Universidad de las Ciencias Informáticas

FACULTAD 6

Título: Visor de reportes para móviles con sistema operativo Android

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

Autores: Miguel Enrique Verdecia Fonseca

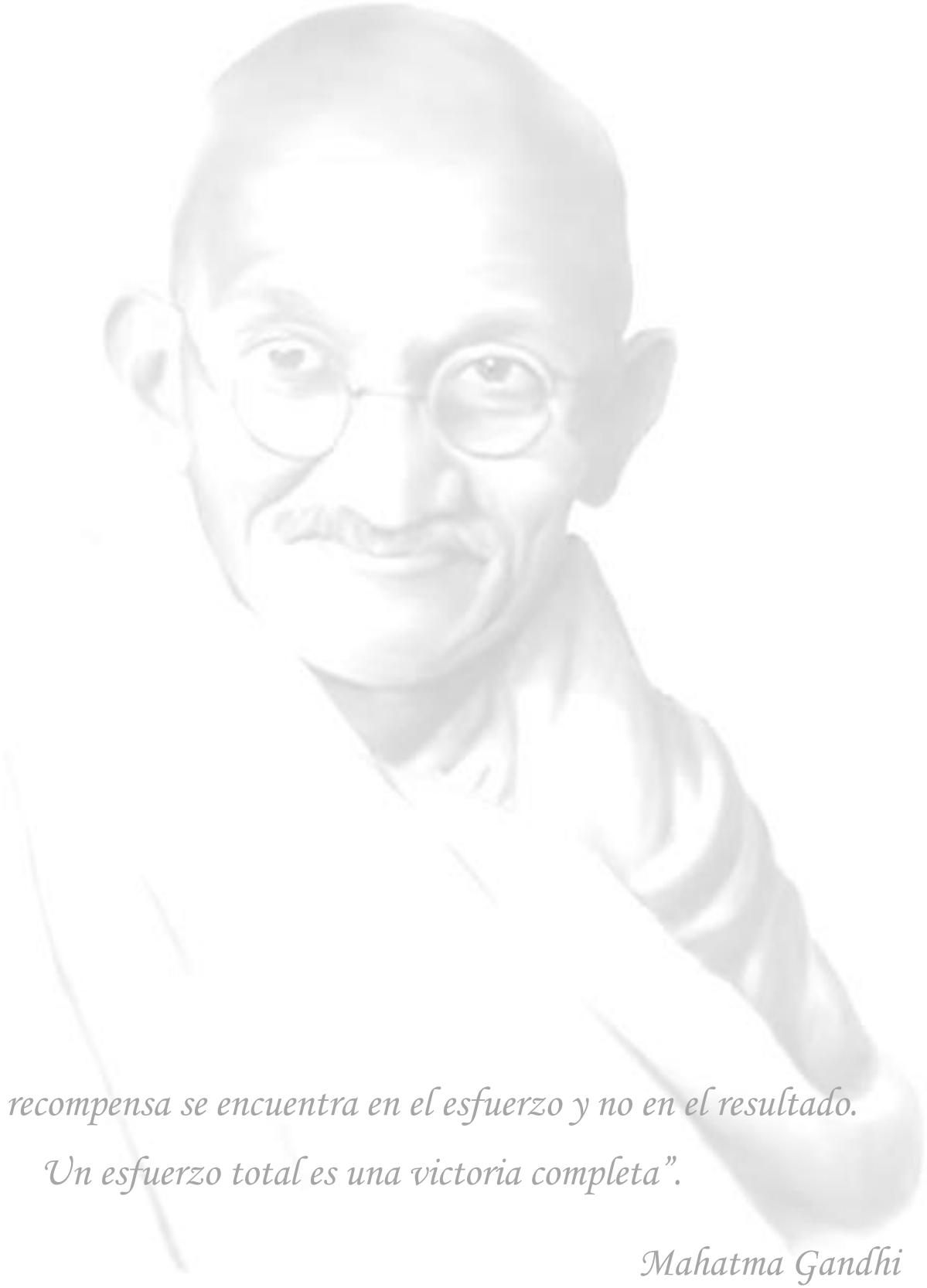
Osvaldo Manuel González Ortiz

Tutores: Ing. Clara Elena Brizuela Figueredo

Ing. Aldis Joan Abreu Medina

La Habana, 17 de junio del 2013

"Año 55 de la Revolución"



"Nuestra recompensa se encuentra en el esfuerzo y no en el resultado.
Un esfuerzo total es una victoria completa".

Mahatma Gandhi

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Miguel Enrique Verdecia Fonseca

Firma del Autor

Osvaldo Manuel González Ortiz

Firma del Autor

Ing. Aldis Joan Abreu Medina

Firma del Tutor

Ing. Clara Elena Brizuela Figueredo

Firma del Tutor

DATOS DE CONTACTO

Autor: Miguel Enrique Verdecia Fonseca

Universidad de las Ciencias Informáticas

E-mail: meverdecia@estudiantes.uci.cu

Autor: Osvaldo Manuel González Ortiz

Universidad de las Ciencias Informáticas

E-mail: omgonzalez@estudiantes.uci.cu

Tutor: Ing. Clara Elena Brizuela Figueredo

Ingeniero en Ciencias Informáticas

E-mail: cebrizuela@uci.cu

Tutor: Ing. Aldis Joan Abreu Media

Ingeniero en Ciencias Informáticas

E-mail: ajabreu@uci.cu

DEDICATORIA

Dedicar este trabajo a todas las personas que han sido partes fundamentales en mi desarrollo profesional sería consumir éstas 80 páginas en solo nombres. Pero, en mi vida, solo enfocado en mi vida, sería lo más sencillo. Por ello, se lo dedico a la luz de mis ojos, al más grande amor de mi vida y responsable de darme fuerzas para poder terminar esta carrera y dedicarme por fin, de entero a ella; a mi bebé, Vanessa. A mis padres, por haberme apoyado durante estos cinco años y haberse sacrificado tanto cada uno por permitirme terminar esta universidad y sobre todo, por apoyarme en cada uno de los pasos que ha dado su nietecita. A mi mujer, por animarme, ayudarme y apoyarme en cada situación que he tenido e impulsarme a hacer lo correcto en cada idea loca que he tenido por retirarme. Mi cielo, ya no tienes que llorar más mi ausencia, mi nené, a partir de ahora podrá tener a su padre cada día de su vida. Por último, a mi mamá: mami, ya puedes dejar de preocuparte, ya tienes un ingeniero en casa. Los quiero.

Osvaldo Manuel González Ortiz

A mi madre Martha Fonseca Cedeño, por ser padre y madre para mí, por el amor y la dedicación con que me ha educado, el constante apoyo y confianza sin la cual no hubiese llegado hasta aquí.
A mi novia Alis Rubio Cuevas por estar siempre a mi lado dándome apoyo en las buenas y en las malas.
A mis abuelos Devora Cedeño y Juan Olimpo Fonseca, por su amor y cariño.
A mi tía Nerza Fonseca, por ser más que una tía para mí.
A toda mi familia por esperar siempre lo mejor de mí.

Miguel Enrique Verdecia Fonseca

RESUMEN

El sistema Generador Dinámico de Reportes luego de un análisis íntegro a sus módulos, identificó algunos factores que repercuten en la disponibilidad de sus reportes por problemas de dependencias de conexiones directas, navegadores y necesidad de poseer en su totalidad el sistema para la captura de estos elementos. Para evitar estas dependencias, se trazó la meta de manejar en tiempo real la información deseada por un usuario, logrando el acceso a estos datos de forma *offline*[1] o desconectado mediante el uso de la telefonía celular. Se implementó la herramienta visor de reportes para móviles con sistema operativo Android, que permite obtener los reportes existentes de cada usuario eliminando las dependencias actuales; con posibilidades de almacenar en el dispositivo los reportes deseados, visualizar su contenido, filtrar los elementos reales de cada reporte, así como exportarlos a formato pdf para su posterior traslado. Todas estas funcionalidades son realizadas localmente en el teléfono móvil sin variación alguna a las existentes en el sistema Generador Dinámico de Reportes. Esto fue posible a raíz de la caracterización de las diferentes herramientas, tecnologías y metodologías empleadas para su desarrollo, permitiendo el diseño de las clases del sistema e implementación de las mismas, aplicando buenas prácticas de los patrones de diseño y de reutilización de componentes.

PALABRAS CLAVE

Android, Generador Dinámico de Reportes, offline, patrones de diseño.

[1] Offline: no posee conexión directa al sistema o recurso solicitado.

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE IMÁGENES

INTRODUCCIÓN

Nos encontramos en una época de cambios donde la sociedad trata cada vez más de independizarse de los medios computacionales. Esto lo ha logrado gracias a la utilización de la tecnología móvil que con el paso del tiempo ha aumentado su nivel de utilidad y funcionalidad.

Esta tecnología nace el pasado siglo desde que Martin Cooper desarrollaba el 3 de Abril de 1973 el primer teléfono móvil, el Motorola DYNA TAC (1). Con su aparición aparecieron varias empresas cuyo aprovechamiento del uso de esta tecnología se ha reflejado en la transformación de la forma de comunicarse entre personas. Con un dispositivo móvil es posible realizar desde una simple llamada telefónica hasta navegar por toda la web. Estos adelantos han traído consigo la búsqueda de la portabilidad de los productos de diferentes empresas, aprovechando los servicios que brinda esta tecnología.

En Cuba a pesar de que la telefonía celular por cuestiones estratégicas, no ha podido ser adquirida por la mayor parte de la población, es cierto que cada día aumenta exponencialmente la cantidad de usuarios interesados en la adquisición de contratos para esta importante vía de comunicación. Es por ello que es de interés de la dirección del país la utilización de las potencialidades de la telefonía celular en el amplio espectro de las Tecnologías de la Informática y las Comunicaciones (TICs).

La Universidad de las Ciencias Informáticas (UCI), cuyo objetivo fundamental es la informatización de la sociedad cubana, constituye la sede de varios centros de producción de software. Cuenta con diferentes líneas de investigación donde se ha comenzado a fomentar el uso y examen de esta nueva tecnología que va incrementando su auge a nivel nacional. Específicamente en la facultad 6 radican los centros de Geoinformática y Señales Digitales (GEYSED) y el Centro de Tecnologías de Gestión de Datos (DATEC). En este último se encuentra la línea de desarrollo Integración de Soluciones, que posee entre sus principales tecnologías de desarrollo; la Plataforma de Ayuda para la Toma de Decisiones y Soluciones Integrales (PATDSI). En dicha plataforma está incluido como uno de sus componentes, el sistema Generador Dinámico de Reportes (GDR) que posee como objetivo garantizar la generación dinámica de reportes partiendo de datos persistentes en algún origen de datos soportado por el sistema (PostgreSQL, MySQL Server, SQLite, Microsoft SQL Server, Oracle).

Entre los módulos que la conforman se encuentra el Visor de Reportes (VR) que permite la visualización de los reportes generados. A raíz del análisis en este módulo se han identificado algunos factores que

constituyen limitaciones en la obtención de los reportes debido a la dependencia existente de una conexión directa y un navegador para lograr el acceso a ellos y dificultades que presenta al mostrar los reportes solicitados, si no se cuenta con el sistema GDR en su totalidad. Estas limitantes imposibilitan el acceso a la información, ya sea por un personal directivo o cliente, impidiendo la disponibilidad de este servicio.

A partir de esta situación surge el siguiente **problema de la investigación**: ¿Cómo contribuir a la visualización de los reportes de manera *offline* en un teléfono móvil con sistema operativo Android a través del Generador Dinámico de Reportes?

Este trabajo tiene como **objeto de estudio** el desarrollo de herramientas para dispositivos móviles con sistema operativo Android y como **campo de acción,** el desarrollo de una herramienta para visualizar los reportes en dispositivos móviles con sistema operativo Android.

Para darle solución al problema planteado se define como **objetivo general** desarrollar una herramienta para dispositivos móviles con sistema operativo Android que permita la visualización de los reportes generados por el Generador Dinámico de Reportes.

Del objetivo general se desglosan los siguientes **objetivos específicos**:

- Construir el marco teórico de la investigación para el desarrollo de aplicaciones Android para la visualización de reportes.
- Realizar el análisis y diseño del visor de reportes para móviles con sistema operativo Android.
- Implementar el visor de reportes para móviles con sistema operativo Android.
- Realizar pruebas funcionales a la herramienta implementada.

Para cumplir con el objetivo propuesto y resolver la situación problemática mencionada, se plantean las siguientes **tareas de la investigación**:

- Caracterización de la metodología, herramientas y tecnologías a utilizar en el desarrollo del visor de reportes para móviles con sistema operativo Android.
- Identificación de los requisitos funcionales y no funcionales para el visor de reportes para móviles con sistema operativo Android.

- Diseño de la herramienta visor de reportes para móviles con sistema operativo Android.
- Implementación de las funcionalidades identificadas.
- Diseño de los casos de pruebas para determinar el correcto funcionamiento de los requisitos
- Realización de pruebas funcionales a la herramienta.

Capítulo 1 Fundamento Teórico: en este capítulo se presenta la definición del marco teórico de la investigación, se fundamenta el uso de los distintos temas referentes a la metodología, herramientas y tecnologías a utilizar para el desarrollo del visor de reportes para móviles con sistema operativo Android.

Capítulo 2 Análisis y Diseño: en este capítulo se describe detalladamente las características del sistema a desarrollar. Se especifica el Modelo de Dominio como vía para describir los principales conceptos del negocio, los requisitos funcionales y no funcionales, así como el diagrama de casos de uso del sistema y las descripciones de los mismos. Se modelan y detallan los diagramas que representan las funcionalidades del sistema, a partir de los patrones de diseño identificados.

Capítulo 3 Implementación y Prueba: en este capítulo se muestra el modelo de implementación como resultado del diseño anteriormente desarrollado, así como el diagrama de componentes de la herramienta. Se describen las pruebas a realizar, con el objetivo de comprobar las funcionalidades de la misma en los diferentes escenarios, para verificar en todos los casos que los resultados de las pruebas sean los esperados y el correcto funcionamiento del producto.

CAPÍTULO 1: FUNDAMENTO TEÓRICO 1

Introducción al capítulo 1

En el siguiente capítulo se abordan los conceptos relacionados con la investigación, que constituyen el conocimiento fundamentalmente teórico sobre los diferentes procesos y tecnologías asociadas al desarrollo de la solución requerida.

1.1. Conceptos Básicos

1.1.1. Tecnología móvil

La tecnología móvil no es más que una de las ramas existentes dentro de los sistemas de comunicación. Esta se basa exclusivamente en sus sistemas de conexión inalámbrica. Posibilita la conversión a señales electromagnéticas de los sonidos que viajan a través del aire hasta ser captadas. Posteriormente son transformadas por antenas repetidoras o vía satélite en mensajes hasta llegar a su receptor.

Los teléfonos celulares, dispositivos de frecuencia dual para el habla y la escucha han redefinido todo nuestro punto de vista de lo que era y ahora se nace en las comunicaciones, como un elemento que la hace brillar dentro de las relaciones humanas modernas.

Innumerables son los cambios que en esta nueva tecnología han surgido, incluyendo accesibilidad, facilidad, servicios y estructura física de los teléfonos celulares, aunque continúen siendo transmisores personales, independientemente de sus características (2).

