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

Facultad 1

Sistema para la publicación de contenido web desde un dispositivo móvil con el sistema operativo Android

Autora: *Daynis Rodriguez Ramos*

Tutor (es):

MSc. Sahilyn Delgado Pimentel

Ing. Nodelvis Hernández Rodríguez

Ing. Abel Salas López

La Habana, junio de 2017

Declaración de Autoría

Declaro por este medio que yo Daynis Rodriguez Ramos, con carné de identidad 94093004994 soy la autora principal del trabajo titulado “Sistema para la publicación de contenido *web* desde un dispositivo móvil con el sistema operativo Android” y autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Para que así conste se firma la presente declaración jurada da autoría en La Habana a los días ___ del mes de_____ del año_____.

Autor

_____

Daynis Rodríguez Ramos

Tutor

_____

MSc. Sahilyn Delgado Pimentel

Tutor

_____

Ing. Nodelvis Hernández Rodríguez

Tutor

_____

Ing. Abel Salas López

Yo rechazo la mentira porque sé que la ignorancia ha sido la gran aliada de la opresión a lo largo de la historia.

Fidel Castro

Agradecimientos

A mi madre por forjarme y convertirme en la mujer que soy. Por guiarme y darme su apoyo cuando las cosas iban mal. Porque que te quiero mami.

A mi abuela por consentirme, preocuparse y ayudarme en todos los momentos de este arduo camino.

A mi hermana por siempre estar presente, por ser parte de este sueño y por sus verdades directas pero muy constructivas.

A mi novio por estar siempre a mi lado brindándome su apoyo y dándome ánimos cuando pensaba que no iba a ser posible este logro.

A todos mis amigos que hicieron que estos 5 años de mi vida fueran inolvidables, y principalmente a Aimet que supo escucharme y soportar todas mis malcriadeces.

Dedicatoria

Dedico el resultado de este trabajo y de toda mi vida como estudiante, especialmente a la persona que siempre fue mi fuente de inspiración, a quien más orgulloso estaría de mí, a ti que aunque no estas a mi lado sé que me acompañaste en todo momento. A mi Abuelo Roberto.

A mi madre querida por darme la vida, su amor y su apoyo incondicional, para lograr este sueño que es su gran ilusión.

Resumen

La presente investigación contribuye a la integración de la tecnología móvil con la publicación de contenido *web*, para mejorar el trabajo de las personas dedicadas a la publicación de contenido *web* en *internet*. Se realizó un análisis de las fuentes bibliográficas y sistemas actuales de publicación de contenido *web* a través de un dispositivo móvil a nivel internacional y nacional. El estudio demostró la necesidad de implementar una nueva solución, que estará guiada por la metodología de desarrollo de software Proceso Unificado Ágil UCI (AUP-UCI). Para el desarrollo de la aplicación se empleó como lenguaje de desarrollo Java en su versión 8.0 y Android Studio en su versión 2.1.3 como entorno integrado de desarrollo lo que permitió obtener un sistema de fácil manejo y alta calidad; *Visual Paradigm* 8.0 como herramienta para el modelado y para el trabajo con la base de dato del dispositivo móvil la herramienta SQLite en su versión 3.9.2. Esta aplicación permite la gestión de contenido y publicación del mismo. Para asegurar la calidad, fiabilidad y robustez del software se realizaron pruebas unitarias, de aceptación, de integración y de compatibilidad, las cuales comprobaron el correcto funcionamiento de la aplicación.

Palabras clave: dispositivo móvil, publicación de contenido *web*.

Índice de Contenido

Índice de Tablas

Índice de Figuras

Introducción

A lo largo de la historia, las Tecnologías de la Información y las Comunicaciones (TIC) evolucionan en cuanto a su diversidad y complejidad. Las TIC, se ajustan a las necesidades del hombre y forman parte de la cultura tecnológica que le rodea y con la que debe convivir. Una de las principales contribuciones que ofrece es el fácil acceso a numerosas fuentes de conocimientos a través de *internet*.

Internet, es una red global a gran escala que permite la interconexión de millones de dispositivos al mismo tiempo. Una manera de acceder a disimiles volúmenes de información, independientemente de la situación geográfica e idioma en que se encuentre, es mediante la *World Wide Web* (WWW) o la *web*, que simboliza una porción grande y popular de la internet (Candón, 2011).

La *web* es hoy un medio extraordinariamente flexible y económico para la comunicación. Ofrece distintos tipos de información, aplicaciones informáticas, y servicios de todo tipo: formularios interactivos, foros de discusión, buscadores, tan sólo por citar algunos ejemplos. El desarrollo de la *web* y su posterior crecimiento exponencial generó un cambio radical en cuanto a la rapidez de difusión y disponibilidad del conocimiento, por lo que está a punto de convertirse en una enciclopedia universal del conocimiento humano (Lapuente, 2013).

Actualmente se vive en lo que se llama, la sociedad de la información y el conocimiento. El desarrollo económico, tecnológico y social de una nación, está cada vez más ligado al desarrollo del conocimiento científico y tecnológico de la misma. Todos estos progresos contribuyen al paradigma de socialización digital, en los diversos países del mundo, en los cuales los elementos conectados ya no son páginas sino personas. Cuba no está exenta de este desarrollo tecnológico.

En Cuba se lleva a cabo un profundo proceso de informatización, en el que tiene un papel fundamental la Empresa de Telecomunicaciones de Cuba S.A. (ETECSA). La misma se orienta a brindar servicios de telecomunicaciones, a respaldar los requerimientos del desarrollo socio-económico, tecnológico, servicios relacionados con el acceso y utilización de internet. Según expone ETECSA (2015), en estos momentos, el país dispone de 361 salas de navegación de internet con tecnología *Wi-fi*, en espacios públicos como son: parques, avenidas y boulevares; bajo el nombre de WIFI_ETECSA. Diariamente en zonas *wifi* en el país se

producen un total de 250 000 conexiones (Radio Habana Cuba, 2015). Esto constituye la primera etapa de una nueva vía de acceso a *internet*, a partir de una estrategia de acceso masivo y seguro a la red.

Con el avance que se ha logrado en el país, los *smartphones* o móviles inteligentes, se posicionan como un elemento transcendente que dirige muchas de las actividades diarias. Según la Oficina Nacional de Estadística e Información (2015), la telefonía móvil sobrepasó los tres millones de usuarios en abril, un sector en franco crecimiento. Los *smartphones* permiten a los usuarios llevar consigo un objeto de dimensiones pequeñas, que cumple funciones similares a las de un computador personal. Una de las funciones más importantes que caracteriza a un dispositivo móvil es la conectividad e interacción con las redes de datos (Malave, 2011). Sin embargo, lo que lo hace atractivo es el sistema operativo, que junto con el *hardware* logran desarrollar operaciones que permiten correr cierta cantidad de aplicaciones, convirtiéndolos en dispositivos más competitivos que otros. Actualmente en Cuba, aunque no se conoce la cifra exacta, se estima que el 70 por ciento de la población que posee telefonía móvil, opera bajo el sistema operativo Android (Pedroso, 2016).

Android tiene la característica de ser una plataforma totalmente libre, que permite desarrollar aplicaciones o modificar las ya existentes. Esto lo sitúa en una opción innovadora capaz de satisfacer los requerimientos de sus usuarios. Su finalidad es enfocarse en la necesidad de los operadores móviles y fomentar el desarrollo de aplicaciones, cualidad que ningún otro incluye en sus conceptos. Báez (2011) plantea que este sistema operativo no sólo es utilizado para dispositivos móviles inteligentes, también es compatible con otras tecnologías como los recién presentados *tablet*, lo que evidencia su adaptación a nuevas tecnologías y dispositivos emergentes.

En la Universidad de las Ciencias Informáticas (UCI) desde su surgimiento, se trabaja en la ardua tarea de incorporar a Cuba en la naciente Industria de Software, convirtiéndola en el sector más eficiente, rentable y que aporte beneficios a la sociedad. En la Universidad, específicamente en el Centro de Ideoinformática (CIDI) perteneciente a la Facultad 1, se trabaja en la creación de portales *web*. El objetivo principal de este centro es proveer soluciones integrales, productos y servicios relacionados con las tecnologías de *internet*, en función de la defensa de la ideología socialista a través del internet y la *web* (Verdecia, 2017). El desarrollo de estos sitios *web* son factores importantes a la hora del intercambio constante de opiniones que se generan entre los visitantes al sitio y los distintos temas que se publiquen (Tramullas, 2010). En los últimos años han emergido diferentes tipos de aplicaciones y plataformas *web* que tratan de ayudar a los científicos en su trabajo diario,

ofreciéndoles herramientas para gestionar sus flujos de trabajo, facilitarles el rastreo de información pertinente o brindarles nuevos medios para comunicar sus hallazgos.

Con todo el auge tecnológico alcanzado en la actualidad, aún publicar en un sitio *web* en ocasiones, es complicado. Los usuarios cuando desean divulgar una noticia deben conectarse directamente al sitio *web* y redactar la información que desean publicar. Debido a que las conexiones son lentas y que no se puede acceder a los sitios *web* desde cualquier lugar, el proceso de gestión del contenido puede resultar extenso, al no poseer una conexión *wifi* disponible en todo momento. Una alternativa es elaborar el contenido con anterioridad, proceso que puede ser molesto al no contar con una vía que brinde una mayor organización y estructura al confeccionar la información. Este proceso debe realizarse con cautela, para no incurrir en errores al introducir los datos en los espacios destinados para cada elemento, en el momento de publicar la información. Parte de los usuarios que se dedican a la publicación de información necesitan viajar para recopilar y buscar diferentes fuentes de información, pudiendo dedicar tiempo en esos momentos a la creación y gestión de la noticia. Con este fin una opción es la utilización de los dispositivos móviles orientada a tareas con este objetivo, facilitando el trabajo de las personas que se dedican a publicar información. Aunque el empleo de los dispositivos móviles va en aumento, no se dispone de un mecanismo que permita crear y gestionar contenidos mediante una interfaz amigable que posea los campos que conforman la noticia, establecer conexión con el sitio, enviarlos, y notificar al usuario la llegada correcta del contenido.

Considerando la situación problemática anteriormente descrita, se plantea como ***problema de investigación:***

¿Cómo gestionar contenido *web* desde un dispositivo móvil, para contribuir a su publicación?

Para la realización de la investigación se define como ***objeto de estudio*** el proceso de gestión de contenido *web* y el ***campo de acción*** se encuentra enmarcado en la gestión de contenido *web* a través de dispositivos móviles con el sistema operativo Android.

Para dar solución al problema planteado, se define como ***objetivo general*** desarrollar un Sistema que permita gestionar contenido *web* desde dispositivos móviles con sistema operativo Android para contribuir a su publicación.

Con el propósito de cumplimentar gradualmente el objetivo general antes mencionado, el mismo se delimita en los siguientes ***objetivos específicos:***

1. Construir los referentes teóricos fundamentales que sustentan la investigación relacionados con el desarrollo de herramientas para la publicación de contenidos *web*[1].
2. Identificar las funcionalidades del sistema para la publicación de contenidos *web*.
3. Diseñar la solución informática a desarrollar.
4. Implementar las funcionalidades del sistema deseado.
5. Validar las funcionalidades del sistema para la publicación de contenidos *web* a través de pruebas de software.

Para hacer efectivo el cumplimiento de los objetivos planteados, quedan definidas las siguientes ***tareas de investigación:***

1. Realización del estudio acerca de las formas en que se publican en sitios *web*.
2. Identificación de las funcionalidades que tendrá el sistema, las técnicas de programación, lenguaje de programación y marco de trabajo para el desarrollo de la aplicación.
3. Diseño de la arquitectura y selección de patrones de diseño que serán empleados en la solución.
4. Realización del diseño de la aplicación.
5. Implementación del sistema.
6. Documentación de las pruebas realizadas.

Después de haber tratado los elementos fundamentales del área de la ciencia a incidir y los objetivos fundamentales, se formula la siguiente hipótesis de investigación: El desarrollo de un sistema para la gestión del contenido *web* desde dispositivos móviles con el sistema operativo Android facilitará su publicación.

[1]Es documento, imagen, animación, texto, sonido, video, aplicación, etc. que puede ser transmitido y ejecutado a través de un navegador en la web (Alegsa, 2010).

Se define como **variable independiente**: Sistema para la gestión de contenido *web* desde dispositivos móviles con el sistema operativo Android. Como **variable dependiente**: Contribuirá al proceso de publicación en la *web*.

Tabla # 1: Operacionalización de las variables.

Fuente: (Elaboración propia)

Variables	**Dimensiones**	**Indicadores**	**Sub-indicadores**	**Unidades de medidas.**
Sistema para la gestión de contenido *web* desde dispositivos móviles con el sistema operativo Android	Aplicación móvil.	Satisfacción de los usuarios.	9-10	Alta
			5-8	Media
			1-4	Baja
Contribuirá al proceso de publicación en la *web*.	Publicación del contenido *web*.	Satisfacción de los usuarios.	9-10	Alta
			5-8	Media
			1-4	Baja

Para realizar la presente investigación se tuvieron en cuenta los siguientes métodos científicos:

Métodos teóricos:

El método Analítico-Sintético se empleó en la búsqueda de los elementos esenciales para la publicación de contenido a través de dispositivos móviles. También en el análisis de la estructura de sistema operativo Android para valorar, conocer sus particularidades y lograr comprender su funcionamiento.

El método Histórico-Lógico permitió mediante el análisis de la evolución de los sistemas de publicación a través de dispositivos móviles, determinar las principales características que debe poseer el sistema que se describe en la presente investigación, así como las herramientas y tecnologías utilizadas.

La Modelación se utilizó con el objetivo de realizar una representación simplificada de los artefactos necesarios, descomponerlos y estudiarlos para conocer nuevas relaciones y cualidades del objeto, para mejorar su comprensión, operar y experimentar con ellos, mediante el uso de diagramas y modelos más simples.

El método Inductivo-Deductivo se aplicó para la determinación de las generalidades y se parte del análisis de casos particulares, para arribar a razonamientos que permitan la fundamentación teórica y elaboración del sistema que se desea.

Métodos empíricos:

Entrevistas: se evidenció en conversaciones planificadas para obtener información con especialistas, trabajadores del departamento de Servicios Informáticos para internet del centro CIDI. Lo que permitió realizar el levantamiento de requisitos, así como un mejor entendimiento acerca del funcionamiento de la tecnología a utilizar.

Observación: fue empleada en los distintos momentos de la investigación para la recogida de la información precisa, real y confiable que ayuden a comprender lo principal de la problemática. Al mismo tiempo esto permitió el planteamiento del problema, dando paso a enmarcar el objeto de estudio y el campo de acción, propiciando enfocar la investigación hacia lo que se necesita alcanzar y cómo alcanzarlo.

