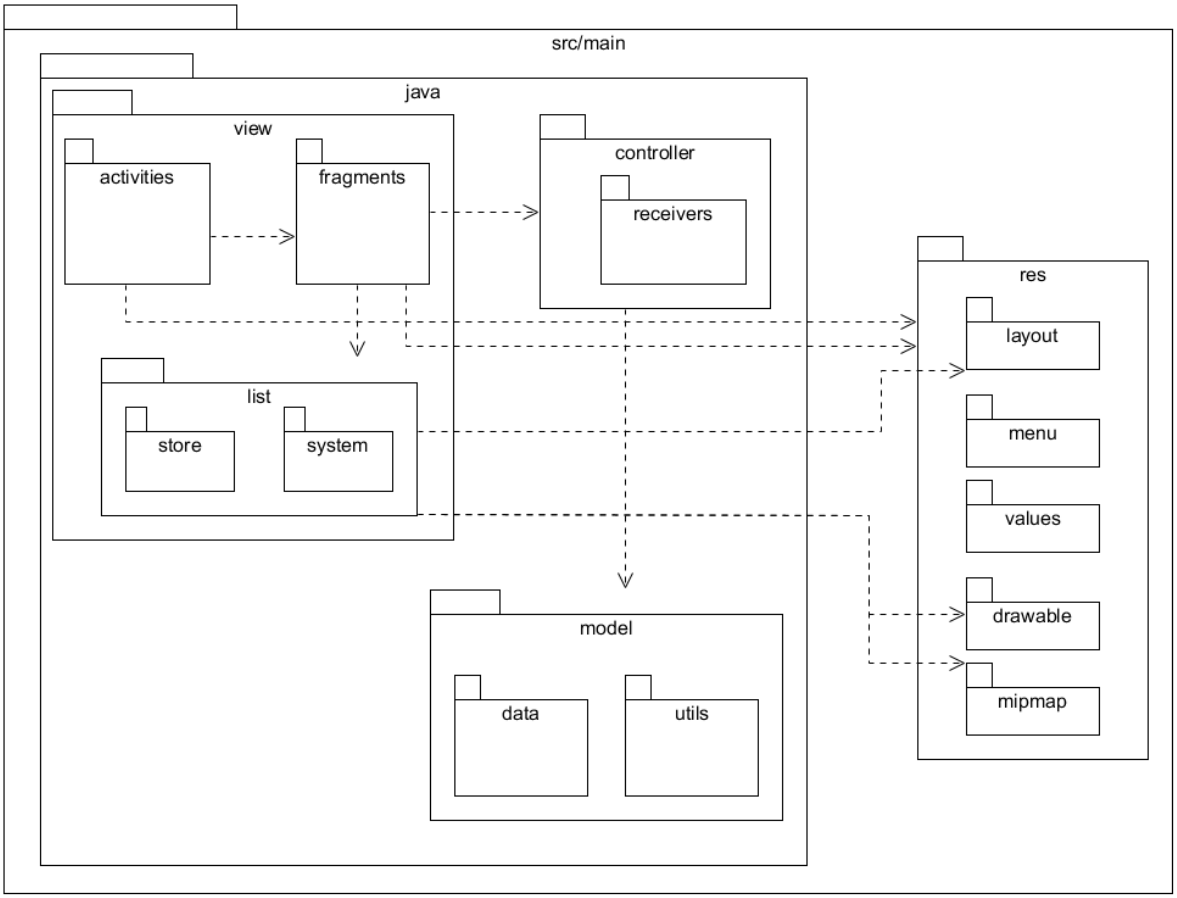



Fig. 3. Diagrama de paquetes que ilustra cómo está estructurada la propuesta de solución basada en MVC.

- **Modelo:** Maneja todo lo referente a la persistencia de datos de la aplicación, las clases entidades, el acceso a la red y los elementos necesarios para manejar dichos elementos (paquete **model**).
- **Vista:** Representa la interfaz gráfica para la interacción con el usuario. Dentro se ubican todos los componentes que intervienen en la visualización del resultado de la comunicación con el Controlador (paquete **view**).
- **Controlador:** Aquí se tendrán las clases que interactúan con el componente Vista recibiendo las solicitudes de eventos de los usuarios y con el Modelo registrando los cambios realizados por el mismo (paquete **controller**).

### *2.5.2 Patrones de diseño*

Un patrón de diseño es una descripción de la comunicación entre objetos y clases, personalizada para resolver un problema de diseño general en un contexto particular. Identifica clases, instancias, roles, colaboraciones y la distribución de responsabilidades (Gamma et al., 1994). Se presentan como pares de problema-solución con nombre, sugiriendo aspectos relacionados con la asignación de responsabilidades.
Los patrones de diseño se caracterizan por:

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

#### 2.5.2.1 Patrones GRASP

Los patrones GRASP[19] describen los principios fundamentales de la asignación de responsabilidades a objetos (Larman, 2004). El nombre se eligió para indicar la importancia de captar[20] estos principios, si se quiere diseñar un software de manera eficaz.

- **Experto:** se usa más que cualquier otro al asignar responsabilidades, es un principio básico que suele utilizarse en el diseño orientado a objetos (Larman, 2004). Consiste en la asignación de una responsabilidad a la clase que cuenta con la información necesaria para llevarla a cabo. El uso de este patrón da pie a un bajo acoplamiento y una alta cohesión, lo

[19] GRASP: Del inglés *General Responsibility Assignment Software Patterns*. Traducido al español Patrones Generales de Software para Asignar Responsabilidades.

[20] Del inglés *grasping*.

que favorece al hecho de tener sistemas más robustos y de fácil mantenimiento. El cumplimiento de una responsabilidad requiere a menudo información distribuida en varias clases de objetos. En la aplicación Cliente para dispositivos móviles de la Tienda de Aplicaciones Android del Caribe este patrón se evidencia en las clases Application, StoreApplication y InstalledApplication, las cuales se encargan de manejar toda la información perteneciente a las aplicaciones, ya sean las que se listan desde la tienda antes mencionada o las instaladas en el dispositivo móvil respectivamente.

- **Bajo Acoplamiento:** el acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Una clase con bajo (o débil) acoplamiento no depende de muchas otras (Larman, 2004). El bajo acoplamiento soporta el diseño de clases más independientes y reutilizables, lo cual reduce el impacto de los cambios y acrecienta la oportunidad de una mayor productividad. Ejemplo del uso de este patrón en la solución propuesta se muestra en las clases Store, Util e InstalledApplication, donde se minimizan las relaciones de estas con el resto de las clases.
- **Alta Cohesión:** en la perspectiva del diseño orientado a objetos, la cohesión (o más exactamente, la cohesión funcional) es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme, clases con responsabilidades moderadas en un área funcional que colaboran con las otras para llevar a cabo las tareas (Larman, 2004). En el diseño de la solución que se propone, se evidencia este patrón en la clase System, cuya funcionalidad GetSortedListByDate utiliza para su implementación la funcionalidad getInstDate de la clase InstalledApplication.
- **Controlador:** un controlador es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema (Larman, 2004). En la solución propuesta se evidencia en las clases Store y System, las cuales se encargan de gestionar los procesos involucrados en la aplicación separándolos en los que tienen que ver con las aplicaciones almacenadas en la Tienda de Aplicaciones Android del Caribe y los que corresponden a las instaladas en el dispositivo móvil en cuestión, respectivamente.

#### 2.5.2.2 Patrones GOF

Los patrones GOF[21] son alternativas de solución a problemas conocidos pero son mucho más específicas las situaciones en las que se aplican. Se clasifican en creacionales, estructurales y de comportamiento (Gamma et al., 1994). A continuación se enuncian brevemente los utilizados en el sistema.

- **Singleton**: Está diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. Su intención consiste en garantizar que una clase solo tenga una instancia y proporcionar un punto de acceso global a ella (Gamma et al., 1994). Se empleó en la creación de las clases Util, BackgroundTask y DBManager para garantizar que no exista duplicación de los objetos que controlan el acceso a la red y a la base de datos, y así no incurrir en problemas de concurrencia.
- **Fachada**: Sirve para proveer de una interfaz unificada sencilla que haga de intermediaria entre un cliente y una interfaz o grupo de interfaces más complejas. Se utiliza ampliamente para hacer más fácil y entendible el trabajo con bibliotecas de software (Gamma et al., 1994). Se empleó en la clase DBManager, la cual sirve para comunicar la lógica de la aplicación con el marco de trabajo de base de datos ADA y provee métodos para facilitar el trabajo con esta librería.
- **Estrategia**: Permite mantener un conjunto de algoritmos entre los cuales el objeto cliente puede elegir aquel que le conviene e intercambiarlo según sus necesidades, incluso en tiempo de ejecución. Cualquier programa que ofrezca un servicio o función determinada, que pueda ser realizada de varias maneras, es candidato a utilizar este patrón (Gamma et al., 1994). Se empleó en la interfaz IListManager como estrategia abstracta y los fragmentos StoreFragment e InstalledFragment como estrategias concretas. La interfaz presenta las operaciones comunes que se realizarán con cada uno de los listados de aplicaciones ya sean de la tienda o instaladas, pero al ser de distintos tipos estos listados se implementarán las mismas operaciones de forma diferente en cada uno de los fragmentos, los cuales son los encargados de dichos listados.

