UNIVERSIDAD NACIONAL DEL ALTIPLANO - PUNO
FACULTAD DE INGENIERÍA ESTADÍSTICA E INFORMÁTICA
ESCUELA PROFESIONAL DE INGENIERÍA ESTADÍSTICA E INFORMÁTICA

Seguridad en archivos externos e internos para dispositivos con sistema operativo Android

TESIS

PRESENTADA POR:

Bach. DARWIN BENIS VALERIANO CALSINA

PARA OPTAR EL TÍTULO PROFESIONAL DE:

INGENIERO ESTADÍSTICO E INFORMÁTICO

PUNO - PERÚ
2017

UNIVERSIDAD NACIONAL DEL ALTIPLANO - PUNO

FACULTAD DE INGENIERÍA ESTADÍSTICA E INFORMÁTICA

ESCUELA PROFESIONAL DE INGENIERÍA ESTADÍSTICA E INFORMÁTICA

Seguridad en archivos externos e internos para dispositivos con sistema operativo Android

TESIS

Presentada por:
Bach. DARWIN BENIS VALERIANO CALSINA

Para optar el Título Profesional de:

INGENIERO ESTADÍSTICO E INFORMÁTICO

APROBADO POR:

PRESIDENTE : _____
M.Sc. Ernesto Nayer Tumi Figueroa

PRIMER MIEMBRO : _____
M.Sc. Charles Ignacio Mendoza Mollocondo

SEGUNDO MIEMBRO : _____
M.Sc. Rudy Alvaro Arpasi Pancca

DIRECTOR : _____
Dr. Juan Reynaldo Paredes Quispe

ASESOR : _____
Dr. Alejandro Apaza Tarqui

AREA : Informática
TEMA : Ingeniería de software e inteligencia artificial
FECHA DE SUSTENTACION : 18 de Mayo de 2017

DEDICATORIA

Dedico la presente tesis a mis seres queridos que tanto apoyo me dieron, a mi Padre y madre, Benito y Paulina por su arduo trabajo y aliento para mi culminación como Ingeniero y por supuesto a todos los amigos y amigas Que desde el inicio de nuestros estudios nos Hemos acompañado incondicionalmente.

Benis Darwin

AGRADECIMIENTOS

A la Universidad Nacional del Altiplano de Puno y a
Mi Facultad, quienes me acogieron y brindaron
El conocimiento necesario para mi vida profesional.

A mis jurados Ing. Charles, Ing. Nayer, gracias por su apoyo.

ÍNDICE

INDICE DE TABLAS

ÍNDICE DE FIGURAS

NDICE DE GRAFICOS

RESUMEN

El sistema operativo android goza de gran popularidad y cada vez más empresas lo adoptan para sus equipos: Smartphone y Tablet; las cuales tienen una increíble funcionalidad para aprovechar la telefonía Móvil 3G y 4G, en un futuro no muy lejano poder administrar dichos recursos que ya es común en las tareas cotidianas como: compras, pagos, revisión de cuentas, envío de archivos, comunicación, entre otras. Es por ello que la investigación se centra en desarrollar la aplicación Android como una alternativa altamente funcional basándose en la necesidad de brindar seguridad a los archivos e información personal de los usuarios. También se busca definir los requerimientos principales para la arquitectura de dicha aplicación realizando pruebas tanto de software para corroborar su funcionalidad y eficiencia así como una interfaz amigable la cual atrae al público en general. En la parte de la metodología de desarrollo del Software se utilizó Xtreme Programming (XP), representa una ventaja y sus nuevas funcionalidades mucho más rápida que las demás metodologías por esa razón es un éxito en los requerimientos; para lo cual se realizó una hipótesis con mcnamer cuyo resultado fue un 69% de aceptación de dicha aplicación. Se logró diseñar y desarrollar la aplicación **“boveda”** para teléfonos inteligentes y Tablet con sistema operativo Android que encriptado y serialización de objetos, se logró establecer que la seguridad de información es una necesidad primordial para los usuarios. Se pudo coincidir con los autores de otras tesis que realmente es importante la seguridad de nuestros dispositivos móviles para que no pueda ver fuga de información.

Palabras clave: Android, App, Metodología Xp, Serialización, Encriptación.

ABSTRACT

The Android operating system is very popular and more and every time more companies adopt it for their equipment: Smartphone and Tablet; which have an incredible functionality to take advantage of 3G and 4G Mobile phones, in the not too distant future to manage these resources that is already common in every day tasks such as: Purchases, payments, accounts review, sending of communication, files, among others. This is why research focuses on developing the Android application as a highly functional alternative based on the need to provide security for use file and personal information. It also seeks to define the main requirements for the architecture of this application by testing both software to corroborate it is functionality and efficiency as well as a friendly interface which attracts the general public. In the part of software development methodology Xtreme Programming (XP) was used, represents an advantage and it is new functionalities much faster than the other methodologies for that reason is a success in the requirements; for this reason a hypothesis was made with mcnamer whose result was a 69% acceptance of this application. It was able to design and develop the application "boveda" for smartphones and Tablet with Android operating system that encryption and serialization of objects, it was established that information security is a primary need for users. It was possible to coincide with he authors of other theses that it is really important the security of our mobile devices so that it can not see leak information.

Keywords: Android, App, Xp Methodology, Serialization, Encryption

INTRODUCCIÓN

La investigación propone el desarrollo de aplicaciones distribuidas para sistema operativo Android, basado en el incremento de demanda que los dispositivos inteligentes en telefonía tienen, sobre todo por que estos utilizan para la interfaz al sistema operativo Android, desde la versión 2.2 en los SmartPhones, hasta la versión 7.0 en las tablets y las versiones más modernas de SmartPhones como Samsung Galaxy S8.

Actualmente la potencia de los dispositivos móviles ha igualado la de un computador portátil, con una capacidad de proceso de hasta 2.7 GHz con sus cuatro núcleos como es caso del Samsung Galaxy Note 4, pero no es posible orientar todas las aplicaciones a este tipo de dispositivo sino a aquellos equipos de gama baja, con capacidad de memoria de 512 MB con velocidad de procesamiento de 800Mz.

Los protocolos inalámbricos más comunes son WAP (Wireless Application Protocol) y Bluetooth, presentes desde los dispositivos más sencillos hasta más modernos tenemos; el Wi-Fi (Wireless Fidelity), la finalidad es ofrecer servicios y contenidos de internet a través de conexiones inalámbricas. Por otro lado, Bluetooth es un estándar diseñado para la transmisión de voz y datos para distancias cortas. Así como la plataforma J2ME (Java to Micro Edition) que tiene más años de experiencia en este campo, cuenta además con conjuntos de APIs (Interfaz Programable para Aplicaciones) para Bluetooth, OpenGLES, Java2D, Java3D, Sockets e Internet conectivity, convirtiéndola en la opción portable más popular para desarrollar aplicaciones móviles.

Lo que se busca demostrar es que en la actualidad la evolución tecnológica, que

se desarrolla a un ritmo alarmante, pero a la par de esta se ha visto ciertas deficiencias o falta de desarrollo en determinados aspectos, por ejemplo la seguridad en la cual los usuarios, no siempre pueden proteger sus datos, archivos o información; esto principalmente en los aplicativos para celulares y tablets.

Teniendo en cuenta esta deficiencia y falta de desarrollo se decidió implementar y buscar qué características implicaría crear y desarrollar en una aplicación la cual permita resguardar y brindar seguridad a nuestros datos y archivos, evitando que cuando pierda el usuario su celular o Tablet; evite que terceras personas puedan hacer uso de la información que éste guardó en dichos aparatos.

Para esto, la investigación se llevará en distintos aspectos dentro de los cuales se verá el desarrollo del software y los requerimientos necesarios para el adecuado funcionamiento; mientras que en otro lado se verá cuáles son las características determinantes que requerirán los usuarios es decir se hará un pequeño análisis de los requerimientos que estos deberían tener; y por último se verá la evaluación de dicho software y se verificará que hayan cumplido con las expectativas de los usuarios.

Dentro de los principales aportes que dará esta investigación es corroborar el hecho de que el factor seguridad en esta era de información y desarrollo tecnológico es primordial para los usuarios ya que cada día hacen uso de la tecnología creando una enorme cantidad de datos, documentos, fotos, videos, archivos, entre otros; siendo muchos de estos de índole personal o primordial para su desarrollo. También a su vez se logrará denotar y dar las características para el diseño y la estructuración del aplicativo, como los requerimientos necesarios para el máximo aprovechamiento en los equipos de diversa gama (tablets, celulares, laptops).

Para esto no se pretende hacer un catálogo de las características y prestaciones de todos estos dispositivos móviles y las empresas o versiones que utilizan, pero se requiere detallar las características más saltantes para esta investigación sobre todo por las versiones de la API de desarrollo.

En los cuales la tesis se divide de la siguiente manera:

En el CAPÍTULO I, se podrá encontrar la descripción general, así como el planteamiento del problema, la justificación, los objetivos, la hipótesis de la investigación y las limitaciones de la investigación. Dentro del CAPÍTULO II, se dará a conocer el marco teórico, la arquitectura de Android, las definiciones de los términos básicos, información relevante acerca de Android SDK, así también como información acerca de los Smartphones.

En el CAPÍTULO III, se procederá a ver el desarrollo de la investigación, se da a conocer los Materiales y Métodos, además de la Población y Muestra de la investigación, también daremos a conocer los Métodos de Recopilación de datos y los métodos de tratamiento de datos; la Prueba de Mcnemar y para finalizar el Modelo y Metodología de Desarrollo De Software.

En el CAPITULO IV de Resultados; se detalla los requerimientos de sistema y a su vez el diseño del mismo siempre buscando la mayor eficiencia del mismo y máxima comodidad y accesibilidad para el adecuado uso de los usuarios; también se realiza el seguimiento y validación de la App información con prueba Estadística y las correspondientes recomendaciones para poder realizar trabajos a futuro.

CAPÍTULO I

PLAN DE INVESTIGACIÓN

1.1 DESCRIPCIÓN GENERAL

El desarrollo de aplicaciones está bastante unida a la disponibilidad de tecnologías para su ejecución para el caso de las aplicaciones de escritorio o Desktop únicamente depende del sistema operativo, actualmente los computadores de bolsillo o PocketPC han sido reemplazados por los Smartphone, convirtiéndose en dispositivos móviles bastantes accesibles de adquisición económica y de prestaciones a los usuarios, es posible desarrollar para estos dispositivos aplicaciones a medida e incluso que sean capaces de crear comunicación cliente/servidor de ser necesario. Android OS se ha convertido en una plataforma de desarrollo adoptada por empresas como Samsung, HTC y fabricantes de Tablets. Los móviles son más potentes y livianos cada vez, permitiendo que nuestra comunicación sea cada vez más eficaz. Su gran número y sus capacidades hacen muy interesante para los proveedores de servicios y contenidos. El disponer de un entorno normalizado que permita ofrecer sus servicios a los usuarios de las redes móviles.

En el trabajo se pretende dar a conocer los pasos para el desarrollo de Aplicaciones Distribuidas en Android OS. Esta aplicación utiliza el framework de SQ Lite que ofrece herramientas para manipular una base de datos que almacenar la información de los contactos, como también los Sockets de conexión, desarrollar que permitan usar las prestaciones gráficas. Con estas herramientas se implementa las diferentes funciones inherentes de estos dispositivos móviles y aprovechar la API de funciones del sistema operativo para todo tipo de Smartphone. Se revisan las plataformas de desarrollo para dispositivos Móviles como Eclipse, NetBeans, MotoDev Studio y finalmente MonoDevelop 3.0.4.7 en la versión Win32 y puede descargarse gratuitamente todo el Kit de Desarrollo desde ***http://android.xamarin.com*** y la versión instalable para comenzar a desarrollar en Windows del Android SDK desde **https://developer.android.com/studio/index.html** en la versión 2.3.0 para desarrollo.

1.2 PLANTEAMIENTO DEL PROBLEMA

En la actualidad la tecnología avanza a un ritmo acelerado donde cada persona o usuario hace uso de diferentes dispositivos (Tablet celulares laptop) que son parte de su vida diaria y satisfacen sus necesidades de transmitir información.

Frente a esto existen diversos tipos y aplicativos creados por empresas grande y pequeño; pero no existe hasta el momento una aplicación que evite la sustracción de información personal y confidencial siendo una ayuda para evitar que los curiosos intenten acceder a nuestra información.

Desarrollar una aplicación distribuida con el software "***Android estudió"*** y el

uso de sockets de conexión sobre una red Wi-Fi del proveedor de telefonía y así intercomunicar dispositivos, personas relacionadas en una empresa mejorando la productividad y comunicación de datos importantes como registros, bases de datos, ordenes de pedido, control de stock entre otros elementos empresariales. Todos estos temas nos permiten plantear la siguiente pregunta de investigación**: *¿Se implementó la Seguridad en archivos externos e internos para dispositivos con sistema operativo Android, que permite a los usuarios administrar sus archivos con seguridad?***

1.3 JUSTIFICACIÓN DE LA INVESTIGACIÓN

Hoy en día los dispositivos móviles se han convertido en una herramienta esencial de comunicación para la sociedad en general; pero junto a esta creciente necesidad también está presente los malos hábitos tales como: robos frecuentes de dispositivos móviles, es por ello la necesidad de implementar un sistema que permita a los usuarios administrar sus archivos con seguridad.

Es aquí donde surge nuestra preocupación, donde uno es vulnerable de los archivos digitales personales, por esa razón se desarrolla una aplicación que sea capaz de resguardar y que se quiera proteger de las personas inescrupulosas.

Esto también, conlleva a que las aplicaciones faciliten a los usuarios para que poco a poco vayan migrando de un sistema de escritorio a un dispositivo móvil, ya que la mayoría de las personas busca consultar información necesaria de forma inmediata desde cualquier lugar por el fácil acceso al internet.

Son tantos casos de robo de información que se da a nivel personal y de su trabajo, vemos como la sociedad por medio de la red circulan dichos archivos sustraídos; que muchas veces hace daño a su persona o su entorno de trabajo

Por lo tanto se espera que sea un buen aporte para los usuarios con su dispositivo móvil Android para fortalecer la seguridad; con esta aplicación desarrollada para vuestro beneficio así evita daños a terceras personas

1.4 OBJETIVOS

1.4.1 Objetivo General

- Desarrollar Sistema de App basado en **Seguridad en archivos externos e internos para dispositivos con sistema operativo Android,** que permita a los usuarios administrar sus archivos con seguridad basados en el uso de Android SDK.

1.4.2 Objetivos Específicos

- Diseñar los requerimientos y la arquitectura de la aplicación para sistema operativo Android.

- Realizar hacer pruebas en diferentes dispositivos con diferentes archivos y emular nuestra aplicación, además testear en los dispositivos móviles.

1.5 HIPOTESIS DE LA INVESTIGACIÓN

La aplicación de seguridad de archivos en el dispositivo móvil mejora la protección de documentos personales.

1.6 LIMITACIONES DE LA INVESTIGACIÓN

La presente investigación presenta las siguientes limitaciones:

- El sistema App de uso exclusivo en dispositivos móviles android.
- El sistema es solo uso para los que puedan tener cuenta en gmail para poder descargar del Google Play

CAPÍTULO II

MARCO TEÓRICO

2.1 ANTECEDENTES DE LA INVESTIGACION

Actualmente dentro del mundo de la programación para dispositivos Android se presenta algunos softwares pero no específicamente lo que el usuario requiere como proteger sus documentos importantes como son Word, Excel, Power point, fotos y entre otros archivos.

El desarrollo de las tecnologías, en la última década, ha dado un impulso notable a nuevos medios de comunicación, los cuales hasta hace pocos años no pasaban de ser experimentos.