1.1.2. Comunicaciones Inalámbricas

Las comunicaciones inalámbricas son un conjunto de sistemas de comunicación y tecnologías asociadas, que utilizan el espectro radioeléctrico como vehículo de comunicación. Dentro de los estándares de estas

comunicaciones, las más utilizadas en la actualidad son las WiFi[2]. Este tipo de comunicación se puede utilizar en oposición al término comunicaciones fijas, debido a que estas últimas utilizan habitualmente un medio físico basado en cable o fibra óptica.

En el ámbito de las telecomunicaciones, pueden ser agrupadas en comunicaciones móviles, por satélite o de acceso inalámbrico fijo.

Brindan numerosas ventajas en comparación con las redes cableadas (3):

- Capacidad para una gran cantidad de suscriptores.
- Uso eficiente del espectro electromagnético debido a la utilización repetida de frecuencias.
- Compatibilidad a nivel internacional y nacional, para que los usuarios puedan utilizar sus equipos en otros países o áreas.
- Prestación de servicios para aplicaciones de voz, video y datos.
- Adaptación a la densidad de tráfico.
- Calidad del servicio en el caso de la voz comparable a servicios telefónicos tradicionales.

1.1.3. Plataforma de software

Una plataforma de software es un elemento crucial en el desarrollo de software. Mediante el marco de trabajo que proporcionan las plataformas se puede crear un nuevo software y permitir su ejecución sobre ellas. Las plataformas de desarrollo típicas incluyen una arquitectura de computadoras, un sistema operativo (S.O.), lenguajes de programación y sus correspondientes librerías o interfaces gráficas (*user interface* o UI) (4).

Las plataformas son comúnmente mencionadas con las APIs (interfaces de programación de aplicaciones que brindan funciones y métodos que ofrece una biblioteca para ser utilizada por otro software como una

[2] WiFi: Fidelidad sin Cables (*Wireless-Fidelity*) es un conjunto de estándares para redes inalámbricas creadas para ser utilizada en redes locales de este tipo, que frecuentemente en la actualidad son utilizadas para acceder a Internet.

capa de abstracción). Un conjunto completo de APIs constituye una plataforma de software. Estas plataformas son normalmente dependientes de los sistemas operativos aunque existen otras en las que no es así.

Java es un ejemplo de plataforma no dependiente del S.O. debido a que Java es tanto el lenguaje como la plataforma de desarrollo, la cual incluye la Máquina Virtual de Java (JVM), cuya función es interpretar el código binario resultante de la compilación del código fuente del programa en Java (5).

1.2. Sistema operativo Android.

1.2.1. Breve historia

En el año 2006 Google comenzó a ofrecer servicios móviles para distintos equipos. Empezó con Mapas de Google (*Google Maps*) ofreciéndolo en versión de web modelos; luego siguió con Java Micro Edición (J2ME), Symbian, BlackBerry e incorporó la búsqueda y el servicio de mensajería instantánea (6).

En noviembre del 2007 Google anunciaba Android como una plataforma móvil a raíz de la adquisición de esta compañía de una pequeña empresa llamada Android Inc que comenzaba el desarrollo de este sistema operativo para móviles. Casi un año después del anuncio de este sistema operativo, abierto y gratuito; así como su compilador y emulador, salió al mercado el primer equipo con SO Android, el HTC G1, precursor de este SO y responsable de la posterior incorporación a este mercado de otras marcas como Samsung, Motorola y Sony Ericson.

Desde un punto comercial, este sistema podría distribuirse en dos modalidades, utilizar Android como un producto de código abierto (*Open Source* por sus siglas en inglés) o convertirlo en un GPhone, un teléfono con servicios de Google (6).

1.2.2. Android

A nivel internacional Android es tratado como un sistema operativo, cuando realmente es un paquete de software que está pensado para instalarlo en cualquier equipo móvil. Este paquete de software incluye un SO Linux, sistema operativo de código abierto más conocido y provee además de una capa de librerías escritas en C y C++, y un marco de trabajo para el desarrollo de aplicaciones. Originalmente este marco de trabajo estaba solo basado en Java y posteriormente se liberó para aplicaciones nativas en C, aunque no

es recomendable a menos que se necesite utilizar de manera excesiva al microprocesador. Incluye además una suite de aplicaciones iniciales, por ejemplo, la aplicación que manejan los contactos y los correos electrónicos (6).

¿Quiénes arman Android?

Google es su principal impulsor, pues es el que lo mantiene, pero en realidad para hacerlo más abierto y que no sea un sistema operativo manejado por una sola empresa, creó una organización sin fines de lucro denominada la Alianza Abierta de Dispositivos (*Open Handset Alliance*). Incluye a más de 65 empresas del sector de fabricación de dispositivos móviles, entre los que se encuentran a Google, fabricantes de equipos Motorola, HTC, LG, Samsung, Alcatel; operadores móviles como Telefónica, Movistar, T-Mobile, Sprint, China Mobile y fabricantes de microprocesadores y placas de video Intel, nVidia, Qualcomm, Texas Intruments, Huawei, entre otros. Estas empresas son unas de las pocas que forman la alianza, responsables de mantener a Android como sistema operativo (6).

1.2.3. Características de Android

- **Marco de trabajo de aplicaciones**: permite el reemplazo y la reutilización de los componentes.
- **Navegador integrado**: basado en los motores *Open Source Webkit*.
- **SQLite**: base de datos para almacenamiento estructurado que se integra directamente con las aplicaciones.
- **Multimedia**: soporte para medios con formatos comunes de audio, video e imágenes planas (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF).
- **Máquina virtual Dalvik**: base de llamadas de instancias muy similar a Java (7).

1.2.4. Ventajas y Desventajas del uso de Android

Ventajas

- Puede instalarse prácticamente en todo tipo de dispositivos, sean móviles, portátiles e incluso microondas.

- Puede adaptarse a la perfección a todo tipo de necesidades.
- Es libre con licencia Apache y código abierto para que un desarrollador no solo pueda modificar su código sino también mejorarlo.
- Garantiza que, en caso de haber un error, sea detectado y reparado con mayor presteza sin depender de nadie para pedir autorización a su cambio.

Desventajas

- Consumo de la batería: debido a la posibilidad de tener varias aplicaciones corriendo a la vez en el sistema esto implica un gasto incremental en el consumo de la batería.
- Poco intuitivo: es un sistema operativo un tanto complicado, sobre todo a la hora de configurar aquellos dispositivos que cuenten con este tipo de sistemas pudiendo llevar mucho tiempo para ello.
- Fragmentación Total: posee algunos problemas de compatibilidad con aplicaciones no creadas por su compañía**,** teniendo que adaptarlos a cada teléfono móvil y dispositivo que utilice esta tecnología (8).

1.2.5. Arquitectura Android

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

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

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

Librerías: Android incluye un conjunto de librerías desarrolladas en C y C++ algunas de las cuales no son desarrolladas por Google o por el proyecto Android, sino que son existentes en el mundo de *Open Source*.

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

Núcleo - Linux: Android depende en la última de sus versiones, del núcleo 3.0.31 de Linux para los servicios base del sistema como seguridad, gestión de memoria, gestión de procesos, pila de red, y modelo de drivers. El núcleo también actúa como una capa de abstracción entre el hardware y el resto de la pila (7).

Figura 1. Arquitectura de componentes de Android (7)

1.2.6. Android SDK

El paquete de Desarrollo de Software de Android (SDK por sus siglas en inglés, *Software Development Kit*), no es más que un conjunto de herramientas de desarrollo que le permite a un programador crear aplicaciones para Android. Estos proporcionan bibliotecas de interfaz de programación de aplicaciones creadas con el fin de permitir un mejor uso de las técnicas a desarrollar en este sistema (9).

Los SDK continuamente contienen códigos de ejemplo y notas técnicas de soporte para ayudar a clarificar ciertos puntos del material de referencia. Este marco de trabajo es tan sencillo como las APIs, creadas con el objetivo de permitir el uso de cierto lenguaje de programación, que puede incluir también hardware sofisticado para comunicarse con este sistema (9).

1.2.7. Distribuciones de versiones Android

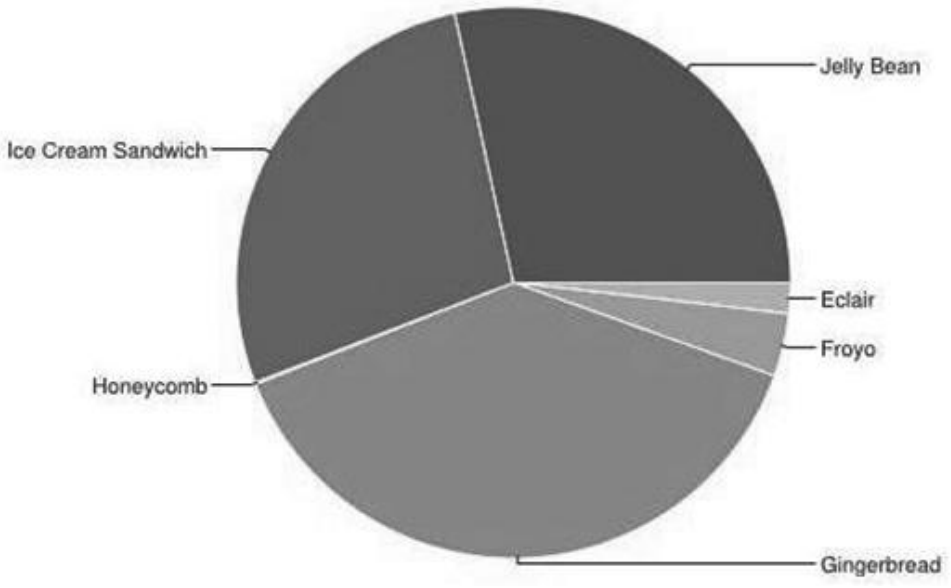


Figura 2. Distribuciones de versiones Android (10)

Plataforma	Nombre	Nivel API	Uso por distribución
Android 2.1	**Eclair**	7	4.1 %
Android 2.2	**Froyo**	8	7 %
Android 2.3 Android 2.3.2	**Gingerbread**	9	33.6 %
Android 2.3.3 Android 2.3.7		10	
Android 3.1	**Honeycomb**	12	1%
Android 3.2		13	
Android 4.0 Android 4.0.2	**Ice Cream Sandwich**	14	25.9 %
Android 4.0.3		15	

Android 4.0.4			
Android 4.1	**Jelly Bean**		28.4 %

Tabla 1. Uso de las distribuciones Android (10)

Teniendo en cuenta las facilidades que brinda Android al desarrollar aplicaciones, se determina el uso de este sistema operativo abierto como base en la línea de investigación de este trabajo. Se selecciona su versión 2.3.3 Gingerbread por su impacto y uso en el mercado, así como la existencia de numerosos dispositivos con este sistema. Se incluye además como marco de trabajo el SDK correspondiente para el desarrollo de la herramienta.

1.3. Metodologías de desarrollo de software

Las metodologías de desarrollo de software tienen como objetivo proveer a los desarrolladores de una guía para la elaboración de un producto de software, incluyendo técnicas, herramientas, soporte documental y procedimientos. Estas se han visto en la obligación de desarrollarse a la par de las necesidades de cada nueva creación debido a la diversificación de requisitos que presenta cada software.

1.3.1. Proceso Unificado Abierto

El Proceso Unificado Abierto (OpenUP por sus siglas en inglés *Open Unified Process*) es un marco de trabajo de proceso de desarrollo de software de código abierto. Está basado en el Proceso Unificado de Racional (RUP por sus siglas en inglés *Rational Unified Process*), desarrollado por el Negocio Internacional de Máquinas o IBM (por sus siglas en inglés *International Business Machines*). El mismo está orientado principalmente al tratamiento y gestión de aquellos proyectos de software que poseen una estructura en su desarrollo incremental, ligero y de manera iterativa. Es un proceso apropiado para proyectos pequeños, de un corto período de desarrollo y de bajos recursos.

Los 4 principios básicos por lo que se caracteriza OpenUP son:

- Colaboración para unificar intereses y compartir conocimientos.
- Balancear las prioridades para maximizar las necesidades de los stakeholders[3].
- Enfoque en la articulación de la arquitectura.
- Desarrollo continuo para obtener realimentación y realizar las mejoras respectivas (11).

Al ser una metodología ágil, orientada al código y ajustable a los continuos cambios del proceso de desarrollo; el grupo de arquitectura de la herramienta visor de reportes para móviles con sistema operativo Android, conjuntamente con el departamento y el grupo de arquitectura del proyecto GDR se decide utilizar OpenUP para guiar todo el proceso de desarrollo de la herramienta.

1.4. Lenguaje de Modelado

Los lenguajes de modelado poseen como objetivo principal visualizar de manera gráfica el sistema que se desarrollará, como ayuda técnica a los ingenieros implicados. Estos pueden utilizarse a la hora de la comunicación con el o los clientes, grupo de desarrollo y otros factores que intervengan en este proceso.

1.4.1. Lenguaje de modelado UML

Lenguaje Unificado de Modelado (UML por sus siglas en inglés, *Unified Modeling Language*) es el lenguaje para la especificación, visualización, construcción y documentación de los artefactos de un proceso de sistema intensivo. Es uno de los lenguajes de modelado de sistemas de software más conocido y utilizado en la actualidad respaldado por el Grupo de Administración de Objetos (OMG, por sus siglas en inglés, *Object Management Group*) (12).

[3] Stakeholder: Usuario, grupo de personas, organización u otra entidad que posee un interés directo o indirecto en un sistema, que puede afectar o afectarse por las acciones u objetivos del mismo.

Características de UML

UML permite la realización de diagramas de casos de uso, identificar la visión estática con diagramas de clases y diagramas de objeto. Define la visión dinámica, tales como los diagramas de secuencia, de actividad, de los estados, de colaboración y el despliegue de componentes que forman el sistema (13).

Proporciona ventajas en la representación del ciclo de vida de un software y de los artefactos específicos del Proceso Unificado de Desarrollo del Software. Posibilita la comunicación sencilla y rápida entre programadores y los clientes del software que se desarrolla, brindando la posibilidad de que estos últimos puedan expresar su conformidad con el producto o las nuevas mejoras que desean ver introducidas (14).

UML es uno de los lenguajes para la modelación de propósito general evolutivo, ampliamente aplicable y soportado por herramientas de modelado. Este se aplica a una multitud de diferentes tipos de sistemas, dominios, métodos y procesos. Debido a estas utilidades que posee, así como por ser el más utilizado y conocido en la actualidad se decide utilizar este lenguaje de modelado, definido dentro de la descripción de la arquitectura de la línea del proyecto.

1.5. Herramientas CASE para la modelación del sistema

La Ingeniería de Software Asistida por Computadoras (CASE, por sus siglas en inglés, *Cumputer Aided Software Engineering*) es un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores durante todos el paso del ciclo de vida de desarrollo de un software. Este puede ser generalmente aplicado a cualquier sistema o colección de herramientas que ayudan a automatizar el proceso de diseño y desarrollo de software (15).

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

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

Una de las herramientas CASE considerada como muy completa y fácil de usar, con un soporte multiplataforma que proporciona excelentes facilidades de interoperabilidad con otras aplicaciones es el Visual Paradigm.

1.5.1. Visual Paradigm 8.0

El Visual Paradigm es una herramienta UML de ingeniería de software que favorece el desarrollo de los programas informáticos desde su planificación hasta el análisis y diseño. Su propósito general es llevar el ciclo de vida completo del desarrollo de software ya sea: análisis y diseño, construcción, pruebas y despliegue a través de la forma de representación de todo tipo de diagramas.