Estructura del trabajo:

Capítulo I. Fundamentación teórica del sistema para la publicación de contenido *web* desde dispositivos móviles.

En este capítulo se realiza una fundamentación teórica de la investigación, se acentúan las bases para entender el problema a solucionar. Se realiza un estudio de las soluciones existentes en el mundo actualmente. Se definen los conceptos fundamentales, técnicas, tendencias, lenguajes, metodologías y herramientas, que serán utilizadas en la construcción de la solución.

Capítulo II. Análisis y diseño del sistema para la publicación de contenido *web* desde dispositivos móviles.

En este capítulo se representa y analiza todo lo relacionado con el modelado del negocio del sistema para la publicación desde dispositivos móviles. Se explica cómo funciona este, se exponen las descripciones completas de los requisitos funcionales y no funcionales con que contará el sistema que se desea implementar y la descripción de los diagramas que representan los requisitos del sistema.

Capítulo III. Construcción y pruebas del sistema para la publicación de contenido *web* desde un dispositivo móvil con el sistema operativo Android.

Este capítulo muestra todo lo relacionado con el modelo de implementación del sistema a partir del diseño anteriormente realizado. Se describen las pruebas a utilizar en el sistema desarrollado y los resultados obtenidos durante las mismas. Al finalizar el capítulo se continúa con las conclusiones, las recomendaciones, las referencias bibliográficas, glosario de términos, acrónimos y los anexos.

Como ***posible resultado*** se pretende que, al concluir la investigación, en Cuba se cuente con una aplicación para dispositivos móviles con el sistema operativo Android que permita asegurar la publicación y distribución del contenido *web* de forma inmediata en los sitios *web*, y así de esta manera satisfacer las necesidades básicas de los editores o periodistas, permitiéndoles la divulgación de la información desde cualquier espacio.

Capítulo 1: Fundamentación teórica del sistema para la publicación de contenido web desde dispositivos móviles.

En este capítulo se realiza una profunda investigación sobre los sistemas homólogos de publicación de contenidos *web* en el ámbito nacional e internacional. Se describen las principales características de las herramientas y lenguajes a utilizar. Se abordan una serie de aspectos que fundamentan teóricamente esta investigación, de modo que resulte evidente su importancia.

1.1 Estudios de los sistemas homólogos

A partir de la necesidad existente de desarrollar un sistema para la publicación de contenido *web* desde un dispositivo móvil con el sistema operativo Android, se hace necesario realizar un estudio de los sistemas que utilizan mecanismos para la publicación de contenido *web*. El análisis de estos permite darle visibilidad, enriquecer y fortalecer esta investigación, de modo que se logre identificar las funcionalidades adecuadas y las mejores prácticas a implementar.

1.1.1 Estudios de los sistemas homólogos en el ámbito internacional

Son varias las herramientas que permiten escribir, editar, corregir la ortografía y gestionar el contenido, aunque no se esté conectado a la red. En este estudio se han seleccionado para su descripción, herramientas *offline* disponibles para la edición de *blog*.

Thingamablog

Thingamablog es una aplicación de escritorio, de código abierto, basada en Java, multiplataforma, con la que se puede crear y mantener los *blogs* sin necesidad de conexión, ni otros elementos extras como una cuenta en Blogger, o un servidor con soporte SQL. El programa funciona como un editor con el que puede crear los artículos de los diarios sin conexión, y es capaz de publicarlos directamente en el servidor FTP que se le indique. Este editor permite publicar de manera remota el contenido vía e-mail (Thingamablog, 2009). Sin embargo, aunque esta herramienta es de código abierto, no es posible utilizarla como solución, ni adaptarla para el desarrollo de una aplicación Android, por ser una herramienta de escritorio. No permite la publicación

de contenido en sitios *web* de noticias solo en *blog*. Además este sistema solo permite el trabajo con el texto del contenido, no con imagen, video y audio necesarios para trabajos publicitarios.

W.bloggar

W.bloggar es un servicio gratuito de *offline blogging* para Microsoft Windows. Representa una aplicación ideal para *bloggers*, pues facilita cualquier acción de actualización, edición y publicación de contenidos sin tener que acceder de forma directa a la página *web* donde están alojados los *blogs*. Soporta diversas herramientas como: Blogger, Blogger Pro, Movable Type, Drupal. Cuenta con una versión portátil, pero no dispone de: ping, etiquetas, ni notificaciones a *Twitter* (Wbloggar, 2016). Aunque es una herramienta gratuita, no es de código abierto. No brinda la facilidad de actualización, edición y publicación en sitios y *blogs* de noticias. Aunque cuenta con una versión portátil, no ofrece en su solución el empleo de la tecnología móvil. Sin embargo ofrece la oportunidad de editar y publicar contenidos de manera *offline* pero no permite eliminar contenido de manera *offline*.

Mailhandler

Mailhandler es un módulo para CMS Drupal que permite a los usuarios registrados en un sitio *web*, crear contenidos y comentarios a través de correo electrónico. El proceso de autenticación en el sitio se basa en hacer coincidir la dirección de correo electrónico y la cuenta de usuario de Drupal, o mediante la utilización de un método basado en contraseña. El módulo proporciona una interfaz de usuario para generar direcciones de correo electrónico únicas para cada usuario del sitio con el permiso de enviar contenido por correo. Posibilita la recuperación de correo desde cualquier buzón: IMAP o POP3 utilizando una variedad de exportar e importar configuraciones de buzón usando comandos (Drupal, 2017). A pesar de las características que posee, no es posible aprovechar este módulo para darle solución debido a que solo permite el envío de contenido para sitios *web* desarrollados en Drupal y tampoco emplea la utilización de dispositivos móviles. Además no es posible el envío de videos y audios a través de este sistema.

1.1.2 Estudios de los sistemas homólogos en el ámbito nacional

Luego de una profunda investigación se identificaron varios sistemas relacionados con la publicación de contenido a nivel nacional, haciendo referencia a tres de ellos.

Sistema para la edición y publicación electrónica de la Gaceta Oficial de la República de Cuba.

Consiste en un sistema *web* que permite consultar la información concerniente a los procesos de edición y publicación de la Gaceta Oficial (GO) de la República de Cuba. Ofrece una plataforma de trabajo colaborativo para las instituciones que intervienen en el proceso legislativo cubano, así como posibilita el proceso de edición y consecuentemente su publicidad digital en el sitio de la GO, lo cual admite introducir mejoras al proceso de las ediciones anteriores, documentos importantes relacionados a la Legislación Cubana. La interacción de los diferentes actores y gestión de la información generada por los procesos de edición y publicación de la GO, de manera que contribuye al establecimiento de un canal de comunicación y servicios a la comunidad de usuarios de dichas entidades. Este sistema se encuentra desarrollado en CMS Drupal, utiliza como gestor de bases de datos para el almacenamiento de la información: PostgreSQL y como servidor *web* para el hospedaje de la aplicación: Apache. Los lenguajes empleados fueron HTML, JavaScript, CSS y PHP (Ivonet, 2014).

Este sistema no brinda la oportunidad de publicar contenidos en diferentes sitios *web* de noticias y tampoco emplea la utilización de la tecnología móvil en su solución. No permite la creación, edición, eliminación y gestión de contenido *web* de manera *offline*. No ofrece la publicación de imágenes, videos y audios en el sitio *web* de la Gaceta Oficial de la República de Cuba.

Herramienta para la actualización de blogs en Wordpress sin acceder a la administración en línea desde sistemas GNU/Linux.

Es una aplicación de escritorio desarrollada sobre la plataforma Wordpress, para gestionar el contenido de los diferentes blogs cuando no se tiene conexión a internet, evitando ingresar constantemente a los servidores para la publicación en los mismos. Permiten hacer todo el trabajo de edición con las mismas funcionalidades del editor en línea de la plataforma de blog, posteriormente se conecta a internet y se sincronizan con los servicios necesarios y los archivos se actualizan de manera rápida. Este sistema fue desarrollado utilizando: NetBeans como IDE, XML-RPC como protocolo de comunicación, Ekit como procesador de textos adaptado al sistema desarrollado y JUnit para las pruebas automatizadas al software siguiendo como principio desarrollar software libre para uso de todos (Perdomo, 2015).

Al ser una herramienta de escritorio, no emplea en sus soluciones el uso de tecnología móvil. Fue desarrollada para gestionar el contenido solo en los *blogs* sobre la plataforma Wordpress. No permite el trabajo con contenido multimedia como las imágenes, videos y audios para enriquecer el contenido a actualizar. No ofrece portabilidad.

Sistema para la Publicación y Distribución de Contenidos Multimedia.

Radica en una aplicación *web* distribuida, que facilita la publicación y distribución de los contenidos multimedia en la red nacional, de manera que las publicaciones se realizan desde servidores locales en el país y con la rapidez requerida. Este sistema admite gestionar, compartir, votar e incluso sugerir contenidos multimedia.

La arquitectura de información del sistema para la publicación y distribución de contenidos multimedia es amigable y fácil de entender para usuarios que tengan conocimientos informáticos básicos. Cuenta con la implementación de una capa de servicios *web* la cual hace posible que el proceso de publicación de contenidos multimedia en aplicaciones *web* se realice de forma transparente al usuario encargado de publicar (Díaz, 2014).

Aunque brinda grandes facilidades para el trabajo con multimedia, no posee portabilidad al ser un sistema *web*. No permite la gestión de contenido *web* de manera *offline*, ni la publicación de contenidos en múltiples sitios *web*. No hace uso en su solución de empleo de tecnología móvil. No es posible reutilizar su código por no ser una herramienta de código abierto y por haber sido desarrollada como aplicación *web*.

Resultados del estudio de los sistemas de homólogos.

Luego de realizar un análisis en los sistemas seleccionados, observando sus características y contribuciones, fue posible resumir que no cumplían con las necesidades de la situación problemática. Estos sistemas no emplean en sus soluciones el uso de tecnología móvil para brindarle al usuario la facilidad de poder gestionar el contenido *web* en todo momento. Sin embargo aunque algunos sistemas permitían la actualización de contenido sin conexión, lo hacían en blog y eran herramientas de escritorio. Solo el Sistema para la Publicación y Distribución de Contenidos Multimedia, permite la distribución de contenido multimedia como imágenes, video y audio, aunque no es posible utilizarlo para la gestión de manera *offline*. Por lo que después de realizar un análisis de cada uno, se puede apreciar que no es posible aprovechar algunas de esas soluciones debido

a que no satisfacen las necesidades de la situación problemática. Po todo lo antes planteado se evidencia la necesidad de implementar una nueva solución para la publicación de contenidos *web*, que integre la utilización de la tecnología móvil y la publicación para mejorar la gestión de estos procesos en el país.

1.2 Análisis de las tecnologías, herramientas y lenguajes a utilizar

En este epígrafe se analizarán las características particulares de las herramientas, tecnologías y lenguajes a utilizar en el desarrollo de la propueta de solución.

1.2.1 Herramientas de Desarrollo

SQLite: herramienta de *software* libre, que permite almacenar información en dispositivos de una forma sencilla, eficaz, potente, rápida y en equipos con pocas capacidades de *hardware*. Esta se puede usar tanto en dispositivos móviles como en sistemas de escritorio, sin necesidad de realizar procesos complejos de importación y exportación de datos, por su compatibilidad entre las diversas plataformas disponibles. No necesita un proceso separado funcionando como servidor ya que lee y escribe directamente sobre archivos que se encuentran en el disco duro. Su base de datos se almacena en un único fichero a diferencia de otros Sistemas Gestores de Base de Datos (SGBD) que hacen uso de varios archivos (Vogel, 2010).

Para el desarrollo de la aplicación en Android se decidió utilizar SQLite en su versión 3.9.2 como herramienta para el almacenamiento, debido a su popularidad y utilidad en el sistema operativo Android, por su pequeño tamaño en memoria y la realización de las operaciones de manera eficiente y rápida; además de precisar poca configuración y ser de código libre.

Visual Paradigm en su versión 8.0: herramienta CASE empleada para visualizar y diseñar elementos de *software*, para ello utiliza el lenguaje UML, proporciona a los desarrolladores una plataforma que les permite diseñar un producto con calidad de forma rápida. Se emplea en la modelación de este proyecto por su característica de ser multiplataforma y por las facilidades que brinda de soportar el ciclo completo de desarrollo de software: análisis, diseño, implementación y pruebas. Permite la generación de bases de datos, conversión de diagramas entidad-relación a tablas de base de datos, mapeos de objetos y relaciones, ingeniería directa e inversa, la gestión de requisitos de software y la modelación de procesos del negocio (Visual Paradigm, 2014).

1.2.2 Entornos de Desarrollo Integrados

Un entorno de desarrollo integrado (IDE), es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Los IDE proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, PHP, Python, Java, C# y otros. En algunos lenguajes, puede funcionar como un sistema en tiempo de ejecución, en donde se permite utilizar el lenguaje de programación en forma interactiva, sin necesidad de trabajo orientado a archivos de texto (Fergarciac, 2013).

Android Studio: entorno de desarrollo basado en *IntelliJ IDEA*. Posee distintos componentes que ayudan en la tarea de la construcción de aplicaciones; sistema de construcción basado en *Gradle*, la construcción de variantes y múltiples archivo APK[2], como también plantillas de código. Contiene una interfaz de usuario que es construida o diseñada previamente, con variados modelos de pantalla, donde los elementos existentes pueden ser desplazados. Adicionalmente se abarca depuradores para emuladores y la posibilidad de trabajo con *Logcat*. Herramienta que mejora el rendimiento, facilita el uso y no producen problemas de compatibilidad con las diferentes versiones. Facilita la reutilización de código, de recursos, posibilitando además configurar, extender y personalizar el proceso (Ibarreche, 2015).

Eclipse: plataforma de desarrollo de código abierto, diseñado para ser extendido de forma indefinida a través de *plug-ins*. Es un entorno de desarrollo multiplataforma para crear, integrar e implementar herramientas de desarrollo de aplicaciones para uso en una amplia gama de tecnología informática. No tiene en mente un lenguaje específico, sino que es un IDE genérico. No es más que un entorno de desarrollo en el que localizarían todas las herramientas y funciones necesarias para el trabajo, recogidas además en una atractiva interfaz que lo hace fácil y agradable de usar (The Eclipse Foundation. 2016).

Selección del entorno de desarrollo integrado

Se eligió Android Studio en su versión 2.1.3 para el desarrollo de la aplicación móvil, pues es el entorno de programación oficial para desarrolladores de Android y está basado en *IntelliJ* IDEA, uno de los IDE para Java.

[2] Aplicación Empaquetada de Android.