[21] GOF: Del inglés *Gang of Four*. Traducido al español La pandilla de los cuatro.

### *2.5.3 Diagrama de clases del diseño*

Un diagrama de clases de diseño (DCD) representa las especificaciones de las clases e interfaces software en una aplicación. Entre la información general encontramos:

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

A diferencia de las clases conceptuales del Modelo del Dominio, las clases de diseño de los DCD muestran las definiciones de las clases software en lugar de los conceptos del mundo real (Larman, 2004). A continuación se muestra el Diagrama de Clases del Diseño de la propuesta de solución:

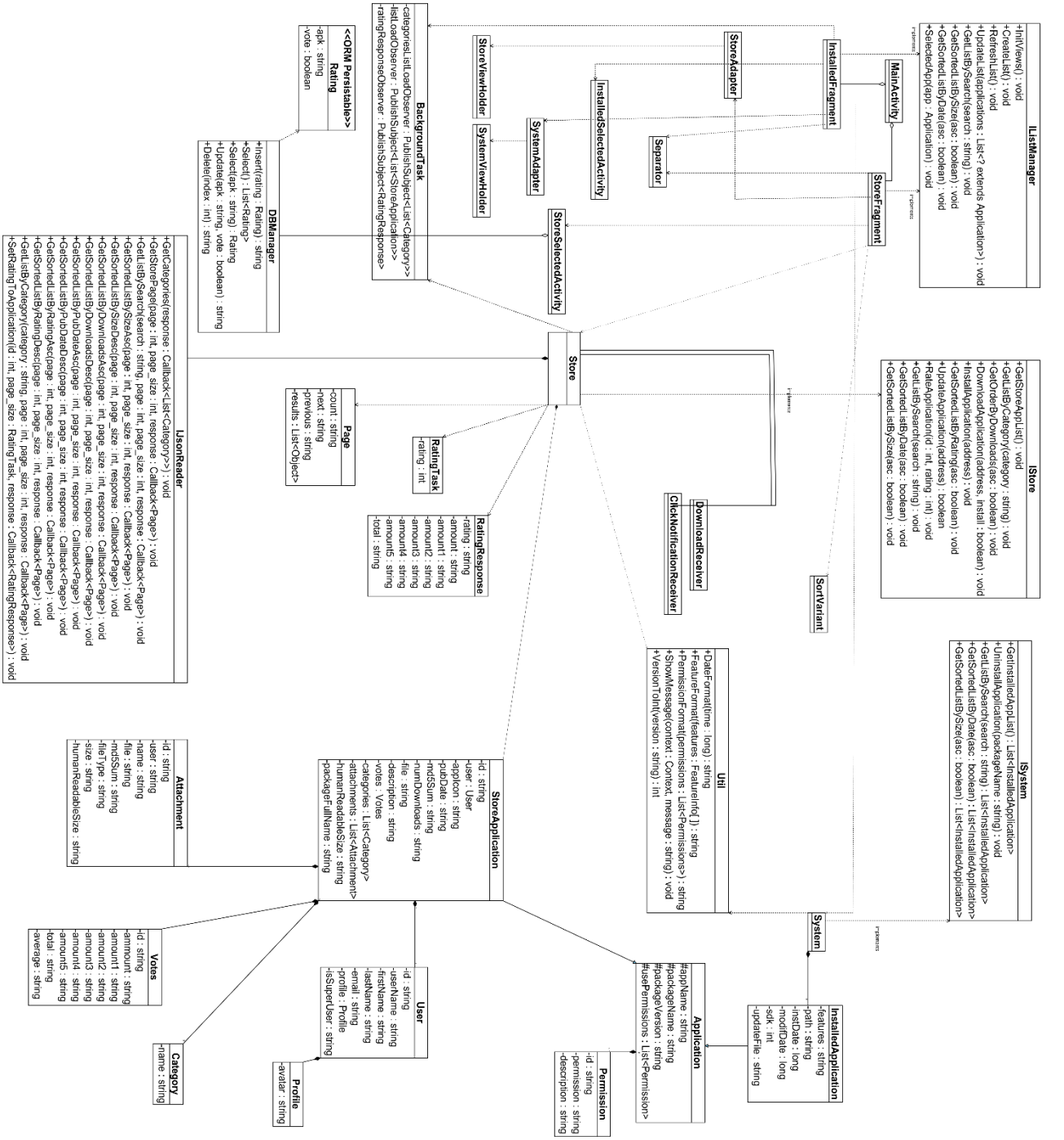



Fig. 4. Diagrama de Clases del Diseño de la solución propuesta.

## 2.6 Conclusiones parciales

A partir del desarrollo del presente capítulo se arribó a las siguientes conclusiones:

1. Se identificaron los requerimientos de la aplicación, para un total de 8 requisitos no funcionales y 16 requisitos funcionales, de los cuales 9 presentan una prioridad Alta para el cliente, 3 con Media y 4 con Baja, y fueron agrupados en 15 historias de usuario. Se realizó la descripción textual de cada uno de ellos, lo cual permitió comprender mejor las funcionalidades de la aplicación que se desea desarrollar, el comportamiento de la misma y la secuencia de actividades que debe de realizar el usuario para lograr su objetivo.
2. La utilización de la arquitectura MVC y los patrones de diseño GRASP y GOF contribuyó al diseño de la aplicación, proporcionando una estructura para la misma y posibilitando el empleo de buenas prácticas de programación y la reutilización de código.
3. Con la realización del diagrama de clases del diseño se obtuvo una visión más exacta de la aplicación en términos de implementación.
4. Como resultado principal de este capítulo quedó plasmada la propuesta de solución para la problemática planteada, detallada a través de los requisitos funcionales y no funcionales, las historias de usuario, la arquitectura y el uso de patrones de diseño en el diagrama de clases del diseño.

# Capítulo 3: Implementación y prueba

## 3.1 Introducción

Se exponen en el presente capítulo las especificaciones asociadas a la implementación de la aplicación. Se describen las pautas de codificación utilizadas y el diagrama de despliegue como resultado de la implementación. Al término de esta, la aplicación resultante será sometida a un proceso de pruebas con el objetivo de validar el cumplimiento de los requerimientos especificados anteriormente.

## 3.2 Implementación

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

### *3.2.1 Estándares de codificación*

Los estándares de codificación permiten un mejor entendimiento del código por parte de todos los miembros del equipo de desarrollo y en consecuencia hacen que el código sea fácil de mantener. El uso de estándares de codificación trae consigo los siguientes beneficios:

- Facilitan el mantenimiento de una aplicación.
- Permiten que cualquier programador entienda y pueda mantener la aplicación.
- Mejoran la legibilidad del código, al mismo tiempo que permiten su rápida compresión (Microsoft, 2016).

En la propuesta de solución se tendrán en cuenta las siguientes convenciones definidas por el equipo de desarrollo:

- Se empleará el idioma inglés para la codificación de la solución.
- El código será tabulado y espaciado a través del formato que aplica la combinación de teclas ALT+Shift+F definida en la configuración del IDE Android Studio, en la sección referente a los atajos de teclado (File/Settings/Keymap).
- Los comentarios que abarquen un bloque de instrucciones se escribirán comenzando con los caracteres “/*” y terminando con “*/”.
- Los comentarios para una línea comenzarán con los caracteres “//”.

- Para la nomenclatura de las clases, interfaces y métodos se utilizará el estilo de capitalización Pascal (Cunningham & Cunningham, Inc., 2014b), con el cual se capitaliza la primera letra de cada palabra. Ejemplo: InstalledApplication.
- Los nombres de las variables y los métodos que permitan acceder a los atributos de clases responderán al estilo de capitalización lowerCamelCase (Cunningham & Cunningham, Inc., 2014a), con el cual se capitaliza la primera letra de las palabras (excepto la primera palabra). Ejemplo: appList.
- No se usarán nombres de variables que coincidan con palabras reservadas.
- No se emplearán caracteres especiales (@, #, $, %, ^, &, * u otros) para la nomenclatura.
- Los nombres de variables globales deberán escribirse todo en mayúsculas con las palabras separadas por un guión bajo ("_"). Todas serán declaradas como *public static*.
- Se inicializarán las variables locales donde se declaran. La única razón para no hacerlo será si su valor inicial depende de cálculos posteriores.