A través de su tesis se puede notar el estudio de la misma manera que con las PC, los dispositivos móviles son un blanco de los ataques contra la seguridad de la información y los problemas de seguridad, hacen que estos sean un objetivo valioso a la hora de realizar tareas como las de espionaje. El uso tan extendido de dispositivos móviles ha hecho que se conviertan de manera activa en una herramienta más de nuestro trabajo, alojando en muchas ocasiones información organizacional crítica o valiosa que en caso de

ser interceptada, conllevaría grandes problemas de seguridad. Dicho uso tan extendido de estos dispositivos ha hecho que los ciber delincuentes lo vean como un mercado a explotar, y al día de hoy, los dispositivos móviles se han convertido en uno de los focos principales ante ataques informáticos, pasando desde el comportamiento enteramente destructivo hasta el robo de información y el chantaje.

La seguridad de los dispositivos móviles constituye una problemática actual de gran interés para las organizaciones cualquiera sea su tamaño. A la hora de diseñar e implementar soluciones y gestionar la seguridad de los dispositivos (Pacheco Veliz, 2016).

Generar una metodología para determinar las acciones a realizar que permitan efectuar una conexión segura entre los diversos dispositivos móviles, como son: laptops, equipos celulares, tabletas, entre otros, para el correcto manejo de los datos que almacenan; considerando las mejores prácticas en la seguridad de los dispositivos móviles que se usan en la empresas; y con ello lograr que la interconectividad sea eficiente y se minimicen los riesgos que el usuario pueda llegar a generar al ser la parte más vulnerable de la operación de estos dispositivos.

Con la investigación realizada dentro de la empresa Estacionamientos Corsa, es notable que, aun teniendo muchas formas de acceso a la información, la gente no se preocupa por las vulnerabilidades dentro de sus dispositivos o no saben el riesgo que pueden enfrentar al no tener las medidas necesarias para salvaguardar información sensible ya sea de carácter personal o del lugar de trabajo.

Es importante señalar que el desarrollo de dicha metodología se aplicará en forma real, teniendo como propósito: reducción de los riesgos y amenazas visibles en la actualidad, los cuales han venido afectando a los dispositivos móviles; mismos que son de interés y preocupación por parte de las organizaciones en cuanto a que su información se encuentre lo más segura posible. (Mejía Pacheco, 2016)

2.2 BASE TEÓRICA

2.2.1 Definición de Login

El login es nombre dado al momento de autentificación al ingresar a un servicio o sistema.

En el momento que se inicia el login, el usuario entra en una sesión, para ello se emplea usualmente un nombre de usuario y contraseña. Actualmente el procedimiento de login puede ser también a través de la lectura de las huellas dactilares (con un lector de huellas), identificación morfológica del rostro (empleando una cámara), por escaneo del iris del ojo, entre otras formas.

Por ejemplo, cuando un foro de internet pide el "login", es porque está pidiendo el nombre de usuario y contraseña que se eligió cuando el usuario se registró en dicho foro de internet.

Login suele usarse como verbo y conjugarse al españolizarse, por ejemplo: "loguearse". En inglés la acción de "loguearse" es "logging in".

Un término más apropiado para "loguearse" sería "Iniciar sesión" o "Autentificarse" o "Autenticarse".

La acción contraria es cerrar sesión o des identificarse (logging out).

Login name o nombre de usuario. Es el nombre que adquiere el usuario para acceder a un determinado servicio. (Alegsa, 2017)

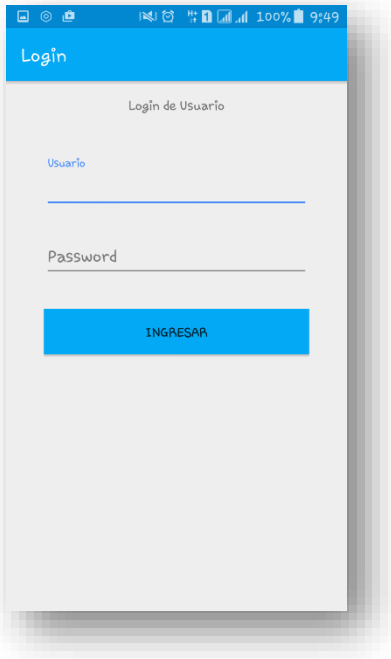


Figura 1 Login para la aplicación. Elaboración Propia

2.2.2 Algoritmo de encriptación AES

AES conocida como Estándar de Encriptación Avanzada (Advanced Encryption Standard). AES es una técnica de cifrado de clave simétrica que remplazará el Estándar de Encriptación de Datos (DES) utilizado habitualmente.

Es el resultado de un llamamiento a nivel mundial por la presentación de solicitudes de los algoritmos de cifrado emitido por el Instituto

Nacional de Estándares y Tecnología (NIST) del Gobierno de EEUU en el año 1997 y completado en el año 2000.

El algoritmo ganador, Rijndael, fue desarrollado por dos criptologistas Belgas, Vincent Rijmen y Joan Daemen.

AES proporciona una encriptación segura y ha sido elegida por NIST como un Estándar de Proceso de Información Federal en Noviembre del 2001 (FIPS-197), y en Junio del 2003 el Gobierno de EEUU (NSA) anunció que AES es lo suficientemente seguro para proteger la información clasificada hasta el nivel ALTO SECRETO, que es el nivel más alto de seguridad y que se definen como información que pudiera causar "daños excepcionalmente graves" a la seguridad nacional en caso de ser divulgada al público.

El algoritmo AES utiliza una de las tres fortalezas de clave de cifrado: una clave de encriptación (contraseña) de 128-, 192-, o 256- bits. Cada tamaño de la clave de cifrado hace que el algoritmo se comporte ligeramente diferente, por lo que el aumento de tamaño de clave no solo ofrece un mayor número de bits con el que se pueden cifrar los datos, sino también aumentar la complejidad del algoritmo de cifrado.

BitZipper soporta claves de cifrado de 128- y 256- bits, las cuales son soportadas por WinZip 9. Ambas claves proporcionan una seguridad mayor respecto a la encriptación estándar ZIP 2.0. Es ligeramente más rápido de encriptar y desencriptar datos protegidos con AES 128-bit, pero con los PCs rápidos de hoy en día la diferencia de tiempo apenas se percibe.(ALEGSA, 2011)

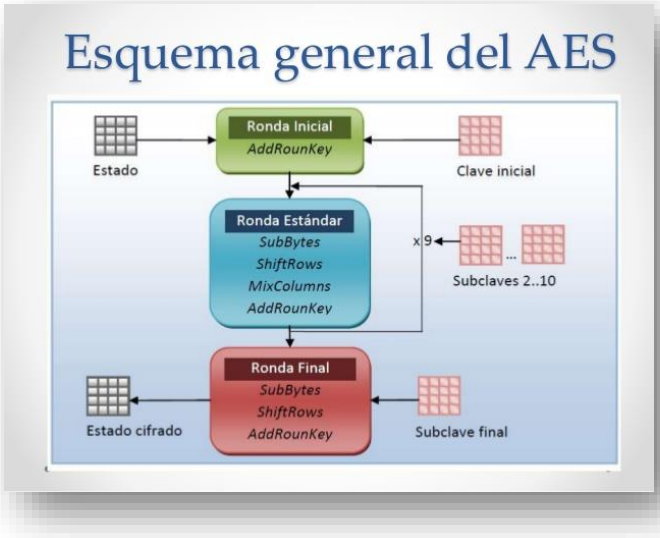


Figura 2 : Esquema General Encriptación AES. Recuperado de: https://es.slideshare.net/Ayares/algoritmo-aes

2.2.3 Serialización de objetos en Java

En ocasiones necesitamos almacenar el estado de un objeto dentro de un dispositivo de almacenamiento, pero el manejar el archivo y escribir cada atributo del objeto es una tarea cansada, pero Java tiene una clase que nos ayuda a guardar el estado del objeto tal y como lo mantenemos en la ejecución del programa. Esta clase es conocida como Serializable.

Serialización de un objeto

La serialización de un objeto consiste en obtener una trama de bytes que represente el estado de un objeto, donde podrá ser escrito dentro de un archivo, o bien, para ser enviado a través de una red. Una característica importante es que esto hace posible la persistencia de

objetos, es decir, el estado de un objeto se almacena para su uso posterior.

¿Qué es Serializable?

Serializable es una clase ubicada en el paquete Java.io.Serializable, la cual no cuenta con ningún método, por lo que es una clase que sirve solamente para especificar que todo el estado de un objeto instanciado podrá ser escrito o enviado en la red como una trama de bytes.

Hay que tomar en cuenta que, si un objeto tiene como atributo otro objeto, entonces se debe declarar ese objeto del tipo Serializable, y estos serán serializados primero. Esto se puede tomar como un árbol, donde las hojas son objetos que forman parte de otro objeto como un atributo, y todos ellos son marcados como serializables, para así entonces serializar primero las hojas del árbol y finalizar con la raíz.

Habrá ocasiones donde querremos exceptuar un atributo del objeto que no queremos serializar, y para esto se encuentra el modificador Transient. Este modificador le indica a la JVM (Java Virtual Machine) que dicho atributo deberá ser exentado de la serialización, en otras palabras, ignorará este atributo.

Los atributos que lleven el modificador Static nunca se tomarán en cuenta al serializar un objeto, ya que este atributo pertenece a la clase y no al objeto.

La serialización es un mecanismo para convertir el estado de un objeto en un flujo de bytes. La deserialización es el proceso inverso en el que se utiliza el flujo de bytes para recrear el objeto Java real en la memoria. Este mecanismo se utiliza para persistir el objeto.

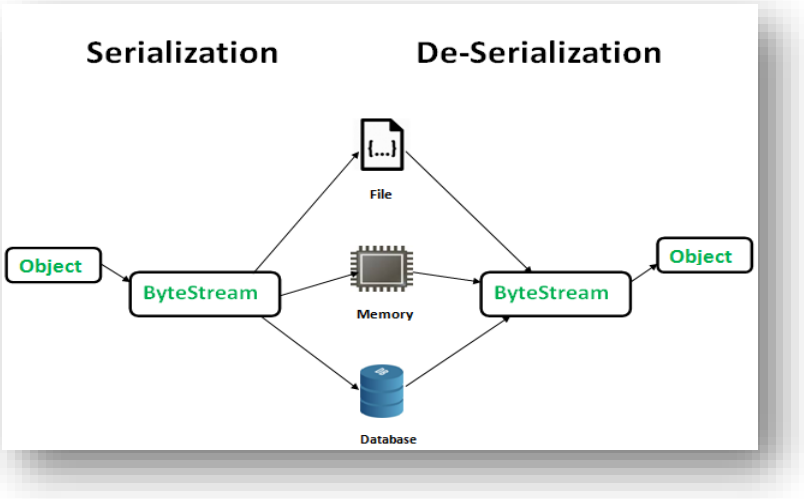


Figura 3: Esquema General Serializacion de-serialización. Recuperado de http://www.geeksforgeeks.org/serialization-in-java/

El flujo de bytes creado es independiente de la plataforma. Por lo tanto, el objeto serializado en una plataforma puede ser deserializado en una plataforma diferente. (Torres, 2014)

2.3 ANDROID OS

Hace algunos años, Google decidió que debía expandir su negocio hacia los móviles, así que su mejor estrategia fue crear un sistema operativo móvil propio, gratis y con varios de los más grandes fabricantes de celulares como respaldo, de esta manera nace Android.

Android es un sistema operativo móvil basado en el kernel de Linux, con una interfaz de programación Java, diseñado para ser utilizado en dispositivos

móviles como teléfonos inteligentes, tabletas, Google TV y otros. Desarrollado por la Open Handset Alliance la cual es liderada por Google

Android permite programar aplicaciones en una variación de Java llamada Dalvik. El sistema operativo proporciona todas las interfaces necesarias para desarrollar aplicaciones que accedan a las funciones del teléfono (GPS, llamadas, sms, agenda, entre otras.) de una forma muy fácil en un lenguaje de programación muy popular como es Java. (Riquelme, 2013)

Figura 4: SO Android en Teléfonos Móviles. Recuperado de http://shmector.com/freevector/phone_with_android_logo/1-0-1058

2.4 BREVE HISTORIA

Fue desarrollado inicialmente por Android Inc., una firma comprada por Google en 2005. Es el principal producto de la Open Handset Alliance, un conglomerado de fabricantes y desarrolladores de hardware, software y operadores de servicio. Las unidades vendidas de teléfonos inteligentes con Android se ubican en el primer puesto en los Estados Unidos, en el segundo y tercer trimestres de 2010, con una cuota de mercado de 43,6% en el tercer trimestre. A nivel mundial alcanzó una cuota de mercado del 50,9% durante el cuarto trimestre de 2011, más del doble que el segundo sistema operativo (iOS de Apple, Inc.) con más cuota.(Jiménez-García & Martínez-Ortega, 2017)

Tiene una gran comunidad de desarrolladores escribiendo aplicaciones para extender la funcionalidad de los dispositivos. A la fecha, se ha llegado ya al 1.000.000 de aplicaciones (de las cuales, dos tercios son gratuitas y en comparación con la App Store más baratas) disponibles para la tienda de aplicaciones oficial de Android: Google Play, sin tener en cuenta aplicaciones de otras tiendas no oficiales para Android como la tienda de aplicaciones Samsung Apps de Samsung. Google Play es la tienda de aplicaciones en línea administrada por Google, aunque existe la posibilidad de obtener software externamente. Los programas están escritos en el lenguaje de programación Java. No obstante, no es un sistema operativo libre de malware, aunque la mayoría de ello es descargado de sitios de terceros.

El anuncio del sistema Android se realizó el 5 de noviembre de 2007 junto con la creación de la Open Handset Alliance, un consorcio de 78 compañías de hardware, software y telecomunicaciones dedicadas al desarrollo de estándares abiertos para dispositivos móviles.Google liberó la mayoría del código de Android bajo la licencia Apache, una licencia libre y de código abierto.

Figura 5. El sistema Operativo Android A nivel mundial alcanzó una cuota de mercado del 50,9% durante el cuarto trimestre de 2011, más del doble que el segundo sistema operativo (iOS de Apple). Recuperado de https://losmuertevideanos.wordpress.com/2013/10/24/android-su-historia/

2.4.1 Adquisición por parte de Google

En julio de 2005, Google adquirió Android Inc., una pequeña compañía de Palo Alto, California fundada en 2003. Entre los cofundadores de Android que se fueron a trabajar a Google están Andy Rubin (co-fundador de Danger), Rich Miner (co-fundador de Wildfire Communications, Inc.), Nick Sears (alguna vez VP en T-Mobile), y Chris White (quien encabezó el diseño y el desarrollo de la interfaz en WebTV).3 En aquel entonces, poco se sabía de las funciones de Android Inc. fuera de que desarrollaban software para teléfonos móviles. Esto dio pie a rumores de que Google estaba planeando entrar en el mercado de los teléfonos móviles.

En Google, el equipo liderado por Rubin desarrolló una plataforma para dispositivos móviles basada en el núcleo Linux que fue promocionado a fabricantes de dispositivos y operadores con la promesa de proveer un sistema flexible y actualizable. Se informó que Google había alineado ya una serie de fabricantes de hardware y software y señaló a los operadores que estaba abierto a diversos grados de cooperación por su parte.

La especulación sobre que el sistema Android de Google entraría en el mercado de la telefonía móvil se incrementó en diciembre de 2006. Reportes de BBC y The Wall Street Journal señalaron que Google quería sus servicios de búsqueda y aplicaciones en teléfonos móviles y estaba muy empeñado en ello. Medios impresos y en línea pronto

reportaron que Google estaba desarrollando un teléfono con su marca.