Como herramienta CASE posee numerosas ventajas como:

- Apoya todo lo básico en cuanto a artefactos generados en las etapas de definición de requerimientos y de especificación de componentes.
- Tiene apoyo adicional en la generación de artefactos automáticos.
- Generación de código e ingeniería inversa: brinda la posibilidad de generar código a partir de los diagramas, para las plataformas como .Net, Java y PHP, así como obtener los diagramas a partir del código.
- Generación de documentación: brinda la posibilidad de documentar todo el trabajo sin necesidad de utilizar herramientas externas.

Visual Paradigm permitir la generación de los diseños centrados en casos de uso y enfocados al negocio, así como la seguridad de que el modelo y el código permanezcan sincronizados en todo el ciclo de desarrollo (16). Por estas características, utilidades, posibilidades, incluyendo las ventajas antes expuestas, se elige esta herramienta de modelado pues cumple con la necesidad de confiabilidad para el desarrollo de la herramienta visor de reportes para móviles además de encontrarse definida dentro de la descripción de la arquitectura de la línea del proyecto.

1.6. Lenguaje de Programación

Un lenguaje de programación es un lenguaje artificial que puede utilizarse para definir una secuencia de instrucciones para su procesamiento por un ordenador o computadora. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión o como modo de comunicación humana (17).

1.6.1. Lenguajes y herramientas de programación para sistemas Android

El desarrollo de aplicaciones para Android ha sido expandido a una serie de herramientas de programación, las mayorías privativas pero capaces de brindar facilidades en este desarrollo. Trabajan con diferentes lenguajes de programación pero, a pesar de las diferencias que poseen entre ellas, existe un gran grupo que utilizan Java para la creación de estas aplicaciones nativas, pues es el lenguaje base de Android y en lo que la mayoría de sus aplicaciones se encuentran creadas.

Entre algunas de las herramientas para el desarrollo de software para Android se encuentran:

Plataforma	Lenguaje	Descripción
Basic4Android	**Visual Basic**	Es una plataforma de programación orientada a aquellas personas sin conocimientos avanzados en procesos de desarrollo. Su sintaxis es basada en la estructura de Microsoft Mobile a pesar de ser en otro lenguaje. No es gratuita, por lo que conlleva a la necesidad de incrementar los costos de desarrollo al tener que comprar la versión deseada para el desarrollo de la herramienta así como a la hora de actualizar alguna que otra mejora (18).
Mono	**Java**	Plataforma de programación que solo cuenta con la necesidad de tener que instalar el SDK de Android, la versión para Android. Trabaja con un lenguaje nativo para Android pues no posee un intérprete al igual que Basic4Android, haciendo su aprendizaje relativamente más sencillo en un tiempo prudente. Al igual que Basic4Android esta plataforma no es libre además de ser altamente costosa en su versión básica (19).

App Inventor	**Java**	Esta plataforma de desarrollo impulsada por Google hace un tiempo con el fin de que más personas se unieran a la familia de Android; está basada en un lenguaje de desarrollo gráfico en donde no escribes ni una sola línea de código por lo que evita cualquier lenguaje actual de programación, tan solo arrastras bloques identificados con la acción que necesitas hacer y listo. A pesar de ser tan ágil, poseer una amplia accesibilidad no es recomendable para el desarrollo de herramientas de mayor escala o estructura lógica debido a la simpleza de su uso y a la exclusión de lenguajes de programación, impidiendo hacer trabajos más detallados de lo que se desee implementar. No es una herramienta libre, pero no posee precio alguno en el mercado para su uso (20).
LiveCode	**Java**	Plataforma de programación abarcadora que posibilita trabajar para Android, Windows, Linux, iPhone, iPad, web y para servidores. Utiliza una programación orientada a eventos. A pesar de las ventajas que posee, es una plataforma privativa cuya licencia solamente sería más costosa que si solo necesitaras programar para Android.

Tabla 2. Plataformas y lenguajes para programar en Android

El desarrollo de programas para Android se hace habitualmente con el lenguaje de programación Java y el conjunto de herramientas de desarrollo SDK, pero existen otras disponibles, a pesar de que este lenguaje es potente y eficaz para el desarrollo de aplicaciones de este tipo. La plataforma Android se ha desarrollado hasta ser una de las más elegidas por los desarrolladores para plataformas móviles (21).

Java

La principal característica de Java es la de ser un lenguaje compilado e interpretado, a la vez de ser de código abierto. Todo programa en Java ha de compilarse y el código generado es interpretado por una máquina virtual. De este modo se consigue la independencia de la máquina, el código compilado se ejecuta en máquinas virtuales que si son dependientes de la plataforma.

Java es un lenguaje orientado a objetos de propósito general. Aunque Java comenzará a ser conocido como un lenguaje de programación de applets[4] que se ejecutan en el entorno de un navegador web, se puede utilizar para construir cualquier tipo de proyecto (22).

Su uso y facilidades para la creación de aplicaciones Android así como compatibilidad, documentación y tendencias actuales, la hacen la solución más óptima para incorporar a la solución propuesta como parte de la línea de investigación que se lleva a cabo.

1.7. Entorno de desarrollo integrado

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

1.7.1. Eclipse 3.7

Eclipse es un entorno de desarrollo integrado, de código abierto y multiplataforma. Generalmente es utilizado para desarrollar lo que se conoce como "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Es un potente y completo entorno de desarrollo que posibilita la creación y compilación de elementos variados como sitios web, programas en C++ o aplicaciones Java (23).

Aplicaciones de cliente enriquecido

Un cliente enriquecido es un término medio entre el cliente liviano y el cliente pesado. El objetivo del cliente enriquecido consiste en proporcionar una interfaz gráfica, escrita con una sintaxis basada en XML, que proporciona funcionalidades similares a las del cliente pesado (arrastrar y soltar, pestañas, ventanas múltiples, menús desplegables) (24).

[4] Applets: programas que pueden incrustarse en un documento HTML.

Cliente liviano

El término "cliente liviano" (a veces llamado "cliente delgado"), en comparación con un cliente pesado, se refiere a una aplicación a la que se puede acceder por una interfaz Web (en HTML), que se puede visualizar con un navegador Web en donde toda la lógica comercial se realiza en el lado del servidor. Por eso, al navegador a veces se le denomina cliente universal (25).

Características

- Dispone de un editor de texto con resaltado de sintaxis donde puedes ver el contenido del fichero en el que estás trabajando.
- Contiene una lista de tareas y otros módulos similares.
- La compilación es en tiempo real.
- Tiene pruebas unitarias con JUnit[5].
- Integración con Ant[6], asistentes para creación de proyectos, clases, pruebas y refactorización.

Ventajas

- El entorno de desarrollo integrado (IDE) de Eclipse emplea módulos (en inglés *plugin*) para proporcionar toda su funcionalidad al frente de la plataforma utilizada, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no.
- Este mecanismo de módulos es una plataforma ligera para componentes de software.
- La arquitectura plugin permite escribir cualquier extensión deseada en el ambiente. Se provee soporte para Java y CVS[7] en el SDK de Eclipse. No tiene por qué ser usado únicamente para soportar otros lenguajes de programación, sino que puede familiarizarse con otros entornos y plataformas (26).

[5] JUnit: entorno que permite ejecutar pruebas de clases Java.

[6] Ant: herramienta usada en programación para la realización de tareas mecánicas y repetitivas.

[7] CVS: Sistemas de versiones concurrentes.

La elección del IDE Eclipse fue basada en las ventajas que brinda y su compatibilidad con el sistema operativo a trabajar. Esta es dada por la adquisición del SDK y las herramientas para el desarrollo de Android (ADT por sus siglas en inglés *Android Development Tools*) como plugin para el mismo, diseñada para darle un ambiente potente e integrado en la construcción de aplicaciones de Android.

1.8. Gestor de base de datos

Un Sistema Gestor de Base de Datos SGBD (DBMS por sus siglas en inglés *DataBase Management System*) es una colección de programas cuyo principal objetivo es servir de interfaz entre las bases de datos, el usuario y las aplicaciones. La elección de datos necesarios para el almacenamiento y búsquedas, ya sea de forma interactiva o a través de un lenguaje de programación forman además parte de su objetividad. Este permite definir los datos a distintos niveles de abstracción y manipularlos, garantizando la seguridad y la integridad de los mismos. (27).

Las principales características de un SGBD son:

- Abstracción de la información.
- Independencia.
- Redundancia mínima.
- Consistencia.
- Seguridad.
- Integridad.
- Respaldo y recuperación.
- Control de la concurrencia.

1.8.1. SQLite 3.0

Sistema completo de bases de datos que soporta múltiples usabilidades, con la característica de ser ágil pero robusto. Es un sistema gestor de bases de datos relacional compatible con Atomicidad, Consistencia, Aislamiento y Durabilidad ACID (por sus siglas en inglés *Atomicity, Consistency, Isolation and Durability*), comprendida en una biblioteca relativamente pequeña. Es una librería escrita en C que implementa un motor de bases de datos para SQL92.

Características

Este gestor no es más que un sistema que soporta tablas, índices y vistas que no necesita de un servidor para su utilización. Es capaz de escribir y leer directamente sobre ficheros que se encuentran en el disco duro. Es multiplataforma e imparcialmente se puede utilizar archivos en sistemas de 32 y 64 bits.

La base de datos se almacena en un único fichero a diferencia de otros SGBD que hacen uso de varios archivos. SQLite emplea registros de tamaño variable de forma tal que se utiliza el espacio en disco que es realmente necesario en cada momento.

Ventajas

- **Tamaño**: posee una pequeña memoria y una única biblioteca necesaria para acceder a bases de datos, lo que lo hace ideal para aplicaciones de bases de datos incorporadas.
- **Rendimiento de base de datos:** SQLite realiza operaciones de manera eficiente y es más rápido que MySQL y PostgreSQL.
- **Portabilidad:** se ejecuta en muchas plataformas y sus bases de datos pueden ser fácilmente portadas sin ninguna configuración o administración.
- **SQL:** implementa un gran subconjunto de la ANSI[8] – 92 SQL estándar, incluyendo sub-consultas, generación de usuarios y vistas (28).

Conclusiones del capítulo 1

En la realización de este capítulo se realizó un estudio de los principales aspectos relacionados con el ambiente de desarrollo. Con el objetivo de crear una herramienta que solucione los problemas planteados se hará uso del sistema operativo Android 2.3.3 como base para la implementación de la solución. Para guiar el proceso de desarrollo del software se seleccionó la metodología OpenUP y como herramienta para el modelado Visual Paradigm 8.0 empleando el lenguaje de modelado UML. Durante la implementación de la solución se hará uso del IDE Eclipse en su versión 3.7 con sus herramientas de integración, utilizando como lenguajes de programación Java. Además para la gestión de los elementos locales se utilizará como SGBD SQLite 3.0.

[8] ANSI: Instituto Nacional Estadounidense de Estándares (por sus siglas en inglés: American National Standards Institute)

CAPÍTULO 2: ANÁLISIS Y DISEÑO 2

Introducción al capítulo 2

En el capítulo correspondiente se describen todos los elementos implicados en el análisis y diseño de la herramienta. Incluye los artefactos correspondientes al flujo de trabajo de la metodología seleccionada para el desarrollo. Se puntualizará el modelo de dominio para representar la estructura estática de la solución, así como la relación entre las diferentes clases, atributos y vistas apoyadas en los diagramas de clases del diseño. Se especifican los requisitos, tanto funcionales como no funcionales, dando paso posteriormente a los diagramas de secuencia realizados a partir de la descripción de los casos de uso arquitectónicamente significativos así como los patrones de diseño utilizados.

2.1. Descripción del sistema

La herramienta a desarrollar es un visor que permite la visualización de los reportes del GDR a los que un usuario tiene determinados permisos. Esto se llevará a cabo a través de un dispositivo móvil con sistema operativo Android. Dicha herramienta permitirá administrar los reportes del usuario, sin poseer conexión directa con el sistema GDR, posibilitándole detallar sus reportes, determinados por el mismo a almacenar. Visualizarlos, filtrarlos y exportarlos a formato PDF, son considerados requisitos fundamentales para el cumplimiento de la problemática y los objetivos planteados.

2.2. Modelo de Dominio

El modelo de dominio es una representación visual estática del entorno real de la herramienta. Tiene como objetivo modelar los términos más relevantes del entorno donde se utilizará la solución, implicando a objetos reales representados mediante clases.

A continuación se presenta el Modelo de Dominio (ver **Figura 3**) de la herramienta visor de reportes para móviles y la descripción de las clases del dominio que intervienen en dicho modelo.

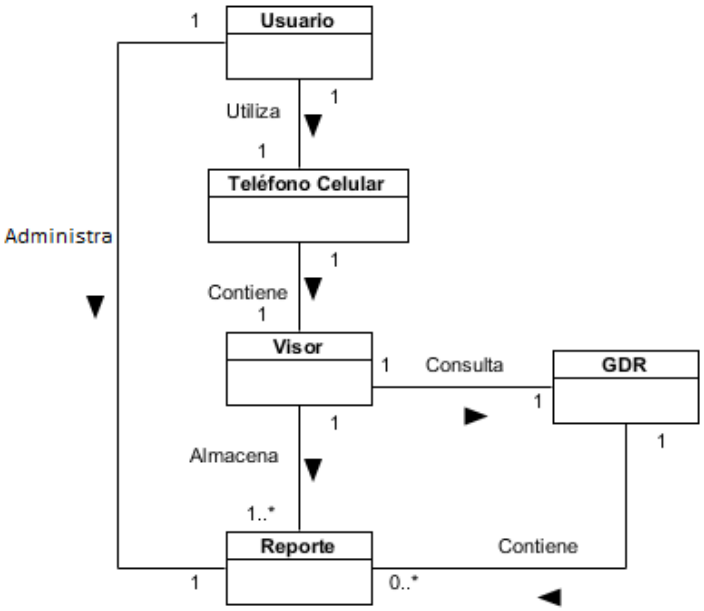


Figura 3. Modelo de Dominio

Descripción de las clases del dominio

Usuario: usuario propietario del teléfono celular que tiene los permisos necesarios para consultar sus reportes.

Teléfono Celular: dispositivo móvil que posee la herramienta visor de reportes.

Visor: herramienta para la visualización de los reportes para móviles con sistema operativo Android.

GDR: sistema que contiene los reportes, los cuales van a ser consultado y llevados al teléfono.

Reportes: representación física de los reportes del usuario. Estos serán tratados de la manera que el usuario determine, desde el dispositivo móvil.

2.3. Especificación de los requisitos del sistema

Los requisitos de un sistema describen los servicios que ha de ofrecer el sistema y las restricciones asociadas a su funcionamiento. Los requisitos no son más que propiedades o restricciones determinadas de forma precisa que deben satisfacerse. Se pueden clasificar en funcionales y no funcionales (29).

2.3.1. Requisitos funcionales

Los requisitos funcionales definen el comportamiento interno de un software, son condiciones que el sistema ha de cumplir. Estos muestran las funcionalidades que deben satisfacerse para cumplir con las especificaciones de software. Para ello se definen los siguientes requisitos funcionales:

RF 1 Configurar Conexión

Descripción: registra en el sistema los datos necesarios para la conexión y obtención de los reportes brindados por los servicios de GDR una vez validados por el sistema.