### *3.2.2 Servidor de Prueba*

Debido a que el servidor, así como los servicios web que debe proveer la Tienda de Aplicaciones Android del Caribe no están completamente implementados, se decidió utilizar para apoyar la implementación y el proceso de prueba de las funcionalidades de la aplicación un servidor de prueba. Este fue desarrollado en otro departamento del centro CESOL, ofrece un repositorio de aplicaciones y provee, a través de una capa de servicios independiente, los servicios web necesarios para realizar operaciones sobre estas aplicaciones; los mismos se adaptan a las necesidades de la solución para la implementación de sus funcionalidades. Para su desarrollo fue empleado Django en su versión 1.9, marco de trabajo para aplicaciones web basado en Python, el cual emplea para la creación y publicación de los servicios necesarios el Django REST Framework en su versión 3.3.2.

### *3.2.3 Diagrama de despliegue*

Un diagrama de despliegue muestra las relaciones físicas que se establecen entre componentes de software y hardware dentro de un sistema de cómputo determinado. Se modela a través de un conjunto de nodos y las relaciones existentes entre ellos. Cada nodo en este tipo de diagrama representa un tipo de unidad computacional, en la mayoría de los casos de tipo hardware (Fowler y Scott, 2002).
A continuación se presenta el diagrama de despliegue de la solución propuesta:

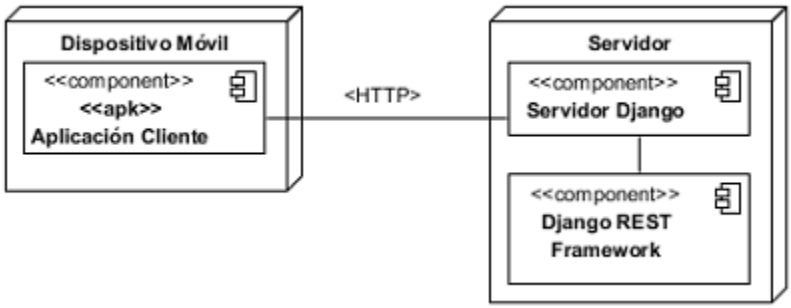



Fig. 5. Diagrama de Despliegue de la propuesta de solución.

## 3.3 Pruebas de software

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

### *3.3.1 Estrategia de pruebas*

Una estrategia de pruebas de software proporciona una guía que describe los pasos que deben realizarse como parte de la prueba, cuándo se planean y se llevan a cabo dichos pasos, y cuánto esfuerzo, tiempo y recursos requerirán. Por tanto, cualquier estrategia de prueba debe incorporar la planificación de la prueba, el diseño de casos de prueba, la ejecución de la prueba y la recolección y evaluación de los resultados (Pressman, 2010).

Según Pressman, las pruebas se dividen en los siguientes niveles principales: pruebas de unidad, de integración, de validación, de sistema y de aceptación (Pressman, 2010). Para la solución propuesta se utilizó una estrategia de pruebas basada en la ejecución de las mismas tomando como guía tres de estos niveles: unidad, validación y aceptación.

#### 3.3.1.1 Pruebas de Unidad

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

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

Con el fin de automatizar este tipo de pruebas sobre la solución se decidió emplear la herramienta **JUnit**, la cual está integrada con el IDE Android Studio que se utilizó para desarrollar la aplicación. Se implementaron los casos de prueba a través de las clases StoreTest y SystemTest, las cuales comprueban el correcto funcionamiento de las principales funcionalidades definidas en las clases Store y System, respectivamente, responsables de la lógica principal de la aplicación.

#### 3.3.1.2 Pruebas de Validación

La prueba de validación proporciona un aseguramiento final de que el software cumple con todos los requisitos funcionales, de comportamiento y desempeño. La prueba se concentra en las acciones visibles para el usuario y en la salida que este puede reconocer. La validación se alcanza cuando el software funciona de tal manera que satisface las expectativas razonables (especificación de requisitos de software) del cliente. Se logra mediante una serie de pruebas que demuestran que se cumple con los requisitos (Pressman, 2010).

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

Para llevarlas a cabo, el método a emplear fue el de **caja negra**, el cual se centra en los requisitos funcionales del software. Es decir, la prueba de caja negra permite al ingeniero del software obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa (Pressman, 2010).

Como técnica se utilizó la de **partición de equivalencia**, o sea, dividir el dominio de entrada de un programa en clases de datos a partir de las cuales pueden derivarse casos de prueba. El diseño de casos de prueba para partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada (Pressman, 2010).

Los casos de prueba se diseñaron según las funcionalidades descritas en las historias de usuario. La intención que se persigue con estos artefactos es lograr una compresión específica de las condiciones que la solución debe cumplir. Cada planilla de casos de pruebas recoge la especificación de una historia de usuario, dividida en secciones y escenarios, detallando las funcionalidades descritas en ella y describiendo cada variable.

A continuación se muestran los casos de prueba correspondientes a las historias de usuario “Listar aplicaciones de la Tienda de Aplicaciones Android del Caribe” y “Descargar aplicación” ya que representan el proceso fundamental del negocio. Los casos de prueba asociados al resto de las historias de usuario son definidos en el Expediente de Proyecto.

Abreviaturas utilizadas:

CP: Caso de Prueba.

HU: Historia de Usuario.

EC: Escenario.

Tabla 5 Caso de Prueba de Validación para la Historia de Usuario Listar aplicaciones de la Tienda de Aplicaciones Android del Caribe.

| | | | |
|---|---|---|---|
| | | | **Caso de Prueba de Validación** |
| **Código:** CP1_HU1 | | **HU_1:** Listar aplicaciones de la Tienda de Aplicaciones Android del Caribe. | |
| **Responsable:** Leonardo Quesada Cruz | | | |
| **Descripción:** El Caso de Prueba se inicia al ejecutarse la aplicación. Se presentan al usuario dos listados de aplicaciones, de los cuales estará seleccionado el primero, el cual contiene un listado con las aplicaciones almacenadas en la Tienda de Aplicaciones Android del Caribe. El Caso de Prueba termina cuando el usuario finaliza la ejecución de la aplicación. | | | |
| **Condiciones de ejecución:** Debe existir al menos una aplicación almacenada en la Tienda de Aplicaciones Android del Caribe. | | | |
| **Escenario** | **Descripción** | **Respuesta del sistema** | **Flujo central** |

| | | | |
|---|---|---|---|
| EC 1.1 Mostrar el listado de aplicaciones de la Tienda de Aplicaciones Android del Caribe. | El usuario ejecuta la aplicación y esta automáticamente se conecta a la Tienda de Aplicaciones Android del Caribe. | La aplicación muestra al usuario el listado de aplicaciones almacenadas en la Tienda de Aplicaciones Android del Caribe. De cada aplicación se muestran los siguientes datos: Ícono, Nombre (incluye la versión), Número de descargas, Promedio de calificaciones y Acciones (Compartir, Descargar e Instalar). | Se inicia la aplicación. |
| EC 1.2 Error de conexión con la Tienda de Aplicaciones Android del Caribe. | El usuario ejecuta la aplicación y esta no logra conectarse a la Tienda de Aplicaciones Android del Caribe. | La aplicación muestra al usuario un mensaje de error indicando que no ha podido conectarse y automáticamente muestra el listado de las aplicaciones instaladas en el dispositivo móvil en cuestión. | Se inicia la aplicación. |

Tabla 6 Caso de Prueba de Validación para la Historia de Usuario Descargar aplicación.