En septiembre de 2007, «InformationWeek» difundió un estudio de Evalueserve que reportaba que Google había solicitado diversas patentes en el área de la telefonía móvil. (Risoto, 2013)

2.4.2 Versiones

El historial de versiones del sistema operativo Android se inició con el lanzamiento de Android beta en noviembre de 2007. La primera versión comercial (de prueba), Android 1.0, fue lanzada en septiembre de 2008.

Tabla 1. Clasificación de versiones de android y fecha de lanzamiento

Nombre código	Número de versión	Fecha de lanzamiento	Nivel de API
Android 1.0	1	23 de septiembre 2008	1
Android 1.1	1.1	9 de febrero 2009	2
Cupcake	1.5	27 de abril de 2009	3
Donut	1.6	15 de septiembre de 2009	4
Eclair	2.0–2.1	26 de octubre de 2009	5–7
Froyo	2.2–2.2.3	20 de mayo 2010	8
Gingerbread	2.3–2.3.7	6 de diciembre 2010	9–10
Honeycomb	3.0–3.2.6	22 de febrero de 2011	11–13
Ice Cream Sandwich	4.0–4.0.4	18 de octubre 2011	14–15
Jelly Bean	4.1–4.3.1	9 de julio de 2012	16–18
KitKat	4.4–4.4.4, 4.4W–4.4W.2	31 de octubre de 2013	19–20
Lollipop	5.0–5.1.1	12 de noviembre de 2014	21–22
Marshmallow	6.0–6.0.1	5 de octubre de 2015	23
Nougat	7.0 - 7.1.2	22 de agosto de 2016	24 - 25
Android O	8	TBA	26

Nota: Adaptado de "Versiones de Android", por Wikipedia. Recuperado de https://es.wikipedia.org/wiki/Anexo:Historial_de_versiones_de_Android

Android es un sistema operativo móvil desarrollado por Google y la Open Handset Alliance, y ha visto un número de actualizaciones a su sistema operativo base desde su lanzamiento original. Estas

actualizaciones típicamente corrigen fallos de programa y agregan nuevas funcionalidades. Desde abril de 2009, las versiones de Android han sido desarrolladas bajo un nombre en clave y sus nombres siguen un orden alfabético: Cupcake, Donut, Éclair, Froyo, Gingerbread, Honeycomb, Ice Cream Sandwich, Jelly Bean, KitKat, Lollipop, Marshmallow, Nougat lanzado en agosto de 2016 y el ya anunciado "Android Oreo". (Lucas, 2016)

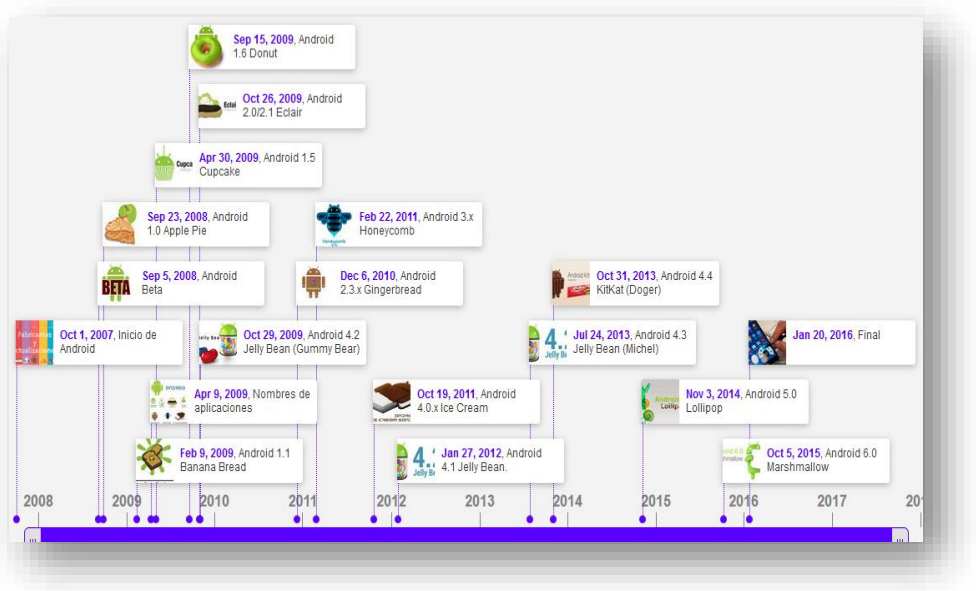


Figura 6. El proceso de Línea de tiempo de Android. Recuperado de https://www.timetoast.com/timelines/versiones-de-android-c110c29d-1479-43c2-b927-396ca908e64b

2.4.3 Lanzamiento de Android

El anuncio del sistema Android se realizó el 5 de noviembre de 2007, junto con la creación de la Open Handset Alliance, un consorcio de 78 compañías de hardware, software y telecomunicaciones dedicadas al desarrollo de estándares abiertos para dispositivos móviles. Google liberó la mayoría del código de Android bajo la licencia Apache, una licencia libre y de código abierto.

La estructura del sistema operativo Android se compone de aplicaciones que se ejecutan en un framework Java de aplicaciones orientadas a objetos sobre el núcleo de las bibliotecas de Java en una máquina virtual Dalvik con compilación en tiempo de ejecución. Las bibliotecas escritas en lenguaje C incluyen un administrador de interfaz gráfica (surface manager), un framework OpenCore, una base de datos relacional SQLite, una Interfaz de programación de API gráfica OpenGL ES 2.0 3D, un motor de renderizado WebKit, un motor gráfico SGL, SSL y una biblioteca estándar de C Bionic. El sistema operativo está compuesto por 12 millones de líneas de código, incluyendo 3 millones de líneas de XML, 2,8 millones de líneas de lenguaje C, 2,1 millones de líneas de Java y 1,75 millones de líneas de C++.

Figura 7. Sistema Android 1.5 su lanzamiento oficial 27 abril 2009. Recuperado de https://developer.android.com/about/versions/android-1.5.html

2.4.4 La Play Store

Tiene una gran comunidad de desarrolladores creando aplicaciones

para extender la funcionalidad de los dispositivos. A la fecha, se ha llegado ya al 1.000.000 de aplicaciones disponibles para la tienda de aplicaciones oficial de Android: Google Play, sin tener en cuenta aplicaciones de otras tiendas no oficiales para Android como la tienda de aplicaciones Samsung Apps de Samsung, slideme de java y amazon appstore. Google Play es la tienda de aplicaciones en línea administrada por Google, aunque existe la posibilidad de obtener software externamente. La tienda F-Droid es completamente de código abierto así como sus aplicaciones, una alternativa al software privativo. Los programas están escritos en el lenguaje de programación Java. No obstante, no es un sistema operativo libre de malware, aunque la mayoría de ello es descargado de sitios de terceros. (Wikipedia, 2016)

Tabla 2. Aplicaciones disponibles por año y sus descargas

Año	Mes	Aplicaciones disponibles	Descargas hasta la fecha
2009	Marzo	2.300	
	Diciembre	16.000	
2010	Marzo	30.000	
	Abril	38.000	
	Agosto	80.000	1.000 millones
	Octubre	38.000	
2011	Mayo	200.000	4.500 millones
	Julio	250.000	6.000 millones
	Octubre	319.000	
	Diciembre	380.297	10.000 millones
2012	Enero	400.000	
	Octubre	675.000	
2013 2017	Enero	800.000	
	Julio	1.000.000	50.000 millones

Nota: Adaptado de "Play Store": Podemos notar cuantas descargas hicieron hasta el 2017. Recuperado de https://es.wikipedia.org/wiki/Google_Play

2.5 ARQUITECTURA DE ANDROID

Android es una pila de software de código abierto basado en Linux creada para una variedad amplia de dispositivos y factores de forma. En el siguiente diagrama se muestran los componentes principales de la plataforma Android.

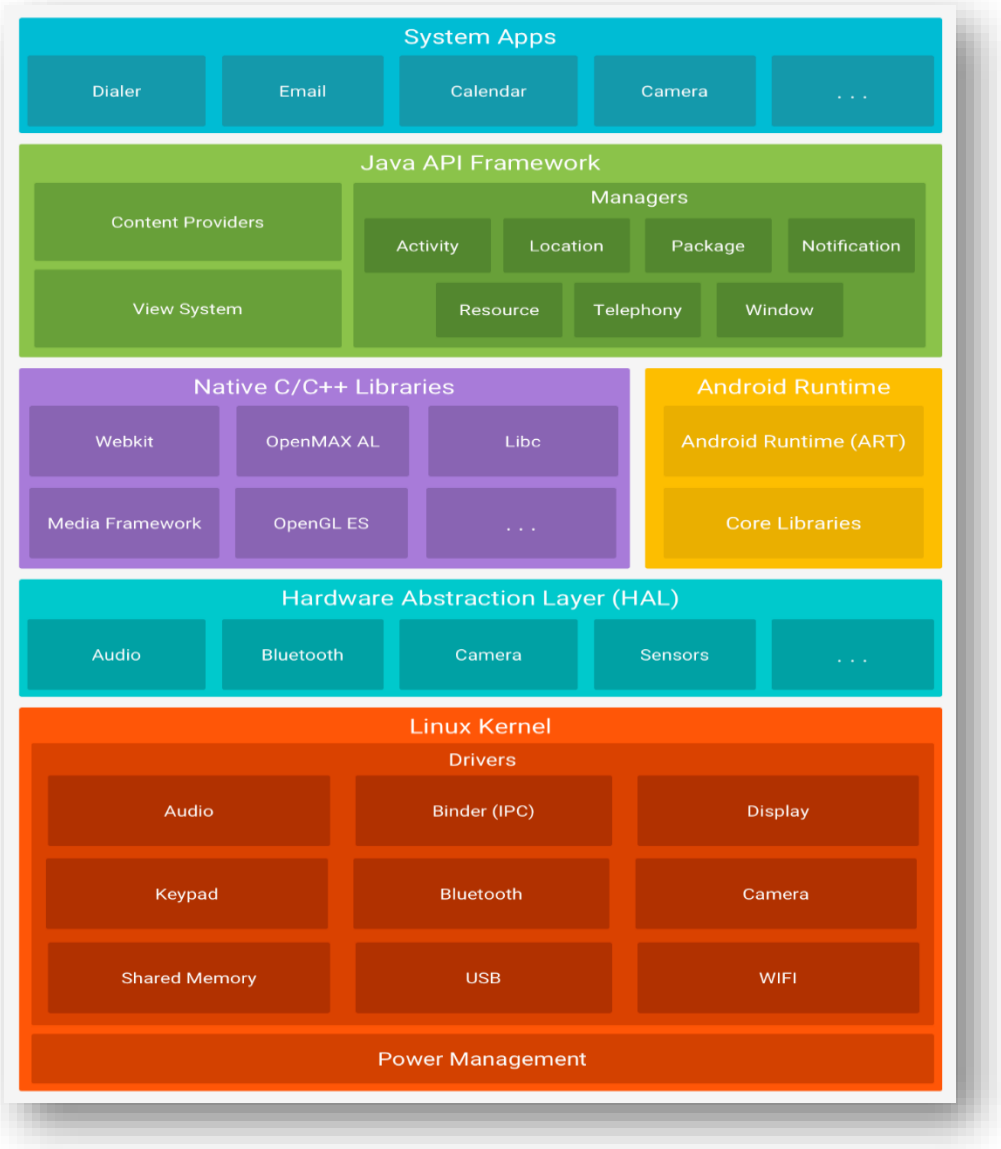


Figura 8. Arquitectura detallada sistema operativo android. Recuperado de https://developer.android.com/guide/platform/index.html?hl=es-419#linux-kernel

2.5.1 Apps del Sistema

En Android se incluye un conjunto de apps centrales para correo electrónico, mensajería SMS, calendarios, navegación en Internet y contactos, entre otros elementos. Las apps incluidas en la plataforma no tienen un estado especial entre las apps que el usuario elije instalar; por ello, una app externa se puede convertir en el navegador web, el sistema de mensajería SMS o, incluso, el teclado predeterminado del usuario (existen algunas excepciones, como la app Settings del sistema).

Las apps del sistema funcionan como apps para los usuarios y brindan capacidades claves a las cuales los desarrolladores pueden acceder desde sus propias apps. Por ejemplo, si en tu app se intenta entregar un mensaje SMS, no es necesario que compiles esa funcionalidad tú mismo; como alternativa, puedes invocar la app de SMS que ya está instalada para entregar un mensaje al receptor que especifiques.

2.5.2 Framework De La Java Api

Todo el conjunto de funciones del SO Android está disponible mediante API escritas en el lenguaje Java. Estas API son los cimientos que necesitas para crear apps de Android simplificando la reutilización de componentes del sistema y servicios centrales y modulares, como los siguientes:

Un sistema de vista enriquecido y extensible que puedes usar para compilar la IU de una app; se incluyen listas, cuadrículas, cuadros de

texto, botones e incluso un navegador web integrable.

Un administrador de recursos que te brinda acceso a recursos sin código, como strings localizadas, gráficos y archivos de diseño.

Un administrador de notificaciones que permite que todas las apps muestren alertas personalizadas en la barra de estado.

Un administrador de actividad que administra el ciclo de vida de las apps y proporciona una pila de retroceso de navegación común.

Proveedores de contenido que permiten que las apps accedan a datos desde otras apps, como la app de Contactos, o compartan sus propios datos.

Los desarrolladores tienen acceso total a las mismas API del framework que usan las apps del sistema Android.

2.5.3 Bibliotecas C/C++ Nativas

Muchos componentes y servicios centrales del sistema Android, como el ART y la HAL, se basan en código nativo que requiere bibliotecas nativas escritas en C y C++. La plataforma Android proporciona la API del framework de Java para exponer la funcionalidad de algunas de estas bibliotecas nativas a las apps. Por ejemplo, puedes acceder a OpenGL ES a través de la Java OpenGL API del framework de Android para agregar a tu app compatibilidad con los dibujos y la manipulación de gráficos 2D y 3D.

Si desarrollas una app que requiere C o C++, puedes usar el NDK de

Android para acceder a algunas de estas bibliotecas de plataformas nativas directamente desde tu código nativo.

2.5.4 Tiempo De Ejecución De Android

Para los dispositivos con Android 5.0 (nivel de API 21) o versiones posteriores, cada app ejecuta sus propios procesos con sus propias instancias del tiempo de ejecución de Android (ART). El ART está escrito para ejecutar varias máquinas virtuales en dispositivos de memoria baja ejecutando archivos DEX, un formato de código de bytes diseñado especialmente para Android y optimizado para ocupar un espacio de memoria mínimo. Crea cadenas de herramientas, como Jack, y compila fuentes de Java en código de bytes DEX que se pueden ejecutar en la plataforma Android. Estas son algunas de las funciones principales del ART:

Compilación ahead-of-time (AOT) y just-in-time (JIT); recolección de elementos no usados (GC) optimizada; mejor compatibilidad con la depuración, como un generador de perfiles de muestras dedicado, excepciones de diagnóstico detalladas e informes de fallos, y la capacidad de establecer puntos de control para controlar campos específicos.

Antes de Android 50 (nivel de API 21), Dalvik era el tiempo de ejecución del sistema operativo. Si tu app se ejecuta bien en el ART, también debe funcionar en Dalvik, pero es posible que no suceda lo contrario.

En Android también se incluye un conjunto de bibliotecas de tiempo de ejecución centrales que proporcionan la mayor parte de la funcionalidad del lenguaje de programación Java; se incluyen algunas funciones del lenguaje Java 8, que el framework de la Java API usa.