Ofrece comodidad para los desarrolladores, permitiendo invocar, durante el desarrollo de aplicaciones, las herramientas necesarias como una forma más ágil de trabajo. En este entorno de desarrollo a través del análisis de código, se destacan los errores de forma inmediata, brindándole soluciones. Su forma de construir los *APK* es más serio y lo más parecido a un proyecto en Java.

1.2.3 Lenguajes utilizados

La propuesta a la situación problemática de esta investigación propone el desarrollo de una aplicación Android para dispositivos móviles, que permita la publicación de la información periodística deseada. Por lo que es necesario la selección de los lenguajes de programación y de modelado a utilizar. En el presente acápite se caracterizan los seleccionados.

Java: lenguaje concurrente, basado en clases, y orientado a objetos, fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Una de las características más importantes es que los programas “ejecutables”, creados por el compilador de Java, son independientes de la arquitectura. Es multiplataforma, un *software* de distribución libre, es completo y poderoso, se pueden realizar muchas tareas con él, pues posee librerías y utilidades muy completas que facilitan la programación (Gironés, 2012).

Es empleado Java en su versión 8.0, debido a que es de código abierto y puede utilizarse en caso de que los contenidos que se necesiten no se encuentren en las librerías nativas de Java. Esta versión contiene importantes mejoras para el rendimiento, estabilidad y seguridad de las aplicaciones, entre las que se destacan: los métodos de extensión virtual y expresión Lambda, por citar tan solo dos ejemplos.

Lenguaje de modelado 8.0: Lenguaje de Modelado Unificado (*UML* por sus siglas en inglés), es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de *software*. Sirve para el modelado completo de sistemas complejos, tanto en el diseño de los sistemas de *software* como para la arquitectura *hardware* donde se ejecuten. Está formado por símbolos que son utilizados por muchas metodologías. Su objetivo principal es entregar un material de apoyo que le permita al lector poder definir diagramas propios como también poder entender el modelamiento de diagramas ya existentes. Mediante el lenguaje UML es posible establecer la serie de requerimientos y estructuras necesarias para plasmar un sistema de *software* previo al proceso intensivo de escribir código (López, 2011).

1.3 Metodología de desarrollo de software

Según Pressman (2010) la metodología indica los diferentes pasos y procedimientos para obtener los distintos productos parciales y finales. Es el conjunto de fases, reglas, técnicas, herramientas, documentación y aspectos de formación para que los desarrolladores tengan claridad y facilidad de comprensión. En sí para el desarrollo de *software*, se necesita aplicar una metodología, de manera que se sepa que hacer y cómo hacerlo para conseguir lo que se quiere y cumplir con las metas planteadas.

Para el desarrollo del sistema se selecciona la metodología Proceso Unificado Ágil de Scott Ambler UCI o *Agile Unified Process* UCI (AUP-UCI) en inglés, es una versión simplificada del Proceso Unificado de *Rational* (RUP). Esta describe una manera simple y fácil de entender la forma de desarrollar aplicaciones de *software* usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. Entre las técnicas ágiles se incluyen el Desarrollo Dirigido por Pruebas, Modelado Ágil, Gestión de Cambios Ágil, y Refactorización de Base de Datos para mejorar la productividad (Rodríguez, 2015).

AUP-UCI se encuentra centrada en actividades de alto valor, esenciales para el desarrollo. Es una metodología adaptada a la Universidad de las Ciencias Informáticas, y utilizado por el departamento Servicios Informáticos para Internet (SENIT) para el cual se desarrolla este sistema. Además es ágil por lo que la prioridad es satisfacer al cliente mediante tempranas y continuas entregas del *software*. Es factible para equipos pequeños, con pocos roles y artefactos.

1.4 Conclusiones parciales

Como resultado obtenido en la investigación realizada en este capítulo puede concluirse que:

- Con el estudio realizado a los diferentes sistemas de publicación tanto nacionales como internacionales, queda acentuada la necesidad del desarrollo de una aplicación Android que logre darle solución a la situación problemática.
- Con el análisis de las tecnologías informáticas se definió la base tecnológica que se utilizará en el desarrollo del sistema, seleccionando a AUP-UCI como metodología para guiar los pasos del desarrollo, UML como lenguaje de representación visual y Visual Paradigm 8.0 como herramienta CASE para el modelado del sistema. Se escogió Java 8.0 como lenguaje para el desarrollo y para el trabajo con base de datos de la

aplicación, la herramienta SQLite. Se definió como entorno de desarrollo integrado Android Studio en su versión 2.1.3.

Capítulo 2: Características del sistema de publicación de contenido web desde un dispositivo móvil con el sistema operativo Android.

En el presente capítulo se describen los aspectos claves relacionados con el diseño del sistema. Se representa el modelo conceptual con el objetivo de capturar y expresar el conocimiento adquirido en el área analizada antes de realizar el diseño. Se definen los requisitos funcionales y no funcionales que el sistema debe cumplir para satisfacer las necesidades del cliente. También se explican los diagramas y artefactos referentes al comportamiento de la aplicación, así como la arquitectura facilitando de esta manera una mayor comprensión del sistema a desarrollar.

2.1 Propuesta de solución

Para darle solución al problema de investigación se definió el desarrollo de un sistema que ofrezca servicio de publicación de contenido *web* haciendo uso de tecnología móvil con sistema operativo Android. La aplicación debe brindarle al usuario la facilidad de gestionar contenido *web* de manera *offline,* a través de la creación, edición, eliminación y muestra del contenido. Con este sistema se le ofrecerá la oportunidad de interactuar con el contenido *web* en todos los momentos del día, por su portabilidad. Les presentará a los usuarios dedicados a la publicación de contenido*,* una herramienta para realizar cómodamente su trabajo, permitiéndoles que redacten el contenido en el momento que sucede la noticia, tomarles fotos, videos y grabaciones a los hechos. Es una manera cómoda, organizada y eficiente de revisar sus redacciones sin necesidad de publicarlas al momento.

Una vez que el usuario considere que su contenido se encuentre en correcto estado para publicarlo, accede a una interfaz de autenticación, donde debe introducir el usuario, la contraseña y la dirección del sitio donde enviará el contenido *web* previamente seleccionado, procedimiento que se debe llevar a cabo con una conexión *wifi*. Para utilizar la aplicación el sitio donde se publicará la información debe tener implementado un servicio *web* para este fin y el usuario tener una cuenta con los privilegios necesarios para realizar esta acción. Cuando se envía la información el usuario tendrá conocimiento de la correcta llegada de los datos o si ha

existido un problema, si la respuesta es satisfactoria, el contenido se mantiene en el dispositivo móvil permitiendo la gestión del mismo.

2.1 Modelo de dominio

Un modelo de dominio o modelo conceptual como también se conoce, es una representación visual de las clases conceptuales u objetos del mundo real en un dominio de interés. Permite comunicar los términos importantes y las relaciones entre ellos, además puede ser tomado como el punto de partida para el diseño del sistema. El objetivo del modelado del dominio es comprender y describir las clases más importantes dentro del contexto del sistema. Estos modelos son construidos con las reglas de UML durante la fase de concepción, en la tarea de construcción del modelo de dominio, presentado como uno o más diagramas de clases y que contiene, no conceptos propios de un sistema de software sino de la propia realidad física (Larman, 2017).

A continuación se presenta el diagrama de modelo de dominio, en el que se muestra las relaciones existentes entre los principales conceptos asociados al sistema de publicación de contenidos, para de esta manera unificar el vocabulario entre los usuarios y los desarrolladores.

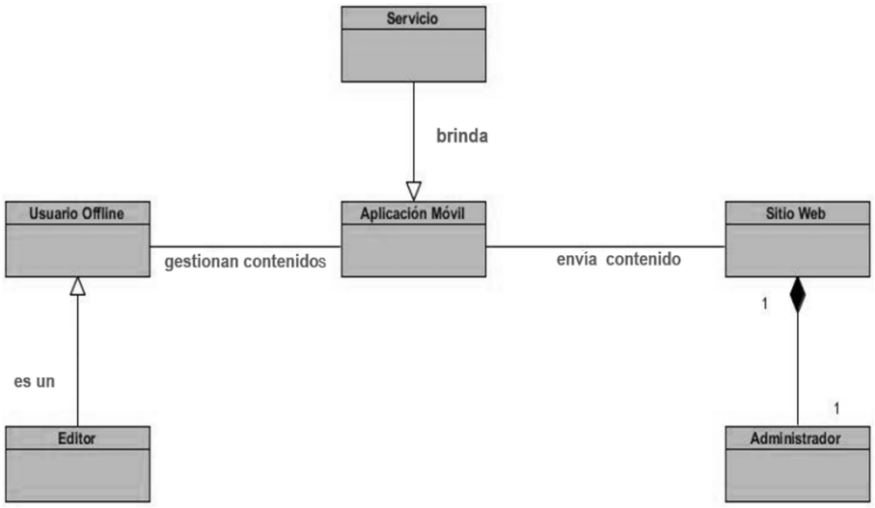


Figura 1: Diagrama de modelo de dominio.

2.2.1 Descripción de las clases del Modelo de dominio

Los usuarios *offline*, entre los que se encuentran los editores, tienen la oportunidad de gestionar el contenido que desean publicar desde la aplicación móvil consumiendo los servicios que le brinda este dispositivo, para luego que se establezca una conexión con el sitio, los editores seleccionen los datos que van a enviar para que sean publicados y el administrador del sitio decide si lo publica o no.

Sitio Web: Conjunto de componentes que ofrecen diferentes tipos de contenidos y servicios.

Servicio: Conjunto de funcionalidades que permiten gestionar el contenido a publicar, con el objetivo de satisfacer las necesidades de los usuarios.

Aplicación Móvil: Combinación de servicios, que le brindan al usuario la facilidad de gestionar el contenido que desean publicar.

Usuario *Offline*: Usuario que interactúa con la aplicación móvil para gestionar el contenido que desea publicar sin tener conexión.

Editor: Usuario que ya se encuentra registrado y posee privilegio de gestionar contenido.

Administrador: Usuario con privilegios para interactuar con las funcionalidades del sitio *web* y para administrarlo.

2.2 Levantamiento de requisitos

El levantamiento de requisitos o requerimientos es la captura de las características que determinará lo que hará el sistema, definiendo las restricciones de sus operaciones e implementación. Este proceso se inicia con continuas interacciones con el cliente, permitiendo establecer una relación e identificación de las necesidades del usuario, además de describir las condiciones necesarias para que los requisitos puedan cumplirse. Estos requerimientos deben ser especificados por escrito, precisos y deben pensarse como las características que convertirán el producto atractivo, usable, rápido o confiable. Los requisitos pueden ser funcionales o no funcionales (Esteller, 2012).

Los requerimientos funcionales son declaraciones de los servicios que debe proporcionar el sistema, de la manera que este debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. En algunos casos también pueden declarar explícitamente lo que el sistema no debe hacer. Los requerimientos no funcionales son aquellos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes como la fiabilidad, tiempo de respuesta y la capacidad de almacenamiento (Decsai, 2017).

2.2.1 Requisitos funcionales y requisitos no funcionales

Luego de definir los principales conceptos relacionados con el dominio, se identificaron durante la entrevista realizada a los clientes pertenecientes al departamento SENIT, 15 requisitos funcionales y 11 requisitos no funcionales, de acuerdo con el objetivo planteado al inicio de esta investigación.

Tabla # 2: Requisitos funcionales y requisitos no funcionales.

Fuente: (Elaboración propia)

Requisitos Funcionales		
Código	**Descripción de requisitos de la Aplicación**	**Prioridad**
RF.1	Crear contenido.	Alta
RF.2	Eliminar contenido.	Alta
RF.3	Editar contenido.	Alta
RF.4	Mostrar contenido.	Alta
RF.5	Guardar contenido.	Alta
RF.6	Mostrar listados de contenidos creados.	Media
RF.7	Eliminar múltiples contenidos.	Alta
RF.8	Adjuntar imagen.	Alta
RF.9	Adjuntar video.	Alta

RF.10	Adjuntar audio.	Alta
RF.11	Establecer conexión.	Alta
RF.12	Autenticar usuario.	Alta
RF.13	Publicar contenido.	Alta
RF.14	Notificar la correcta publicación del contenido.	Alta
Requisitos No Funcionales		
	Usabilidad	
RNF.1	La aplicación móvil debe permitir acceder a sus diferentes interfaces con una profundidad máxima de 2 clics.	
RNF.2	El sistema debe poseer interfaces amigables, con botones que tengan nombres sugerentes para que usuarios inexpertos puedan interactuar fácilmente con el *software*.	
	Eficiencia	
RNF.3	La aplicación debe responder en un tiempo menor de 3 segundos a las solicitudes de los usuarios.	
RNF.4	La aplicación debe notificar al usuario en un tiempo menor o igual a un 1 minuto la respuesta generada al enviar el contenido *web*.	
	Escalabilidad	
RNF.5	El sistema debe ser escalable permitiendo que se le puedan agregar nuevas funcionalidades sin afectar las anteriores implementadas.	
	Hardware	
RNF.6	El dispositivo móvil debe contar como mínimo con 3 megas de almacenamiento que la aplicación sea instalada y funcione correctamente.	

	Software
RNF.7	El dispositivo móvil debe tener sistema operativo Android 4.1 o superior para poder utilizar la aplicación móvil.
	Seguridad
RNF.8	Los errores de autenticación deben mostrar la menor cantidad de detalles posible, para evitar brindar información que comprometa la seguridad e integridad del sistema.
RNF.9	El envío del contenido desde la aplicación hacia el sitio debe realizarse a través del protocolo HTTPS siempre y cuando el sitio posea certificado de SSL[3].
	Diseño o Implementación.
RNF.10	Como lenguaje de programación para el desarrollo de la aplicación se deberá utilizar Java 8.0.
RNF.11	Para utilizar la aplicación el sitio en el que se va a publicar el contenido *web* tiene que tener implementado un servicio *web*.

2.3 Historias de usuarios

Para especificar los requisitos del sistema identificado se emplean las historias de usuario con el objetivo de describir las salidas necesarias, características y funcionalidades del sistema a desarrollar. Cada una de estas historias es descrita por el cliente, colocada en una tarjeta y en cualquier momento es posible escribir nuevas historias de usuario.

Cada historia de usuario debe ser lo suficientemente comprensible y delimitada para que se pueda implementar en pocas semanas y no superar el tamaño de una iteración, que es el tiempo estimado para

[3] Protocolo diseñado para permitir que las aplicaciones web transmitan información de manera segura.

una entrega de un componente de desarrollo de manera incremental. También son utilizadas para poder crear las pruebas de aceptación (Pressman, 2010).