| **Caso de Prueba de Validación** | |
|---|---|
| **Código:** CP3_HU3 | **HU_3:** Descargar aplicación. |
| **Responsable:** Leonardo Quesada Cruz | |
| **Descripción:** El Caso de Prueba se inicia luego de iniciada la aplicación. Para cada aplicación presente en el listado de las aplicaciones de la Tienda de Aplicaciones Android del Caribe se muestran las siguientes acciones disponibles: Compartir, Descargar e Instalar. El usuario escoge la acción Descargar de una en específico. El Caso de Prueba termina cuando la descarga ha finalizado y el usuario cambia la lista actual por la lista de aplicaciones instaladas en el dispositivo móvil. | |

| **Condiciones de ejecución:** Debe existir al menos una aplicación almacenada en la Tienda de Aplicaciones Android del Caribe. | | | |
|---|---|---|---|
| **Escenario** | **Descripción** | **Respuesta del sistema** | **Flujo central** |
| EC 3.1 Descargar una aplicación de la Tienda de Aplicaciones Android del Caribe hacia el dispositivo móvil. | El usuario selecciona la acción Descargar de una aplicación de la Tienda de Aplicaciones Android del Caribe y descarga dicha aplicación hacia el dispositivo móvil. | La aplicación muestra al usuario un mensaje para indicar el inicio de la descarga de la aplicación de la Tienda de Aplicaciones Android del Caribe. Luego muestra una notificación que informará acerca del progreso de la descarga. Finalmente mostrará un mensaje indicando que la descarga se ha completado satisfactoriamente. | El usuario selecciona la acción Descargar de una aplicación de la Tienda de Aplicaciones Android del Caribe específica. |
| EC 3.2 Error en la descarga de una aplicación de la Tienda de Aplicaciones Android del Caribe hacia el dispositivo móvil. | El usuario selecciona la acción Descargar de una aplicación de la Tienda de Aplicaciones Android del Caribe específica y no puede completar la descarga desde la Tienda de Aplicaciones Android del Caribe. | La aplicación muestra al usuario un mensaje de error indicando que ha fallado la descarga y la causa por la cual no se ha podido completar la acción. | El usuario selecciona la acción Descargar de una aplicación de la Tienda de Aplicaciones Android del Caribe específica. |

#### 3.3.1.3 Pruebas de Aceptación

Esta es la etapa final en el proceso de pruebas, antes de que el sistema se acepte para uso operacional. El sistema se pone a prueba con datos suministrados por el cliente del sistema, en vez de datos de prueba simulados. Las pruebas de aceptación revelan los errores y las omisiones en la definición de

requerimientos del sistema, ya que los datos reales ejercitan el sistema en diferentes formas a partir de los datos de prueba. Asimismo, las pruebas de aceptación revelan problemas de requerimientos, donde las instalaciones del sistema en realidad no cumplan las necesidades del usuario o cuando sea inaceptable el rendimiento del sistema (Sommerville, 2011).

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

### *3.3.2 Ejecución y resultados de las pruebas de software*

Para la ejecución de las pruebas de validación y de aceptación de tipo alfa se utilizaron los siguientes terminales:

- Teléfono celular:
  - Modelo BLU DASH JR 4.0 K
  - Versión de SO Android 4.4.2 (Kitkat)
  - SDK 19
  - CPU ARMv7 Processor rev 4 (v7l)
  - RAM 226 Mb
  - Almacenamiento interno 196,3 Mb
  - Almacenamiento externo 7,3 Gb
- Teléfono celular:
  - Modelo Jianke v11
  - Versión de SO Android 4.4.3 (Kitkat)
  - SDK 19
  - CPU Octa Core8
  - RAM 512 Mb
  - Almacenamiento interno 2,0 Gb
  - Almacenamiento externo 15,8 Gb
- Emulador Genymotion:
  - Modelo Genymotion_vbox86p_4.3_150216_21300

- Versión de SO Android 4.3 (Jelly Bean)
- SDK 18
- Intel® Core™ i3-2120 @3.30 GHz
- RAM 512 Mb
- Almacenamiento interno 5039 Mb
- Almacenamiento externo 8189 Mb

Para ilustrar el desarrollo del proceso de pruebas unitarias se muestran imágenes de los resultados arrojados por JUnit para los casos de pruebas implementados.

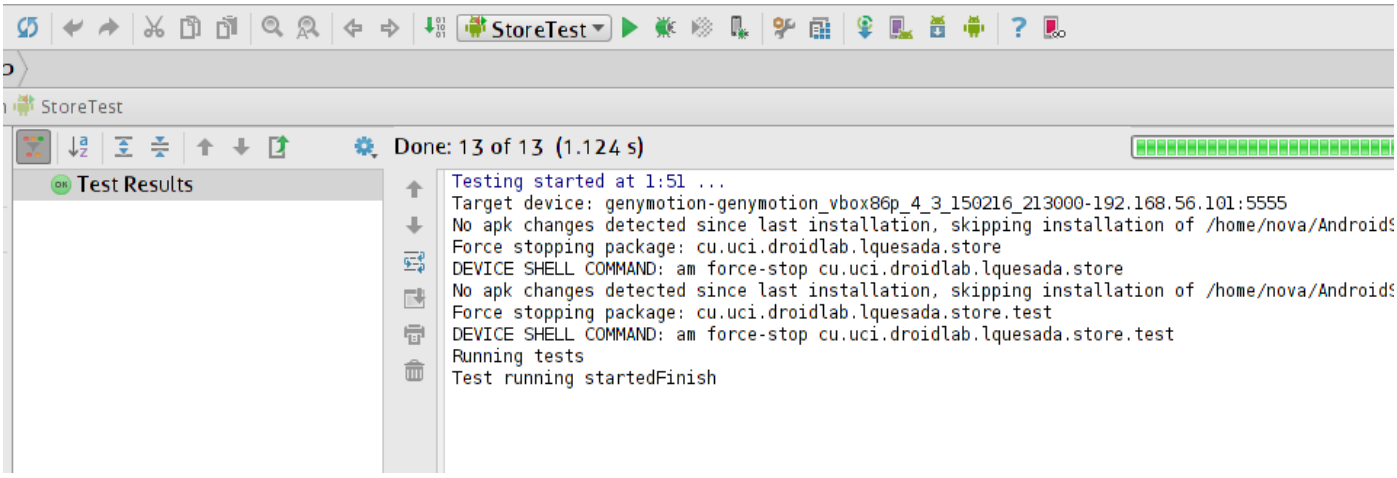



Fig. 6 Resultado de las pruebas a la clase Store con JUnit.

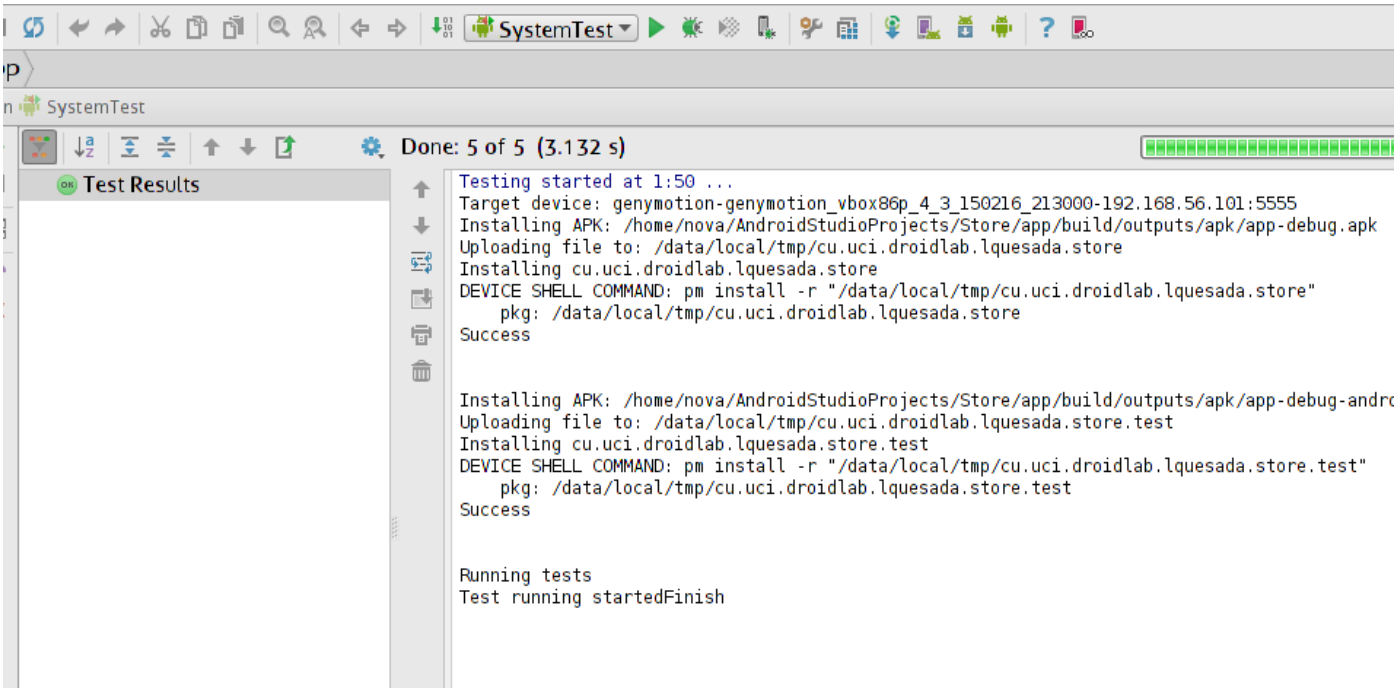



Fig. 7 Resultado de las pruebas a la clase System con JUnit.