2.5.5 Capa de abstracción de hardware (HAL)

La capa de abstracción de hardware (HAL) brinda interfaces estándares que exponen las capacidades de hardware del dispositivo al framework de la Java API de nivel más alto. La HAL consiste en varios módulos de biblioteca y cada uno de estos implementa una interfaz para un tipo específico de componente de hardware, como el módulo de la cámara o de bluetooth. Cuando el framework de una API realiza una llamada para acceder a hardware del dispositivo, el sistema Android carga el módulo de biblioteca para el componente de hardware en cuestión.

2.5.6 Kernel de Linux

La base de la plataforma Android es el kernel de Linux. Por ejemplo, el tiempo de ejecución de Android (ART) se basa en el kernel de Linux para funcionalidades subyacentes, como la generación de subprocesos y la administración de memoria de bajo nivel.

El uso del kernel de Linux permite que Android aproveche funciones de seguridad claves y, al mismo tiempo, permite a los fabricantes de dispositivos desarrollar controladores de hardware para un kernel conocido. (Google, s.f.-a)

2.6 ANDROID SDK

La introducción de Java MIDlets ha expandido las expectativas de los desarrolladores pero las actuales capacidades de ejecución de los dispositivos móviles está superando las capacidades de los MIDlets por lo que hace falta una tecnología que permita aprovecharlas en todo su potencial, que permita desarrollar aplicaciones pequeñas, rápidas y capaces de manipular la plataforma tan igual como en las aplicaciones nativas para los sistemas operativos de escritorio.

Android se basa en una nueva onda de sistemas operativos móviles, diseñados para incrementar la potencialidad de estos equipos, en Android todas las aplicaciones están escritas usando la misma API creando una unicidad para que el sistema operativo pueda ejecutarlos en tiempo real a diferencia de una MIDLet, las aplicaciones Android son nativas y funcionales con las que contiene los siguientes elementos:

2.6.1 SQLLite Database

Rápido y eficiente para almacenar y recuperar datos, el ser compacto y efectivo es una sus mejores características.

Android provee de esta base de datos ligera con lo que sus aplicaciones pueden tomar ventaja frente a otras.

2.6.2 Shared Data and Interapplication Communication

Comunicación entre Aplicaciones y Compartición de Datos, Como recordamos las notificaciones son el modo estándar en que el

dispositivo móvil alerta al usuario. Usando la API se puede usar alertas audibles entre otros.

Finalmente el proveedor de contenidos es un medio para acceder a otras aplicaciones nativas, como Contact Manager al cual podemos incluso modificar el contenido.

2.6.3 P2P Services with Google Talk

Basada en las primeras versiones del SDK permite la comunicación estructurada peer-to-peer con los servidores que lo soporten, es una de las capacidades que menos ha sido explotada pero disponible para desarrolladores que deseen probar las capacidades de Android.

2.6.4 Extensive Media Support and 2D/3D Graphics

Pantallas más grandes y más brillantes, pantallas de mayor resolución han ayudado a hacer que los dispositivos móviles multimedia.

Para aprovechar al máximo el hardware disponible, Android proporciona bibliotecas de gráficos para el dibujo 2D Canvas y los gráficos 3D con OpenGL.

Android también ofrece amplias bibliotecas para la manipulación de imágenes fijas, vídeo y audio fi-les-incluyendo el MPEG4, H.264, MP3, AAC, AMR, JPG, PNG y GIF

2.6.5 Optimized Memory and Process Management

Los Procesos de Android y la gestión de la memoria son un poco

inusual. Al igual que Java y NET, Android utiliza su propio tiempo de ejecución y la máquina virtual para gestionar la memoria de la aplicación. A diferencia de cualquiera de estos marcos, el tiempo de ejecución Android también maneja los tiempos de vida del proceso. Android se asegura de respuesta de las aplicaciones, deteniendo y matando a los procesos cuando sea necesario para liberar recursos para las aplicaciones de mayor prioridad.

En este contexto, la prioridad se determina dependiendo de la aplicación con la que el usuario está interactuando. Asegurar que las aplicaciones están preparados para una muerte rápida, pero todavía son capaces de seguir respondiendo y actualizar o reiniciar en el fondo si es necesario, es una consideración importante en un entorno que no permite que las aplicaciones para el control de sus propias vidas. (Gironés, 2012)

2.7 ANDROID COMPARADO CON OTRAS PLATAFORMAS

Hoy en día, Android destaca dentro de las principales plataformas móviles y es el sistema operativo instalado en la mayoría de dispositivos móviles en el mercado. Se muestra las principales plataformas móviles disponibles en la actualidad.

Tabla 3. Comparativa de las principales plataformas móviles.

	Apple ¡OS 9	Android 7.0	Windows Phone8	Black Berry 10
Compañía	Apple	Open Handset Alliance	Microsoft	6l3ckBerry
Núcleo del SO	Mac OS X	Linux	Wndows NT	QNX
Licencia de software	Propietaria	Lfcrey abierto	Propietaria	Propietaria
Ano de lanzamiento	2007	2008	2010	1999
Fabricante único	Si	No	No	Si
Variedad de disoositivos	Modelo único	Muy Alta	Media	Baja
Soporte memoria externa	No	Sí	Si	Si
Motor del navegador web	WebKit	WebKit/Chromu m (Blnk)	Trident	WebKit
Tienda de aplicaciones	App Store	Google Play	Windows Marketplace	BlaCkBerry World
Número de aplicaciones	2 400.000 (seot 20161	2.000.000 fwn. 20161	700.000 foct 20161	270.000 (2016)
Coste publicar	$99 ¡ año	$25 una vez	$99/año	Sin coste
Otras tiendas sin supervisión	No	Si	No	Si
Familia CPU soportada	ARM	ARM. MIPS. x88	ARM	ARM
Soporte 84 bits	Si	Si	No	No
Máquina virtual	No	Dahrik/ART	.net	No
Lenguaje de programación	Objeaive-C. Swift	Java. C++	C#. Visual Basic. C++	C. C++. Java
Plataforma de desarrollo	Mac	Windows. Mac. Linux	Windows	Windows. Mac
Multi usuario	No	Si	No	No
Modo invitado	Si	Si	No	No

Nota: Tomado de"Comparativa con otras plataformas", por la Universidad Politécnica de Valencia. Recuperado de http://www.androidcurso.com/index.php/tutoriales-android/31-unidad-31-vision-general-y-entorno-de-desarrollo/98-comparativa-con-otras-plataformas

Otro aspecto fundamental a la hora de comparar las plataformas móviles es su cuota de mercado. En la siguiente gráfica podemos ver un estudio realizado por la empresa Gartner Group, donde se muestra la evolución del mercado de los sistemas operativos para móviles según el número de terminales vendidos. Podemos destacar la desaparición de la plataforma Symbian de Nokia, el declive continuo de BlackBerry, el estancamiento de la plataforma de

Windows, que parece que no despega, y el afianzamiento de la cuota de mercado de Apple en torno al 15 %. En la gráfica se puede apreciar como Apple consigue anualmente un aumento significativo de ventas coincidiendo con el lanzamiento de un nuevo terminal. Finalmente, cabe señalar el espectacular ascenso de la plataforma Android, que en cinco años ha alcanzado una cuota de mercado superior al 80%. (Valencia, 2017)

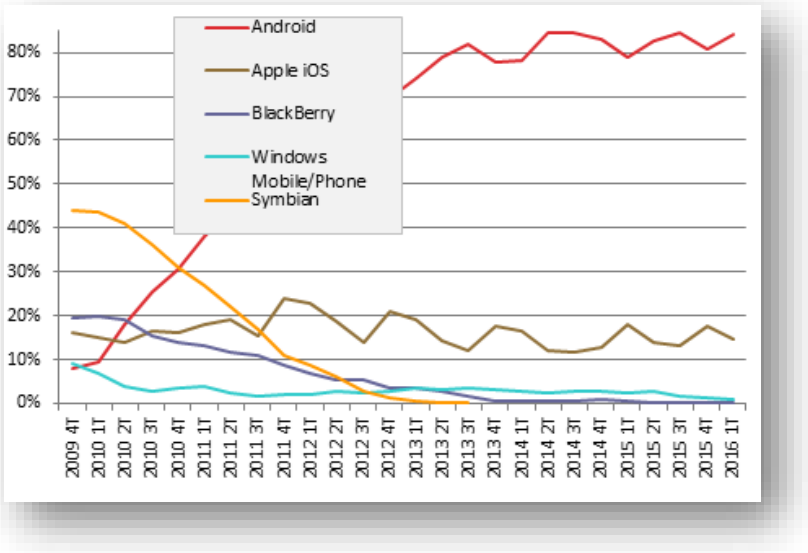


Figura 9. Variación porcentual de Android de los 5 años alcanzó un 80% en primer trimestre de 2016 y declive de symbian." Comparativa con otras plataformas". Recuperado de http://www.androidcurso.com/index.php/tutoriales-android/31-unidad-31-vision-general-y-entorno-de-desarrollo/98-comparativa-con-otras-plataformas

2.8 ANDROID EN EL MERCADO MUNDIAL

La compañía de investigación y análisis de tecnología Gartner reveló en su último informe respecto al estado del mercado de teléfonos móviles en el segundo cuarto de 2015, que los dispositivos Android son los más dominantes ocupando un 82.2% de la cuota de mercado, seguido por Apple con un 14.6% y con Microsoft cerrando el podio, aunque reduciendo su participación respecto al año pasado, con un 2.5%.

Tabla 4. Ventas mundiales de teléfonos inteligentes de diferente sistema operativo 2015

Worldwide Smartphone Sales to End Users by Operating System in 2Q15 (Thousands of Units)

Operating System	2Q15 Units	2Q15 Market Share (%)	2Q14 Units	2Q14 Market Share (%)
Android	271,010	82.2	243,484	83.8
iOS	48,086	14.6	35,345	12.2
Windows	8,198	2.5	8,095	2.8
BlackBerry	1,153	0.3	2,044	0.7
Others	1,229.0	0.4	1,416.8	0.5
Total	**329,676.4**	**100.0**	**290,384.4**	**100.0**

Source: Gartner (August 2015)

Nota: tomado de la "Empresa Gartner" en agosto 2015. Podemos ver como lidera en ventas el sistema android 2014. Recuperado de https://www.wayerless.com/2015/08/android-sigue-dominando-el-mercado-de-smartphones/

Según las estadísticas entregadas por Gartner entre la cuota de mercado perdida por Microsoft y lo ganado por Huawei, el fabricante chino podría proyectarse que de aquí a final de año se convierte en el tercer fabricante con mayor cantidad de unidades vendidas. Habrá que ver de qué forma afecta la llegada de Windows 10 Mobile para una tratar de recuperar terreno.

Para efectos de esta investigación se ha dispuesto que las unidades vendidas de Lenovo agrupen aquellas vendida por este fabricante en conjunto con Motorola. Junto con la dominación del sistema operativo para dispositivos móviles desarrollado por Google, Gartner advierte de un lento ritmo de crecimiento de venta de Smartphone a niveles que no se veían desde 2013. Al igual que otros analistas, una vez más el culpable de la desaceleración en el mercado de ventas apunta al mercado chino. (Mattassi, 2015)

Tabla 5.Compañías que vendió más Smartphone durante el año 2015

Table 3

Worldwide Mobile Phone Sales to End Users by Vendor in 2Q15 (Thousands of Units)

Company	2Q15 Units	2Q15 Market Share (%)	2Q14 Units	2Q14 Market Share (%)
Samsung	88,739	19.9	97,418	21.9
Apple	48,086	10.8	35,345	8.0
Microsoft	27,690	6.2	43,814	9.9
Huawei	26,119	5.9	18,219	4.1
LG Electronics	17,622	4.0	18,310	4.1
Lenovo*	16,626	3.7	19,266	4.3
Xiaomi	16,065	3.6	12,541	2.8
TCL Communication	15,733	3.5	13,923	3.1
ZTE	14,560	3.3	12,629	2.8
Micromax	9,884	2.2	8,578	1.9
Others	164,634.7	36.9	164,148.3	37.0
Total	**445,758.8**	**100.0**	**444,190.4**	**100.0**

Source: Gartner (August 2015)

*The results for Lenovo include sales of mobile phones by Lenovo and Motorola both in 2Q15 and 2Q14.

Nota: Tomado de la Empresa Gartner 2015, podemos ver en la nota que la empresa Samsung con 19.9% lidera en venta de sus dispositivos móviles luego seguido por Apple con 10.8 %. Recuperado de www.wayerless.com/2015/08/android-sigue-dominando-el-mercado-de-smartphones/

2.9 ANDROID EN EL PERÚ

Durante la primera mitad de año, han llegado al Perú, entre Smartphone y tablets más de 3.3 millones de dispositivos móviles, lo que demuestra la fortaleza de la penetración de dispositivos Smart en el Perú y lo competitivo que se ha vuelto el mercado de telefonía móvil.

En el caso de los Smartphone, son más de 2.9 millones de dispositivos que han llegado al Perú de enero a julio del 2015, de los cuales el 22.3% corresponden a Smartphone 4G LTE, funcionando el resto con redes 3G.

En cuanto a fabricantes, Samsung es la marca con mayor número de equipos (el 21.2% del total). Destaca, durante la primera mitad del 2015, el gran crecimiento que ha tenido Alcatel en el Perú, ocupando el segundo lugar del ranking con un 12.1% del total de Smartphone. En tercer lugar tenemos a LG con un 11.5% de los teléfonos que han llegado.

En el caso de las tablets, parece que hay una pequeña desaceleración en el número de tablets que van a ingresar este 2015. En la primera mitad del año han ingresado más de 410 mil tablets, por lo que es posible que este año tampoco lleguemos al Perú al millón de tablets, como ocurrió el 2013.

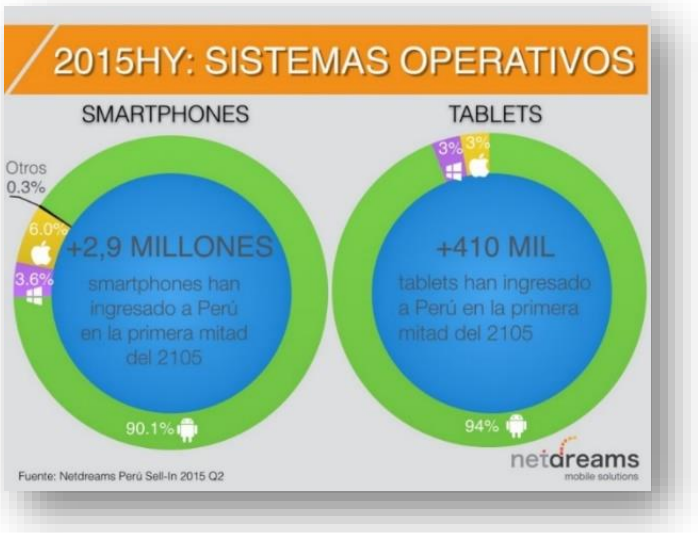


Figura 10. La relación de diferentes sistemas operativos: Android es el sistema operativo dominante. Recuperado de http://netdreams.pe/blog/mundo-movil/dispositivos-moviles-peru-sell-in-primer-semestre-2015-half-year-mas-de-3-3-millones-de-smartphones-y-tablets-llegan-al-peru/

Más del 90% de los dispositivos móviles que ha ingresado al Perú durante el 2015, utilizan el sistema operativo Android, con lo que sigue consolidándose como el sistema operativo de referencia. IOS tiene un pequeño crecimiento, llegando al 6% del total de Smartphone que han ingresado, a costa de Windows Phone que ve su cuota caer al 3.6%. BlackBerry casi ha desaparecido del mundo Smartphone y Firefox OS durante el primer semestre. Entre ambos ocupan el 0.3% del total de equipos.