Una vez identificados los requisitos se establecieron las prioridades del negocio: alta cuando las funcionalidades son esenciales para el negocio, medias cuando son importantes pero no influyen en el desarrollo del negocio y baja cuando su ausencia no perjudica al sistema.

A continuación, en las tablas 4 y 5 se muestran las historias de usuario para crear contenido y mostrar listado de contenidos creados. El resto de las historias de usuario se encuentran descritas en los anexos (ver Anexo 2).

Tabla # 3: HU_Crear contenido.

Fuente: (Elaboración propia)

Historia de usuario	
Número: HU_1	**Nombre Historia de Usuario:** Crear contenido.
Prioridad en negocio: Alta	
Descripción: Comienza cuando el usuario selecciona la opción Crear (icono con el símbolo de más), en la interfaz principal. Se muestra una nueva interfaz con los campos título, fecha, resumen, contenido, autor, fuente, palabras clave, además de la opción de adjuntar imagen, video y audio; luego de redactar el contenido, se selecciona la opción guardar y queda creado el nuevo contenido.	
Observaciones: El nuevo contenido no puede crearse con ningún campo vacío. La opción de adjuntar video, imagen y sonido no es obligatoria.	
Prototipo:	

Tabla # 4: HU_Mostrar listado de contenidos creados.

Fuente: (Elaboración propia)

Historia de usuario	
Número: HU_6	**Nombre Historia de Usuario:** Mostrar listado de contenidos creados
Prioridad en negocio: Media	
Descripción: Muestra un listado de los contenidos creados en la interfaz principal, los cuales también pueden ser seleccionados por los usuarios para gestionarlos posteriormente.	
Prototipo:	

2.4 Estilo arquitectónico

Un estilo arquitectónico es una transformación que se impone al diseño de todo sistema. Cada estilo arquitectónico describe una categoría del sistema que contiene: un conjunto de componentes, conectores, restricciones y modelos semánticos. El sistema construido para la publicación de contenido desde dispositivos móviles con el sistema operativo Android también usa uno de muchos estilos arquitectónicos.

Como se ilustra en la figura 2, Android está dividida en cuatro capas, facilitando la descomposición de problemas en varios niveles de abstracción. En las siguientes líneas se dará una visión global de cada capa que posee la aplicación desarrollada, donde cada una ellas utiliza servicios ofrecidos por los niveles anteriores y están basadas en *software* libre (Blanco, 2009).

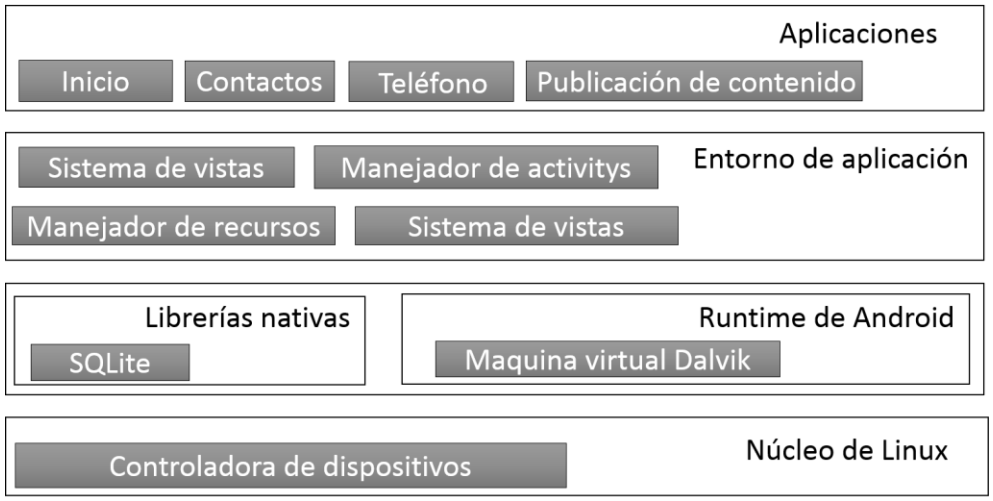


Figura 2: Arquitectura de la aplicación Android.

Núcleo Linux: contiene los controladores de dispositivos de bajo nivel para los distintos componentes del hardware de un dispositivo Android. Por lo tanto, en esta capa se incluyen los servicios de seguridad, gestión de memoria, gestión del procesador, entre otros. Actúa como capa de abstracción, siendo esta la única dependiente del *hardware* y su núcleo está formado a partir del sistema operativo *Linux* versión 2.6.

Librerías nativas: contiene las librerías utilizadas por Android. Junto al núcleo basado en Linux, estas librerías constituyen el corazón de Android.

***Runtime* de Android**: en la misma capa que las librerías se encuentra el tiempo de ejecución Android (*runtime*) que contiene las bibliotecas del núcleo que permite que los desarrolladores ejecuten sus aplicaciones Android utilizando Java. Incluye la máquina virtual *Dalvik*, con la que cada aplicación se puede ejecutar en un proceso independiente, con una instancia de esta máquina virtual.

Entorno de aplicación: esta capa ha sido diseñada para simplificar la reutilización de componentes. Las aplicaciones pueden publicar sus capacidades y otras pueden hacer uso de ellas. Representa fundamentalmente el conjunto de herramientas de desarrollo de cualquier aplicación.

Aplicaciones: en el nivel superior de la arquitectura se encuentran las aplicaciones que se integran en el dispositivo Android. Este nivel está formado por el conjunto de aplicaciones instaladas en una máquina Android. Todas las aplicaciones han de correr en la máquina virtual *Dalvik* para garantizar la seguridad del sistema. Todas estas aplicaciones utilizan los servicios, las API y librerías de los niveles anteriores.

2.5 Patrones de diseño

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí, adaptada para resolver un problema de diseño general en un contexto particular. Proponen una forma de reutilizar experiencias de los desarrolladores, para ello clasifican y describen formas de solucionar problemas frecuentes durante el desarrollo de *software* (Pressman, 2010). En otras palabras, brindan una solución ya probada y documentada a problemas de desarrollo de *software* que están sujetos a contextos similares.

En el desarrollo de la propuesta de solución se tuvo en cuenta la utilización de los Patrones Generales de Software para Asignación de Responsabilidades (GRASP, por sus siglas en inglés) y los patrones *Gang of Four* (*GoF*, por sus siglas en inglés).

2.5.1 Patrones GRASP

Se encuentran enfocados en los principios fundamentales de asignación correcta de responsabilidades en el diseño orientado a objetos. Describen los patrones con que cuenta la solución y su aplicación en el desarrollo de los componentes. A continuación se muestran los patrones utilizados y un ejemplo de cómo se evidencia en la aplicación:

Creador: su implementación permite identificar quién debe ser el responsable de la creación o instanciación de nuevos objetos o clases. Dicho de otra manera este patrón guía la asignación de responsabilidades relacionadas con la creación de objetos, con lo que se logra menos dependencia y mayores oportunidades de reutilización de código (Visconti, 2012). En el desarrollo de la solución planteada se trabaja con clases que utilizan esta función, como es la clase CrearContenido y se muestra un ejemplo en la Figura 3.

```
Calendar calendar = new GregorianCalendar();
date = calendar.getTime();
SimpleDateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy");
fechhha.setText(dateFormat.format(date));
```

Figura 3: Ejemplo de utilización del patrón Creador.

Controlador: este patrón tiene como objetivo asignar la responsabilidad a una clase de recibir o manejar un mensaje de evento del sistema generado por un actor externo, por lo general a través de una interfaz

gráfica de usuario a la que accede una persona para realizar ciertas operaciones en el sistema (Bustacara, 2017).

```
if (((ti.isEmpty() || C.isEmpty() || r.isEmpty() || a.isEmpty() || fue.isEmpty()) || pa.isEmpty()) || (resultado == "Es menor ")) {
    if ((ti.isEmpty() || C.isEmpty() || r.isEmpty() || a.isEmpty() || fue.isEmpty() || pa.isEmpty())) {
        Toast.makeText(this, "Debe completar todos los campos", Toast.LENGTH_SHORT).show();
    } else
        Toast.makeText(this, "Verifique la fecha de publicación", Toast.LENGTH_SHORT).show();
```

Figura 4: Ejemplo de utilización del patrón Controlador.

Experto: este patrón plantea que se debe asignar una responsabilidad al experto en información, en otras palabras, a la clase que cuenta con los datos necesarios para cumplir la responsabilidad. De esta forma se conserva el encapsulamiento de la información (Visconti, 2012). En la figura 5 se muestra el uso de este patrón con el que se pretende que los objetos realicen las acciones relacionadas con la información que posee.

```
BdContenido connector = new BdContenido(this);
try {
    connector.open();
    connector.InsertarBaseDato(objeto);
    connector.close();
    finish();
} catch (Exception e) {
    e.printStackTrace();
    Log.w("ERROR", "Insert publication into Database FAILED: " + e.getMessage());
    Toast.makeText(this, "Error al adicionar publicación", Toast.LENGTH_SHORT).show();
}
```

Figura 5: Ejemplo del empleo del patrón Experto.

Polimorfismo: se emplea para determinar el tipo de comportamiento específico en cada una de las clases, asignando responsabilidades de comportamiento a la misma clase utilizando operaciones polimórficas.

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == Variable_Camara && resultCode == RESULT_OK) {
        try {
            imag.setImageBitmap(BitmapFactory.decodeFile(localizacion));
            picture = localizacion;
            imag.setScaleType(ImageView.ScaleType.CENTER_CROP);
        } catch (Exception e) {
            Toast.makeText(this, "No se puede capturar Imagen", Toast.LENGTH_SHORT).show();
        }
    }
```

Figura 6: Ejemplo de Polimorfismo.

Alta cohesión: caracteriza a las clases con responsabilidades estrechamente relacionadas, que colaboran entre sí y con otros objetos para simplificar su trabajo. Siguiendo este principio se han diseñado las clases de este sistema y sus relaciones (Bustacara, 2017).

Bajo acoplamiento: principio que se debe tener en cuenta durante las decisiones de diseño. En el diseño de la solución propuesta las clases se encuentran lo menos relacionadas, de tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la re-utilización y disminuyendo la dependencia entre las clases (Bustacara, 2017).

2.5.2 Patrones GoF

Los patrones GoF describen las formas comunes en que diferentes tipos de objetos pueden ser organizados para trabajar unos con otros. Tratan la relación entre clases, la combinación de clases y la formación de estructuras de mayor complejidad (Guerrero, 2013). A continuación se muestran los patrones GoF que se utilizan:

Observador: permite captar dinámicamente las dependencias entre objetos, de tal forma que un objeto notificará a los objetos dependientes de él cuando cambia su estado, siendo actualizados automáticamente (Guerrero, 2013). En la solución propuesta, este patrón es utilizado al crear clases que extiendan de la clase nativa de Android.