A partir de la interpretación de dichos resultados se puede afirmar que los mismos garantizan el correcto funcionamiento de los métodos de las clases Store y System, los cuales constituyen las unidades comprobables más pequeñas con que consta la solución implementada.

Los problemas detectados en el período de pruebas de validación y aceptación se clasificaron en: No conformidades significativas (NCS) y en No conformidades no significativas (NCNS). A continuación se describen los aspectos que se tuvieron en cuenta en cada clasificación.

**NCS:** son las no conformidades referentes a las funcionalidades de la aplicación: validaciones incorrectas o respuestas de la aplicación diferentes a lo descrito previamente en las historias de usuario. **NCNS**: son las no conformidades en cuanto al diseño de la propuesta de solución y errores ortográficos.

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

En la primera iteración se detectaron 7 NCS y 5 NCNS. Las mismas fueron resueltas satisfactoriamente en la misma iteración.

No Conformidades Significativas:

1. La aplicación se detuvo al tratar de acceder al servidor sin tener conexión.

2. La aplicación mostró la información de una aplicación almacenada en la tienda distinta a la seleccionada en el listado correspondiente de la interfaz principal.
3. La aplicación mostró la información de una aplicación instalada en el dispositivo móvil distinta a la seleccionada en el listado correspondiente de la interfaz principal.
4. La aplicación no mostró un cuadro de diálogo con información acerca de la aplicación y su desarrollador al seleccionarse la opción “Acerca de”, la cual está presente tanto en la barra de herramientas de la aplicación como en el panel desplegable.
5. La aplicación no mostró ningún resultado al ejecutar una búsqueda en el listado de aplicaciones almacenadas en la tienda, conociéndose que existen aplicaciones que coinciden con el criterio de búsqueda introducido.
6. La aplicación no actualizó el listado de aplicaciones instaladas en el dispositivo móvil al desinstalarse una aplicación.
7. La aplicación se detuvo al intentar desinstalar una aplicación que ya había sido desinstalada, pero que todavía no había sido removida del listado correspondiente.

No Conformidades No Significativas:

1. La aplicación tiene faltas de ortografía en los nombres de algunas de las categorías presentes en el panel desplegable.
2. La aplicación no muestra completamente el texto de las acciones a realizar que presenta cada elemento de la lista de aplicaciones de la tienda.
3. La aplicación muestra el nombre y la versión de una aplicación del listado de aplicaciones de la tienda sin un espacio entre ambos textos.
4. La aplicación muestra incorrectamente los textos presentes en el cuadro de diálogo Filtros que se visualiza al seleccionar esta opción en la barra de herramientas de la aplicación.
5. La aplicación muestra el botón “Refrescar” en la interfaz principal de forma que este queda por encima del último elemento de la lista de aplicaciones que se esté mostrando, lo que impide la total visualización de dicho elemento por parte del usuario.

En la segunda iteración se detectaron 2 NCS y 2 NCNS, siendo solucionadas de la misma forma que las anteriores.

No Conformidades Significativas:

1. La aplicación no mostró un mensaje de error cuando no pudo conectarse al servidor.

2. La aplicación no mostró ningún resultado al seleccionar la categoría UCI del panel desplegable, conociéndose que existen aplicaciones con esta categoría en el listado de aplicaciones almacenadas en la tienda.

No Conformidades No Significativas:

1. La aplicación muestra incorrectamente los textos presentes en el cuadro de diálogo que se visualiza al seleccionar la opción “Acerca de“, que se encuentra tanto en la barra de herramientas de la aplicación como en el panel desplegable.
2. La aplicación presenta faltas de ortografía en el cuadro de diálogo que se muestra cuando se ejecuta la acción “Refrescar” al presionar el botón presente en la interfaz principal.

En la tercera iteración no se detectaron NCS ni NCNS, por lo que se demostró que la aplicación cumple con los requisitos funcionales establecidos y fue considerada concluida. A continuación se muestra un gráfico con un resumen de los resultados obtenidos tras la realización de las pruebas:

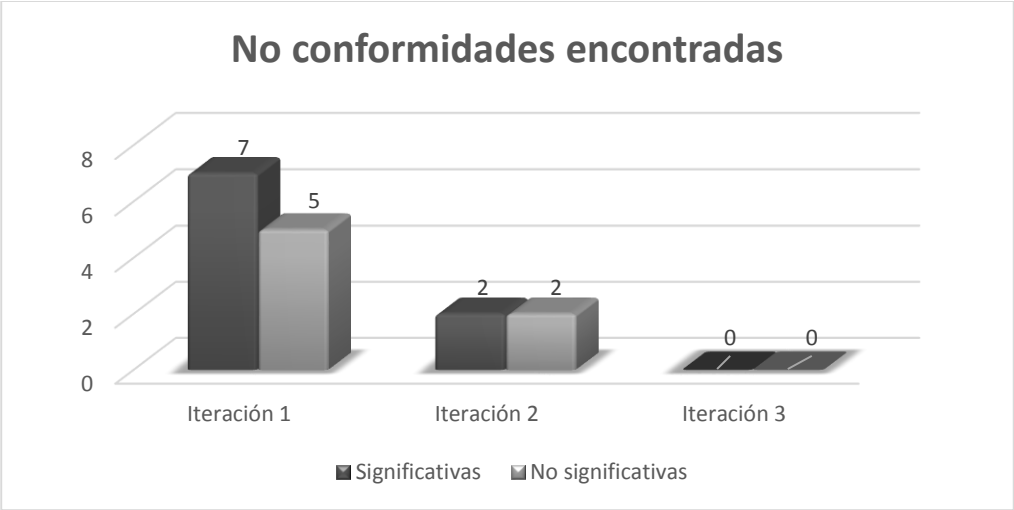



Fig. 8. Gráfica correspondiente a las no conformidades encontradas por iteración en el proceso de pruebas.

## 3.4 Conclusiones parciales

A partir del desarrollo del presente capítulo se arribaron a las siguientes conclusiones:

1. La descripción del proceso de implementación de la aplicación, a través de la definición de las convenciones utilizadas para la codificación, posibilitó una mejor legibilidad del código, haciéndolo más comprensible y estandarizado.
2. A través del diagrama de despliegue se obtuvo la especificación de las características que deben cumplir la aplicación y el servidor de aplicaciones de la Tienda de Aplicaciones Android del Caribe para su correcto funcionamiento, así como la distribución final que tendrán ambos.
3. Las pruebas de caja blanca, caja negra y alfa, permitieron comprobar el correcto funcionamiento del código de la aplicación, validar la completitud de los requisitos y determinar la aceptación por parte del cliente, respectivamente.

## Conclusiones Generales

A raíz de los resultados obtenidos y tomando como base el cumplimiento de los objetivos trazados al iniciarla, se puede arribar a las siguientes conclusiones:

1. El análisis de los elementos teóricos relacionados con las aplicaciones clientes de tiendas de aplicaciones para dispositivos móviles posibilitó una mejor comprensión de los conceptos asociados.
2. El estudio de las aplicaciones similares permitió un mayor entendimiento del funcionamiento de las mismas, identificándose además las principales funcionalidades presentes en ellas, las cuales se tuvieron en cuenta para la propuesta de solución al problema planteado.
3. La identificación de los requisitos funcionales y no funcionales, descritos a través de historias de usuario, facilitaron la descripción, comprensión y posterior codificación por parte del equipo de desarrollo de los requerimientos que precisaba el cliente.
4. La implementación de la aplicación Cliente para dispositivos móviles de la Tienda de Aplicaciones Android del Caribe posibilitó darle respuesta a los requisitos definidos y con ello a las necesidades del cliente.
5. La ejecución de pruebas al Cliente para dispositivos móviles de la Tienda de Aplicaciones Android del Caribe permitió la validación de todos los requisitos identificados y la aceptación por parte del cliente.

## Recomendaciones

Tomando como base la investigación realizada y el análisis de los resultados obtenidos se recomienda:

1. Agregar una funcionalidad que permita al usuario compartir el fichero de una aplicación seleccionada por él, utilizando los medios presentes en el dispositivo móvil, dígase Bluetooth, Zapya u otros.
2. Agregar una funcionalidad que permita a la aplicación filtrar los listados de aplicaciones teniendo en cuenta varios criterios a la vez.
3. Agregar una funcionalidad que permita al usuario cambiar el tema de la interfaz de la aplicación, escogiendo entre varios disponibles.
4. Agregar una funcionalidad que permita al usuario cambiar el directorio donde se almacenarán las Aplicaciones descargadas desde la Tienda de Aplicaciones Android del Caribe.