En el caso de tablets, la supremacía de Android aumenta, siendo el sistema operativo del 94% de las tablets que han ingresado al Perú y el otro 6% se lo reparten IOS y Windows. (netdreams, 2015)

2.10 ENTORNO DE DESARROLLO EN ANDROID STUDIO

Android Studio es un entorno de desarrollo integrado para la plataforma Android. Fue anunciado el 16 de mayo de 2013, en la conferencia Google I/O, y reemplazó a Eclipse como el IDE oficial para el desarrollo de aplicaciones para Android. La primera versión estable fue publicada en diciembre de 2014.

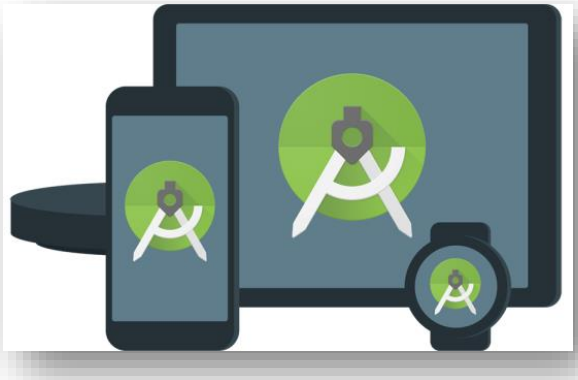


Figura 11. Android Studio. El lanzamiento oficial fue el 8 de diciembre 2014. Recuperado de https://developer.android.com/studio/features.html

Está basado en el software IntelliJ IDEA de JetBrains, y es publicado de forma gratuita a través de la Licencia Apache 2.0. Está disponible para las plataformas Microsoft Windows, Mac OS X y GNU/Linux.(IntelliJ, 2014)

2.11 BASE TEÓRICA PARA NUESTRA APLICACIÓN

2.11.1 Librería para la Encriptación en Java

Cipher: Esta clase proporciona la funcionalidad de un cifrado criptográfico para cifrado y descifrado. Forma el núcleo del marco de Java Criptographic Extensión (JCE).

Para crear un objeto Cipher, la aplicación llama al método getInstance del cifrado y le pasa el nombre de la transformación solicitada. Opcionalmente, se puede especificar el nombre de un proveedor.

Una transformación es una cadena que describe la operación (o conjunto de operaciones) que se realizará en la entrada dada, para producir alguna salida. Una transformación siempre incluye el nombre de un algoritmo criptográfico (por ejemplo, DES), y puede estar seguido por un modo de realimentación y un esquema de relleno.

Una transformación es de la forma:

- "algorithm/mode/padding" or
- "algorithm"

(En este último caso, se utilizan los valores por defecto específicos del proveedor para el modo y el esquema de relleno). Por ejemplo, lo siguiente es una transformación válida:

```
Cipher c =Cipher.getInstance("DES/CBC/PKCS5Padding");
```

Utilizando modos como CFB y OFB, los cifrados de bloque pueden cifrar datos en unidades más pequeñas que el tamaño de bloque real del cifrado. Cuando solicite dicho modo, puede especificar opcionalmente el número de bits a procesar a la vez añadiendo este número al nombre del modo como se muestra en las transformaciones "DES / CFB8 / NoPadding" y "DES / OFB32 / PKCS5Padding". Si no se especifica ningún número, se utiliza un valor predeterminado específico del proveedor. (Por ejemplo, el proveedor SunJCE utiliza un valor predeterminado de 64 bits para DES). Así, los cifrados de bloque pueden convertirse en cifrados de flujo orientados a bytes

utilizando un modo de 8 bits como CFB8 u OFB8.

Modos como el cifrado autenticado con datos asociados (AEAD) proporcionan garantías de autenticidad tanto para datos confidenciales como para datos asociados adicionales (AAD) que no están cifrados. (Para obtener más información sobre los algoritmos AEAD y AEAD como GCM / CCM, consulte el RFC 5116). Los datos confidenciales y AAD se pueden utilizar al calcular la etiqueta de autenticación (similar a una Mac). Esta etiqueta se añade al texto cifrado durante el cifrado y se verifica al descifrar.

Los modos AEAD como GCM / CCM realizan todos los cálculos de autenticidad AAD antes de iniciar los cálculos de autenticidad del texto cifrado. Para evitar que las implementaciones tengan que intermediar internamente el texto cifrado, todos los datos AAD deben ser suministrados a implementaciones GCM / CCM (a través de los métodos updateAAD) antes de procesar el texto cifrado (a través de los métodos update y doFinal).

Tenga en cuenta que el modo GCM tiene un requisito de unicidad en IVs utilizado en el cifrado con una clave determinada. Cuando se repiten los IVs para el cifrado GCM, tales usos están sujetos a ataques de falsificación. Por lo tanto, después de cada operación de cifrado utilizando el modo GCM, los llamantes deben reinicializar los objetos cifrados con parámetros GCM que tienen un valor IV diferente.

```
GCMParameterSpec s =...;
    cipher.init(..., s);
    // If the GCM parameters were generated by the provider, it can
    // be retrieved by:
    //
cipher.getParameters().getParameterSpec(GCMParameterSpec.class);
    cipher.updateAAD(...);  // AAD
    cipher.update(...);     // Multi-part update
    cipher.doFinal(...);    // conclusion of operation
   // Use a different IV value for every encryption
    byte[] newIv =...;
    s =newGCMParameterSpec(s.getTLen(), newIv);
    cipher.init(..., s);
    ...
```

javax.crypto: Proporciona las clases e interfaces para operaciones criptográficas. Las operaciones criptográficas definidas en este paquete incluyen cifrado, generación de claves y acuerdo de claves, y generación de Código de autenticación de mensajes (MAC).

El soporte para el cifrado incluye codificaciones simétricas, asimétricas, de bloque y de flujo. Este paquete también admite flujos seguros y objetos sellados.

Muchas de las clases proporcionadas en este paquete están basadas en proveedor. La propia clase define una interfaz de programación a la que pueden escribir las aplicaciones. Las implementaciones mismas pueden ser escritas por proveedores independientes de terceros y conectadas de forma transparente según sea necesario. Por lo tanto, los desarrolladores de aplicaciones pueden aprovechar cualquier número de implementaciones basadas en proveedores sin tener que agregar o volver a escribir código.(google, s.f.-b)

SecretKey	Una clave secreta (simétrica).

Classes

Cipher	Esta clase proporciona la funcionalidad de un cifrado criptográfico para cifrado y descifrado.decryption.
CipherInputStream	Un CipherInputStream se compone de un InputStream y un Cipher para que los métodos read () devuelvan datos que se lean en el InputStream subyacente pero que hayan sido procesados adicionalmente por el Cipher.
CipherOutputStream	Un CipherOutputStream se compone de un flujo de salida y un cifrado para que los métodos write () procesen primero los datos antes de escribirlos en el OutputStream subyacente.
CipherSpi	Esta clase define la interfaz de proveedor de servicios (SPI) para la clase Cipher.
EncryptedPrivateKeyInfo	Esta clase implementa el tipo EncryptedPrivateKeyInfo como se define en PKCS # 8.
ExemptionMechanism	Esta clase proporciona la funcionalidad de un mecanismo de exención, ejemplos de los cuales son la recuperación de claves, el debilitamiento de claves y el bloqueo de claves.
ExemptionMechanismSpi	Esta clase define la interfaz de proveedor de servicios (SPI) para la clase ExemptionMechanism.
KeyAgreement	Esta clase proporciona la funcionalidad de un acuerdo clave (o intercambio de claves).
KeyAgreementSpi	Esta clase define la interfaz de proveedor de servicios (SPI) para la clase KeyAgreement.
KeyGenerator	Esta clase proporciona la funcionalidad de un generador de claves secreto (simétrico).
KeyGeneratorSpi	Esta clase define la interfaz de proveedor de servicios (SPI) para la clase KeyGenerator.
Mac	Esta clase proporciona la funcionalidad de un algoritmo "Message Authentication Code" (MAC).
MacSpi	Esta clase define la interfaz de proveedor de servicios (SPI) para la clase Mac
NullCipher	La clase NullCipher es una clase que proporciona un "cifrado de identidad" - uno que no transforma el texto sin formato.
SealedObject	Esta clase permite a un programador crear un objeto y proteger su confidencialidad con un algoritmo criptográfico.
SecretKeyFactory	Esta clase representa una fábrica de claves secretas.
SecretKeyFactorySpi	Esta clase define la SPI (Service Provider Interface) para la clase SecretKeyFactory.

Excepciones

AEADBadTagException	Esta excepción se activa cuando un Cipher que opera en un modo AEAD (como GCM / CCM) no puede verificar la etiqueta de autenticación suministrada.
BadPaddingException	Esta excepción se activa cuando se espera un mecanismo de relleno particular para los datos de entrada, pero los datos no se rellenan correctamente.
ExemptionMechanismException	Esta es la excepción genérica de ExemptionMechanism.
IllegalBlockSizeException	Esta excepción se activa cuando la longitud de los datos proporcionados a un cifrado de bloque es incorrecta, es decir, no coincide con el tamaño de bloque del cifrado.
NoSuchPaddingException	Esta excepción se activa cuando se solicita un mecanismo de relleno particular pero no está disponible en el entorno.
ShortBufferException	Esta excepción se activa cuando un búfer de salida proporcionado por el usuario es demasiado corto para contener el resultado de la operación.

2.11.2 Librería para la Serializable en Java

Serializable Interfaz pública Serializable

Java.io.Serializable

Serializabilidad de una clase es habilitada por la clase que implementa la interfaz java.io.Serializable. Las clases que no implementan esta interfaz no tendrán ninguno de su estado serializado o deserializado. Todos los subtipos de una clase serializable son ellos mismos serializable. La interfaz de serialización no tiene métodos o campos y sólo sirve para identificar la semántica de ser serializable.

Para permitir la serialización de subtipos de clases no serializables, el subtipo puede asumir la responsabilidad de guardar y restaurar el estado de los campos públicos, protegidos y (si son accesibles) del paquete del supertipo. El subtipo puede asumir esta responsabilidad

sólo si la clase que se extiende tiene un constructor no-arg accesible para inicializar el estado de la clase. Es un error declarar una clase Serializable si este no es el caso. El error se detectará en tiempo de ejecución.

Durante la deserialización, los campos de las clases no serializables se inicializarán utilizando el constructor público o protegido no-arg de la clase. Un constructor no-arg debe ser accesible a la subclase que es serializable. Los campos de las subclases serializables se restaurarán de la secuencia.

```
private void writeObject(java.io.ObjectOutputStream out)
    throws IOException
private void readObject(java.io.ObjectInputStream in)
    throws IOException, ClassNotFoundException;
private void readObjectNoData()
    throws ObjectStreamException;
```

El método writeObject es responsable de escribir el estado del objeto para su clase particular para que el método readObject correspondiente pueda restaurarlo. El mecanismo predeterminado para guardar los campos del objeto se puede invocar llamando a out.defaultWriteObject. El método no necesita ocuparse del estado perteneciente a sus superclases o subclases. Estado se guarda escribiendo los campos individuales en ObjectOutputStream utilizando el método writeObject o utilizando los métodos para tipos de datos primitivos admitidos por DataOutput.

El método readObject es responsable de leer el flujo y restaurar los campos de clases. Puede llamar a in.defaultReadObject para invocar

el mecanismo predeterminado para restaurar los campos no estáticos y no transitorios del objeto. El método defaultReadObject utiliza la información de la secuencia para asignar los campos del objeto guardado en la secuencia con los campos correspondientemente nombrados en el objeto actual. Esto maneja el caso cuando la clase ha evolucionado para agregar nuevos campos. El método no necesita ocuparse del estado perteneciente a sus superclases o subclases. Estado se guarda escribiendo los campos individuales en ObjectOutputStream utilizando el método writeObject o utilizando los métodos para tipos de datos primitivos admitidos por DataOutput.

El método readObjectNoData es responsable de inicializar el estado del objeto para su clase particular en el caso de que la secuencia de serialización no enumera la clase dada como una superclase del objeto que está siendo deserializado. Esto puede ocurrir en los casos en que la parte receptora usa una versión diferente de la clase de la instancia deserializada que la parte que envía, y la versión del receptor extiende las clases que no son extendidas por la versión del remitente. Esto también puede ocurrir si el flujo de serialización ha sido manipulado; Por lo tanto, readObjectNoData es útil para inicializar correctamente objetos deserializados a pesar de un flujo de fuente "hostil" o incompleta.

Las clases serializables que necesitan designar un objeto alternativo para ser utilizado al escribir un objeto en el flujo deben implementar este método especial con la firma exacta:

```
ANY-ACCESS-MODIFIER Object writeReplace() throws ObjectStreamException;
```

Este método writeReplace es invocado por serialización si el método existe y sería accesible desde un método definido dentro de la clase del objeto que se está serializando. Por lo tanto, el método puede tener acceso privado, protegido y paquete privado. El acceso de subclase a este método sigue las reglas de accesibilidad de java.

Las clases que necesitan designar un reemplazo cuando se lee una instancia del mismo deben implementar este método especial con la firma exacta.

```
ANY-ACCESS-MODIFIER Object readResolve() throws ObjectStreamException;
```

Este método readResolve sigue las mismas reglas de invocación y reglas de accesibilidad que writeReplace.

El tiempo de ejecución de serialización asocia a cada clase serializable un número de versión, denominado serialVersionUID, que se utiliza durante la deserialización para verificar que el remitente y el receptor de un objeto serializado han cargado clases para ese objeto que son compatibles con respecto a la serialización. Si el receptor ha cargado una clase para el objeto que tiene un serialVersionUID distinto al de la clase del remitente correspondiente, la deserialización dará lugar a una *InvalidClassException*. Una clase serializable puede declarar su propio serialVersionUID explícitamente declarando un

campo llamado "serialVersionUID" que debe ser estático, final y de tipo long:

```
ANY-ACCESS-MODIFIER static final long serialVersionUID = 42L;
```

Si una clase serializable no declara explícitamente un serialVersionUID, entonces el tiempo de ejecución de serialización calculará un valor serialVersionUID predeterminado para esa clase basado en varios aspectos de la clase, tal como se describe en la Especificación de serialización de objetos Java (TM). Sin embargo, se recomienda encarecidamente que todas las clases serializables declaren explícitamente los valores serialVersionUID, ya que la computación serialVersionUID predeterminada es muy sensible a los detalles de la clase que pueden variar dependiendo de las implementaciones del compilador y, por lo tanto, puede resultar en InvalidClassExceptions inesperadas durante la deserialización. Por lo tanto, para garantizar un valor serialVersionUID coherente a través de diferentes implementaciones de compilador java, una clase serializable debe declarar un valor serialVersionUID explícito. También se recomienda encarecidamente que las declaraciones explícitas serialVersionUID usen el modificador privado cuando sea posible, ya que dichas declaraciones se aplican sólo a la clase inmediatamente declarante - los campos serialVersionUID no son útiles como miembros heredados. Las clases de matrices no pueden declarar un serialVersionUID explícito, por lo que siempre tienen el

valor calculado por defecto, pero el requisito de coincidencia de los valores serialVersionUID no se aplica a las clases de matrices. La implementación de Android del cómputo serialVersionUID cambiará ligeramente para algunas clases si está orientado a android N. Para preservar la compatibilidad, este cambio solo se habilita si la versión del SDK de destino de la aplicación está establecida en 24 o superior. Se recomienda utilizar un campo serialVersionUID explícito para evitar problemas de compatibilidad. (google, s.f.-c).