```
public class EditarContenido extends AppCompatActivity implements View.OnClickListener {
```

Figura 7: Ejemplo del patrón Observador.

Adaptador: se emplea para generar los elementos de los componentes visuales de la lista de contenidos, que requiere de un adaptador para crear los contenidos que se muestran a los usuarios (Guerrero, 2013). Se evidencia en las clases ContenidoAdapter y SeleccionarAdapter.

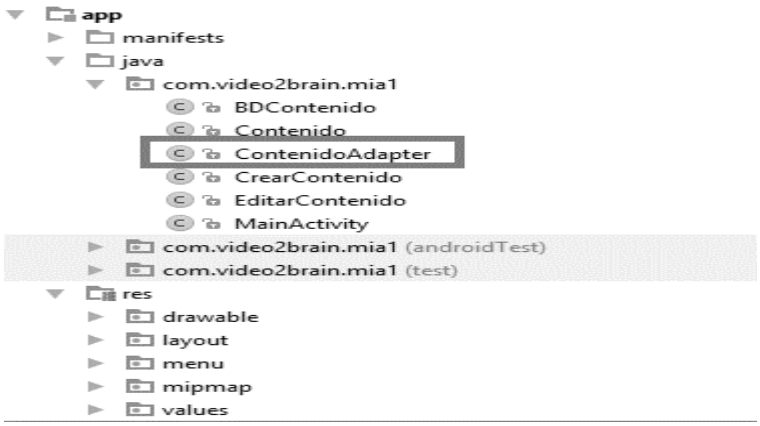


Figura 8: Ejemplo de patrón Adaptador.

2.6 Diagramas de clase del diseño

Los diagramas de clases se utilizan con el propósito de visualizar las relaciones existentes entre las clases que involucran el sistema. Estos diagramas son el pilar básico del modelado con UML, siendo utilizados para mostrar tanto lo que el sistema puede hacer como para mostrar cómo puede ser construido. A continuación se representa el diagrama de clases del sistema:

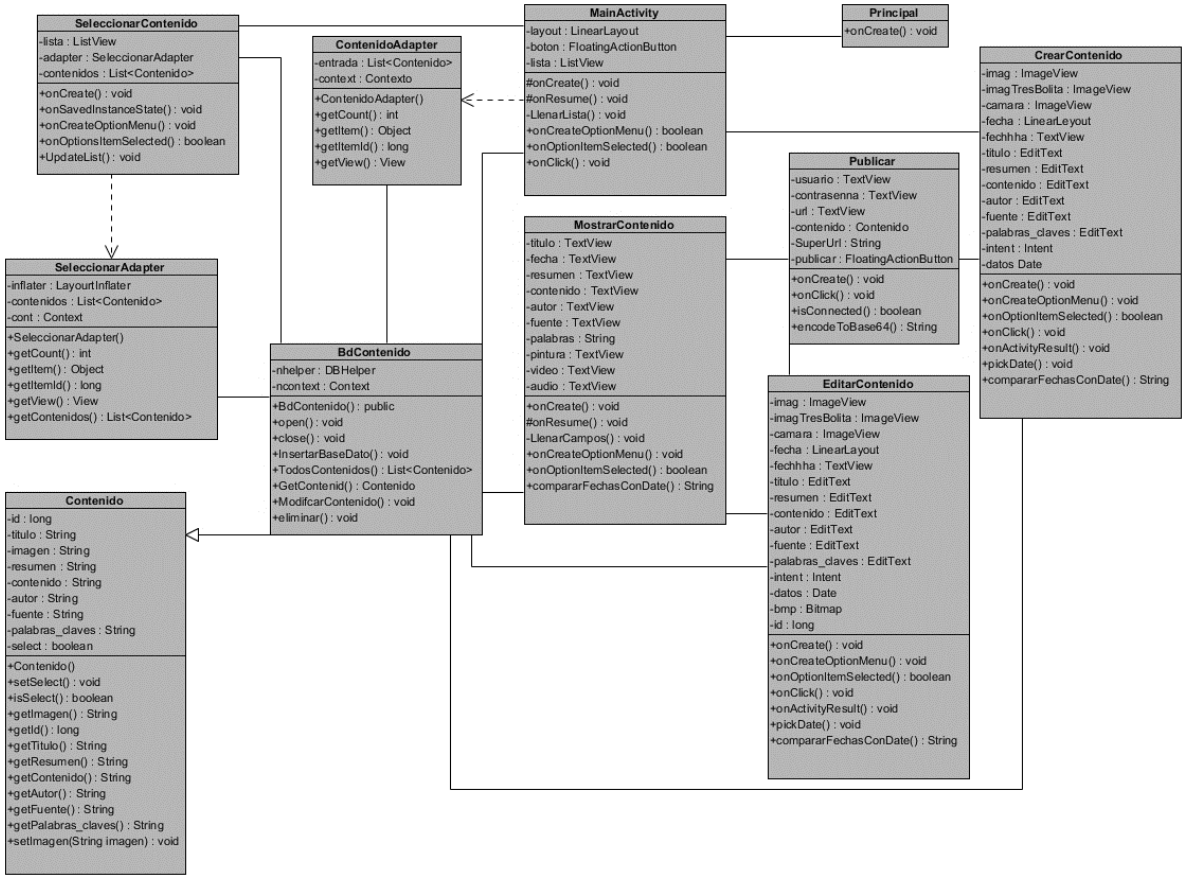


Figura 9: Diagrama de clases de la aplicación Android.

2.7 Modelo de datos

Para el desarrollo del sistema de publicación de contenido desde un dispositivo móvil se definió el siguiente modelo de datos apoyado en el diagrama Entidad-Relación, el cual define todos los datos que se introducen.

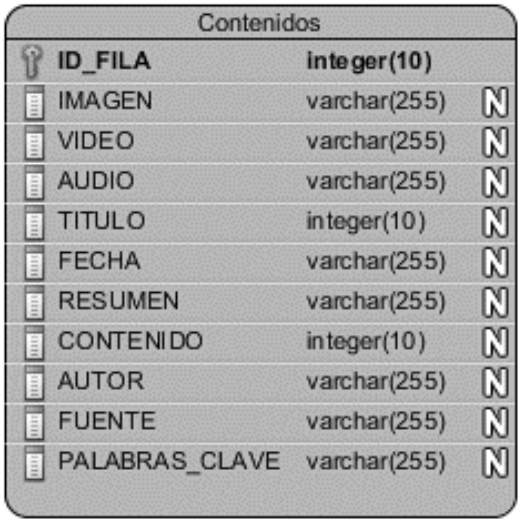


Figura 10: Modelo de datos.

La base de datos del sistema contiene una tabla nombrada Contenidos, en la cual se almacenará y se transformará el contenido que se genere dentro de la aplicación. La columna ID_FILA es el identificador del contenido, los campos IMAGEN, VIDEO y AUDIO almacena la dirección de donde se encuentra ubicada la imagen, el video y el audio. La columna FECHA hace referencia a la fecha en la que el contenido fue creado, el RESUMEN, CONTENIDO, AUTOR, FUENTE y PALABRAS CLAVE, contiene los datos que el usuario introduce, el resumen, el contenido, la fuente de donde obtuvo la información, las palabras claves del contenido a publicar y el autor de la noticia.

2.8 Modelo de despliegue

Un diagrama de despliegue representa la relación física que se establece entre los distintos componentes o nodos que describen la topología de un sistema. Detalla las capacidades de red, las especificaciones del servidor, los requisitos de *hardware* y la información relacionada con la forma en la que los componentes se comunicarán a lo largo de la infraestructura del sistema. También se utiliza para visualizar la distribución de los componentes de *software* en los nodos físicos. El mismo está compuesto por: nodos, dispositivos y conectores (Pressman, 2010).

El diagrama de despliegue propuesto está compuesto por 3 nodos y muestra cómo será desplegada la aplicación: **Dispositivo Móvil** representa el móvil con sistema operativo Android de los usuarios que se

conectarán para publicar en un sitio indicado, realizando una conexión al **Servidor Web (Apache)** a través del protocolo *HTTP*, para aquellos sitios que no posean *SSL* y *HTTPS* para los que sí. Este a su vez realiza las conexiones al **SGBD (MySQL)** mediante el protocolo TCP.

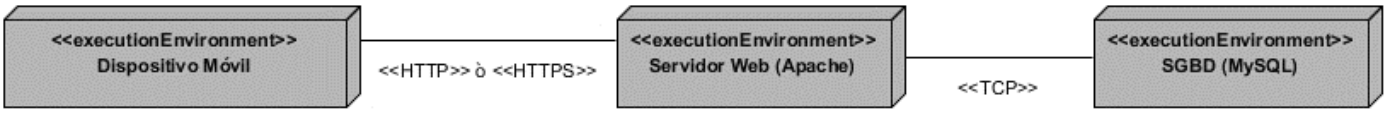


Figura 11: Diagrama de despliegue.

2.9 Conclusiones parciales

Con el estudio de los temas referentes al análisis y diseño de la propuesta del Sistema para la publicación de contenido *web* desde dispositivo móvil con el sistema operativo Android se pudo llegar a las siguientes conclusiones:

- Los requisitos funcionales y no funcionales obtenidos a partir del levantamiento de requisitos garantizan que la propuesta de solución responda a las necesidades planteadas por el cliente.
- Los artefactos generados según la metodología AUP-UCI, la arquitectura y los patrones de diseño crearon las bases necesarias para la construcción de la propuesta de solución.

Capítulo 3: Construcción y pruebas del sistema para la publicación de contenido web desde un dispositivo móvil con el sistema operativo Android.

En el presente capítulo se caracterizan los elementos definidos en el proceso de desarrollo, para obtener un producto exitoso, teniendo en cuenta cada uno de los atributos de calidad. Se describen los elementos físicos del sistema y sus relaciones a través del diagrama de componentes. Se exponen los estándares de codificación utilizados para el desarrollo del sistema y por último se ejecutan las pruebas de *software* para verificar el correcto funcionamiento de la aplicación.

3.1 Diagrama de componentes

Un diagrama de componente describe los elementos físicos del sistema y sus relaciones. Está conformado por componentes y paquetes, donde se muestra la estructura del sistema en términos de implementación a un alto nivel. Un componente de *software* representa una parte de un sistema modular, desplegable y reemplazable que se encuentra en la computadora. Estos son utilizados para modelar la vista estática y dinámica de un sistema. Su principal punto es el potencial que tienen para volver a ser utilizados (Stevens, 2007).

A continuación se describe los componentes del diagrama:

Paquete *layout*: cada uno de los componentes que contiene este paquete, pertenecen a las interfaces que son creadas en el editor gráfico de Android Studio, las cuáles son visualizadas en la aplicación y vinculadas con las clases Javas que contienen las funcionalidades que debe realizar el sistema.

MainActivity.java: contiene la funcionalidad de mostrar los contenidos que han sido guardados por el usuario, brindando la oportunidad de crear, mostrar y seleccionar múltiples contenidos.

CrearContenido.java: con este componente se puede crear un nuevo contenido, el cual puede ser publicado o guardado en la base de datos.

EditarContenido.java: permite seleccionar un contenido para editarlo en caso de que se haya producido un error al crearlo, para publicarlo o guardarlo nuevamente en la base de datos.

MostrarContenido.java: posibilita mostrar los datos del contenido que ha sido seleccionado por el usuario.

Selection.java: ofrece la funcionalidad de seleccionar múltiples contenidos para realizar cualquier acción de publicación o eliminación.

ContenidoAdapter.java: este componente actúa como un puente entre el MainActivity.java y los datos de la vista; proporcionando acceso a cada elemento de la lista y otorgándole una vista a cada producto en el conjunto de datos.

SeleccionarAdapter.java: actúa como un puente entre el Selection.java y los datos de la vista; proporcionando acceso a cada elemento de la lista y otorgándole una vista a cada producto en el conjunto de datos.

SQLite.java: es el componente que contiene todo lo relacionado con la base de datos, los métodos que posibilitan el proceso de gestión de contenidos.

Paquete HttpURLConnection: este paquete está compuesto por la clase Javas que permite establecer la conexión con el sitio en el cual se va a publicar. La clase Publicar.java se encuentra vinculada a los componentes EditarContenido.java, CrearContenido.java y MostrarContenido.java los cuales envían el contenido a publicar, esta clase lo recibe y lo envía hacia el sitio.

En la figura 12 se muestra el diagrama de componentes del sistema para una mayor comprensión del mismo:

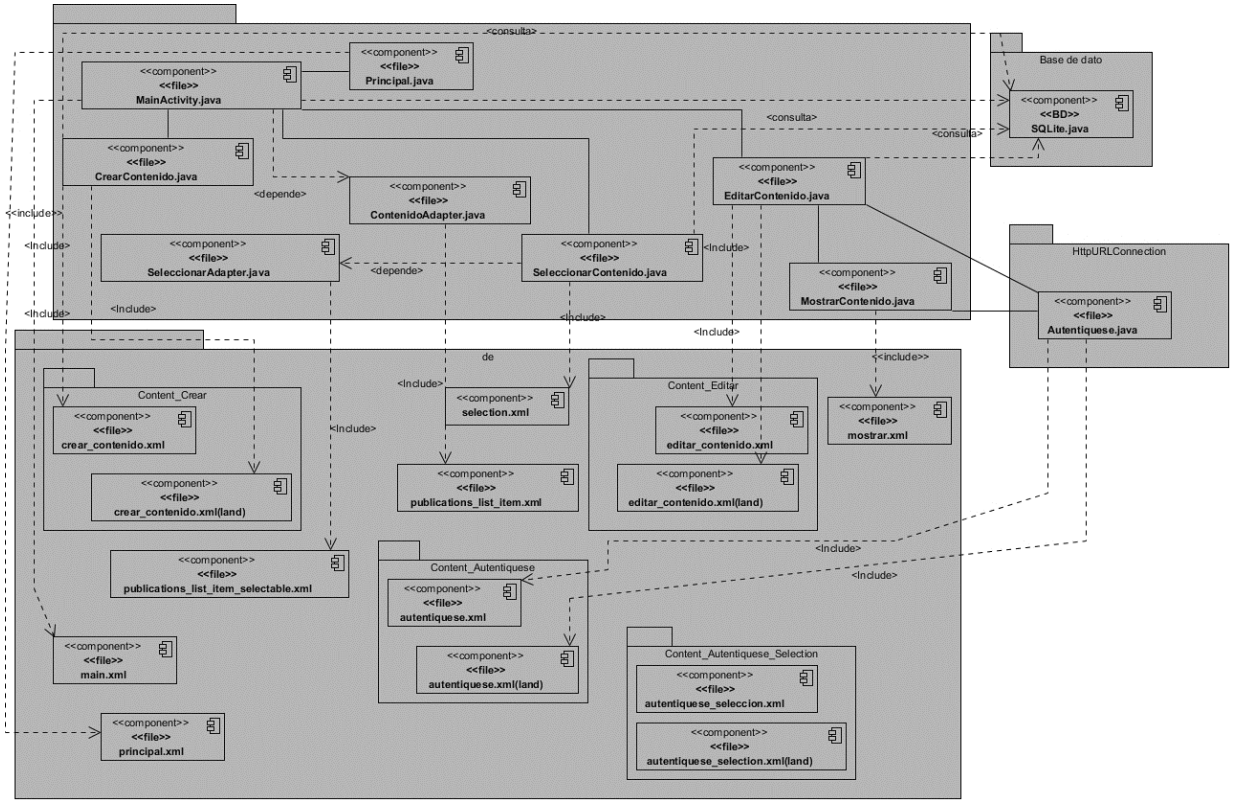


Figura 12: Diagrama de componente.

3.2 Estándares de codificación

Los estándares de codificación son un conjunto de reglas o patrones de codificación a seguir por los desarrolladores con el objetivo de establecer un orden y un formato común en el código fuente del sistema en desarrollo. Estos estándares deben cumplir con dos principios: legibilidad y mantenibilidad, lo cual repercute en lo bien que el programador entienda el código y la facilidad con que el sistema puede modificarse (Microsoft, 2016). Con el propósito de estandarizar las nomenclaturas en la implementación de los componentes y obtener un buen producto, se utilizan los siguientes estándares de codificación:

- **Asignación de nombres:** cada tipo de elemento debe nombrarse con una serie de reglas determinadas.

Clases, Métodos, Variables

Para nombrar estos elementos se utiliza el mecanismo CamelCase, que indica que para todos los nombres simples la primera letra debe de ser Mayúscula y si son varias palabras se debe de intercalar entre mayúsculas y minúsculas (García, 2011).

```
public class MainActivity extends AppCompatActivity implements View.OnClickListener {

    private ListView Lista;
    private RelativeLayout Layout;
    private FloatingActionButton Boton;

    private void LlenaLista() throws Exception {
        BdContenido connector = new BdContenido(this);
        connector.open();
        List<Contenido> contenidos = connector.TodosContenidos();
        connector.close();
        if (contenidos.size() > 0) {
            Lista.setAdapter(new ContenidoAdapter(this, contenidos));
            Lista.setVisibility(View.VISIBLE);
            Layout.setVisibility(View.GONE);
        }
        else {
            Lista.setVisibility(View.GONE);
            Layout.setVisibility(View.VISIBLE);
        }
    }
```

Figura 13: Fragmento del código fuente de la clase MainActivity.

Constantes

Los nombres de constantes deberían escribirse todo en mayúsculas con las palabras separadas por subrayados ("_"). Todas serán declaradas como *public static final.*

```
private static final int Variable_Camara = 2;
public static final String VARIABLE_ID_CONTENODO = "id_contenido";
```

Figura 14: Fragmento del código fuente de la clase EditarContenido.

- **Número de declaraciones por línea:** se debe declarar cada variable en una línea distinta, de esta forma cada variable se puede comentar por separado.

```
//Declaracion de los componetes visuales
private ImageView imag;
private LinearLayout fecha;
private TextView fechhha;
```

Figura 15: Fragmento de código de la clase CrearContenido.

- **Añadir un único espacio a ambos lados de operadores como: =, ==, != , etc.**

```
if (savedInstanceState != null) {

    picture = savedInstanceState.getString("IMAGEN");
    if (picture == null) {
        imag.setScaleType(ImageView.ScaleType.FIT_CENTER);
        imag.setImageDrawable(getResources().getDrawable(R.drawable.ic_crear));
```

Figura 16: Ejemplo de este estándar de codificación en la clase CrearContenido.

- **Sentencias**

Las sentencias pertenecientes a un bloque de código estarán ubicadas a un nivel más a la derecha con respecto a la sentencia que las contiene. El carácter inicio de bloque "{" debe situarse al final de la línea que inicia el bloque y "}" debe situarse en una nueva línea tras la última línea del bloque y alineada con respecto al primer carácter de dicho bloque. Todas las sentencias de un bloque deben encerrarse entre llaves "{...}", aunque el bloque contenga solo una sentencia. Esta práctica permite añadir código sin cometer errores accidentalmente al olvidar añadir las llaves.