## Bibliografía Referenciada

**1 MOBILE MARKET, 2015.** 1 Mobile Market. [En línea]. [Consulta: 3 noviembre 2015]. Disponible en: http://www.1mobile.com/mobile-market-free-android-store-79873.html.

**AMBLER, S.W., 2014.** The Agile Unified Process (AUP) Home Page. [En línea]. [Consulta: 17 enero 2016]. Disponible en: http://www.ambysoft.com/unifiedprocess/agileUP.html.

**ANDROID DEVELOPERS, 2014.** La oportunidad de Google Play | Android Developers. [En línea]. [Consulta: 20 octubre 2015]. Disponible en: https://developer.android.com/distribute/googleplay/about.html.

**ANDROID DEVELOPERS, 2015a.** Download Android Studio and SDK Tools | Android Studio. [En línea]. [Consulta: 13 diciembre 2015]. Disponible en: https://developer.android.com/studio/index.html?hl=es.

**ANDROID DEVELOPERS, 2015b.** Meet Android Studio | Android Studio. [En línea]. [Consulta: 13 diciembre 2015]. Disponible en: https://developer.android.com/studio/intro/index.html?hl=es.

**APPLE INC., 2015.** App Store Review Guidelines - Apple Developer. [En línea]. [Consulta: 2 noviembre 2015]. Disponible en: https://developer.apple.com/app-store/review/guidelines/.

**APTOIDE, 2015a.** Acerca de | Aptoide. [En línea]. [Consulta: 4 noviembre 2015]. Disponible en: http://m.aptoide.com/about.

**APTOIDE, 2015b.** Aptoide. [En línea]. [Consulta: 6 noviembre 2015]. Disponible en: http://m.aptoide.com/.

**BALSAMIQ, 2016.** Balsamiq Mockups | Balsamiq. [En línea]. [Consulta: 3 mayo 2016]. Disponible en: https://balsamiq.com/products/mockups/.

**COHN, M., 2008.** Agile Estimating Planning. Agile Development Practices. [En línea]. [Consulta: 17 enero 2016]. Disponible en: https://www.mountaingoatsoftware.com/uploads/presentations/Agile-Estimating-Planning-Agile-Development-Practices-2008.pdf.

**COLLAZO LINARES, A., 2014a.** Android UCLV - Acerca de. [En línea]. [Consulta: 6 noviembre 2015]. Disponible en: http://android.uclv.edu.cu/index.php?/about.

**COLLAZO LINARES, A., 2014b.** Android UCLV - Todas. [En línea]. [Consulta: 6 noviembre 2015]. Disponible en: http://android.uclv.edu.cu/.

**CUELLO, J. y VITTONE, J., 2015.** Capítulo 1: Las aplicaciones – Diseñando apps para móviles. [En línea]. [Consulta: 25 octubre 2015]. Disponible en: http://appdesignbook.com/es/contenidos/las-aplicaciones/.

**CUNNINGHAM & CUNNINGHAM, INC., 2014a.** Lower Camel Case. [En línea]. [Consulta: 23 marzo 2016]. Disponible en: http://c2.com/cgi/wiki?LowerCamelCase.

**CUNNINGHAM & CUNNINGHAM, INC., 2014b.** Pascal Case. [En línea]. [Consulta: 23 marzo 2016]. Disponible en: http://c2.com/cgi/wiki?PascalCase.

**F-DROID, 2015a.** F-Droid. [En línea]. [Consulta: 5 noviembre 2015]. Disponible en: https://f-droid.org/wiki/page/Main_Page.

**F-DROID, 2015b.** F-Droid – Free and Open Source Android App Repository. [en línea]. [Consulta: 5 noviembre 2015]. Disponible en: https://f-droid.org/.

**FOWLER, M. y SCOTT, K., 2002.** *UML distilled: a brief guide to the standard object modeling language*. 2. ed., 10. Print. Boston: Addison-Wesley. Object technology series. ISBN 978-0-201-65783-8.

**GAMMA, E., HELM, R., JOHNSON, R. y VLISSIDES, J., 1994.** *Design Patterns : Elements of Reusable Object-Oriented Software* [en línea]. S.l.: Addison-Wesley. [Consulta: 23 junio 2016]. ISBN 0-201-63361-2. Disponible en: http://www.uml.org.cn/c++/pdf/DesignPatterns.pdf.

**GARLAN, D. y SHAW, M., 1993.** An introduction to software architecture. *Advances in software engineering and knowledge engineering* [en línea], vol. 1, no. 3.4. [Consulta: 23 junio 2016]. Disponible en: https://books.google.com/books?hl=en&lr=&id=k8bsCgAAQBAJ&oi=fnd&pg=PA1&dq=%22beyond+the+algorithms+and+data+structures+of+the+computation:%22+%22and+frameworks+for+systems+that+serve+the+needs+of+specific%22+%22as+industry+and+scientific%22+%22choices+among+design+alternatives.+Fourth,+an+architectural%22+&ots=m19z6kaXRh&sig=s0C_vyvt79gGYT30mj5T5lqJ-EA.

**GENYMOTION, 2015a.** Genymotion – Fast and Easy Android Emulation. [En línea]. [Consulta: 15 diciembre 2015]. Disponible en: https://www.genymotion.com/.

**GENYMOTION, 2015b.** Genymotion Plugin for Android Studio Guide - Plugin-for-Android-Studio-1.0.7-Guide.pdf. [En línea]. [Consulta: 15 diciembre 2015]. Disponible en: https://docs.genymotion.com/pdf/PDF_Plugin_for_Android_Studio/Plugin-for-Android-Studio-1.0.7-Guide.pdf.

**GITHUB, 2015.** The RxJava Android Module · ReactiveX/RxJava Wiki · GitHub. [En línea]. [Consulta: 13 diciembre 2015]. Disponible en: https://github.com/ReactiveX/RxJava/wiki/The-RxJava-Android-Module.

**GITHUB, 2016.** Consuming APIs with Retrofit · codepath/android_guides Wiki · GitHub. [En línea]. [Consulta: 7 febrero 2016]. Disponible en: https://github.com/codepath/android_guides/wiki/Consuming-APIs-with-Retrofit.

**GONZÁLEZ, M.O., 2012.** *Módulo Salud para el Sistema Informativo de la Administración Provincial* [en línea]. Artemisa: Universidad de las Ciencias Informáticas. [Consulta: 17 diciembre 2015]. Disponible en: http://bibliodoc.uci.cu/RDigitales/2013/marzo/27/TD_06096_12.pdf.

**GOOGLE, 2015.** Aplicaciones de Android en Google Play. [En línea]. [Consulta: 12 octubre 2015]. Disponible en: https://play.google.com/store?hl=es.

**GOOGLE SUPPORT, 2014.** Países en los que se pueden comprar dispositivos - Ayuda de Google Store. [En línea]. [Consulta: 20 octubre 2015]. Disponible en: https://support.google.com/store/answer/2462844?hl=es&rd=1.

**GOSLING, J., 2000.** *The Java language specification* [en línea]. 7th. S.l.: Addison-Wesley Professional. [Consulta: 23 junio 2016]. Disponible en: https://books.google.com/books?hl=en&lr=&id=Ww1B9O_yVGsC&oi=fnd&pg=PA1&dq=%22Escape+Sequences+for+Character+and+String+Literals%22+%22Variables+of+Reference+Type%22+%22String+Conversion%22+%22The+Causes+of+Exceptions%22+%22Initialize+Test:+Execute+Initializers%22+%22Implementing+Finalization%22+%22Java%C2%AE+Language%22+&ots=Sf4IepS9pE&sig=ZH6qFKxzpgeyyOY565AfVlhv0hc.

**GRADLE INC., 2015a.** Enterprise Build Automation l Why Gradle #WhyGradle. [En línea]. [Consulta: 14 diciembre 2015]. Disponible en: https://gradle.org/whygradle-build-automation/.

**GRADLE INC., 2015b.** Getting Started with Gradle for Android Build l Gradle. [En línea]. [Consulta: 14 diciembre 2015]. Disponible en: http://gradle.org/getting-started-android-build/.

**HEADWAYS MEDIA, 2015.** Definición: Tienda virtual | Glosario de Mercadotecnia Online y Offline. [En línea]. [Consulta: 20 octubre 2015]. Disponible en: http://www.headways.com.mx/glosario-mercadotecnia/definicion/tienda-virtual/.

**IDC, 2015.** IDC: Smartphone OS Market Share 2015, 2014, 2013, and 2012. [En línea]. [Consulta: 18 diciembre 2015]. Disponible en: http://www.idc.com/prodserv/smartphone-os-market-share.jsp.