Entrada: usuario, contraseña y dirección de los servicios.

Salida: almacenamiento de los datos de configuración.

RF 2 Cargar reportes del sistema

Descripción: carga los reportes existentes en el servidor permitiéndole al usuario elegir de una lista cuál de ellos desea almacenar en el dispositivo. Esta lista de reportes posee la característica que tiene seleccionado los reportes existentes en el dispositivo móvil.

Entrada: conexión al GDR a partir de los datos introducidos en la configuración de la conexión.

Salida: listado de reportes seleccionados por el usuario.

RF 3 Descargar reportes

Descripción: brinda la posibilidad de descargar los reportes que el usuario determine a la base de datos SQLite del teléfono celular. Esto lo realiza solo para los reportes permitidos por la herramienta, informándole al usuario si encuentra en su selección reportes inválidos. Previo a esto, elimina los reportes existentes en la base de datos local.

Entrada: listado de reportes a insertar elegidos por el usuario.

Salida: reportes almacenados.

RF 4 Listar Reportes

Descripción: el sistema debe brindar la posibilidad de listar los reportes existentes en el dispositivo móvil.

Entrada: reportes almacenados.

Salida: lista de reportes persistentes.

RF 5 Filtrar lista de reportes

Descripción: garantiza que el usuario pueda filtrar los elementos de la lista de reportes, a visualizar solo los que desee el mismo.

Entrada: lista de reportes seleccionados.

Salida: actualización visual de la lista de reportes.

RF 6 Buscar reporte

Descripción: permite una búsqueda por los elementos del nombre de un reporte con el objetivo de buscar todas las coincidencias y actualizar el listado visual.

Entrada: criterio de búsqueda por el nombre del usuario.

Salida: actualización visual de la lista de reportes.

RF 7 Visualizar los reportes

Descripción: el sistema debe brindar la posibilidad de acceder a los reportes detalladamente.

Entrada: reporte seleccionado.

Salida: visualización de los datos del reporte.

RF 8 Filtrar datos de reporte

Descripción: permite filtrar el contenido de un reporte por sus columnas y descripción de sus elementos.

Entrada: nombre del contenido a filtrar y columnas a visualizar.

Salida: actualización de la vista de los reportes, filtrando sus campos.

RF 9 Exportar Reporte

Descripción: el sistema debe brindar la posibilidad de exportar el reporte seleccionado ya sea filtrado o no, a formato PDF**.**

Entrada: reporte a exportar.

Salida: almacenamiento del reporte en formato PDF en la memoria interna del dispositivo.

2.3.2. Requisitos no funcionales

Los requisitos no funcionales son propiedades y características que hacen a la herramienta atractiva, usable y confiable. Se refieren a todos los requisitos que ni describen información a guardar, ni funciones a realizar. Para la solución propuesta se definen los siguientes requisitos no funcionales:

Usabilidad

- La herramienta debe ajustarse al entorno donde será aplicada y utilizar el castellano como lenguaje base. Debe ser capaz de aprovechar la pequeña infraestructura de red inalámbrica creada por la institución para su uso. La sencillez y claridad de sus acciones debe permitir su utilización por cualquier usuario con conocimientos básicos en la tecnología móvil, brindándole la capacidad de trabajar de manera eficaz en la herramienta.

Confiabilidad

- La veracidad de la información adquirida (reportes) ya sea descargados o extraídos de la base de datos local, deben corresponder sin ninguna alteración con los elementos obtenidos de la base de datos de GDR.

Eficiencia

- La herramienta debe mantener tiempos de respuestas razonables a la hora de procesar los reportes. Para descargar un reporte, no más de un minuto; así como un tiempo límite de tres segundos para la

visualización de los reportes. Para los restantes procesos deberá mantener un tiempo inferior al segundo. Todo dependiendo de la dimensión del reporte y los recursos del dispositivo.

Restricciones de Diseño e Implementación

- Sistema operativo Windows o Linux.
- El sistema deberá ser implementado en el lenguaje de programación Java.
- Se utilizará el marco de trabajo de desarrollo Android en su versión 2.3.3.
- Se empleará la herramienta de desarrollo Eclipse 3.7 y el sistema gestor de base de datos SQLite 3.0.

Software

Requisitos mínimos para el dispositivo móvil:

- Soporte para conexiones WIFI.
- Sistema operativo Android 2.3.3.

Hardware

Requisitos mínimos del teléfono móvil:

- Capacidad disponible de 6Mb en la memoria del teléfono para la instalación y ejecución de la herramienta.

2.4. Modelo de Caso de Uso del Sistema

El modelo de caso de uso describe las funcionalidades que se proponen en la nueva herramienta. Este permite a través de los diferentes casos de uso, la relación existente entre un usuario o máquina y la herramienta (30).

2.4.1. Definición de los actores del sistema

Los actores no son más que usuarios o elementos externos del sistema que interactúan con la herramienta. Estos no tienen la obligación de ser humanos, también se pueden entender por sistemas informáticos, empresas y otros. Los actores utilizan los casos de usos para desarrollar partes del trabajo que poseen valor para el negocio. Aquella cantidad de casos de usos con los que tiene relación un actor define su rol global en la herramienta y al alcance de su acción.

Se identificó como actor para la herramienta propuesta:

Actor	Descripción
Propietario	Usuario del teléfono móvil que posee los privilegios de almacenar en su dispositivo los reportes que determine, teniendo en cuenta los reportes a los cuales tiene acceso en el GDR. Es responsable de configurar la conexión al GDR, permitiéndole el trabajo con sus reportes, administrarlos así como manejarlos individualmente.

Tabla 3. Descripción de los actores del sistema

2.4.2. Diagrama de Caso de Uso del Sistema (DCUS)

Para documentar el comportamiento y las funcionalidades de un software es utilizado el siguiente diagrama de caso de uso del sistema (ver **Figura 4**). Este enmarca los requisitos funcionales de la herramienta, representando las funciones que la herramienta puede efectuar.

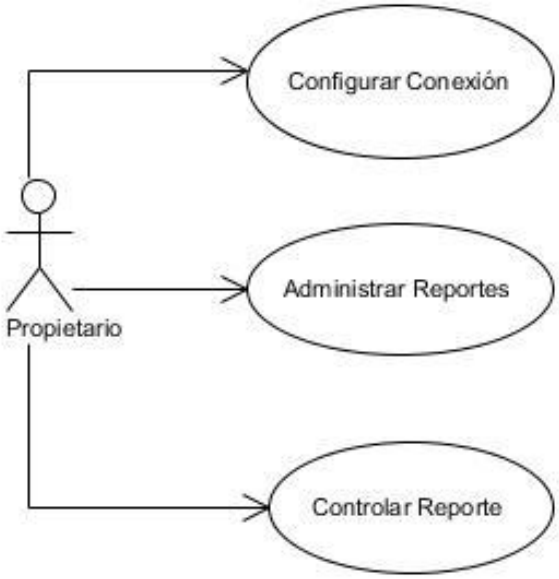


Figura 4. Diagrama de casos de uso del sistema

En este DCUS, el actor Propietario es el responsable de inicializar el caso de uso ***Administrar Reportes***, considerado crítico por el impacto que proporciona en la operatividad.

2.5. Descripción textual del caso de uso crítico del sistema

AdministrarReportes

Caso de Uso:	Administrar Reportes
Actor:	Propietario
Resumen:	El caso de uso se inicia cuando el actor Propietario ejecuta la herramienta en el dispositivo celular. Esta se conecta con el GDR en caso de existir una conexión directa o con la base de datos local en caso contrario y muestra o ejecuta las posibles acciones sobre los reportes del usuario.
Precondiciones:	Deben existir datos de conexión almacenados y reportes en el sistema GDR así como reportes almacenados en caso contrario.
Referencias	RF2, RF3, RF4, RF5, RF6
Prioridad	Crítico
Flujo Normal de Eventos	

Sección "Administrar Reportes"		
	Acción del Actor	**Respuesta del Sistema**
1.	Inicia la herramienta desde la vista principal.	
2.		El sistema muestra la actividad **ListReports** y realiza las operaciones correspondientes para los siguientes contextos: • Establecer Conexión. Ver Sección 1: "Establecer Conexión". • Cargar reportes del sistema GDR. Ver Sección 2: "Cargar reportes del sistema" • Descargar reportes. Ver Sección 3: "Descargar reportes" • Listar reportes. Ver Sección 4: "Listar reportes". • Filtrar la lista de reportes almacenados. Ver Sección 5. "Filtrar lista de reportes" • Buscar reporte. Ver Sección 6. "Buscar Reporte"
Flujo Normal de Eventos		
Sección 1 "Establecer Conexión"		
Acción del Actor		**Respuesta del Sistema**
1.		Extrae los datos de conexión de la base de datos y comprueba que la conexión puede ser establecida.
2		Muestra un mensaje "Se ha establecido la conexión". En caso contrario ver flujo alterno 1a.
3		Carga los reportes existentes en el servidor GDR ver Sección 2: "Cargar reportes del sistema".
Prototipo de Interfaz		

Se ha establecido la conexión
Flujos Alternos
1a. La conexión no puede ser establecida.

	Acción del Actor	Respuesta del Sistema
1.		Muestra un mensaje "Imposible establecer la conexión".
2.		Lista los reportes que se encuentran almacenados de manera local. Ver sección 3: "Listar reportes almacenados".

Prototipo de Interfaz
Imposible establecer la conexión

Sección 2 "Cargar reportes del sistema"		
	Acción del Actor	**Respuesta del Sistema**
1.		Establece la conexión con el GDR y extrae el listado de los reportes a los que el usuario tiene permisos a través de una respuesta json[9].
2.		Muestra una ventana con el listado de los reportes, seleccionando además los existentes en el dispositivo.
3.	Selecciona los reportes de interés a descargar.	
4.	Selecciona la opción Aceptar. En caso de seleccionar Cancelar, ver flujo alterno 4a.	

[9] JSON: acrónimo de *JavaScript Object Notation*. Es un formato alternativo de envío y recepción de datos.

5.		Captura los reportes seleccionados por el propietario.
6.		Procede a descargar los reportes elegidos. Ver Sección 3: “Descargar reportes”.

Flujos Alternos

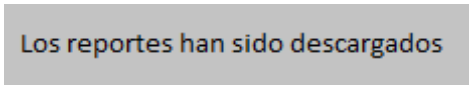
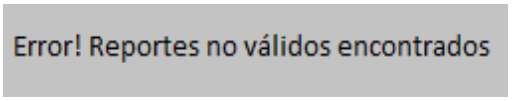
4a. Descarga cancelada.

	Acción del Actor	Respuesta del Sistema
1.		Extrae y lista los reportes persistentes en el dispositivo ver sección: 4 “Listar reportes”.

Prototipo de Interfaz

Sección 3 “Descargar reportes”

	Acción del Actor	Respuesta del Sistema
1.		Captura los reportes seleccionados por el propietario.
2.		Establece la conexión con la base de datos local.
3.		Elimina las tablas dinámicas creadas por cada reporte.
4.		Elimina de la tabla de reportes, todos sus valores.
5.		Inserta los nuevos reportes en la tabla reportes de la base de datos local.

6.		Establece la conexión con el GDR y extrae los valores reales de cada reporte seleccionado. Captura un código html[10] como repuesta del sistema.
7.		Interpreta el código para identificar si el reporte es válido y extraer sus valores.
8.		Crea una tabla dinámica para cada reporte válido.
9.		Inserta los valores de cada reporte válido en sus tablas correspondientes.
10.		Lista los reportes almacenados. Ver Sección 3: "Listar reportes almacenados".
11.		Muestra un mensaje de confirmación "Los reportes han sido descargados" si no encontró reportes inválidos en el proceso de descarga. En caso contrario ver flujo alterno 11a.
Prototipo de Interfaz 		
Flujos Alternos		
11a. Se encontraron reportes inválidos en el proceso de descarga		
	Acción del Actor	**Respuesta del Sistema**
1.		El sistema muestra un mensaje "Error! Reportes no válidos encontrados"
Prototipo de Interfaz 		
Sección 4 "Listar reportes"		
	Acción del Actor	**Respuesta del Sistema**

[10] HTML: es un lenguaje que se utiliza fundamentalmente en el desarrollo de páginas web.

1.		El sistema extrae de la base de datos local los reportes existentes.
2.		Lista los reportes.

Prototipo de Interfaz

Flujos Alternos

1a. No existen datos almacenados.

	Acción del Actor	**Respuesta del Sistema**
1.		Muestra un mensaje "No existen reportes. La aplicación será cerrada."
2.	Selecciona la opción "Aceptar".	
3.		Cierra la aplicación.

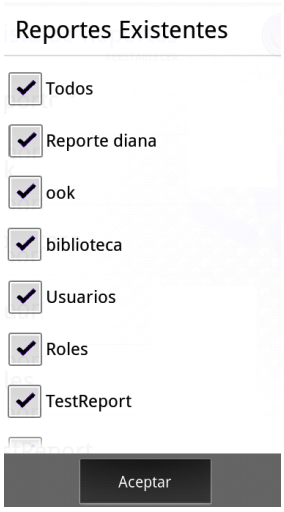
Prototipo de Interfaz

Sección 5 "Filtrar lista de reportes"

	Acción del Actor	**Respuesta del Sistema**
1.	Selecciona del menú la opción Filtrar.	

2.		Genera y muestra una ventana con los reportes visibles.
3.	Selecciona los reportes que desea visualizar de los mostrados.	
4.	Presiona el botón Aceptar.	
5.		Captura los atributos seleccionados y genera una nueva lista con los reportes a visualizar.
6.		Muestra la lista de reportes filtrada.

Prototipo de Interfaz



Sección 6 "Buscar Reportes"

	Acción del Actor	**Respuesta del Sistema**
1.	Inserta el criterio de búsqueda de coincidencias por nombre.	
2.		Captura los caracteres insertados y genera una nueva lista con los reportes que contengan dicho criterio.

Prototipo de Interfaz

	
Pos-condiciones	Existencia en el teléfono móvil de los reportes necesarios para el propietario, con la posibilidad de listarlos y filtrarlos.

Tabla 4. Descripción del caso de uso Administrar Reportes

2.6. Modelo del Diseño

Para un mejor entendimiento lógico y detallado de los requisitos funcionales, se selecciona como entrada indispensable en la fase de diseño, los modelos de diseño. Estos no son más que refinamientos del análisis ocupados del cómo serán implantados los requisitos y restricciones que se le suponen. Sus principales objetivos se basan en transformar los requisitos en un diseño del sistema, evolucionar hacia una arquitectura sólida y adaptar el diseño para que se ajuste al entorno de implementación del cual depende así como del lenguaje de programación.

2.6.1. Diagrama de clases del diseño

Aquellas funcionalidades y características de cada clase en el lenguaje de desarrollo seleccionado, son representadas a través de los diagrama de clases del diseño (DCD). Estos proporcionan el objetivo y trabajo de cada clase usada en la herramienta. El siguiente diagrama de clase del diseño (ver **Figura 5**) contiene las principales clases con sus respectivos métodos y atributos para el caso de uso Administrar Reportes.