```
private void LlenaLista() throws Exception {

    BdContenido neda = new BdContenido(this);
    neda.open();
    List<Contenido> publication = neda.TodosContenidos();
    neda.close();

    if (publication.size() > 0) {
        lista.setAdapter(new ContenidoAdapter(this, publication));
        lista.setVisibility(View.VISIBLE);
        layout.setVisibility(View.GONE);
    }
    else {
        lista.setVisibility(View.GONE);
        layout.setVisibility(View.VISIBLE);
    }

}
```

Figura 17: Fragmento de código de la clase MainActivity

3.3 Pruebas de software

Según Navarro (2014) las pruebas de software comprenden una fase del proceso de desarrollo que se centra en asegurar la calidad, fiabilidad y robustez de un software, dentro de un contexto o escenario donde está previsto que este sea utilizado. Se encuentra encaminado a medir el cumplimiento de las funcionalidades establecidas por el cliente, reduciendo de esta manera el número de errores no detectados. Esta fase es una de las más costosas del ciclo de vida del *software*. En sentido estricto, deben realizarse pruebas a todos los artefactos generados durante la construcción de un producto, lo que incluye especificaciones de requisitos, casos de uso, diagramas de diversos tipos y, por supuesto, el código fuente y el resto de productos que forman parte de la aplicación.

La estrategia a seguir para la realización de las pruebas a la aplicación móvil implementada, contempla cuatro niveles de pruebas: pruebas Unitarias, de Aceptación, de Integración, de Compatibilidad.

3.3.1 Pruebas Unitarias

Las pruebas unitarias se realizan para controlar el funcionamiento de pequeñas porciones de código como métodos o clases. Generalmente son realizadas por los mismos programadores puesto que al conocer con mayor detalle el código, se les simplifica la tarea de elaborar conjuntos de datos de prueba para testearlo. Por último, es importante que las funcionalidades de cada componente unitario sea cubierta, por al menos, dos casos de prueba, los cuales deben centrarse en probar al menos una funcionalidad positiva y una negativa (Myers, 2011).

Las pruebas de unidad siempre están orientadas a caja blanca y le permiten al programador conocer si determinada funcionalidad se puede agregar al sistema sin afectar su funcionamiento. Facilitan que el programador cambie el código para mejorar su estructura (refactorización), puesto que permiten hacer pruebas sobre los cambios y asegurarse de que no han introducido errores.

Para la realización de las pruebas unitarias es utilizada la extensión de Android JUnit, que es una parte integral del SDK de Android. A cada una de las clases de la aplicación se le realizaron varias pruebas con el objetivo de obtener y corregir los errores existentes antes de realizar la entrega al cliente.

Resultado de las pruebas unitarias

Entre algunas de las insuficiencias identificadas en las iteraciones se encuentran:

- No se actualizaban los contenidos que eran editados.
- Cuando se creaba un nuevo audio no se generaba la dirección física.
- No se guardaban los contenidos seleccionados cuando se rotaba la pantalla del móvil.
- No se guardaban las fotos que eran tomadas cuando se creaba un nuevo contenido.
- Se detenía la aplicación cuando se trataba de acceder a los videos que se encontraban en la galería.
- Existencia de error cuando se borraban múltiples contenidos.

A estas deficiencias se le dieron solución, como se puede apreciar en siguiente figura, obteniendo finalmente resultados con el éxito esperado en las siguientes etapas de iteración de pruebas.

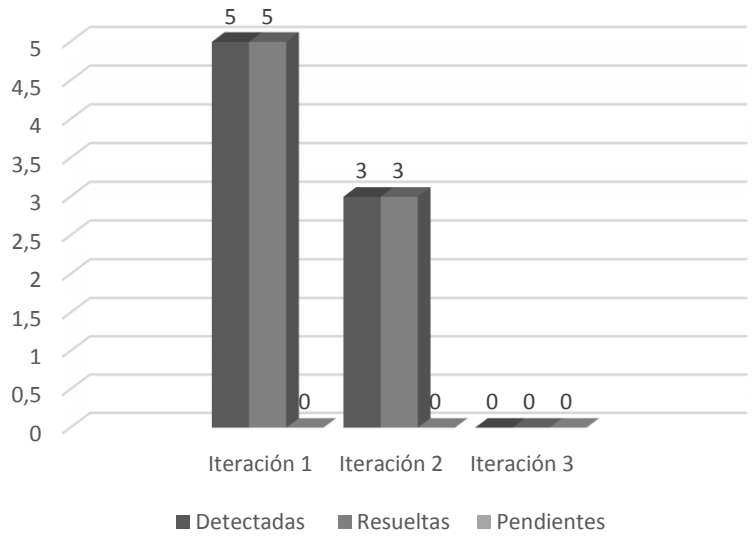


Figura 18: Comportamiento de las no conformidades en las prueba unitarias.

3.3.2 Prueba de Aceptación

Las pruebas de aceptación son básicamente pruebas funcionales sobre el sistema completo, ya que tienen como objetivo obtener la aceptación final del cliente antes de la entrega del producto para su utilización. Son realizadas por el usuario final permitiendo la valoración del producto, donde el cliente confirma que las funcionalidades exigidas y descritas en la HU funcionan correctamente. Intentan encontrar errores de las

siguientes categorías: funciones incorrectas o ausentes, errores de interfaz, errores en estructuras de datos o en accesos a bases de datos, errores de rendimiento y errores de inicialización y terminación (Fontela, 2003).

Cuando se realizan este tipo de pruebas, el producto está listo para implantarse en el entorno del cliente. Para su desarrollo se emplearon casos de prueba de aceptación donde se describe el proceso a realizar y en los cuales tuvieron destacada participación miembros del centro para el cual se desarrolla esta aplicación. A continuación se describe uno de los casos de pruebas de aceptación, para el requisito Crear contenido, los demás están accesible en el Anexo 3.

Tabla # 5: Casos de prueba #1. Crear contenido.

Fuente: (Elaboración propia)

Escenario	**Descripción**	**Campos (Imagen, Video, Audio, Título, Fecha, Resumen, Contenido, Fuente, Autor y Palabras Clave)**	**Respuesta del Sistema**	**Flujo central**
EC 1.1 Crear un nuevo contenido de manera válida.	Se selecciona la opción de crear un nuevo contenido. Se llenan los campos y se guardan los contenidos.	Se introducen correctamente todos los valores de los campos que existen en el contenido, sin dejar ningún campo sin completar. Los únicos campos opcionales son	El sistema mostrará una interfaz donde se muestran los campos que se necesita llenar de un contenido para que sea creado. Luego cuando se selecciona la	1. En la interfaz principal se selecciona la opción de crear contenido (ícono con el símbolo de más). **2.** El sistema muestra una interfaz con todos

		imagen, audio y el video.	opción de guardar, el sistema verifica que todos los campos estén llenos, de ser así se almacenará en la base de datos.	los campos necesarios para crear el contenido. 3. El sistema permite capturar una imagen de la propia cámara colocarla como imagen del contenido o seleccionar la imagen de la galería, todo a través del icono de la cámara. 4. El sistema permite adjuntar videos que se encuentren en la galería (ícono de video). 5. La aplicación facilita grabar sonido y guardarlo. 6. Completados todos los campos se guarda el
EC 1.2 Crear un nuevo contenido de manera inválida.	Se selecciona la opción de crear un nuevo contenido.	No se completan correctamente todos los valores de los campos que existen en el contenido, dejando algunos campos sin completar.	El sistema mostrará una interfaz donde se muestran los campos que se necesita llenar de un contenido para que sea creado. Luego cuando se selecciona la opción de guardar, el sistema verifica que todos los campos estén llenos, de no ser así se mostrará un mensaje “Debe llenar todos los campos”.	

				contenido correctamente.

Resultados de las pruebas de aceptación

Para verificar el correcto funcionamiento de las funcionalidades del sistema se realizaron inicialmente dos iteraciones de las pruebas de aceptación, las cuales arrojaron un número de no conformidades, las que posteriormente, fueron corregidas de manera satisfactoria. Los siguientes gráficos muestran el comportamiento de estas pruebas sobre las funcionalidades del sistema y las no conformidades arrojadas por las mismas.

Figura 19: Resumen de errores en las pruebas de aceptación.

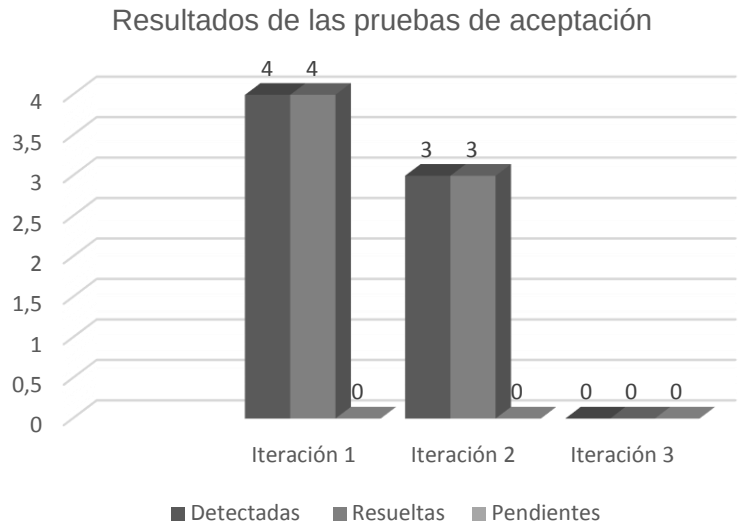


Figura 20: Comportamiento de las no conformidades en las prueba de aceptación.

3.3.3 Pruebas de Integración

Las pruebas de integración tienen como base las pruebas unitarias y consisten en una progresión ordenada de testeos para los cuales los distintos módulos van siendo ensamblados y probados hasta haber integrado el sistema completo. Las mismas validan que estos componentes realmente funcionan juntos, son llamados correctamente y, además, transfieren los datos correctos en el tiempo preciso y por las vías de comunicación establecidas. Si bien se realizan sobre módulos ya probados en forma individual, no es necesario que se terminen todas las pruebas unitarias para comenzar con las de integración. Existen dos tipos de integración: incremental y no incremental (Kostakopoulou, 2011).

Una vez realizada las pruebas de aceptación a la propuesta de solución, donde fue posible verificar el correcto funcionamiento de la aplicación, se hace necesario verificar su correcta integración y compatibilidad a partir de pruebas de integración.

Resultado de las pruebas de integración

Para la ejecución de las pruebas de integración se tuvieron en cuenta diferentes acciones, a continuación se mencionan las más fundamentales:

- Verificación del establecimiento de la conexión.
- Correspondencia entre los datos enviados desde la aplicación y los recibidos en el sitio a publicar.
- Confirmación de la correcta publicación de los datos.

A través de la ejecución de estas pruebas se hizo posible detectar 9 errores entre los que se destacan: error en la obtención de los datos enviados por *post* en el servicio *web* desarrollado en Python, los datos enviados como una cadena de *bytes* como las imágenes, los videos y el audio llegaban incompletos; los campos opcionales para la aplicación móvil eran campos obligatorios en el servicio *web*, existían ocasiones en las que no se recibía notificación de que los datos habían llegado. A continuación se presenta la figura 18 donde se puede apreciar con una mayor precisión el comportamiento de los errores detectados en cada una de las iteraciones.

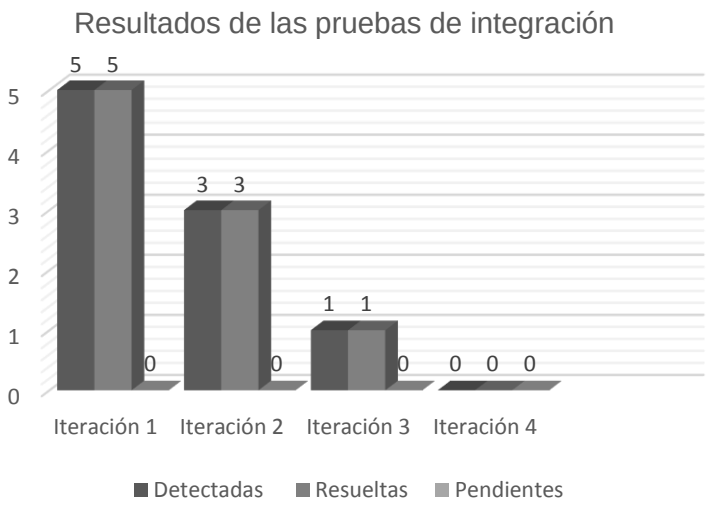


Figura 21: Resultados de las pruebas de integración.

Luego de haber corregido los errores detectados, se realizó una cuarta iteración la cual no arrojó ningún error, lo que permite afirmar la correcta integración de la aplicación móvil con el sitio en donde se va a publicar, logrando de esta manera el buen funcionamiento del sistema en general.

3.3.4 Prueba de Compatibilidad

Las pruebas de compatibilidad son muy significativas para mostrar una calidad adecuada del sistema en diferentes entornos. Se le realizan al *software*, para comprobar que son compatibles con todos los navegadores de *internet* o todos los sistemas operativos. Estas pruebas son realmente importantes para que el producto llegue a todos los usuarios, permitiendo descubrir problemas del sistema antes de que se encuentre en línea (Pressman, 2010).

Para el desarrollo de estas pruebas se utiliza el emulador de Android Studio, Android Virtual Device (AVD), creando diferentes prototipos de teléfonos móviles, con diferentes versiones y resoluciones de pantalla. Se comprobará la compatibilidad de la aplicación para algunos niveles de API superior al valor especificado minSdkVersion[4].