2.12 ESTADÍSTICA: PRUEBA DE MCNEMAR

La prueba de McNemar se utiliza para decidir si puede o no aceptarse que determinado "tratamiento" induce un cambio en la respuesta dicotómica o dicotomizada de los elementos sometidos al mismo, y es aplicable a los diseños del tipo "antes-después" en los que cada elemento actúa como su propio control.

Los resultados correspondientes a una muestra de elementos se disponen en una tabla de frecuencias 2 x 2 para recoger el conjunto de las respuestas de los mismos elementos antes y después. El aspecto general de dicha tabla, en la que los signos + y - se utilizan para representar las diferentes respuestas, es el siguiente:

Antes/Después	-	+
-	a	b
+	c	d

En las celdas de la tabla, a es el número de elementos cuya respuesta es la misma, -; b es el número de elementos cuya respuesta es - antes del "tratamiento" y + después de éste; c es el número de elementos que han cambiado de + a -; y d es el número de elementos que mantienen la respuesta +.

Por tanto, b + c es el número total de elementos cuyas respuestas han cambiado, y son los únicos que intervienen en el contraste. La hipótesis nula es que el "tratamiento" no induce cambios significativos en las respuestas, es decir, los cambios observados en la muestra se deben al azar, de forma que es igualmente probable un cambio de + a - que un cambio de - a +. Así pues, si H_0 es cierta, de los b + c elementos cuya respuesta ha cambiado es de esperar que (b + c)/2 hayan pasado de + a -, y (b + c)/2 hayan pasado de - a +. En otras palabras, si H_0 es cierta, la frecuencia esperada en las correspondientes celdas es (a + b)/2.

La hipótesis alternativa puede ser no direccional, cuando postula que la probabilidad de un cambio de + a - tiene distinta probabilidad que un cambio de - a +, o direccional, cuando predice que un cambio de - a + es más (o menos) probable que un cambio de + a -.

Si H_0 es cierta, el estadístico tiene distribución aproximadamente chi-cuadrado con 1 grado de libertad. La aproximación es más precisa si se realiza la corrección de continuidad de Yates, quedando el estadístico. Nuestra formula es la siguiente:

$$X^2 = \frac{|A - D| - 1^2}{A + D}$$

La hipótesis nula, de que ambos tipos de cambio son igualmente probables, se rechaza si el valor del estadístico se encuentra en la región crítica.(Gómez-Biedma)

2.13 MODELO Y METODOLOGÍA DE DESARROLLO

2.13.1 Metodologías Ágiles

Los enfoques ágiles se utilizan normalmente en el desarrollo de software para ayudar a las empresas a responder a la imprevisibilidad, proponen alternativas a la gestión tradicional de proyectos y ofrecen oportunidades para evaluar la dirección en todo el ciclo de vida de desarrollo. Las metodologías ágiles se caracterizan por ser interactivas e incrementales, ya que se centran en la repetición de los ciclos de trabajo abreviados así como del producto funcional.

Los principios de las metodologías ágiles son:

- Satisfacción del cliente por la entrega de software útil.
- Bienvenidos cambios en los requisitos, incluso en etapas tardías de la programación.
- El software funcional es entregado con frecuencia (semanas en lugar de meses).
- El software funcional es la principal medida de progreso.

- Desarrollo sostenible, capaz de mantener un ritmo constante.
- Cooperación diaria y cercana entre los encargados de negocios y desarrolladores.
- Conversación cara a cara es la mejor forma de comunicación.
- Los proyectos se construyen en torno a individuos motivados, en los cuales se debe confiar.
- Atención constante a la excelencia técnica y al buen diseño.
- Simplicidad.
- Equipos auto-organizados.
- Adaptación regular a circunstancias cambiantes.

Algunas de las metodologías ágiles más relevantes son:

2.13.2 Proceso de desarrollo Extreme Programming (XP)

XP es una metodología que está centrada en las mejores prácticas de programación, las cuales se describen a continuación:

Planificación del juego. El cliente decide el alcance y el calendario de versiones basados en tiempos estimados por los programadores.

Pequeñas versiones. Que se realizan diaria o mensualmente, lo que permite poner el sistema en producción en pocos meses.

La metáfora. La forma del sistema está definida por una metáfora o un conjunto de metáforas compartidas entre el cliente y los programadores.

Diseño simple. En cada momento el diseño ejecuta todas las pruebas, comunica todo lo que los programadores quieren comunicar, no contiene código duplicado y tiene el menor número de clases y métodos posibles.

Las pruebas. Los programadores escriben pruebas unitarias minuto a minuto, y todas deben ser ejecutadas correctamente.

La refactorización. El diseño del sistema está desarrollado a través de transformaciones del diseño existente que mantiene todas las pruebas en funcionamiento.

Programación en pares. Todo el código de producción está escrito por dos personas en una pantalla/teclado/mouse.

Integración continúa. Lo que permite que nuevo código sea integrado al sistema actual en unas pocas horas.

Propiedad colectiva. Todos los programadores pueden mejorar cualquier código en cualquier lugar del sistema en cualquier momento si tienen la oportunidad.

El cliente en su lugar. Lo que implica tener al cliente todo el tiempo con el equipo.

40 horas semanales. Nadie puede trabajar una segunda semana de sobretiempo pues sino sería una señal de problemas que deben ser resueltos.

Espacio de trabajo abierto. El equipo trabaja en una amplia sala con pequeños cubículos.

Solo reglas. El equipo puede cambiar las reglas en cualquier momento siempre y cuando estén de acuerdo en la forma en que evaluarán los efectos de la variación.(Vélez, 2015)

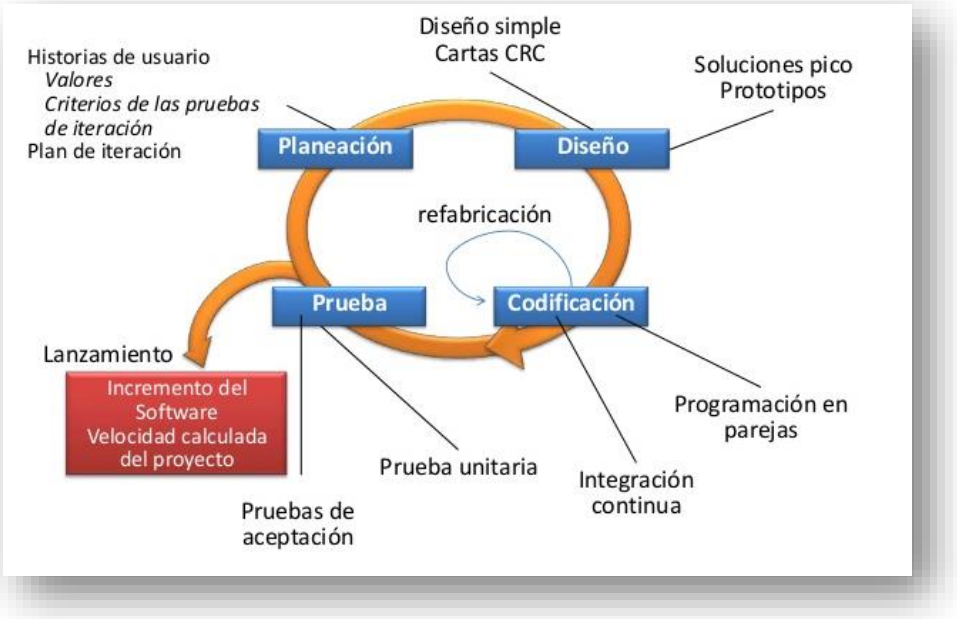


Figura 12. Programación Extrema. Tomado de Desarrollo ágil de software, por Luis Fernández. Recuperado de https://es.slideshare.net/coesiconsultoria/4-desarrollo-gil-del-software

XP (Extreme programming) es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo XP se basa en realimentación

continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todo los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos muy cambiantes.

La metodología ágil XP expone ciertos criterios que indican por qué se debe utilizar esta metodología para el desarrollo de software, estos criterios son:

- Software que funcione es más importante que documentación exhaustiva, es decir si el software no funciona los documentos no valen nada, o la colaboración con el clientes más importante que la negociación de contratos.
- El éxito de los proyectos se basa en una realimentación permanente.
- La respuesta ante el cambio es más importante que el seguimiento de un plan.

XP se eligió como la metodología ágil entre otros motivos porque:

Los requerimientos no se tenían exactamente definidos y fueron cambiando a lo largo del proceso de desarrollo, o Los clientes estuvieron involucrados directamente con el proyecto todo el tiempo.

El proyecto fue realizado bajo prácticas que fomentaron la comunicación, tanto cliente-desarrollador como entre desarrolladores-desarrolladores, sencillez, el proyecto se realizaba según las

necesidades del módulo que se fuera implementado, realimentación, lo que permitía corregir errores inmediatos para evitar complicaciones a futuro, y valentía que se refiere a la aceptación al cambio, ya que los requerimientos fueron variando a lo largo del proyecto, era necesario estar dispuestos a aceptar dichos cambios, a continuación se muestra la tabla de evaluación de la calidad del producto de software que se aplicó para la validación del sistema de información.

CAPÍTULO III

MATERIALES Y MÉTODOS

3.1 MATERIALES

Se utilizó una ficha de encuesta para nuestros resultados de nuestra aplicación *(anexo 1)*

3.2 POBLACIÓN

La población de investigación estuvo constituida por todas las personas adquirieron un equipo en el centro de atención y pagos comunica sur de claro que se encuentra en nuestra localidad del Jr. Los Incas N° 210 la prueba ser realizo dentro del mes de enero.

3.3 MUESTRA

El tipo de muestreo utilizado fue probabilístico empleándose para ello un Muestreo aleatorio simple y estuvo constituido por dueños de los celulares

La población fue finita es decir (< 100 unidades de análisis) se calculó el tamaño de la muestra:

$$n = \frac{Z^2 p * q * N}{Ne^2 + Z^2 p * q}$$

Donde:
N =100.
n = tamaño de la muestra,
E => 0.10.
Z= desviación estándar (para un margen de confianza de 95% es 1.96).
p = 0.90.
q= 0.90.

Entonces se tiene: n= 49

Nuestro tamaño de muestra para la investigación fue de 49 usuarios.

Si trabajamos con un 10% de error tendríamos aproximadamente 49 usuarios, de los cuales tomado en cuenta los factores de inclusión, es decir, solo aquellos usuarios que les importa la seguridad de sus archivos y que manejan un **SMARTPHONES**, reducimos a 15 usuarios por la encuesta que se plateo quedo satisfecho con nuestra aplicación por eso se redujo la muestra.

3.4 MÉTODO DE RECOLECCIÓN DE LA INFORMACIÓN

3.4.1 Recolección de datos para variables independientes

Se utilizó un cuestionario (VER ANEXO 01).

La recolección de datos para la presente investigación se realizó a través de la adquisición de dispositivos móviles, cuyos datos se recolecto de la tienda centro de atención y pagos comunica ***sur de claro*** que se encuentra en nuestra localidad del Jr. Los Incas N° 210 donde el estudio duró un mes cuyos dueños adquieren su Smartphone nuevo y se les instaló la aplicación **BÓVEDA** aplicación,

durante ese mes se vendieron 100 nuevos Smartphone con sus respectivos números.

3.4.2 Material Experimental

El presente trabajo se desarrolló con la información de todos los usuarios que adquirieron un Smartphone de muestra

3.5 METODOLOGÍA DE LA INVESTIGACIÓN

La metodología ágil elegida para desarrollar la investigación – acción del presente proyecto es Extreme Programming (XP). XP es un proceso de desarrollo de software, el cual busca resolver frecuentes problemas de la ingeniería del software.

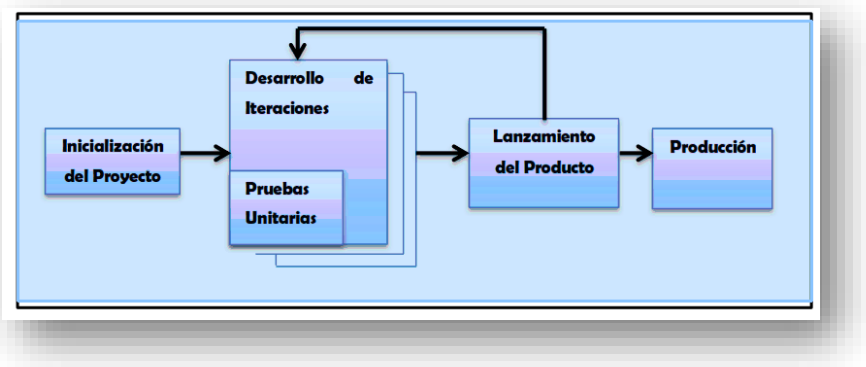


Figura 13. Metodología XP

En XP, el proceso de desarrollo está dividido en:

Iteraciones, las cuales duran aproximadamente de 1 a 4 semanas y cuyo fin es entregar software útil con funcionalidades adicionales al cliente.

Al principio de cada iteración, se lleva a cabo una etapa de “en el que se plantean los requisitos de los clientes y las historias de usuarios, que serán

elegidos e implementados en la siguiente iteración. Después de ello, se llevan a cabo pruebas de aceptación para cada historia de usuario y asegurarse de que todas las historias han sido implementadas correctamente.

Cabe mencionar, que en el presente Proyecto no se cubrirán las etapas de Lanzamiento del producto ni la Producción.

3.5.1 Análisis de los Roles

Hay que tener en cuenta que los desarrolladores del proyecto era una persona por lo que los roles definidos en XP fueron ocupados por mi persona.

Programador: El código fuente fue hecho solo por mi persona. Pero solo el desarrollador asignó la prioridad a las historias de usuario y decidió cuales se implementaron en cada interacción.

Entrenador (Coach): Mi persona fue el responsable del proceso global también de realizar las guías para las practicas XP y se siguiera el proceso correctamente.

Algunos roles definidos en XP (consultor y gestor) no fueron tomados en cuenta para este proyecto, bien porque no fuese necesario o porque el tamaño tan reducido del equipo de trabajo no lo permita. (Mejía Pacheco, 2016)

3.6 MÉTODOS DE TRATAMIENTO DE DATOS

3.6.1 Prueba de Mcnemar

		Despues	
		sin uso app	con app
Antes	con app	34	15
	sin app	48	3

$$X^2 = \frac{|A - D| - 1^2}{A + D}$$

Donde:
A=34
B=3

$$X = 24.32$$

3.7 MÉTODOS DE RECOPILACIÓN DE DATOS

Para nuestra investigación la validación de nuestros datos como nos indica mcnemar en el antes y después de utilizar muestra aplicación para ello se hizo una pequeña encuesta a la población.

3.8 DESARROLLO DE APLICACIONES MÓVILES

Para desarrollar aplicaciones para dispositivos Android, que se utilizó la herramienta android Studio y Java Development. Una vez que se haya descargado e instalado, ya que pueden invocar directamente las herramientas que necesita, mientras que el desarrollo de aplicaciones.