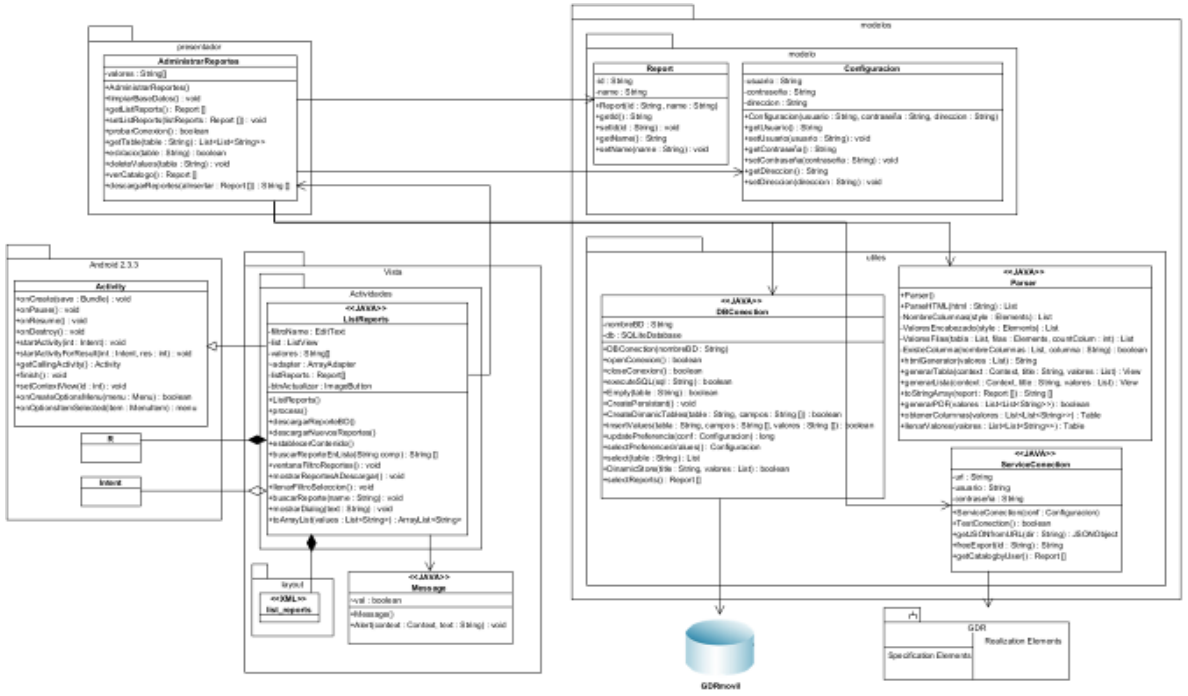


Figura 5. Diagrama de clases de diseño para el caso de uso Administrar Reportes

En este diagrama los elementos del modelo corresponden a las clases implicadas en el caso de uso Administrar Reportes, considerado significativo para el sistema.

Android basa su desempeño en el uso de clases Actividades que poseen la responsabilidad de controlar aquellos formularios generados en estructura XML. En el caso particular de Administrar Reportes se cuenta con la actividad **ListReport,** responsable de la captura de los componentes de la interfaz **list_reports.xml.** Esta actividad posee una instancia de la clase **AdministrarReportes,** encargada de la gestión de todo los elementos implicados en la administración de reportes, conexión a la base de datos con la utilización de la clase **DBConection**, a los servicios del sistema GDR con la clase **ServiceConection** e interpretación de los elementos extraídos del sistema, apoyándose en la clase **Parser**.

A continuación en la siguiente tabla se verá reflejada la descripción de las clases del diagrama de clases de diseño para el caso de uso Administrar Reportes.

Clases	Descripción
Activity	Es una parte importante del ciclo de vida total de una aplicación y del modelo de aplicación de la plataforma por la manera en que son lanzadas a procesar y trabajar en conjunto. Una actividad es una clase enfocada a lo que el usuario puede hacer. Poseen una interacción directa con el usuario y son las encargadas de crear las ventanas necesarias para el correcto uso de usabilidad en la elaboración de esta herramienta.
Intent	Descripción abstracta de una operación a ejecutar. Puede ser usada a través de la función startActivity() con el objetivo de lanzar una aplicación, al igual que servicios a comunicar con la herramienta. Provee facilidades para lograr el uso de diferentes códigos en diferentes actividades o aplicaciones. Su uso más significativo es para el tránsito entre actividades.
R	Clase controladora de identificadores de Android que se encarga de poseer una referencia de todos los componentes, acciones, visuales, actividades, entre otros. Todos los elementos que son creados en un proyecto Android almacenan sus identificadores en esta clase con el objetivo de que todas las restantes actividades tengan estas referencias.
ListReports	Actividad encargada de la captura y muestreo de los datos de manera visual. Recibe los componentes generados por el XML correspondiente a esta actividad.
list_report.xml	Fichero XML que cuenta con los componentes visuales necesarios para su actividad correspondiente para el desarrollo e interacción del usuario con la herramienta.
Message	Clase de utilidad que contiene la sintaxis del mensaje mostrados desde las actividades.

AdministrarReportes	Clase presentadora encargada de la gestión de la información referente a este caso de uso. Cuenta con el listado de los reportes existentes, así como los objetos correspondientes para la conexión tanto a la base de datos como a los servicios que brinda GDR.
DBConection	Se encarga de la conexión a la base de datos. Cuenta con elementos para la realización de todas las posibles consultas a la base de datos local SQLite.
ServiceConection	Conexión a servicios. Encargada de la relación entre la herramienta y la captura de los elementos brindados por los servicios generados por el GDR.
Parser	Es la responsable del ajuste de la información obtenida luego de la captura de los servicios, ya sea JSON, o HTML, así como la generación de la tabla de los reportes con los elementos interpretados.
Report	Clase entidad que refleja el contenido básico de un reporte.
Configuracion	Clase entidad que posee los elementos de configuración para el acceso a los servicios.
GDRmovil	Base de datos local, que almacena persistentemente los valores necesarios para la herramienta.
GDR	Subsistema externo que brinda los elementos necesarios de los reportes a los que el usuario tiene permiso.

Tabla 5. Descripción de las clases del diagrama de clases del diseño

2.7. Patrones utilizados en la solución

Un patrón describe un problema que ocurre varias veces así como el núcleo de la solución al inconveniente, de forma que puede utilizarse en ilimitadas ocasiones sin tener que hacer dos veces lo mismo (31).

2.7.1. Patrones de diseño GRASP

En la solución del sistema se aplicaron principalmente los siguientes patrones de asignación de responsabilidades (GRASP) del diseño:

Controlador: es utilizado por todas las clases implicadas en la capa de presentación de la lógica del negocio. Poseen la responsabilidad de controlar el flujo de eventos mediante las actividades correspondientes. En la arquitectura de la herramienta se definieron tres clases presentadoras cada una encargada de controlar la lógica del negocio en lo que respecta a su caso de uso correspondiente. (Ver **Figura 5**).

Bajo acoplamiento: este patrón es utilizado en la creación de clases independientes para la lógica de los diferentes tipos clases; actividades y entidades. Esto trae como ventaja que solo se realicen acciones sobre el tipo de entidad que se solicite o formulario correspondiente y no sobre todo el conjunto. Estas clases (entidades) se encargan de la representación de los elementos reales de cada uno respectivamente y las actividades, de manejar los componentes visuales de cada funcionalidad dentro de los diferentes casos de uso.

Alta cohesión: caracteriza las clases con responsabilidades similares. Cada elemento del diseño debe realizar una labor única dentro del sistema como es el caso de la clase de presentación Administrar Reporte cuya responsabilidad es la de administrar los reportes del usuario y no desempeña responsabilidades externa a esta, así mismo a las clases utilidades. (Ver **Figura 5**).

2.7.2. Patrones Arquitectónicos

Es un conjunto de patrones que proporcionan un marco de referencia necesario para guiar la construcción de un software, permitiendo a los programadores, analistas y todo el conjunto de desarrolladores del software compartir una misma línea de trabajo y cubrir todos los objetivos y restricciones de la aplicación. Es considerada el nivel más alto en el diseño de la arquitectura de un sistema puesto que establecen la estructura, funcionamiento e interacción entre las partes del software (32).

A continuación se presenta el modelo de arquitectura uniforme para un proyecto Android basada en el Modelo Vista Presentador (MVP por sus siglas en inglés *Model View Presenter*) familia del patrón conocido como Modelo Vista Controladora (MVC por sus siglas en inglés *Model View Controller*) (ver **Figura 6**).

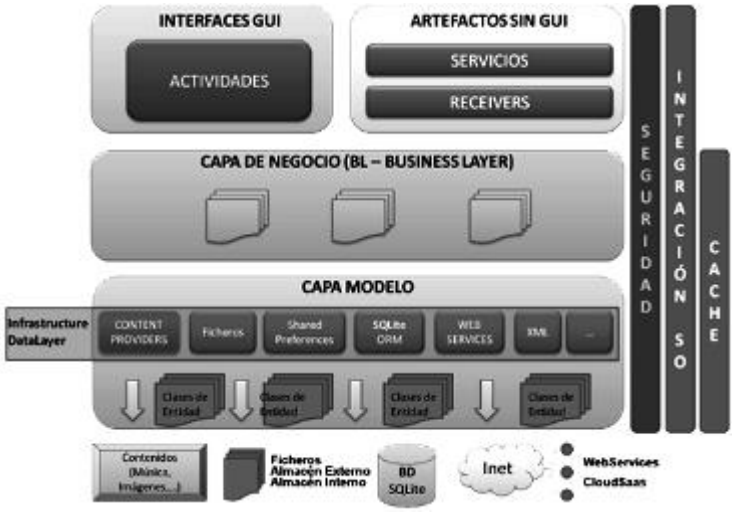


Figura 6. Arquitectura basada en MVP para un proyecto Android (33)

En la imagen se pueden ver la siguiente separación de capas basada en MVP:

- **Capa Vista:** Incluye las interfaces de usuario (Actividades, recursos, layouts, entre otros), y aquellos artefactos de Android sin interfaz gráfica (utilidades visuales). Las clases actividades contenidas en esta capa son las responsable de la gestión de los componentes visuales, ListReports, PreferenceConfig y ViewReports. En los formularios (layouts), se incluyen los archivo XML list_reports, main y preference_config y como clase de utilidad visual la clase Message.

- **Capa Presentador o Capa de Negocio:** Incluye las clases del negocio, que hacen de puente entre la vista y el modelo, incluyendo clases con métodos que alberguen la lógica de la herramienta. Siempre los artefactos de la capa Vista invocarán a las clases albergadas en esta capa, nunca accederán al modelo directamente. Las clases presentadoras contenidas en esta capa de negocio son las que albergan el control y gestión sobre todo el contenido referente a cada caso de uso, AdministrarReportes, ConfigurarConexion y ControlarReporte.

- **Capa Modelo:** Dos partes importantes:

  Clases de Entidad: incluyen la definición lógica del modelo de datos a utilizar. Estas clases serán manejadas por la capa de negocio. Contiene las clases Report y Configuracion,

encargadas de controlar los elementos de los reportes y datos de configuración respectivamente.

Capa de Infraestructura: Abstrae la complejidad tecnológica de la gestión del almacenamiento físico. En el caso de la herramienta propuesta, contiene la base de datos local SQLite GDRmovil.

- En la base de la pila de capas se encuentran las fuentes físicas de información: Ficheros en SD / RAM interna, Bases de datos, Contenidos gestionados por otras aplicaciones, Servicios Web, entre otros (33).

2.7.3. Patrones de diseño GOF

Los patrones de diseño GOF, son clasificados según su propósito en creacionales, estructurales y de composición, mientras que respecto a su ámbito se clasifican en clases y objetos (34).

- Según su propósito:

1. **De Creación**: Resuelven problemas relativos a la creación de objetos.

2. **Estructurales**: Resuelven problemas relativos a la composición de objetos.

3. **De Comportamiento**: Resuelven problemas relativos a la interacción entre objetos.

- Según su ámbito:

1. **Clases**: Relaciones estáticas entre clases.

2. **Objetos**: Relaciones dinámicas entre objetos.

Entre los existentes se utilizó en la solución del problema el patrón Singleton el cual se puede emplear en proyectos simples para contener todo el código bajo una única variable global. En grandes proyectos, puede ser usado para agrupar código relacionado para simplificar la permanencia del mismo y localizarlo en un único lugar, facilitando la modularidad. Incluso en los proyectos de mayor envergadura, Singleton ayuda a optimizar la aplicación, facilitando la carga asíncrona de partes del código que no sean usadas

frecuentemente (35). Este patrón es utilizado en las clases DBConection, Message y Parser para evitar varios intanciamientos en las diferentes clases presentadoras y de actividades.

2.8. Diagrama de paquetes

Los diagramas de paquetes son utilizados para reflejar la organización de paquetes y sus componentes. Cuando se usan para representaciones, los diagramas de paquete se aprovechan para proveer una visualización de estructura física de la herramienta en desarrollo. Los usos más comunes para los diagramas de paquete son para organizar diagramas de casos de uso y diagramas de clase (36). A continuación el diagrama de paquete correspondiente al sistema (ver **Figura 7**):

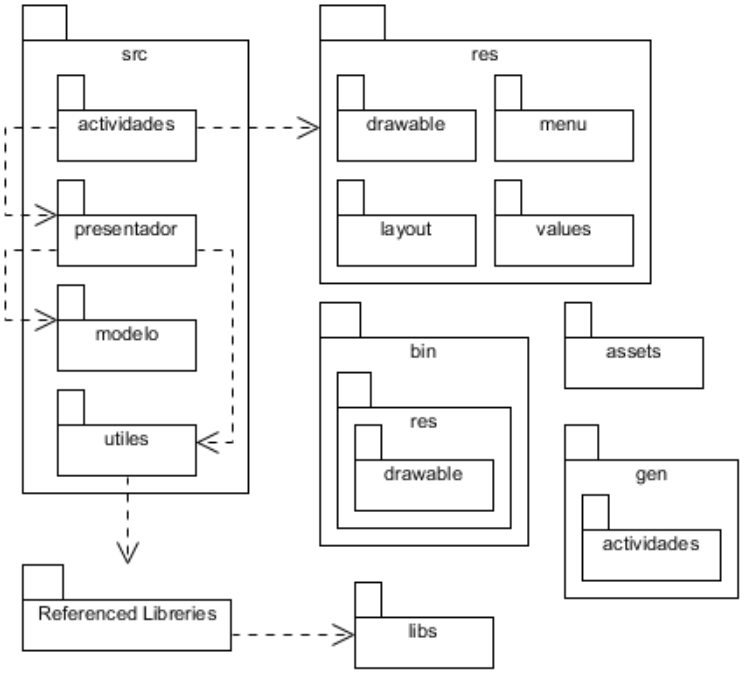


Figura 7. Diagrama de paquetes de la herramienta

Los proyectos Android eventualmente van construyendo en los archivos .apk todo el código fuente, recursos y elementos necesarios para el funcionamiento del software. Estos archivos .apk son los ejecutables de instalación de las aplicaciones. Algunos de estos ficheros .apk son generados por el desarrollador por defecto, mientras que otros solo en caso de ser necesario. La estructura de paquetes y archivos de un proyecto de este tipo es la siguiente:

src/: Contiene los archivos de clases y actividades. Estos son almacenados en la dirección:

src/your/package/namespace/ActivityName.Java.

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

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

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

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

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

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

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

2.9. Diagrama de interacción del diseño

La comunicación existente entre el actor y las diferentes clases, es representada a través de los diagramas de secuencia, los cuales responden a un determinado evento o escenario de un caso de uso particular. Guía todo el proceso lógico de funcionamiento de estos eventos modelando así aspectos dinámicos del sistema.