Resultado de las pruebas de compatibilidad.

Este sistema fue desarrollado con una versión de SDK igual a 4.1, lo que posibilita que gran cantidad de usuarios tengan soporte para la aplicación. Con el propósito de saber el grado de compatibilidad de esta aplicación con otros dispositivos móviles, se realizaron cuatro iteraciones, las cuales arrojaron varias no conformidades relacionadas todas al diseño de la aplicación. En las siguientes figuras se muestran cómo se visualizaban las interfaces del sistema en diferentes dispositivos Android y a los dispositivos que no se realizaron las pruebas se garantiza la correcta visualización de la aplicación al utilizar el código <*supports-screens*> en el desarrollo del sistema.

[4] Valor entero que designa el nivel de API mínimo que se requiere para ejecutar la aplicación.

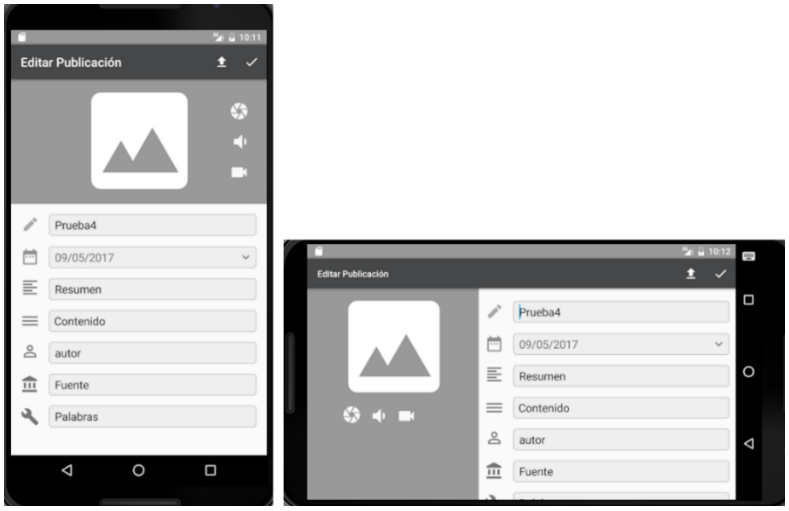


Figura 22: Interfaz de EditarConteido en dispositivos Samsung. Versión de Android 5.7. Resoluciones de pantalla 1440x2560: 560dpi.

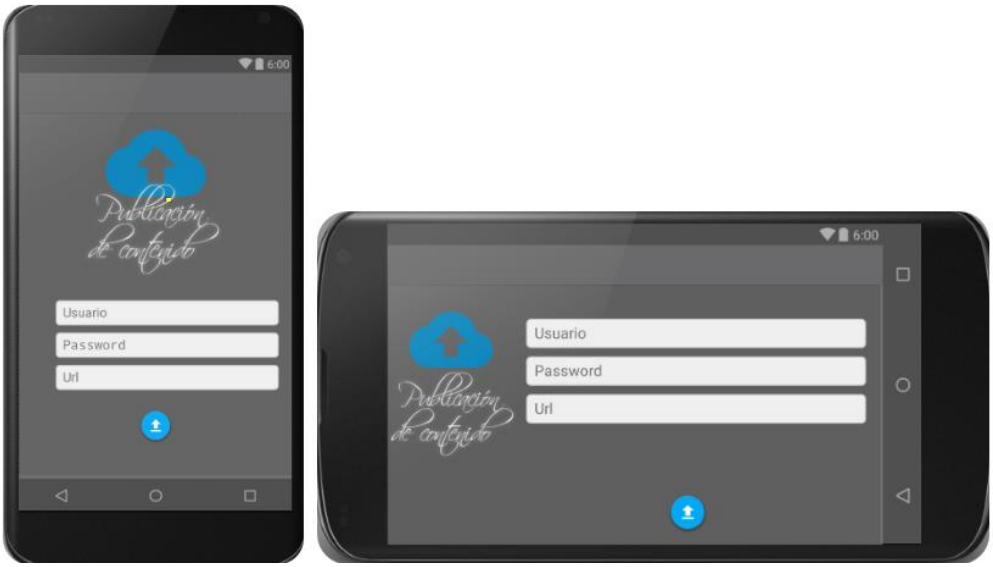


Figura 23: Interfaz de Publicar en dispositivo móvil Nexus 4. Versión de Android 4.7. Resoluciones de pantalla 768x1280: xhdpi.

En la tabla y en el gráfico que a continuación se presenta, se exponen los resultados obtenidos en cada una de las iteraciones, así como la corrección de cada uno de los errores.

Tabla # 6: Resumen de los resultados de la prueba de compatibilidad.

Fuente: (Elaboración propia)

No conformidades	Iteración 1	Iteración 2	Iteración 3	Iteración 4
Detectadas	14	7	3	1
Resueltas	14	7	3	1
Pendientes	0	0	0	0

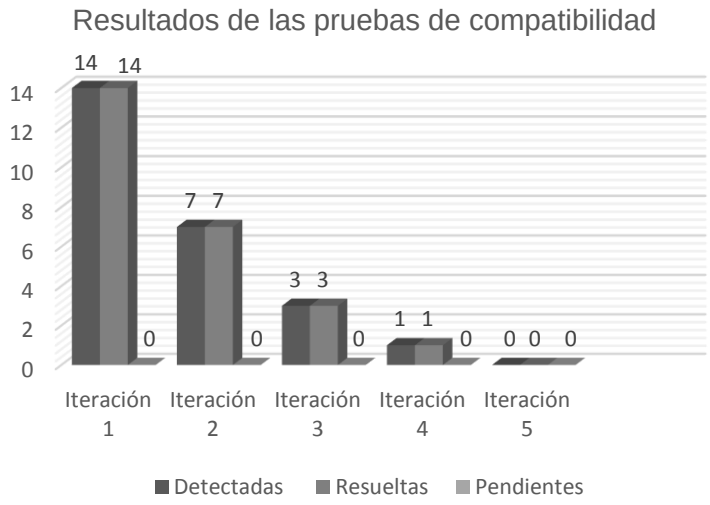


Figura 24: Resultados de la prueba de compatibilidad.

Luego de haber corregido cada una de las no conformidades encontradas, se realizó una última iteración en la cual se obtuvieron resultados satisfactorios, al no encontrarse ningún error de compatibilidad.

3.3.5 Validación de la hipótesis de la investigación

La validación del sistema se realizó utilizando el método de criterio de expertos empleando el escalamiento de Likert y el de satisfacción de usuario utilizando como procedimiento científico para el procesamiento de los resultados la Técnica de IADOV.

3.3.5.1 Validación a través de criterio de experto

El método de criterio de expertos es una validación útil para verificar la fiabilidad de una investigación y obtener valoraciones sobre la propuesta de solución (Cuervo-Martínez, 2008). El criterio de expertos se realizará empleando el escalamiento de Likert. Es una escala psicométrica utilizada principalmente en la investigación de mercados para la comprensión de las opiniones y actitudes de un consumidor. Nos sirve principalmente para realizar mediciones y conocer el grado de conformidad de una persona o encuestado hacia determinada oración afirmativa o negativa.

Para formar parte de este juicio de expertos fueron seleccionados cinco especialistas, sus descripciones pueden verse en el Anexo 5, los cuales cumplieron con los siguientes criterios de selección: título universitario, grado científico, años de experiencia en el área, y nivel de dominio sobre el tema que se encuesta. Los planteamientos enfocados a obtener las valoraciones de los expertos en función de los indicadores definidos fueron:

- Se considera que existe una correcta distribución de la estructura de los campos cuando se crea un nuevo contenido.
- Se tienen en cuenta el posible error cometido por los usuarios al crear un contenido con algún campo vacío.
- La facilidad que se brinda al crear un contenido, le permite al usuario la rápido familiarización con la aplicación.
- La aplicación permite obtener una rápida experiencia de usuario.

Y los indicadores se resumen en: 5- Totalmente acuerdo (TA), 4- De acuerdo (DA), 3- Ni de acuerdo ni en desacuerdo (Sí-No), 2- En desacuerdo (ED) y 1- Totalmente en desacuerdo (TD).

La tabla 8 muestra la distribución de las tendencias múltiples de la aplicación de acuerdo con las afirmaciones identificadas.

Tabla 7: Distribución de las tendencias múltiples.

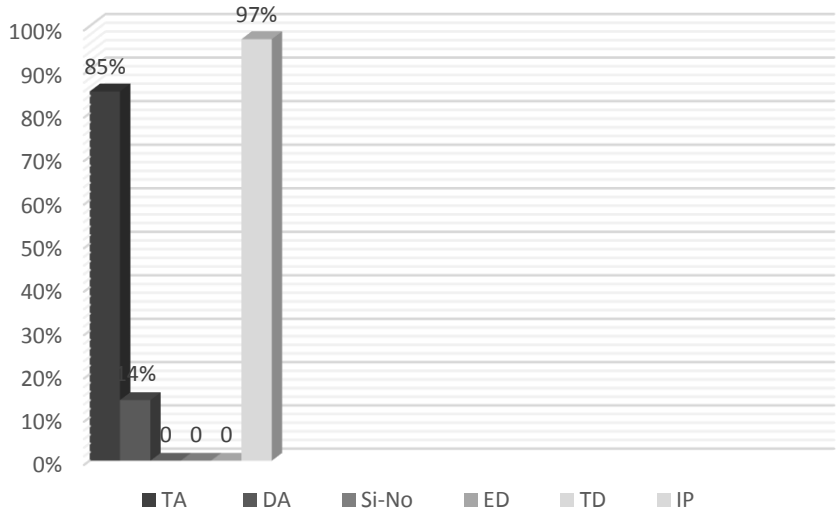
Fuente: (Elaboración propia)

	Afirmaciones realizadas				
Indicadores	A1	A2	A3	A4	Total
TA	5	3	4	5	17
DA	0	2	1	0	3
Si-NO	0	0	0	0	0
ED	0	0	0	0	0
TD	0	0	0	0	0
Total	5	5	5	5	20

Una vez realizada la encuesta y obtenido los porcientos de concordancia de los expertos con cada una de las respuestas para los planteamientos formulados, mostrados en el Anexo 8, se calcula un índice porcentual (Ramírez, 2016). Este índice porcentual (IP) integra en un solo valor la aceptación de cada planteamiento por los evaluadores mediante la siguiente fórmula:

$$IP = \frac{5(\%\ de\ TA) + 4(\%\ de\ DA) + 3(\%\ de\ Si - No) + 2(\%\ de\ ED) + 1(\%\ de\ TD)}{5}$$

La figura 25 muestra que el índice porcentual relacionado con la valoración de los expertos, sobre los aspectos planteados, es superior a 85%.



.

Figura 25: Resumen de la valoración de los expertos.

El método de validación realizado a través del escalamiento de Likert evidencia que las características y funcionalidades de la aplicación cumplen con los objetivos trazados y satisface las necesidades de los usuarios, al tener una alta valoración por parte de los expertos.

3.3.5.2 Satisfacción de usuarios a través de la técnica de IADOV

Esta técnica tiene como objetivo validar el grado de satisfacción de los usuarios con la implementación del sistema para la publicación de contenido *web* desde un dispositivo móvil con el sistema operativo Android.

En correspondencia con el criterio de selección antes esgrimido, se seleccionaron un total de 5 profesionales del Departamento de Operaciones Web y Análisis de Información (DOWAI), por su alta experiencia en la publicación de contenido. El cuestionario empleado para determinar el grado de satisfacción de los usuarios con la propuesta desarrollada y medir su impacto contó con las siguientes tres preguntas cerradas:

- ¿Se siente satisfecho con el resultado que se obtuvo con la aplicación teniendo en cuento el objetivo planteado?
- ¿Siente Usted que esta herramienta le es útil para ser utilizada desde el mismo proyecto?

- ¿Le gusta la forma en que se diseñó el sistema para la publicación de contenido *web* desde un dispositivo móvil con el sistema operativo Android?

Las preguntas cerradas se relacionan a través del "Cuadro lógico de IADOV", mostrado en la Tabla 9, el cual posibilita determinar posteriormente el nivel de satisfacción del usuario y del grupo.

Tabla 8: Cuadro lógico de IADOV.

Fuente: (Elaboración propia)

	¿Se siente satisfecho con el resultado que se obtuvo con la aplicación teniendo en cuento el objetivo planteado?								
	Sí			No Sé			No		
	¿Siente Usted que esta herramienta le es útil para ser utilizada desde el mismo proyecto?								
¿Le gusta la forma en que se diseñó el sistema para la publicación de contenido *web* desde un dispositivo móvil con el sistema operativo Android?	Sí	No Sé	No	Sí	No Sé	No	Sí	No Sé	No
Clara satisfacción;	1	2	6	2	2	6	6	6	6
Más satisfecho que insatisfecho	2	2	3	2	3	3	6	3	3
No definida	3	3	3	3	3	3	3	3	3
Más insatisfecho que satisfecho	6	3	6	3	4	4	3	4	4

Clara insatisfacción	6	6	6	6	4	4	6	4	5
Contradictoria	2	3	6	3	3	3	6	3	4

El número resultante de la interrelación de las preguntas nos indica la posición de cada sujeto en la escala de satisfacción, o sea su satisfacción individual. La escala de satisfacción utilizada es la siguiente: 1. Total satisfacción; 2. Más satisfecho que insatisfecho; 3. No definida; 4. Más insatisfecho que satisfecho; 5. Total insatisfacción; 6. Contradictoria.

Luego de aplicado el cuestionario y haber triangulado las preguntas cerradas, el número resultante de la interrelación indica la posición de cada cual en dicha escala de satisfacción. El índice de satisfacción grupal (ISG) se expresa en la escala numérica que oscila entre +1 y -1 de la siguiente forma:

Tabla 9: Escala numérica del ISG.

Fuente: (Elaboración propia)

Escala	**Resultado**
1	Máximo de satisfacción
0.5	Más satisfecho que insatisfecho
0	. No definido y contradictorio
-0.5	Más insatisfecho que satisfecho
-1	. Máxima insatisfacción

La satisfacción grupal se calcula por la siguiente fórmula:

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

El índice grupal arroja valores entre + 1 y - 1. Los valores que se encuentran comprendidos entre - 1 y - 0,5 indican insatisfacción; los comprendidos entre - 0,49 y + 0,49 evidencian contradicción y los que caen entre 0,5 y 1 indican que existe satisfacción.

$$\mathrm{ISG} = \frac{3(+1) + 2(+0.5) + 0(0) + 0(-0.5) + 0(-1)}{5} = 0.8$$

El valor obtenido al aplicar la técnica fue 0.8, el cual se encuentra en el intervalo de satisfacción, por lo que se puede afirmar que existe un alto grado de satisfacción con el sistema desarrollado. En la Figura 26 se muestran los porcientos de satisfacción obtenidos luego de calculado el ISG.

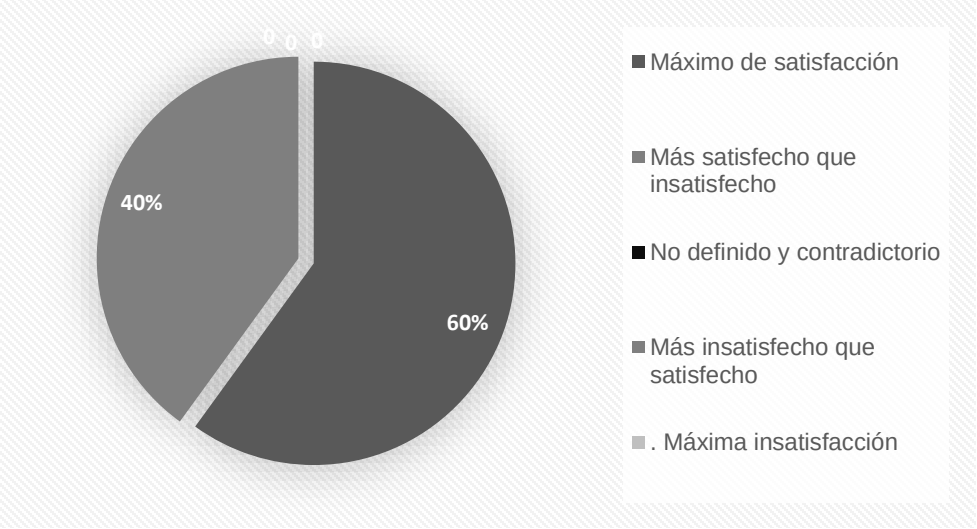


Figura 26: Satisfacción de usuarios con el sistema.

3.4 Conclusiones

Durante el desarrollo del capítulo se realizó la validación del sistema a través de los tipos de pruebas realizadas, obteniendo las siguientes conclusiones:

- La construcción del diagrama de componentes y la descripción de los mismos, brindó una mayor compresión del sistema desarrollado.
- Al aplicar los estándares de codificación se logró una mayor legibilidad, limpieza y organización del código.
- Las diferentes pruebas realizadas permitieron demostrar la calidad de la aplicación Android desarrollada.

Conclusiones

Luego de finalizada la investigación, se pudo arribar a las siguientes conclusiones:

- El análisis de los sistemas homólogos en los diferentes ámbitos arrojaron que no existía un sistema que brindara la facilidad de publicación de contenido *web*, lo que permitió una mayor comprensión del objeto de estudio, quedando acentuada la necesidad de implementación de un nuevo sistema.
- La caracterización de varias herramientas y tecnologías ayudaron a escoger las adecuadas para la realización de la aplicación, lo que contribuyó a un correcto proceso de desarrollo, siendo guiado por la metodología AUP-UCI.
- La identificación de los requisitos funcionales garantizó que la propuesta de solución respondiera a las necesidades planteadas por el cliente, quedando explícito lo que el sistema debía hacer y con los requisitos no funcionales se evidenció claramente las cualidades del producto.
- La modelación de los artefactos facilitó el soporte a la implementación de los requisitos previamente expresados por el cliente y garantizó la base para la organización lógica del código fuente.