**JUNIT, 2016.** JUnit - About. [En línea]. [Consulta: 10 mayo 2016]. Disponible en: http://junit.org/junit4/.

**KAISLER, S.H., 2005.** *Software Paradigms*. New Jersey: John Wiley & Sons. ISBN 0-471-48347-8.

**LARMAN, C., 2004.** *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, Third Edition* [en línea]. S.l.: Addison Wesley Professional. [Consulta: 24 junio 2016]. ISBN 0-13-148906-2. Disponible en: https://aanimesh.files.wordpress.com/2013/09/applying-uml-and-patterns-3rd.pdf.

**MICROSOFT, 2015.** Aplicaciones para Windows Phone - Microsoft Store. [En línea]. [Consulta: 2 noviembre 2015]. Disponible en: https://www.microsoft.com/es-es/store/apps/windows-phone.

**MICROSOFT, 2016.** Microsoft. MSDN. Revisiones de código y estándares de codificación. [En línea]. [Consulta: 24 marzo 2016]. Disponible en: http://msdn.microsoft.com/es-es/library/aa291591(v=vs.71).aspx.

**MOB&ME, 2013.** javaHispano - Android - Tutorial ADA Framework (parte I). [En línea]. [Consulta: 14 diciembre 2015]. Disponible en: http://www.javahispano.org/android/2013/1/9/tutorial-ada-framework-parte-i.html.

**MTTKAY, 2013.** Functional Reactive Programming on Android with RxJava. [En línea]. [Consulta: 12 diciembre 2015]. Disponible en: https://mttkay.github.io/blog/2013/08/25/functional-reactive-programming-on-android-with-rxjava/.

**NÚÑEZ, K., 2013.** Sistema Operativo Android; versiones, historia. [En línea]. [Consulta: 20 octubre 2015]. Disponible en: http://es.slideshare.net/karenonunez/sistema-operativo-android-versiones-historia.

**OBJECT MANAGEMENT GROUP, 2005.** Introduction to OMG's Unified Modeling Language™ (UML®). [En línea]. Disponible en: http://www.omg.org/gettingstarted/what_is_uml.htm.

**ORACLE CORPORATION, 2015.** ¿Qué es Java? [En línea]. [Consulta: 11 diciembre 2015]. Disponible en: https://www.java.com/es/about/whatis_java.jsp.

**ORTIZ, J.C., 2014.** Repositorios: definición, características y ejemplos. [En línea]. [Consulta: 17 diciembre 2016]. Disponible en: http://es.slideshare.net/juliocabrejos1/repositorios-definicin-caractersticas-y-ejemplos.

**PRESSMAN, R.S., 2010.** *Software Engineering. A practitioner´s Approach.* 7th. New York: McGraw Hill. ISBN 978-0-07-337597-7.

**PUROMARKETING, 2015.** PuroMarketing. Nuevos datos demuestran la dependencia a los dispositivos móviles. [En línea]. [Consulta: 18 diciembre 2015]. Disponible en: http://www.puromarketing.com/12/19353/datos-demuestran-dependencia-dispositivos-moviles.html.

**QORE, 2014.** Apple implementa nuevas características en su App Store para iOS - Qore. [En línea]. [Consulta: 2 noviembre 2015]. Disponible en: http://www.qore.com/noticias/23284/Apple-implementa-nuevas-caracteristicas-en-su-App-Store-para-iOS.

**REACTIVEX, 2015.** ReactiveX. Intro. [En línea]. [Consulta: 12 diciembre 2015]. Disponible en: http://reactivex.io/intro.html.

**SÁNCHEZ, T.R., 2015.** Metodología UCI [en línea]. 8 abril 2015. S.l.: s.n. [Consulta: 18 enero 2016]. Disponible en:

http://excriba.prod.uci.cu/proxy/alfresco/api/node/content/workspace/SpacesStore/a622adab-eac5-4fb3-ba08-a266767fff5f/Metodologia%20UCI.pdf?a=true.

**SHIN, L.Y., 2014.** Journal of European Psychology Students. A Comparative Study of Mobile Internet Usage between the U.S. and Korea. [En línea]. [Consulta: 16 diciembre 2015]. Disponible en: http://jeps.efpsa.org/articles/10.5334/jeps.cg/.

**SOMMERVILLE, I., 2011.** *Ingeniería de Software*. 9na. México: Addison Wesley. Pearson Education, Inc. ISBN 978-607-32-0603-7.

**STATISTA, 2015.** Devices targeted by app developers 2014. [En línea]. Disponible en: http://www.statista.com/statistics/256570/devices-targeted-by-app-developers/.

**UNIFIED MODELING LANGUAGE, 2015.** Unified Modeling Language (UML). [En línea]. [Consulta: 9 diciembre 2015]. Disponible en: http://www.uml.org/.

**VARANASI, B. y BELIDA, S., 2014.** *Introducing Maven* [en línea]. S.l.: Apress. [Consulta: 17 diciembre 2015]. Disponible en: https://books.google.com/books?hl=en&lr=&id=a2MnCgAAQBAJ&oi=fnd&pg=PP3&dq=%22elements+of+a+pom.xml+file.+Then+you+learn+about+testing+the+project+using%22+%22and+documentation+such+as+Javadocs,+test+coverage+reports,+and%22+%22the+Source%22+%22welcome+reader+feedback.+If+you+have+any+questions+or+suggestions,+you+can%22+&ots=wmD0E2GAzv&sig=0noM4dFvHQLps32QykPsTco1td4.

**VISUAL PARADIGM, 2015.** What is Visual Paradigm? [En línea]. [Consulta: 15 diciembre 2015]. Disponible en: https://www.visual-paradigm.com/features/.

**WINDOWS CENTRAL, 2015.** Windows Phone Store - Everything you need to know! [En línea]. [Consulta: 2 noviembre 2015]. Disponible en: http://www.windowscentral.com/windows-phone-store.

## Bibliografía Consultada

**ABRAN, A., 2004.** Software Engineering Body of Knowledge. S.l.: s.n. ISBN 0-7695-2330-7.

**ACEBO, Y.B., 2012.** Gestión de la Calidad en el Ciclo de Desarrollo del Software de proyectos que usan metodologías ágiles. Revista Granma Ciencia [en línea], vol. 16. [Consulta: 5 febrero 2016]. ISSN 1027-975X. Disponible en: http://www.grciencia.granma.inf.cu/vol%2016/2/2012_16_n2.a9.pdf.

**AMARO SORIANO, J.E., 2012.** El gran libro de programación avanzada en Android. México, D.F.: AlfaOmega. ISBN 978-607-707-551-6.

**AMBYSOFT, 2006.** The Agile Unified Process. [En línea]. [Consulta: 17 enero 2016]. Disponible en: http://www.cc.una.ac.cr/AUP/.

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

**ANDROID OPEN SOURCE PROJECT, 2016.** Code Style for Contributors | Android Open Source Project. [En línea]. [Consulta: 23 marzo 2016]. Disponible en: http://source.android.com/source/index.html.

**AQUINO, F.R., 2015.** Metodología AUP. [En línea]. [Consulta: 17 enero 2016]. Disponible en: https://es.scribd.com/doc/117964344/Metodologia-AUP.

**BOGGS, M. y BOGGS, W., 2002.** UML with Rational Rose. New York: Sybex. ISBN 0-7821-4017-3.

**BOTTA, A., 2010.** Resumen de Teoría de: Diseño de Sistemas. [En línea]. Disponible en: http://es.scribd.com/doc/85235594/9/FLUJO-DE-TRABAJO-DE-CAPTURA-DEREQUISITOS.

**CABALLERO, I., GARCÍA, F.O. y VIZCAÍNO, A., 2015.** Prácticas Ingeniería del Software. Una Herramienta CASE para ADOO: Visual Paradigm. Análisis y Diseño Orientado a Objetos. S.l.: s.n.

**CANTILLO, R.A. y LEÓN, S.H., 2011.** El proceso de investigación científica. . La Habana:

**CELULARIS, 2010.** Celularis. Las compañías móviles más grandes del mundo. [En línea]. Disponible en: http://www.celularis.com/mercado/las-companias-moviles-mas-grandes-del-mundo.php.

**CENTER FOR HISTORY AND NEW MEDIA, 2016.** Guía rápida Zotero. [En línea]. [Consulta: 21 junio 2016]. Disponible en: http://zotero.org/support/quick_start_guide.

**COCKBURN, A., 2006.** Agile software development: the cooperative game. S.l.: Pearson Education.

**DHWAJ, A., 2015.** Design Pattern: MVC vs. MVP vs. MVVM | LinkedIn. [En línea]. [Consulta: 21 junio 2016]. Disponible en: https://www.linkedin.com/pulse/design-pattern-mvc-vs-mvp-mvvm-arun-dhwaj.