En el presente trabajo se utilizan diagramas de secuencia, para organizar los eventos con la sucesión temporal en que se desencadenan. A continuación se representa el escenario “Descargar nuevos reportes” (ver **Figura 8**) perteneciente al caso de uso Administrar Reportes.

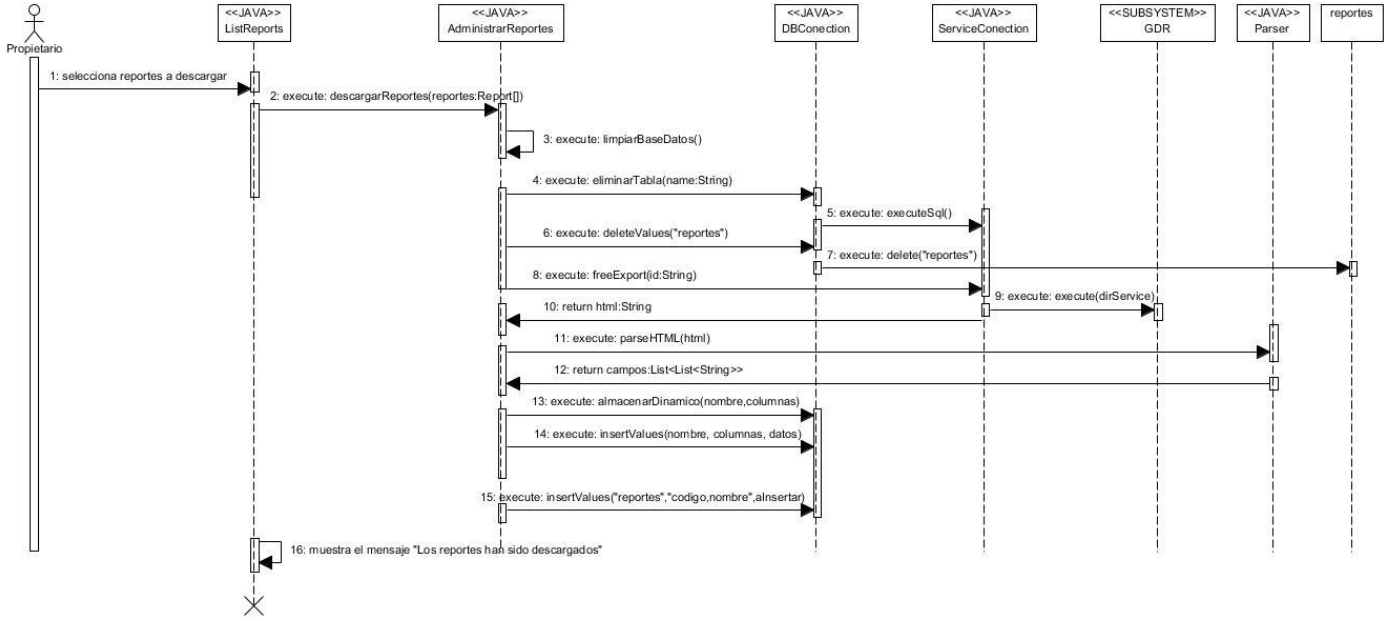


Figura 8. Diagrama de secuencia del escenario Descargar reportes

Este escenario es inicializado una vez que el usuario selecciona los reportes que desee descargar del sistema. La herramienta captura los reportes seleccionados por el usuario y establece la conexión con los servicios para obtener sus valores reales. Antes de este proceso, limpia todos los reportes existentes en la base de datos, incluyendo los nuevos posteriormente y listándolos al finalizar la descarga. La eliminación de los reportes persistentes así como la inclusión de los nuevos elementos es realizada a través de la clase **DBConection**, responsable de la conexión con la base de datos SQLite **GDRmovil**. La captura de los valores de los reportes es realizada con el uso de la clase **ServiceConection** que posee todas las funciones necesarias para la captura de los servicios que brinda el GDR, en este case en particular freeExport para la captura del html correspondiente de cada reporte con sus valores reales. Este html es interpretado a través de la clase **Parser**. Disponiendo de todos estos elementos, la herramienta crea de manera dinámica una tabla para cada reporte que contendrá los elementos reales del mismo, los que una vez almacenados de manera persistente, son listados. Al finalizar, comunica al usuario de la descarga de los reportes.

2.10. Modelo de datos

Un modelo de datos es una estructura abstracta que documenta y organiza la información para la comunicación. En la informática, se centra en el planeamiento del desarrollo de aplicaciones y la decisión de cómo se almacenarán los datos y cómo se accederá a ellos (37). El modelo relacional se caracteriza a muy grandes rasgos por disponer que toda la información debe estar contenida en tablas, y las relaciones entre datos deben ser representadas explícitamente en esos mismos datos (ver **Figura 9**).

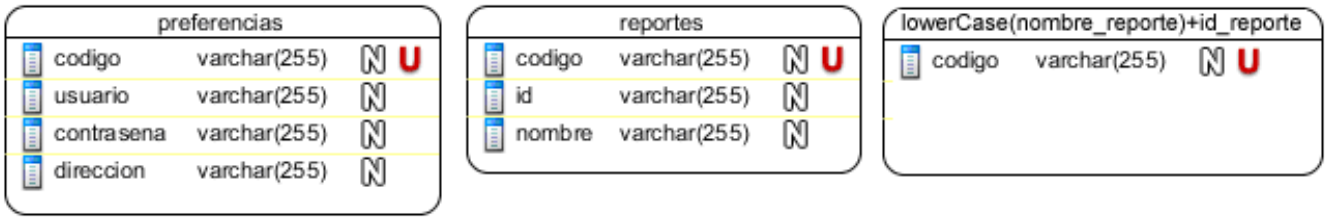


Figura 9. Modelo de Datos

A continuación se muestra la descripción de las tablas del modelo de datos representado:

Nombre	Descripción
preferencias	Es la tabla responsable del almacenamiento de los valores de configuración que permiten el acceso a los servicios de GDR.
reportes	Encargada de almacenar los reportes del usuario una vez consumido el servicio respectivo.
lowerCase (nombre_reporte)+id_reporte	Tabla encargada de almacenar los valores reales de cada reporte. Esta es creada una vez obtenido el reporte y determinado la cantidad de campos que posee. El nombre de dichas tablas estará caracterizado por el nombre del reporte más su identificador y poseerán una cantidad de campos indefinida según los determinados luego de la interpretación del HTML recibido del reporte.

Tabla 6. Descripción de las Tablas

2.11. Diagrama de Despliegue

El diagrama de despliegue modela la topología del hardware sobre el que se ejecuta un sistema, el cual muestra la configuración de los nodos que participan en la ejecución de los componentes que residen en ellos. Además, representa el despliegue físico de un componente. La misma será desplegada en un teléfono celular con sistema operativo Android que poseerá una base de datos local SQLite para almacenar de manera persistente los datos necesarios, así como una conexión directa con el Visor de Reportes para la obtención de los elementos necesarios para el usuario (reportes). A continuación se muestra el diagrama de despliegue para la herramienta visor de reporte para móviles (ver **Figura 10**).

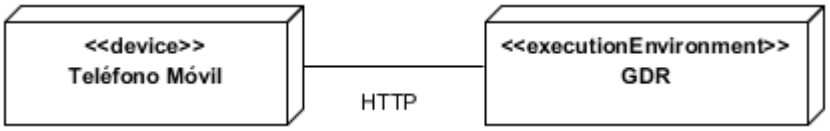


Figura 10. Diagrama de Despliegue

<<device>> Teléfono Móvil: Es el dispositivo móvil el cual va a tener la herramienta visor de reporte.

<<executionEnvironment>> GDR: Sistema que va a tener los reportes, la cual el dispositivo móvil se va a conectar por vía HTTP para obtener dichos reportes.

Conclusiones del capítulo 2

En el capítulo se logró un mejor entendimiento del entorno real de la herramienta a través de la realización del modelo de dominio. Se determinaron las diferentes funcionalidades y cualidades que el sistema debe cumplir mediante la especificación de requisitos. En la confección del diagrama de casos de uso se evidenció la relación entre el actor y los casos de uso. Con el propósito de representar la organización de la herramienta se desarrollaron los diagramas de clases; mostrando la relación entre clases, atributos e interfaces. Evitando problemas en el diseño del software, se aplicaron los patrones correspondientes. Para definir la estructura general se aplicó el patrón arquitectónico Modelo Vista Presentador. Los diagramas de secuencia elaborados permitieron describir gráficamente cada escenario de cada caso de uso. El modelo de datos relacional especificó las clases persistentes de la base de datos y finalmente con la confección del diagrama de despliegue se logró modelar la distribución física de la herramienta.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA 3

Introducción al capítulo 3

En el capítulo correspondiente a la fase de implementación y prueba, se analizarán los artefactos principales, como el modelo de implementación y las pruebas. Se realizará una descripción de como los elementos del modelo de diseño se implementan en términos de componentes, que serán distribuidos por los nodos de configuración y se realizarán los casos de pruebas para validar el correcto funcionamiento de la herramienta.

3.1. Modelo de implementación

El modelo de implementación describe cómo los elementos de diseño se implementan en componentes. Además, describe los componentes a construir y su organización en nodos físicos en los que funcionará el sistema. Está compuesto por los diagramas de despliegue y componentes (38). Es de gran utilidad a la hora de implementar el sistema, pues facilita la organización del producto y lo hace más entendible a los desarrolladores.

3.1.1. Diagrama de componentes

El diagrama de componentes es usado para estructurar el modelo de implementación. Describe los elementos físicos del sistema y sus relaciones; además de mostrar las opciones de realización incluyendo código fuente, binario y ejecutable. Un componente representa un elemento físico que forma parte del sistema, se puede representar por nodos y sus operaciones solo se pueden alcanzar a través de interfaces (39). A continuación se representa el diagrama de componente para el caso de uso Administrar Reportes (ver **Figura 11**):

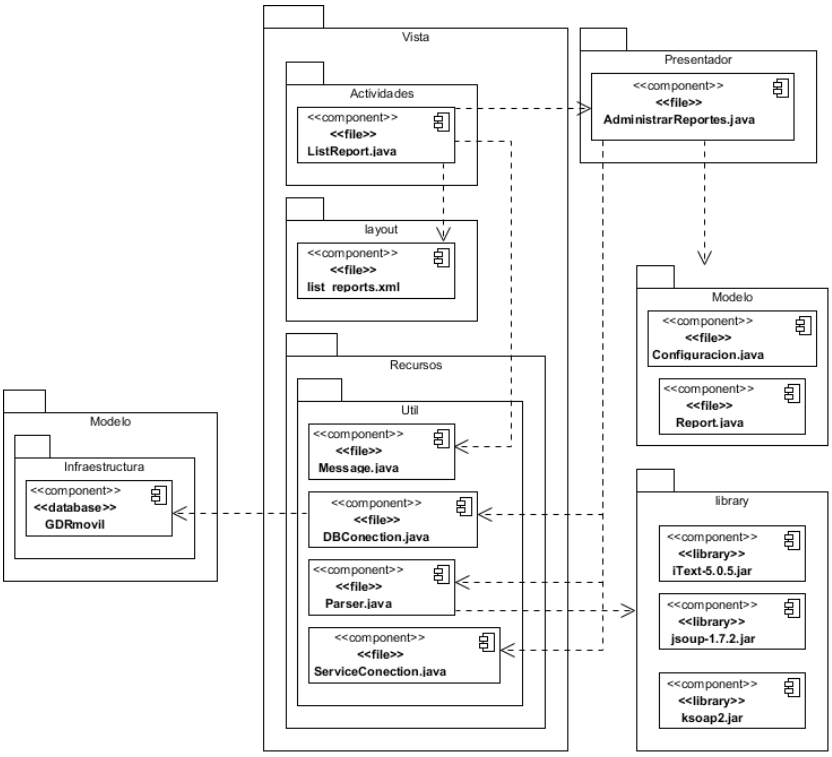


Figura 11. Diagrama de Componentes para el caso de uso Administrar Reportes

En este diagrama se pueden observar los diferentes componentes que intervienen en la realización del CU_Administrar Reportes, el cual contiene en la capa presentador al componente **AdministrarReportes**, encargado de toda la gestión de los reportes en la aplicación cuyos valores son necesitados por las actividades controladoras visuales que engloban todos los elementos necesarios para la generación visual del contenido administrado.

3.2. Implementaciones relevantes

Dentro de las funcionalidades realizadas para el cumplimiento de los requisitos planteados, se determinaron algunas implementaciones relevantes durante el desarrollo de la aplicación. A continuación se detalla la clase presentadora AdministrarReportes (ver **Figura 12**), **descargarReportes ()** ejecutada únicamente si se establece una conexión con los servicios que brinda GDR, se cargan los reportes existentes y el usuario determina la inserción de nuevos reportes en el dispositivo móvil. Esta funcionalidad descarga los valores

reales de los reportes seleccionados por el usuario y los adiciona a la base de datos local en caso de ser reportes válidos para la herramienta.

```
public void descargarReportes(Report[] aInsertar) throws Exception
{   getPreference();

    limpiarBaseDatos();

    listReports = aInsertar;

    int errores=0;

    for (int i = 0; i < listReports.length; i++) {
        Report rep = listReports[i];

        try {
        String html = serv.freeExport(rep.getId() + "");

        List<List<String>> campos = null;

            campos = parser.ParseHTML(html);

        int cant = campos.get(0).size();

        String[] colum = new String[cant];

        for (int j = 0; j < cant; j++) {
            colum[j] = campos.get(0).get(j);
        }

        String name = rep.getName().replaceAll(" ", "");
        name = name.replaceAll(" ", "");
        name += rep.getId();
        name = name.toLowerCase();

        con.CreateDimanicTables(name, colum);

        for (int j = 1; j < campos.size(); j++) {
            List<String> fila = campos.get(j);

            String[] data = new String[colum.length];

            for (int k = 0; k < colum.length; k++) {
                data[k] = fila.get(k);
            }
            con.insertValues(name, colum, data);
        }

        con.insertValues("reportes", new String[] { "codigo", "nombre" },
                new String[] { rep.getId(), rep.getName() });

        } catch (Exception e) {
            errores++;
        }
    }

    if(errores>0)
    {
        throw new Exception();
    }
}
```

Figura 12. Funcionalidad para la descarga de nuevos reportes

3.3. Estándar de codificación

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez. Al comenzar un software, se establece un estándar de codificación para asegurarse de que todos los programadores del proyecto trabajen de forma coordinada. Cuando el proyecto de software incorpore código fuente previo, o bien cuando realice el mantenimiento de un sistema de software creado anteriormente, el estándar de codificación debería establecer cómo operar con la base de código existente (40).

Para el desarrollo de la herramienta se definió los siguientes estándares de codificación los cuales se explica a continuación.

Número de declaraciones por línea

Se debe declarar cada variable en una línea distinta, de esta forma cada variable se puede comentar por separado.

Ejemplo:

```
private IDialog dialog;
private ViewManager manager;
private FactoryController factory;
private Usuario user;
private ConnectionSource con;
private DefaultTableModel modelElem;
private DefaultTableModel modelReq;
```

Espacio en blanco

Se debe usar una línea en blanco entre:

- Métodos.
- Variables locales de un método y la primera sentencia.
- Entre diferentes secciones lógicas dentro de un fichero (más legibilidad).