- El sistema desarrollado permitió la obtención de una aplicación móvil para la gestión de contenido *web* y así facilitar la gestión de la información, sin necesidad de conexión directa con el sitio, disminuyendo el consumo de recurso y tiempo.
- La realización de las pruebas de *software* erradicó las insuficiencias detectadas en la aplicación, permitiendo su corrección en aras de lograr un producto con calidad y que cumpliera las necesidades del cliente.

Recomendaciones

Una vez concluida la investigación e implementada la aplicación se recomienda:

- El desarrollo de una funcionalidad que permita al usuario utilizar las conexiones de datos móviles para el envío del contenido *web*.
- El desarrollo de la aplicación para otros sistemas operativos de dispositivos móviles.

Referencias Bibliográfica

ALEGSA, L. Alegsa.com.ar. [En línea]. 12 de mayo de 2010. [Citado el: 1 de diciembre de 2016.]. Disponible en: http://www.alegsa.com.ar/Dic/sgbd.php.

BÁEZ, M. Introducción a Android. EME Madrid. España. 2012.

BLANCO, P.; CAMARERO, J., et al. Metodología de desarrollo ágil para sistemas móviles. Introducción al desarrollo con Android y el iPhone. Universidad Politécnica de Madrid. Madrid. 2009.

BUSTACARA, C.; DÍAZ, C. L., et al. Análisis y diseño O.O. Arquitectura de software. [En línea]. 2007. [Citado el: 23 de mayo de 2017]. Disponible en*: https://sophia.javeriana.edu.co/~lcdiaz/ADOO2006-3/grasp_cpaternostro-lvargas-jviafara.pdf.*

CANDÓN, M. J. Internet en movimiento: Nuevos movimientos sociales y nuevos medios en la sociedad de la información. Tesis Doctoral. Universidad Complutense de Madrid. 2011.

CUERVO, M.; ESCOBAR, P. Validez de contenido y juicio de expertos: una aproximación a su utilización. Facultad de Psicología Universidad El Bosque. Bogotá. Colombia. 2008.

DECSAI. Departamento de Ciencias de la Computación e I.A. [En línea]. Especificación de requerimientos. Diseño de bases de datos. [Citado el: 27 de enero de 2017]. Disponible en*:* http://elvex.ugr.es/idbis/db/docs/design/2-requirements.pdf.

DÍAZ, E. A.; SOTO, E. A. Sistema para la Publicación y Distribución de Contenidos Multimedia. Trabajo de Diploma, Universidad de las Ciencias Informática, Cuidad de la Habana, 2014.

DRUPAL. Mailhandler. [En línea]. DrupalCon Baltimore: 161 sessions, many voices, infinite possibilities. Earlybird rate ends Friday. [Consultado el: 1 de enero de 2017]. Disponible en: https://www.drupal.org/project/mailhandler.

ESTELLER, V.; MEDINA, E. Procesos de desarrollo de software y materiales educativos computarizados. Venezuela: Universidad de Carabobo, 2012.

ETECSA. Empresa de Telecomunicaciones de Cuba S.A. [En línea]. 2016. [Citado el: 25 de mayo de 2016.] Disponible: http://www.etecsa.cu/internet_conectividad/areas_wifi/.

FERGARCIAC. Entorno de Desarrollo Integrado (IDE). [En línea]. 2013. [Citado el: 17 de noviembre de 2016]. Disponible en: https://fergarciac.wordpress.com/2013/01/25/entorno-de-desarrollo-integrado-ide/.

FONTELA, C.; SUAREZ, P. Documentación y pruebas Antes del paradigma de objetos.2003

FRANGANILLO, **J**. "Html5: el nuevo estándar básico de la Web". *Anuario ThinkEPI*, 2011, v. 5, pp. 261-265.

GARCÍA, O. Nomenclatura Estándar del Código Java. [En línea]. [Consultado el: 29 de mayo de 2017]. Disponible en: http://www.elclubdelprogramador.com/2011/08/22/java-nomenclatura-estandar-del-codigo-java/.

GUERRERO, A.C.; SUAREZ, M. J., et al. GOF (The Gang of Four) Design Patterns in the context of Process Development of Web-Oriented Applications. [En línea]. [Consultado el: 15 de mayo de 2017]. Disponible en: http://www.scielo.cl/scielo.php?pid=S0718-07642013000300012&script=sci_arttext.

GIRONÉS, J.T. El gran libro de Android. México, Marcombo, S.A, 2012. 9-11p.

IBARRECHE, J. M. Creación de una plataforma de desarrollo de aplicaciones para Android. Escuela Técnica Superior de Ingeniería de Madrid. España. Tesis. 2010.

IVONET, A.; HERNÁNDEZ, L. Sistema para la edición y publicación electrónica de la Gaceta Oficial de la República de Cuba. Trabajo de Diploma, Universidad de las Ciencias Informática, Cuidad de la Habana, 2014.

LAPUENTE, M. J. Hipertexto, el nuevo concepto de documento en la cultura de la imagen. Tesis Dortoral, Universidad Complutense de Madrid, Madrid. 2013.

KOSTAKOPOULOU, D. Matters of Control: Integration Tests, Naturalisation Reform and Probationary Citizenship in the United Kingdom. Journal of Ethnic and Migration Studies, Vol. 36. 2010, 829p-847p.

LARMAN, C. UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado. [En línea]. [Consultado el: 26 de enero de 2017]. Disponible en: http://somosprogramacion.blogspot.com/2015/03/uml-y-patrones-2ed-craig-larman.html#.

LÓPEZ, P.; RUIZ, F. Lenguajje Uificado de Modelado-UML. Ingeniería del Software I, tema 2. Universidad Cantabria. 2011.

MALAVE, K. "Android" el sistema operativo de Google para dispositivos móviles. Venezuela, Fundación Miguel Unamuno y Jugo, 2011. 5-19p.

MICROSOFT. Revisiones de código y estándares de codificación. [En línea] Revisiones de código y estándares de codificación, 2016. [Citado el: 15 marzo 2017]. Disponible en: https://msdn.microsoft.com/es-es/library/aa291591(v=vs.71).aspx.

MOLINA, B. Integración de un proxy inverso NGINX con un panel de control Virtualmin para crear una plataforma de servicios de hospedaje Web. Tesis Doctoral, Universidad Politécnica de Valencia, España, 2015.

MYERS, G. J. The Art of Software Testing, Second Edition. Badgett and Todd M. Thomas with Corey Sandler. 2011

NAVARRO, P. L.; PÉREZ, G. M., et al. Open HMI Tester: un Framework Open-source para herramientas de Pruebas de Software. Universidad de Murcia. España. 16 de mayo del 2014.

OFICINA NACIONAL DE ESTADÍSTICAS E INFORMACIÓN. ONEI. [En línea]. 2015. [Citado el: 25 de mayo de 2017.]. Disponible en: http://www.one.cu/aec2014/17%20Tecnologias%20de%20la%20Informacion.pdf.

ORACLE. NetBeans IDE. [En línea]. 2015. [Citado el: 15 de noviembre de 2016.]. Disponible en: https://netbeans.org.

PALAQUIBAY, Q.; AZUCENA, P. Diseño e Implementación de la Arquitectura de Datos Basada en Comparativas de Rendimiento entre Sistemas de Gestión de Bases de Datos. [En línea] 24 de marzo de 2010. [Citada en: 20 de noviembre de 2016]. Disponible en: http://dspace.espoch.edu.ec/handle/123456789/97.

PAVÓN, J. Servidores Web – Apache. [En línea]. Aplicaciones Web/Sistemas Web, 2012. [Citado en: 21 de noviembre de 2016]. Disponible en: https://www.fdi.ucm.es/profesor/jpavon/web/31-ServidoresWeb-Apache.pdf.

PEDROSO, Z. "Informe diario de operaciones". División De Tecnología De La Información, 2016.

PERDOMO, V. M. Herramienta para la actualización de blogs en Wordpress sin acceder a la administración en línea desde sistemas GNU/Linux. Trabajo de Diploma, Universidad de las Ciencias Informática, Ciudad de la Habana, 2015.

PostgresSQL. [En línea]. 2016. [Citado el: 16 de noviembre de 2016.]. Disponible en: http://www.postgresql.org/about/.

PUIG, J. CSS3 y Javascript avanzado.Barcelona, 2013, p.7.

RADIO HABANA CUBA. Una voz de amistad que recorre el mundo. [En línea]. 2015. [Citado el: 25 de mayo de 2017.]. Disponible en: http://www.radiohc.cu/especiales/search?query=internet&total=151.

RAMÍREZ, J. F. Modelo para la selección de equipos de trabajo quirúrgico en sistemas de información en la salud aplicando técnicas de inteligencia organizacional. Tesis Doctoral. Universidad de la Ciencias Informáticas. La Habana. 2016.

RODRIGUÉZ, T. Metodología de desarrollo para la Actividad productiva de la UCI. Programa de Mejoras, Universidad de las Ciencias. Cuidad da la Habana.

PRESSMAN, S. R. Ingeniería del software un enfoque práctico. México. McGRAW- HILL INTERAMERICANA EDITORES, S.A. DE C.V. 2010. 57-70p.

STEVENS, P. P., et al. Utilización de UML en Ingeniería del Software con Objetos y Componentes. Pearson Addison Wesley. [En línea]. 2007.[Citado el: 10 de marzo de 2017]. Disponible en: http://dspace.ucbscz.edu.bo/dspace/bitstream/123456789/612/1/4000.pdf.

THE ECLIPSE FOUNDATION. Ecilpse. *About the Eclipse Foundation.* [En línea]. 2016. [Citado el: 2016 de noviembre de 09.] Disponible en: https://eclipse.org/org/.

THINGAMABLOG. Desktop Blogging for Everyone. [En línea]. 2009. [Citado el: 2 de diciembre de 2016]. Disponible en: http://www.thingamablog.com/

TRAMULLAS, J.; MARQUINA, J., et al. . Drupal para bibliotecas y archivos. [En línea]. Fund. Zaragoza Ciudad del Conocimiento, 2010. [Citado el: 2 de octubre de 2016]. Disponible en: http://eprints.rclis.org/14400.

URRA, X. Desarrollo de una comunidad web de montañeros basada en la plataforma Drupal. Trabajo Fin de Estudios, Universidad Pública de Navarra, Navarra, 2013.

VALBUENA, A. M. Guía comparativa de Frameworks para los lenguajes HTML 5, CSS y JavaScript para el desarrollo de aplicaciones Web. Trabajo de Grado, Universidad Tecnológica de Pereira, 2014.

VERDECIA, Y.J. Manual de funcionamiento interno. Centro de Ideoinformática. Universidad de las Ciencias Informáticas. La Habana. 6 de febrero del 2017.

VISCONTI, M.; ASTUDILLO, H. Fundamentos de Ingeniería de Software. 2012.

VISUAL PARADIGM. Visual Paradigm for UML - Software design tools for agile software development. [En línea]. 2014. [Citado el: 30 de octubre de 2016]. Disponible en: http://www.visual-paradigm.com/product/vpuml.

VOGEL, L. Android SQLite database and content provider-tutorial. Java, Eclipse, Android and Web programming tutorials, 2010.

WBLOGGAR. [En línea]. 2016. [Citado el: 2 de nodiciembre de 2016]. Disponible en: http://wbloggar.com/.

Glosario de términos

IntelliJ IDEA: entorno de desarrollo integrado que ha sido empaquetado como un programa de aplicación, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica.

Logcat: listado de los mensajes emitidos por el teléfono. Muy útil para los programadores con el fin de encontrar qué causa los errores.

Gradle: sistema de compilación que reúne en uno solo las mejores prestaciones de otros sistemas de compilación.

Plugin*:* aplicación o programa informático que se relaciona con otra para agregarle una función nueva y generalmente muy específica.

Scripts*:* lenguaje de programación que ejecuta diversas funciones en el interior de un programa de computador.

RSS: servicio que permite tener constancia de la actualización de gran cantidad de páginas directamente en nuestro escritorio, cliente de correo o a través de la *Web*, justo al poco de ser actualizadas por su autor.

Acrónimos

A	**APK**: Aplicación Empaquetada de Android. **API:** Interfaz de Programación de Aplicaciones.
B	**BSD**: Distribución de Software Berkeley.
G	**GPL**: Licencia Pública General. **GNU**: GNU no es Unix.
S	**SGML**: Lenguaje de Marcas Generalizado.