Sin embargo, usted se optar por desarrollar con otro IDE o un editor de texto simple e invocar las herramientas de la línea de comandos o con scripts. Esta

es una forma menos eficiente para el desarrollo, ya que a veces tendrá directo que llamar a las herramientas de línea de comandos de forma manual

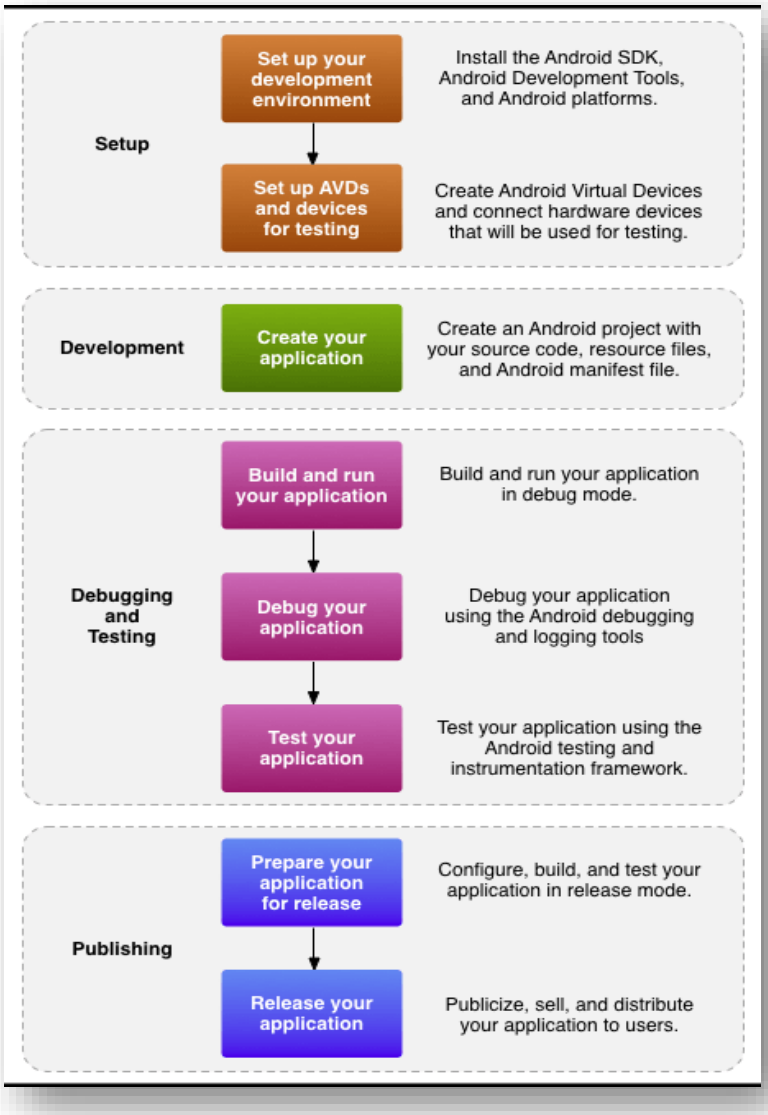


Figura 14. Pasos Básicos para nuestra Aplicación. Recuperado de https://androidfeed.wordpress.com/2012/09/10/workflow/

3.8.1 Instalación.

Durante esta fase de instalar y configurar el entorno de desarrollo. También se crea Android Virtual Devices (AVDs) y conectar los dispositivos de hardware en el que se pueden instalar las

aplicaciones. Consulte Administración de dispositivos virtuales y Uso de dispositivos de hardware para obtener más información.

3.8.2 Desarrollo

Durante esta fase de configurar y desarrollar su proyecto Android, que contiene todo el código fuente y los archivos de recursos para su aplicación. Para obtener más información, vea Crear un proyecto Android.

3.8.3 Depuración y Prueba

Durante esta fase se genera el proyecto en un paquete depurable. Apk que se puede instalar y ejecutar en el emulador o en un dispositivo con Android, construye se generan cada vez que se guarda proyecto. Si se está utilizando otro IDE, usted puede construir su proyecto utilizando Ante e instalarlo en un dispositivo que utiliza adb. Para obtener más información, vea Crear y ejecutar la aplicación.

A continuación, depurar la aplicación utilizando un depurador JDWP compatible con las herramientas de depuración y registro que se proporcionan con el SDK de Android. Eclipse ya viene empaquetado con un depurador compatible. Para obtener más información, vea Depurar la aplicación con el SDK y las herramientas de depuración de registro.

Por último, probar la aplicación utilizando diversas herramientas de pruebas de Android SDK. Para obtener más información, consulte Comprobación de la aplicación de la prueba y el marco de

instrumentación.

3.8.4 Publicación

Durante esta fase se configura construye tu solicitud de liberación y distribuir su aplicación a los usuarios mediante la play strory. (Androidfeed, 2012)

3.9 REQUERIMIENTOS DE SISTEMA

Requerimientos funcionales:

Se han definido para el sistema, los siguientes puntos más relevantes debe poder realizar:

- R1: Deben existir la seguridad que se aplicó para cada usuario
- R2: El sistema debe estar en la capacidad restaurar los archivos protegidos

Requerimientos no funcionales:

- Aplicación multiplataforma.
- Interfaz agradable para un fácil entendimiento de la App, Disponibilidad en la play Store.
- Rendimiento de la App brindará un servicio óptimo ya que es diseñado para que funcione.

3.10 RECURSOS COMPUTACIONALES:

Hardware

- Computador con Procesador i3 o superior.
- Equipo Celular con Android OS 4.3.0 o superior.
- Cable de datos.
- Memoria RAM de 8GB o superior.

Software

- Microsoft® Windows® 8/7/Vista/2003 (32 o 64-bit)
- 2 GB RAM mínimo, 4 GB RAM recomendado
- 400 MB espacio de disco duro
- 1280 x 800 resolución de pantalla mínimo
- Java Development Kit (JDK) 7

CAPITULO IV

RESULTADOS

4.1 PROCESO DE DESARROLLO DE LA APP

4.1.1 Análisis de Sistema

4.1.1.1 Descripción del Sistema Actual

En la actualidad tenemos muchos problemas con las pérdidas de nuestros celulares, por ejemplo, perdida, robo, en estos casos el app se diseñó con el objetivo de proteger nuestros archivos personales que no caigan en manos dudosas y cuyos archivos no sean utilizados de forma maliciosa.

Lo que lleva a esta investigación fue diseñar una app amigable donde cada usuario pueda manipular de manera fácil y sencilla.

En esta app lo que se utilizó fue encriptación y serializar para su mayor protección de nuestro archivos documentos imágenes.

Que nos diferencia de las demás App es la protección en nuestra memoria externa como son conocidos las micro SD, en este caso se hizo una investigación cuyas App de competencia solo se centra en la memoria interna cuya capacidad no es tan grande como son las memorias externas.

4.1.1.2 Análisis de Viabilidad

- Viabilidad Operativa: la App tiene la capacidad de poner usuario y clave personal solo conocido por el usuario.
- Viabilidad Técnica: los usuarios tiene que tener Android en su Smartphone para su correcta manipulación.

4.1.1.3 Análisis de Planificación

Historias de usuario

En estas los clientes describieron brevemente las características que la aplicación debía poseer:

Tabla 6. Historial de usuario y numero de tareas

NUMERO	HISTORIA DE USUARIO	TAREA
1	Creación de la Base de datos, donde se almacenara la información	Diseño e implementación de la base de datos
2	La aplicación debe ser amigable y de fácil uso para los posibles interesados en ella	Buscar un diseño amigable y agradable para todos los usuarios
3	Creación de una aplicación que guarde la información personal SD	Diseño e implementación de los módulos de inserción de datos
4	La aplicación debe tener las restricciones necesarias para evitar errores en la base de datos	Se deben diseñar e implementar las validaciones respectivas en los diferentes campos que Se ingresaran en la base de datos
5	La aplicación debe estar disponible en la play store	Elegir una herramienta de desarrollo para aplicaciones app

Nota: Análisis de Planificación. Vemos brevemente que debía poseer nuestra aplicación. Elaboración Propia.

4.2 ENTORNOS DE DESARROLLO

4.2.1 Proceso de Instalación de Android

Android ofrece en su web developer el instalable de la instalación, solo tienes que hacer click en "Download Android Studio for Windows", aceptar las condiciones de uso y seleccionar un destino del fichero instalable.

Si usa otro sistema operativo (Mac OS o Linux) tiene estos ejecutables al final de la página, donde pone "All Android Studio Packages"

Android Studio. https://developer.android.com/studio/index.html

https://developer.android.com/studio/index.html

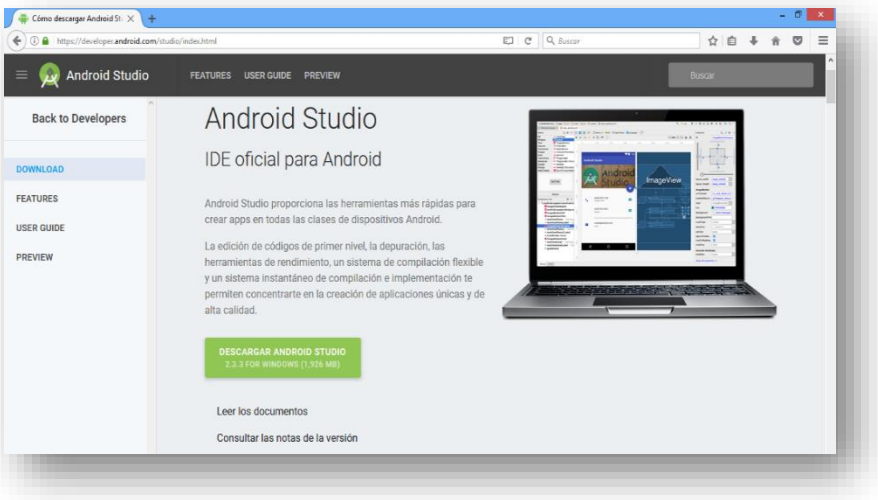


Figura 15. Android Studio interfas de la web. Elaboracion propia

Java JDK 6.

http://www.oracle.com/technetwork/java/javase/downloads/index.html

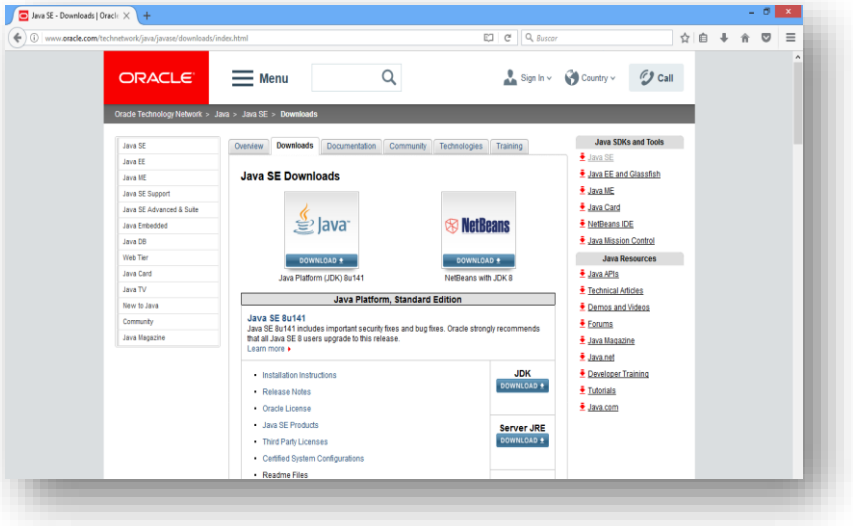


Figura 16. Java JDK 6 Interfas de la Web. Elaboración Propia

También se necesita instalar Java JDK, el link que dejé arriba e instálalo, es sencillo, pulsa siguiente, siguiente, siguiente y Finalizar (instala esto primero).

Descargado Android Studio

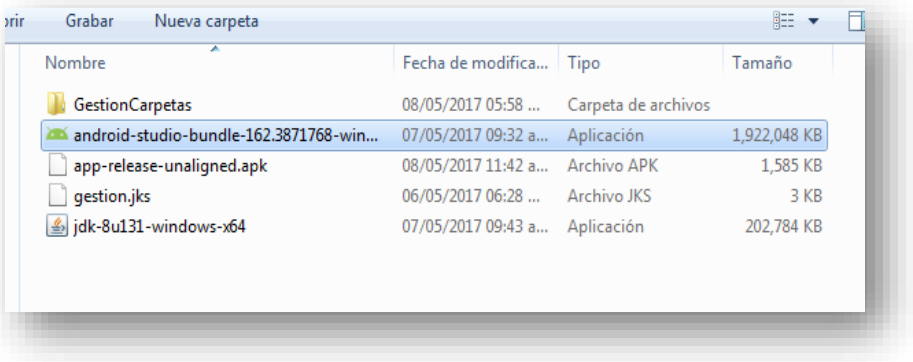


Figura 17. Proceso para la instalación Android Studio Paso 1. Elaboración Propia

Cuando se termine la descarga del fichero de Android Studio da doble click, se iniciará un asistente, no es complicado, da a siguiente, y deja las opciones por defecto.

Figura 18. Proceso para la instalación Android Studio Paso 2. Elaboración Propia

Para aceptar el acuerdo de licencia hacemos clic en el botón I Agree.

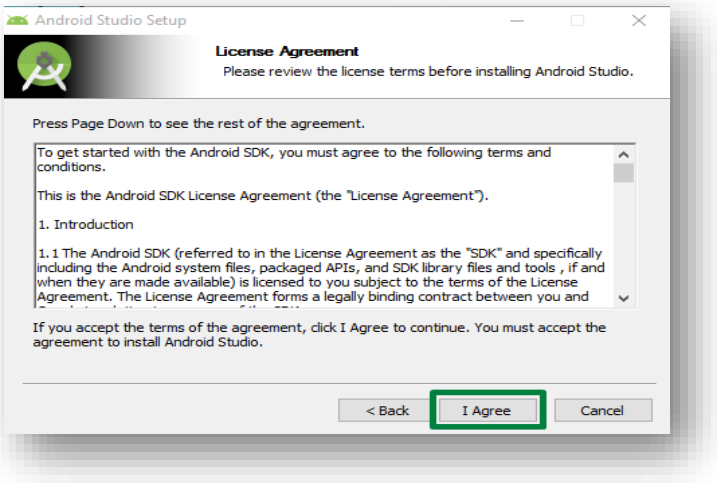


Figura 19, Proceso para la instalación Android Studio Paso 3. Elaboración Propia

Llegará a una pantalla donde se piden las localizaciones del programa en si (Android Studio) y la ubicación del SDK, que es donde se descargarán y guardarán todas las APIs y complementos que se necesitan para programar, es recomendable que se memorice.la ubicación del SDK

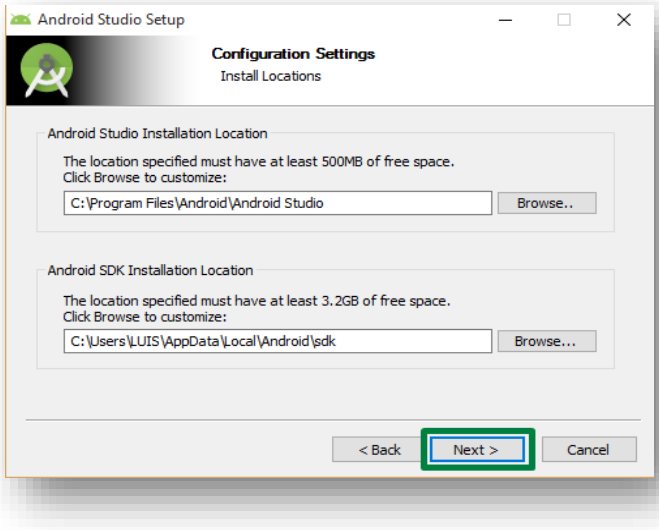


Figura 20. Proceso para la instalación Android Studio Paso 4. Elaboración Propia.

Una vez que acabe, seguir pulsando siguiente, hasta que llegue a esta ventana, si es la primera vez que instala Android Studio, selecciona la segunda opción (que es la opción por defecto), si tenías alguna otra versión, tendrá que seleccionar la primera.