Ejemplo:

```
public String[] toStringArray(Report[] report)
{
    String[] result=new String[report.length];

    for (int i = 0; i < report.length; i++) {
        result[i]=report[i].getName();
    }

    return result;
}
```

Asignación de nombres

Cada tipo de elemento debe nombrarse con una serie de reglas determinadas.

Clases e interfaces: Nombres. La inicial en mayúscula ya sea simple o compuesta su nombre.

Ejemplo:

```
public class PreferenceConfig extends Activity
public class ServiceConection
```

Métodos: Deben ser verbos. La primera letra de la primera palabra en minúsculas, el resto de las palabras empiezan por mayúsculas.

Ejemplo:

```
public Report[] getCatalogbyUser() throws JSONException
```

Variables: Deben comenzar por minúscula. No se utilizará en ningún caso el carácter "_".

Ejemplo:

```
private String url;
private String usuario;
private String contraseña;
```

3.4. Pruebas

La prueba del software es un elemento crítico para la garantía de la calidad del software. El objetivo de la etapa de pruebas es garantizar la calidad del producto desarrollado. Estas son utilizadas para identificar posibles fallos durante el proceso de desarrollo. Los casos de prueba son actividades en las cuales un sistema o componente es ejecutado bajo condiciones o requerimientos específicos. También son un proceso que se enfoca sobre la lógica interna del software y las funciones externas, es un proceso de ejecución de un programa con la intención de descubrir un error.

3.4.1. Estrategia de pruebas

Una estrategia para pruebas de software integra las técnicas de diseño de casos de prueba en una serie de pasos bien planificados que llevan a la evaluación correcta del software. También proporcionan un mapa que describe los pasos que se darán como parte de la prueba mostrada y cuando se planean y darán dichos pasos. Incluye los niveles de prueba a ser diseccionados, el tipo de prueba a ser ejecutada y los casos de

prueba diseñados para lograr los objetivos. Define consideraciones especiales relacionadas con los recursos necesarios para realizar esta tarea (41).

Nivel de prueba

Los niveles de pruebas verifican y validan un producto de software en diferentes ángulos con la capacidad de determinar no conformidades en relación con el uso que se le vaya a dar a la herramienta. La realización de los casos de prueba a la solución estuvo enfrascada a nivel de desarrollador guiado durante todo el ciclo de desarrollo de la herramienta.

Tipo de prueba

Las pruebas en conjunto tienen como objetivo general verificar y validar un software, independientemente de las características y el entorno donde se desarrollen, además de los recursos y los factores vinculados al proceso de desarrollo (42). Para comprobar las operaciones a realizar por la herramienta, se utilizó el tipo de prueba de Funcionalidad así como de Rendimiento para la verificación de los tiempos de respuestas de la herramienta.

Método de prueba

La prueba de caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. Estas pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene (43).

Técnica de prueba

Dentro del método de Caja Negra la técnica de la Partición de Equivalencia es una de las más efectivas pues permite examinar los valores válidos e inválidos de las entradas existentes en la herramienta, descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico (44).

3.4.2. Diseño de casos de pruebas

Un caso de prueba se diseña según las funcionalidades descritas en los casos de uso. La intención que se persigue con este artefacto es lograr una compresión específica de las condiciones que la solución debe cumplir. Cada planilla de casos de pruebas recoge la especificación de un caso de uso, dividido en secciones y escenarios, detallando las funcionalidades descritas en él y describiendo cada variable. También quedan plasmados las revisiones realizadas al caso de prueba; así como un registro de todo aquello que no corresponde a la calidad del software. A continuación se presentan las tablas de la sección probada para el caso de uso Configurar Conexión.

Nombre de la sección	Descripción de la funcionalidad
SC1 Configurar Conexión	El visor de reportes inserta los reportes que se encuentra en el subsistema GDR a la base de datos local.

Tabla 7. Secciones de prueba para el caso de uso Configurar Conexión

A continuación se detallan las variables que se encuentran asociadas al caso de uso Configurar Conexión.

No.	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	usuario	Campo de Texto	No	Alfanumérico de 1 a 255 caracteres. Admite a-z
2	contraseña	Campo de Texto	No	Alfanumérico de 1 a 255 caracteres. Admite 0-9 a-z A-Z - áéíóú ñÑ ÓÁÍÉÚ üÜ _ ( ) , ; . : / ^ - [ ] {}% $ # @ * " &
3	direccion	Campo de Texto	No	Alfanumérico. Admite 0-9 a-z A-Z - _ ( ). : /

Tabla 8. Descripción de las variables

Las anteriores representaciones permitieron la realización de una matriz de datos. Esta permite evaluar y probar la veracidad de la información introducida por el usuario, enfrascándose únicamente en la sección que se toma como referencia. Utilizando un juego de datos correctos e incorrectos se registraron, con el empleo de la técnica de partición de equivalencia, los resultados de las pruebas.

Una partición equivalente es una técnica de prueba de Caja Negra que divide el dominio de entrada de un programa en clases de datos basándose en una evaluación de las clases de equivalencia para una condición

de entrada. Una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones de entrada.

Escenario	Descripción	1	2	3	Respuesta del sistema	Flujo central
EC 1.1 Configurar Conexión	Se adicionan correctamente los datos de configuración.	*V*	*V*	*V*	Se adiciona correctamente la configuración en la BD. Muestra un mensaje de éxito en el almacenamiento	1- Se inicia la Actividad 2- Se llenan los datos correctamente. 3- Se seleciona guardar preferencias 4- Guarda los elementos
EC 1.2 Configurar Conexión con campos en blanco	Se adicionan los datos de configuración dejando campos en blancos. Se muestra un mensaje indicando el error.	*N/A*	*V*	*V*	Se muestra un mensaje indicando que que debe llenar todos los campos.	1- Se inicia la Actividad. 2- Se escoge la opción guardar preferencias 3- Se muestra un mensaje de error.
		V	*N/A*	*V*		
		V	*V*	*N/A*		
		N/A	*N/A*	*N/A*		
EC 1.3 Configurar Conexión con datos no válidos	Se adicionan los datos de conexión con valores inválidos. Se muestra un mensaje indicando el error.	*I (espacios)*	*N/A*	*V*	Se muestra un mensaje indicando que debe insertar correctamente los datos.	1- Se inicia la Actividad. 2- Se llenan los datos. 3- Se escoge la opción guardar preferencias 4- Se muestra un mensaje de error.
		I (valores numéricos)	*N/A*	*V*		
EC 1.3 Configurar Conexión con usuario o dirección inválida	Se adicionan los datos de conexión con un usuario o dirección inválidos. Se muestra un mensaje indicando el error.	*I (pepe)*	*N/A*	*V*	Se muestra un mensaje indicando que el usuario o la dirección están incorrectos.	1- Se inicia la Actividad. 2- Se llenan los datos. 3- Se escoge la opción guardar preferencias 4- Se muestra un mensaje de error.
		V (administrador)	*N/A*	*I*		
		I (pepe)	*N/A*	*I*		

Tabla 9. Caso de prueba y sección Configurar Conexión

3.4.3. Resultados de las pruebas

Pruebas de funcionalidad

Luego de realizar las pruebas de caja negra a los casos de pruebas asociados a los diferentes casos de uso, se obtuvo como resultado el correcto funcionamiento de la herramienta así como la validación de los campos de entrada asociados a cada una de estas. Al finalizar, se lograron detectar 12 no conformidades en el caso de prueba Administrar Reporte. Las mismas fueron corregidas a medida que fue avanzando el proceso de prueba.

Diseño de caso de prueba	No Conformidades Pendientes		
	Iteración 1	Iteración 2	Iteración 3
Administrar Reportes	12	6	0
Total	**19**	**10**	**0**

Tabla 10. Resumen de los resultados de las pruebas aplicadas

El resultado de algunas de las pruebas realizadas por escenario a las secciones identificadas en los casos de pruebas definidos. Ver (**Anexo 2, 3 y 4**).

Pruebas de rendimiento

Con el objetivo de comprobar el tiempo de respuesta de la herramienta para las acciones permitidas en la misma, se realizaron pruebas de rendimiento, utilizando para ello el IDE seleccionado y el emulador proporcionado por el ADT del marco de trabajo de Android. Los resultados obtenidos en estas pruebas demostraron que, dependiendo de las características del dispositivo móvil así como de los reportes tratados correspondientemente; el tiempo de respuesta cumplió con las características propuestas de rendimiento.

Para ver los resultados de estas pruebas y la respuesta de la herramienta. Ver (**Anexo 5**).

Conclusiones del capítulo 3

El presente capítulo permitió la realización de la descripción de la implementación de la herramienta, representando las dependencias que existen entre los principales componentes, a través del diagrama de componentes. Se brindó una descripción de los códigos más significantes de la herramienta así como los estándares de codificación. Finalmente luego de la implementación del sistema, se desarrollaron las pruebas de caja negra utilizando la técnica de partición equivalente para lograr la calidad del módulo implementado y validar la completitud de los requisitos. Posteriormente fueron determinados los resultados alcanzados durante la realización de las pruebas, para determinar la calidad del software, comprobando la aceptación del mismo y cumplimiento de los requisitos definidos.

CONCLUSIONES

Una vez culminado el desarrollo de dicho trabajo se arriban a las siguientes conclusiones:

- Se definió el marco teórico de la investigación para el desarrollo de la herramienta de visualización de reportes para móviles con sistema operativo Android, permitiendo escoger una metodología que garantizara una correcta estructura en todo el proceso de desarrollo del software, facilitando la selección de las herramientas necesarias para el desarrollo satisfactorio de la solución propuesta.
- Se definió el análisis y diseño de las clases del visor de reportes para móviles con sistema operativo Android, logrando un mejor entendimiento del proceso de negocio.
- Se realizó la implementación del diseño propuesto para el visor de reportes para móviles con sistema operativo Android, obteniendo una herramienta capaz de suplir con las deficiencias encontradas al módulo visor de reportes.
- Se aplicaron las pruebas funcionales de caja negra, descritas en los casos de pruebas realizados, validando con estas el correcto funcionamiento del visor de reportes para móviles con sistema operativo Android.

RECOMENDACIONES

Al concluir el desarrollo de la herramienta visor de reportes para móviles en sistema operativo Android se plantean las siguientes recomendaciones:

- Desarrollar la v2.0 del visor de reportes para móviles con sistema operativo Android con la posibilidad de observar los reportes de mayor complejidad.
- Investigar el uso de los protocolos WAP para la extensión de la web a la telefonía celular y el uso de la herramienta desde cualquier punto del territorio nacional.
- Investigar sobre la utilización de las herramientas de desarrollo SenchaTouch y PhoneGab con la tecnología JQuery Mobile para el desarrollo de aplicaciones web para telefonía celular.

REFERENCIAS BIBLIOGRÁFICAS

1. **Nacho.** microsiervos. *La primera llamada desde un teléfono móvil*. [En línea]. 22 de mayo de 2007. [Citado el: 22 de mayo de 2013] Disponible en:

   http://www.microsiervos.com/archivo/curiosidades/primera-llamada-telefono-movil.html.

2. **The State University of New Jersey**. [En línea]. [Citado el: 29 de noviembre de 2012] Disponible en: http://www.eden.rutgers.edu.

3. **Revista RED Edición especial.** "El ABC de las Telecomunicaciones". [En línea] Diciembre del 2002. [Citado el: 29 de noviembre de 2012].

4. **Pérez Rodríguez, Iván.** Ingeniería Técnica de Telecomunicaciones. 2009.

5. **Java**. [En línea] http://www.Java.com/.

6. **Lic. Firtman, Maximiliano.** Video2Brain. *Curso Básico Android*. [En línea] 2013. http://www.video2brain.com.

7. **Vílchez, Ángel.** configurarequipos. *Qué es Android: Características y Aplicaciones*. [En línea] 2 de abril de 2009. [Citado el: 11 de enero de 2013.] Disponible en: http://www.configurarequipos.com/doc1107.html.

8. **Bello, A.** Notadiario. *Ventajas y Desventajas de un Android*. [En línea] 14 de diciembre de 2011. [Citado el: 11 de enero de 2013.] Disponible en: http://www.notadiario.com/sci-tech/ventajas-y-desventajas-de-un-android/.

9. **ecured.cu**. ecured.cu. [En línea]. 2010. [Citado el: 5 de diciembre de 2012]. Disponible en: http://www.ecured.cu/index.php/SDK

10. **xatak.** xatakandroid. [En línea]. [Citado el: 22 de Mayo de 2013]. Disponible en: http://www.xatakandroid.com_tag_distribucion-de-versiones-android

11. **epf.eclipse.org**. epf.eclipse.org. *Mapa: Derrotero del OpenUP/Basic*. [En línea]. [Citado el: 5 de diciembre de 2012]. Disponible en:

http://epf.eclipse.org/wikis/openupsp/openup_basic/guidances/roadmaps/openup_basic_roadmap,_vEruwN-rEdqiM_wFaqLjNg.html.

12. **Ingeniería de Sistemas e Informática**. *Lenguaje Unificado de Modelado.* [En línea] 2011. [Citado el: 26 de noviembre de 2012]. Disponible en: http://www.slideshare.net/ligtningfleeting/lenguaje-unificado-de-modelado-6895413.

13. **González Cornejo, José Enrique.** DocIRS. *¿Cuáles son las características que debe tener una herramienta UML?* [En línea] [Citado el: 11 de enero de 2013.] Disponible en: http://www.docirs.cl/caracteristica_herramienta_uml.htm.

14. **ecured.cu**. ecured.cu. [En línea] 2010. [Citado el: 11 de enero de 2013.] Disponible en: http://www.ecured.cu/index.php/Lenguaje_de_Modelaje_Unificado#Ventajas.

15. **ecured.cu**. ecured.cu. [En línea]. 2010. [Citado el: 5 de diciembre de 2012]. Disponible en: http://www.ecured.cu/index.php/Herramientas_CASE.

16. **Conran, Aaron.** Sencha. [En línea] 3 de mayo de 2009. [Citado el: 28 de noviembre de 2011]. Disponible en: http://www.sencha.com.

17. **ecured.cu**. ecured.cu. [En línea]. 2010. [Citado el: 5 de diciembre de 2012]. Disponible en: http://www.ecured.cu/index.php/Lenguaje_de_Programaci%C3%B3n.

18. **Fernández, Belmonte.** *Introducción al lenguaje de programación Java.: Una guía básica.* 2004.

19. **Jaime.** Androideity. *Androideity.* [En línea] 16 de julio de 2012. [Citado el: 13 de diciembre de 2012.] Disponible en: http://androideity.com/2012/07/16/5-lenguajes-para-programar-en-android/.

20. **McClure, Wallace B., y otros, y otros.** *Professional Android Programming with Mono for Android and .Net/C#.* s.l. : Xamarin.

21. **Wolber, David, y otros, y otros.** *App Inventor.* Sebastopol : O´Reilly Media Inc, 2011.

22. **Holgate, Colin.** *LiveCode Mobile Development Beginner´s Guide.* Birmingham : Packt Publishing Ltd., 2012.

23. **ecured.cu**. ecured.cu. [En línea]. 2010. [Citado el: 5 de diciembre de 2012]. Disponible en: http://www.ecured.cu/index.php/Eclipse,_entorno_de_desarrollo_integrado.

24. **es.kioskea.net**. es.kioskea.net. [En línea]. [Citado el: 13 de diciembre de 2012]. Disponible en: http://es.kioskea.net/contents/cs/client-riche.php3.