**DJANGO PROJECT, 2016a.** Django REST framework. [En línea]. Disponible en: ttp://www.django-rest-framework.org/.

**DJANGO PROJECT, 2016b.** Empezando | Django documentation | Django. [En línea]. Disponible en: https://docs.djangoproject.com/es/1.9/intro/.

**DJANGO PROJECT, 2016c.** The Web framework for perfectionists with deadlines | Django. [En línea]. Disponible en: https://www.djangoproject.com/.

**DOGTIEV, A., 2016.** App Development Marketplace. Mobile App Developer Statistics Roundup. [En línea]. Disponible en: http://appindex.com/app-development/mobile-app-developer-statistics-roundup/.

**DUQUE, A., 2014.** METODOLOGIAS DE DESARROLLO AUP. [En línea]. [Consulta: 17 enero 2016]. Disponible en: https://prezi.com/0w76rs1pojdq/metodologias-de-desarrollo-aup/.

**EDUARDO RIVERA, 2014.** Arquitectura de Software II - Diagrama de Componentes y Despliegue. [En línea]. Disponible en: http://es.scribd.com/doc/7884665/Arquitectura-de-Software-IIDiagrama-de-Componentes-y-Despliegue.

**ESTRADA, A.F., GARCÍA, T.C., PERDOMO, Y.L., CINTRA, A.V., MARTÍNEZ, R.D. y DÍAZ, R.C., 2011.** Una experiencia novedosa para el testing desarrollada por un departamento de pruebas de software. Revista Cubana de Ciencias Informáticas, vol. 5, no. 2. ISSN 1994 - 1536.

**FIESTAS, J., 2014.** ElevenPaths. [En línea]. Disponible en: http://blog.elevenpaths.com/2014/09/qa-pruebas-para-asegurar-la-calidad-del.html.

**FUENTES, J.L.D., ANILLO, M. del C.S. y GAMBERO, F.M.S., 2013.** Proceso unificado ágil (aup) [en línea]. 13 marzo 2013. S.l.: s.n. [Consulta: 17 enero 2016]. Disponible en: http://es.slideshare.net/joseluisdifu/proceso-unificado-gil-aup-17171038.

**GENBETA, 2014.** Genbeta: dev. [En línea]. Disponible en: http://www.genbetadev.com/metodologias-de-programacion/patrones-de-diseno-que-son-y-porque-debes-usarlos.

**GITHUB, 2015.** The RxJava Android Module · ReactiveX/RxJava Wiki · GitHub. [En línea]. [Consulta: 13 diciembre 2015]. Disponible en: https://github.com/ReactiveX/RxJava/wiki/The-RxJava-Android-Module.

**GITHUB, 2016.** Consuming APIs with Retrofit · codepath/android_guides Wiki · GitHub. [En línea]. [Consulta: 7 febrero 2016]. Disponible en: https://github.com/codepath/android_guides/wiki/Consuming-APIs-with-Retrofit.

**GLOBE TESTING, 2015.** Pruebas de regresión - Globe Testing. [En línea]. [Consulta: 27 junio 2016]. Disponible en: http://www.globetesting.com/pruebas-de-regresion/.

**GONZÁLEZ, J.P., 2014.** Pruebas de aceptación orientadas al usuario: contexto ágil. Sevilla: s.n.

**GUTIÉRREZ, D., 2009.** UML Diagrama de Paquetes. [En línea]. Disponible en: http://www.codecompiling.net/files/slides/UML_clase_05_UML_paquetes.pdf.

**HOLGATE, C., 2012.** LiveCode Mobile Development Beginner´s Guide. Birmingham: Packt Publishing Ltd.

**HOMMEL, S., 1999.** Convenciones de código para el lenguaje de programación Java. .

**IBM, 2006.** IBM Corp. Ayuda del Rational (Español). Rational Unified Process Versión 7.0.1. [En línea]. Disponible en: http://www.ibm.com/developerworks/ssa/rational/newto/.

**ISO 25000, 2015.** ISO 25010. [En línea]. [Consulta: 23 junio 2016]. Disponible en: http://iso25000.com/index.php/normas-iso-25000/iso-25010.

**ITIL® FOUNDATION, 2011.** ITIL. Herramientas y metodologías. [En línea]. Disponible en: http://itilv3.osiatis.es/proceso_mejora_continua_servicios_TI/herramientas_metodologias.php.

**JEREMY DICK, JACKSON, K. y HULL, E., 2011.** Requirements Engineering. 3rd. Londres: Springer. ISBN 978-1-84996-404-3.

**LAMANCHA, B.P., 2007.** EVA. Gestión de las Pruebas Funcionales. [En línea]. 2007. S.l.: s.n. Disponible en: http://eva.uci.cu/file.php/158/Documentos/Recursos_bibliograficos/Libros_y_articulos_UD_2/Comun/Gestion_de_las_Pruebas_Funcionales.pdf.

**LINARES, J., VIDAL, M. y LEÓN, M., 2013.** Diagrama de clases. UNIVERSIDAD BICENTENARIA DE ARAGUA VICERRECTORADO ACADÉMICO ESCUELA DE INGENIERIA TURMERO– ESTADO ARAGUA. [En línea]. Disponible en: http://es.slideshare.net/nedowwhaw/diagrama-de-clases-16208245.

**LONDOÑO, J.H.A., 2005.** Ingeniería de Software. [En línea]. Disponible en: http://ing-sw.blogspot.com/2005/04/tipos-de-pruebas-de-software.html.

**MARKETING CLOUD, 2015.** 2014 Mobile Behavior Report. [En línea]. [Consulta: 23 junio 2016]. Disponible en: https://www.marketingcloud.com/sites/exacttarget/files/deliverables/etmc-2014mobilebehaviorreport.pdf.

**MASTER MAGAZINE, 2010.** Definición de Arquitectura de Software. [En línea]. Disponible en: http://www.mastermagazine.info/termino/3916.php.

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

**MORGILLO, I., 2015.** RxJava Essentials. 1st. Birmingham: Packt Publishing Ltd. ISBN 978-1-78439-910-8.

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

**PRUEBAS DE SOFTWARE., 2015.** Gestión de Calidad y Pruebas de Software. [En línea]. Disponible en: http://www.pruebasdesoftware.com/laspruebasdesoftware.htm.

**RICHTER, F., 2016.** Statista. Smartphone Infographics. [En línea]. Disponible en: https://www.statista.com/chartoftheday/Smartphone/.

**SALUNKHE, S., 2015.** Difference between MVC, MVP and MVVM | LinkedIn. [En línea]. [Consulta: 21 junio 2016]. Disponible en: https://www.linkedin.com/pulse/difference-between-mvc-mvp-mvvm-swapneel-salunkhe.

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

**SCRIBD INC., 2015.** Scribd Inc. Requerimientos funcionales y no funcionales. [En línea]. Disponible en: http://es.scribd.com/doc/37187866/Requerimientos-funcionales-y-no-funcionales#scribd.

**SLONIM, J., NORMAN, R.J. y MULLER, H.A., 2011.** Computer Aided Software Engineering. New York: Springer. ISBN 978-4612-8626-4.

**SQUARE, 2016.** GitHub | Retrofit. [En línea]. Disponible en: http://square.github.io/retrofit/.

**STACK OVERFLOW, 2016.** Stack Overflow. Developer Survey 2016 Results. [En línea]. Disponible en: http://stackoverflow.com/research/developer-survey-2016.

**T.EMMATTY, J., 2011.** Differences between MVC and MVP for Beginners - CodeProject. [En línea]. [Consulta: 21 junio 2016]. Disponible en: http://www.codeproject.com/Articles/288928/Differences-between-MVC-and-MVP-for-Beginners.

**TUYA, J., ROMÁN, I.R. y COSÍN, J.D., 2007.** Técnicas cuantitativas para la gestión en la ingeniería del software. [En línea]. La Coruña: NetBiblo, S. L. [Consulta: 27 junio 2016]. ISBN 978-84-9745-204-5. Disponible en: https://books.google.com.cu/books?id=PZQoZ9KTNaEC&printsec=frontcover&hl=es&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false.

**UNIVERSIDAD DE PUERTO RICO, 2011.** Zotero: Guía básica. [En línea]. [Consulta: 22 junio 2016]. Disponible en: http://www.uprm.edu/library/docs/tutorias/zotero_guia4taEd.pdf.

**VILLEGAS, J.O.P., 2014.** Comunidad de Android de la UCI. ¿Qué es Material Design? [En línea]. [Consulta: 15 diciembre 2015]. Disponible en: http://android.uci.cu/2014/11/que-es-material-design/.