25. **es.kioskea.net**. es.kioskea.net. [En línea]. [Citado el: 13 de diciembre de 2012]. Disponible en: http://es.kioskea.net/contents/cs/client-leger.php3.

26. **ecured.cu**. ecured.cu. [En línea]. 2010. [Citado el: 14 de enero de 2013]. Disponible en: http://www.ecured.cu/index.php/Eclipse,_entorno_de_desarrollo_integrado#Datos_y_cifras_relacionadas_con_Eclipse.

27. **¿Qué es un Sistema de Gestor de Bases de Datos o SGBD?** CAVSI. [Citado el: 28 de noviembre de 2011]. Disponible en: http://www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-de-bases-de-datos-o-sgbd.

28. **ecured.cu**. ecured.cu. [En línea]. 2010. [Citado el: 5 de diciembre de 2012]. Disponible en: http://www.ecured.cu/index.php/SQLite.

29. **Monferrer Agut , Raúl.** Especificación de Requisitos Software según el estándar IEEE 830. 2008.

30. **sparxsystems.** sparxsystems. [En línea] 2007. [Citado el: 24 de mayo de 2013.] Disponible en: http://www.sparxsystems.com.ar/resources/tutorial/use_case_model.html.

31. **Craig Larman**. UML y Patrones. 2003, Disponible en: http://archivos5.movistar.cl/comunidad/Book-UMLyPatrones-CraigLarman/UMLyPatrones-CraigLarman.rar.

32. **ecured.cu**. ecured.cu. [En línea]. 2010. [Citado el: 21 de marzo de 2013]. Disponible en: http://www.ecured.cu/index.php/Arquitectura_de_software.

33. **Soto, Jesus.** negomobile.es. *negomobile.es.* [En línea] 21 de julio de 2012. [Citado el: 3 de abril de 2013.] Disponible en: http://www.negomobile.es/es/node/221.

34. **Ingeniería, Facultad de Informática-Universidad Politécnica de Madrid-Unidad Doc** de.Patrones del “Grang of Four ”. España-Madrid : s.n.

35. **Almécija, Sergio.** blog.intelligenia. blog.intelligenia. [En línea] 20 de diciembre de 2012. [Citado el: 15 de enero de 2013.] http://blog.intelligenia.com/2012/12/el-patron-de-diseno-singleton-basico.html.

36. **sparxsystems**. sparxsystems. [En línea] 2007. [Citado el: 8 de abril de 2013.] Disponible en: http://www.sparxsystems.com.ar/resources/tutorial/uml2_packagediagram.html.

37. **definicion.de**. definicion.de. [En línea] 2008. [Citado el: 25 de abril de 2013.] Disponible en: http://definicion.de/modelo-de-datos/.

38. **I.Jacobson, G.Booch, J.Rumbauggh.** El Proceso Unificado de Desarrollo. s.l. : Addison Wesley, 2000.

39. **Daniele, Marcela.** Teoría 11: El Arte de Modelar UML. 2007.

40. **Becerra Tristán, Fernando**. kualtus.com. kualtus.com. [En línea] [Citado el: 28 de marzo de 2013.] Disponible en: http://serk.kualtus.com/codigo.htm.

41. **ecured.cu**. ecured.cu. [En línea]. 2010. [Citado el: 4 de abril de 2013]. Disponible en: http://www.ecured.cu/index.php/Estrategia_de_pruebas_de_software.

42. **ecured.cu**. ecured.cu. [En línea]. 2010. [Citado el: 4 de abril de 2013]. Disponible en: http://www.ecured.cu/index.php/Pruebas_de_Calidad_de_Software#Tipos_de_Pruebas_de_Software.

43. **ecured.cu**. ecured.cu. [En línea]. 2010. [Citado el: 4 de abril de 2013]. Disponible en: http://www.ecured.cu/index.php/Flujo_de_pruebas_de_un_software#Tipos_de_Prueba

44. **PRESSMAN, Roger** "*Ingeniería del Software. Un enfoque práctico*".2002. McGraw-Hill/Interamericana de España

BIBLIOGRAFÍA

Almécija, Sergio. blog.intelligenia.com. *blog.intelligenia.com.* [En línea] intelligenia, 20 de diciembre de 2012. [Citado el: 9 de mayo de 2013.] Disponible en : http://blog.intelligenia.com/2012/12/el-patron-de-diseno-singleton-basico.html.

Becerra Tristán, Fernando. kualtus.com. kualtus.com. [En línea] [Citado el: 28 de marzo de 2013.] Disponible en: http://serk.kualtus.com/codigo.htm.

Becerra Tristán, Fernando. kualtus.com. kualtus.com. [En línea] [Citado el: 28 de marzo de 2013.] Disponible en: http://serk.kualtus.com/codigo.htm.

Bello, A. Notadiario. [En línea] 14 de diciembre de 2011. [Citado el: 11 de enero de 2013.] Disponible en: http://www.notadiario.com/sci-tech/ventajas-y-desventajas-de-un-android/

Brito Acuña, K (2009). Selección de Metodologías de Desarrollo Cienfuegos

Brito Rodríguez, Julio César. *Módulo diseñador de modelos para el Generador Dinámico de Reportes v2.0.* La Habana : s.n., 2012.

Carrillo Pérez, I., Pérez González, R., Rodríguez, M., & Aureliano, D. Metodología de Desarrollo del Software. Universidad Politécnica de Valencia.

Colectivo de autores. Manual de capacitación del radioaficionado. Federación de Radioaficionados de Cuba. [En línea] 2010.

Conran, Aaron. Sencha. [En línea] 3 de mayo de 2009. [Citado el: 28 de noviembre de 2011]. Disponible en: http://www.sencha.com.

Craig Larman. UML y Patrones. 2003, Disponible en: http://archivos5.movistar.cl/comunidad/Book-UMLyPatrones-CraigLarman/UMLyPatrones-CraigLarman.rar.

Daniele, Marcela. Teoría 11: El Arte de Modelar UML. 2007.

developer.android.com. developer.android.com. [En línea]. [Citado el: 28 de noviembre de 2011]. Disponible en: http://developer.android.com/tools/adk/index.html.

developer.android.com. developer.android.com. [En línea]. [Citado el: 4 de diciembre de 2012]. Disponible en: http://developer.android.com/tools/sdk/eclipse-adt.html.

ecured. ecured. [En línea] 2010. [Citado el: 11 de enero de 2013.] Disponible en: http://www.ecured.cu/index.php/Lenguaje_de_Modelaje_Unificado#Ventajas.

ecured. ecured. [En línea]. 2010. [Citado el: 21 de marzo de 2013]. Disponible en: http://www.ecured.cu/index.php/Flujo_de_Trabajo_de_Implementaci%C3%B3n.

ecured.cu. ecured.cu. [En línea]. 2010. [Citado el: 14 de enero de 2013]. Disponible en: http://www.ecured.cu/index.php/Eclipse,_entorno_de_desarrollo_integrado#Datos_y_cifras_relacionadas_con_Eclipse.

ecured.cu. ecured.cu. [En línea]. 2010. [Citado el: 21 de marzo de 2013]. Disponible en: http://www.ecured.cu/index.php/Arquitectura_de_software.

ecured.cu. ecured.cu. [En línea]. 2010. [Citado el: 4 de abril de 2013]. Disponible en: http://www.ecured.cu/index.php/Estrategia_de_pruebas_de_software.

ecured.cu. ecured.cu. [En línea]. 2010. [Citado el: 4 de abril de 2013]. Disponible en: http://www.ecured.cu/index.php/Pruebas_de_Calidad_de_Software#Tipos_de_Pruebas_de_Software.

ecured.cu. ecured.cu. [En línea]. 2010. [Citado el: 4 de abril de 2013]. Disponible en: http://www.ecured.cu/index.php/Flujo_de_pruebas_de_un_software#Tipos_de_Prueba

ecured.cu. ecured.cu. [En línea]. 2010. [Citado el: 5 de diciembre de 2012]. Disponible en: http://www.ecured.cu/index.php/SDK.

ecured.cu. ecured.cu. [En línea]. 2010. [Citado el: 5 de diciembre de 2012]. Disponible en: http://www.ecured.cu/index.php/Herramientas_CASE.

ecured.cu. ecured.cu. [En línea]. 2010. [Citado el: 5 de diciembre de 2012]. Disponible en: http://www.ecured.cu/index.php/Eclipse,_entorno_de_desarrollo_integrado.

ecured.cu. ecured.cu. [En línea]. 2010. [Citado el: 5 de diciembre de 2012]. Disponible en: http://www.ecured.cu/index.php/SQLite.

epf.eclipse.org. epf.eclipse.org. [En línea]. [Citado el: 5 de diciembre de 2012]. Disponible en: http://epf.eclipse.org/wikis/openupsp/openup_basic/guidances/roadmaps/openup_basic_roadmap,_vEruwN-rEdqiM_wFaqLjNg.html.

es.kioskea.net. es.kioskea.net. [En línea]. [Citado el: 13 de diciembre de 2012]. Disponible en: http://es.kioskea.net/contents/cs/client-riche.php3.

es.kioskea.net. es.kioskea.net. [En línea]. [Citado el: 13 de diciembre de 2012]. Disponible en: http://es.kioskea.net/contents/cs/client-leger.php3.

Fernández, Belmonte. Introducción al lenguaje de programación Java.: Una guía básica. 2004.

González Cornejo, José Enrique. DocIRS. [En línea] [Citado el: 11 de enero de 2013.] Disponible en: http://www.docirs.cl/caracteristica_herramienta_uml.htm.

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

Hull, Elizabeth, Ken Jackson y Jeremy Dick. Requirements Engineering Third Edition. Londres: Springer, 2011

Ingeniería de Sistemas e Informática. Lenguaje Unificado de Modelado [En línea] 2011. [Citado el: 26 de noviembre de 2012]. Disponible en: http://www.slideshare.net/ligtningfleeting/lenguaje-unificado-de-modelado-6895413.

Ingeniería, Facultad de Informática-Universidad Politécnica de Madrid-Unidad Doc de.Patrones del "Grang of Four ". España-Madrid : s.n.

Jaime. Androideity. Androideity. [En línea] 16 de julio de 2012. [Citado el: 13 de diciembre de 2012.] Disponible en: http://androideity.com/2012/07/16/5-lenguajes-para-programar-en-android/.

l.Jacobson, G.Booch, J.Rumbauggh. El Proceso Unificado de Desarrollo. s.l. : Addison Wesley, 2000.

McClure, Wallace B., y otros, y otros. *Professional Android Programming with Mono for Android and .Net/C#.* s.l. : Xamarin.

Monferrer Agut , Raúl. Especificación de Requisitos Software según el estándar IEEE 830. 2008.

Nacho. microsiervos. [En línea]. 22 de mayo de 2007. [Citado el: 22 de mayo de 2013] Disponible en: http://www.microsiervos.com/archivo/curiosidades/primera-llamada-telefono-movil.html.

PRESSMAN, Roger "Ingeniería del Software. Un enfoque práctico".2002. McGraw-Hill/Interamericana de España

Qué es un Sistema de Gestor de Bases de Datos o SGBD CAVSI. [Citado el: 28 de noviembre de 2011]. Disponible en: http://www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-de-bases-de-datos-o-sgbd.

Revista RED Edición especial. "El ABC de las Telecomunicaciones". [En línea] Diciembre del 2002.

Revista RED Edición especial. "El ABC de las Telecomunicaciones". [En línea] Diciembre del 2002. [Citado el: 29 de noviembre de 2012].

Revista RED Edición Especial. (2012). El ABC de las Telecomunicaciones.

Soto, Jesus. negomobile.es. *negomobile.es.* [En línea] 21 de julio de 2012. [Citado el: 3 de abril de 2013.] Disponible en: http://www.negomobile.es/es/node/221.

sparxsystems. sparxsystems. [En línea] 2007. [Citado el: 24 de mayo de 2013.] Disponible en: http://www.sparxsystems.com.ar/resources/tutorial/use_case_model.html.

sparxsystems. sparxsystems. [En línea] 2007. [Citado el: 8 de abril de 2013.] Disponible en: http://www.sparxsystems.com.ar/resources/tutorial/uml2_packagediagram.html.

The State University of New Jersey. [En línea]. [Citado el: 29 de noviembre de 2012] Disponible en: http://www.eden.rutgers.edu.

Universidad de las Ciencias Informáticas. Manual de Usuario v1.8.0 del Generador Dinámico de Reportes.

Vílchez, Ángel. configurarequipos. [En línea] 2 de abril de 2009. [Citado el: 11 de enero de 2013.] Disponible en: http://www.configurarequipos.com/doc1107.html.

Wolber, David, y otros, y otros. *App Inventor.* Sebastopol : O´Reilly Media Inc, 2011.

xatak. xatakandroid. [En línea]. [Citado el: 22 de Mayo de 2013]. Disponible en: http://www.xatakandroid.com_tag_distribucion-de-versiones-android.

ANEXOS

Anexo 1: Acta de Aceptación

UCI
Universidad de las Ciencias Informáticas

Centro de Tecnologías de Gestión de Datos
DATEC

La Habana, 17 de Mayo del 2013
"Año 54 de la Revolución".

ACTA DE ACEPTACIÓN

De una parte, el Centro de Tecnologías de Gestión de Datos, en lo sucesivo DATEC, de la Universidad de las Ciencias Informáticas, representado en este acto por: Adrian Rosales Cruz, y de **otra parte** los estudiantes: Osraldo M. González Ortiz y Miguel E. Verdecia Fonseca.

Primero: Que en cumplimiento de los requisitos funcionales han sido efectuadas las implementaciones correspondientes.

CONSIDERANDO: Que los hitos realizados han sido desarrollados con la calidad requerida y bajo las condiciones pactadas y aprobadas por **Las Partes**.

CONSIDERANDO: Que los hitos que se han ejecutado cumplen con los requerimientos establecidos.

POR TANTO: Las Partes acuerdan formalizar mediante la presente Acta, la aceptación del producto: Visor de Reportes para Móviles con Sistema Operativo Android

Y para que así conste, se extiende la presenta **Acta** en dos (2) ejemplares, rubricados por **Las Partes**.

Osraldo M. González Ortiz
Miguel E. Verdecia Fonseca
Entregan

UCI Universidad de las Ciencias Informáticas Decanato Facultad

Recibe

Figura 13. Acta de Aceptación de la herramienta

Anexo 2: Caso de prueba Configurar Conexión

Figura 14. Escenario 1.1 Datos correctos

Figura 15. Escenario 1.2 Datos en blanco

Figura 16. Escenario 1.3 Datos incorrectos

Figura 17. Escenario 1.4 Usuario o dirección no válida

Anexo 3: Caso de prueba Controlar Reporte

Figura 18. Escenario 2.1 Filtrar sin criterio

Figura 19. Escenario 2.3 Filtrar por columna y criterio

Figura 20. Escenario 3.2 Exportar reporte sin insertar el nombre

Figura 21. Escenario 3.1 Exportar reporte insertando el nombre

Anexo 4: Caso de prueba Administrar Reporte:

Figura 22. Escenario 2.3 Filtrar al menos un reporte

Figura 23. Escenario 3.1 Buscar reporte

Anexo 5: Respuestas de la herramienta (uso del Eclipse)

Figura 24. Tiempo de respuesta (descarga de reporte)

Figura 25. Tiempo de respuesta (visualizar reporte)