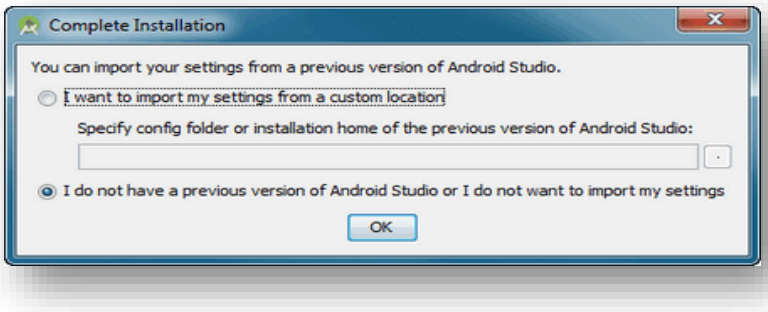


Figura21. Proceso para la instalación Android Studio Paso 5. Elaboración Propia.

Si todo salió bien, debe tener algo parecido a esto, pulsa "Finish" y habrá acabado.

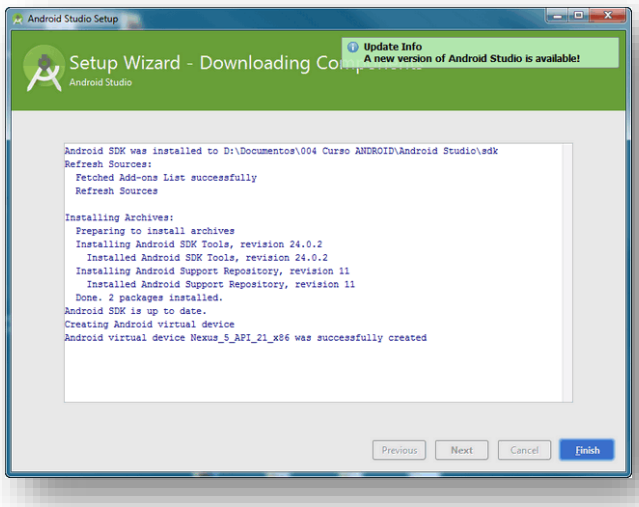


Figura 22 Proceso para la instalación Android Studio Finalizando. Elaboración Propia.

Esta es la pantalla principal de Android Studio, desde aquí se hacen todas las operaciones, es bastante intuitivo, para instalar otras APIs y buscar actualizaciones, dale a "Configure" y veras el "SDK Manager" que es el "buscador" de APIs y actualizaciones necesarias para programar Android.

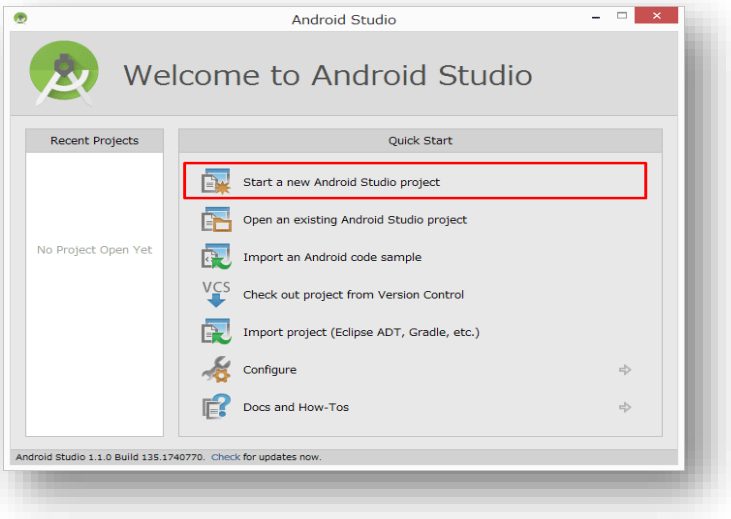


Figura 23. Iniciando Android Studio Agregar nuevo proyecto. Elaboración Propia.

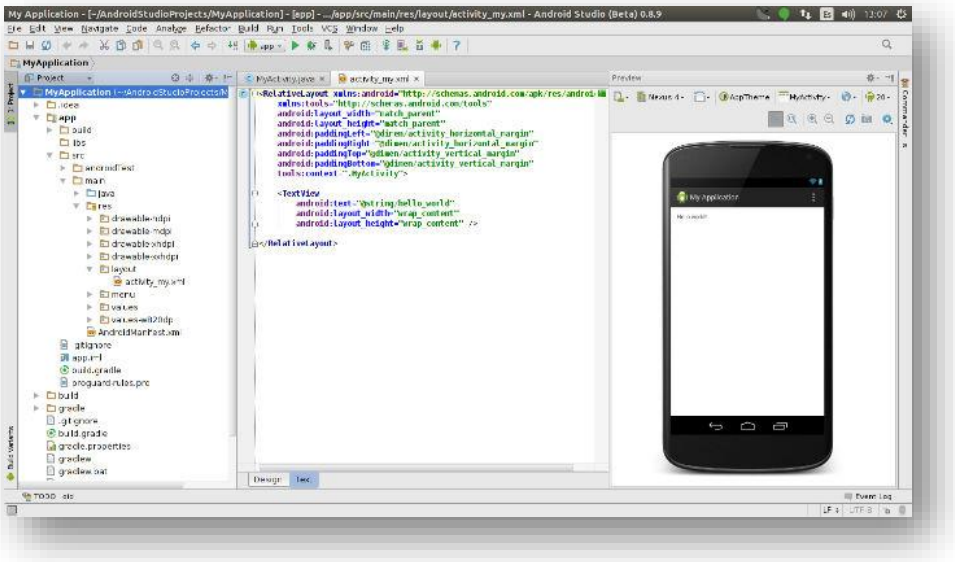


Figura 24.Menú de iniciación de componentes Android para compilar y Ejecutar. Elaboración Propia.

4.3 DISEÑO DEL SISTEMA

Para el diseño de la aplicación el equipo de trabajo sigue las recomendaciones de la metodología XP, siempre se trató de evitar las soluciones complejas, y se trabajó en una sola iteración, sin pensar en los que vendrían más adelante.

Otro aspecto importante en el diseño, es la constante reestructuración del código. El principal objetivo de la Reestructuración es evitar la duplicación del código, simplificarlo y hacerlo más flexible para facilitar los posteriores cambios. Esto se hizo constantemente en la programación.

4.4 DIAGRAMAS UML

El análisis y diseño orientados a objetos pueden ofrecer una metodología que facilita los métodos lógicos, rápidos y detallados para crear sistemas que respondan a un panorama de negocios en evolución. Las técnicas orientadas a objetos funcionan bien en situaciones en las que los sistemas de información

complicados pasan a través de un proceso continuo de mantenimiento, adaptación y rediseño. (Kendall, 2011)

4.5 MODELAMIENTO DEL SISTEMA DE INFORMACIÓN

4.5.1 Diagrama Caso de uso Sistema Cliente

Casos de uso: Administrar el sistema.

Actores: Usuario

Propósito: Mostrar las funciones de la App.

Descripción: En el diagrama observamos las diferentes funciones de la App.

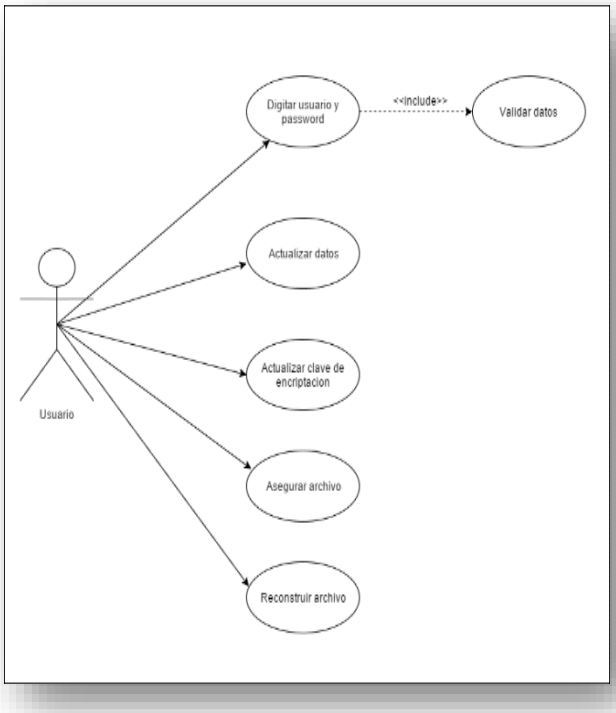


Figura 25. Caso de uso del Aplicación. Elaboración Propia.

4.5.2 Diagrama Actividad para Usuario y Password

Para acceder pone usuario y clave para luego ingresar a la interfaz de la app.

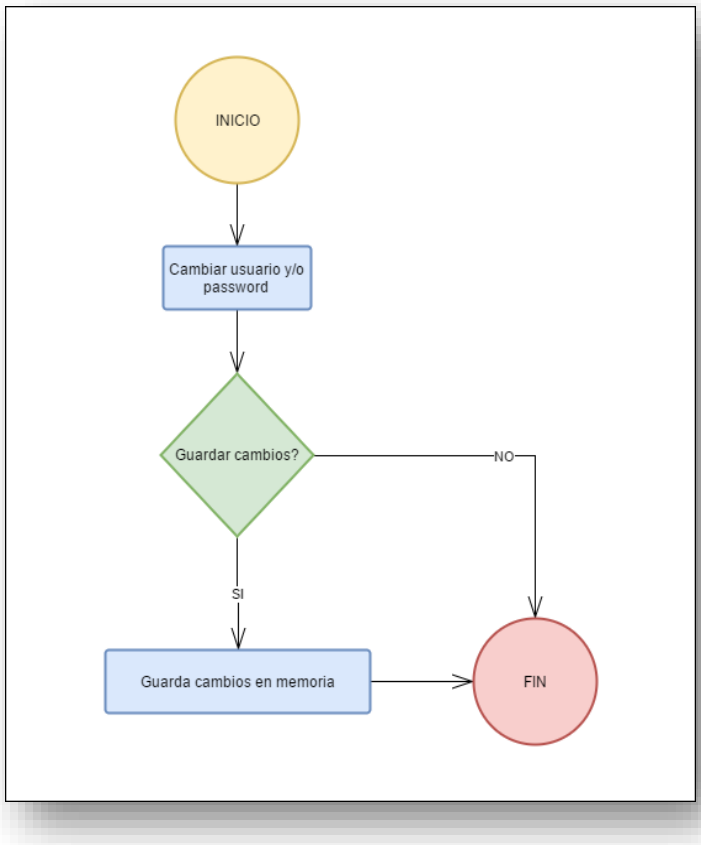


Figura 26. Diagrama de Actividad para Cambio de usuario password.

4.5.3 Diagrama de Actividad Encriptación y Serialización

El USUARIO de la App una vez ingresado podrá ver todos los archivos en la SD luego podrá encriptarlo y serializarlo podrá ver los archivos con extensión .dvc y hacer el proceso inverso para luego visualizar los archivos.

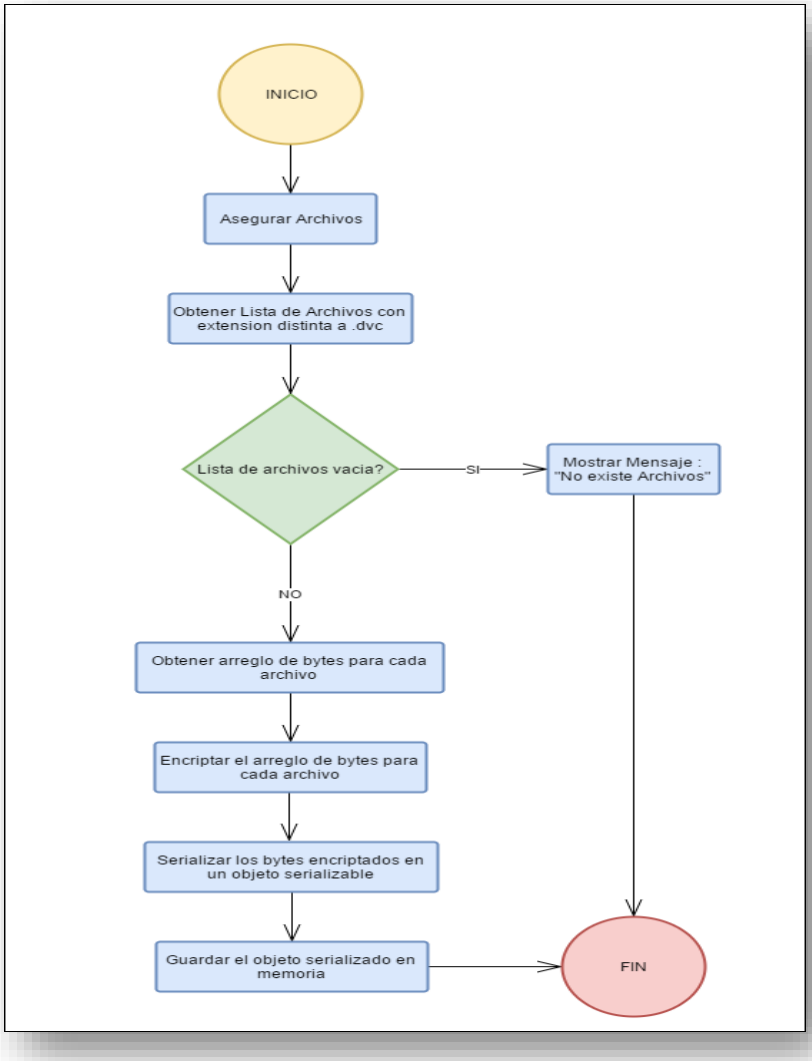


Figura 27.Diagrama de Actividad de Encriptación Y Serialización.

4.5.4 Diagrama de Actividad, Revertir la Encriptación y Serialización

En el proceso de revertir la encriptación y serialización en base a los datos.

Ingresamos a la APP los datos nos devuelve la información original haciendo el proceso de encriptación y serialización.

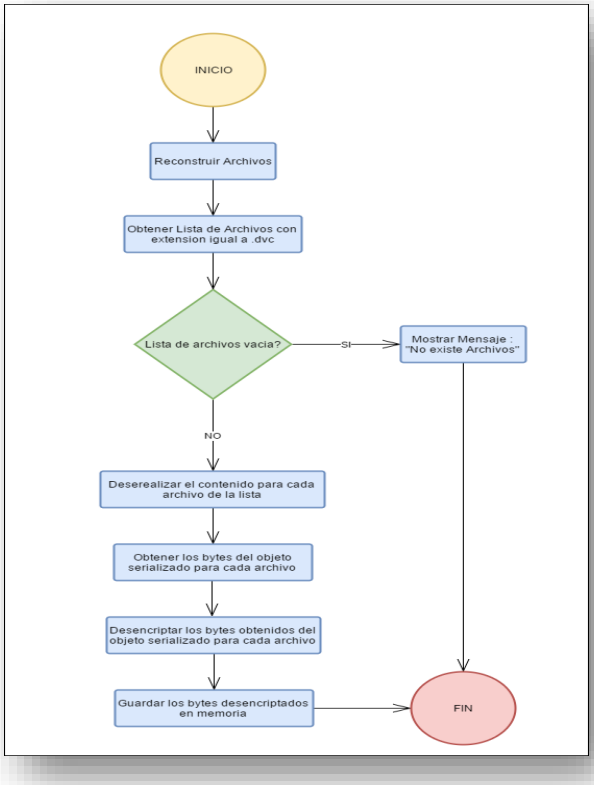


Figura 28.Diagrama de Actividad de para la revertir la encriptación y serialización.

4.5.5 Diagrama de Cambio Password de Encriptación clave Maestra

Vemos como se inicia, vamos cambio de clave, encriptación, luego guardar cambios, luego finalizamos.

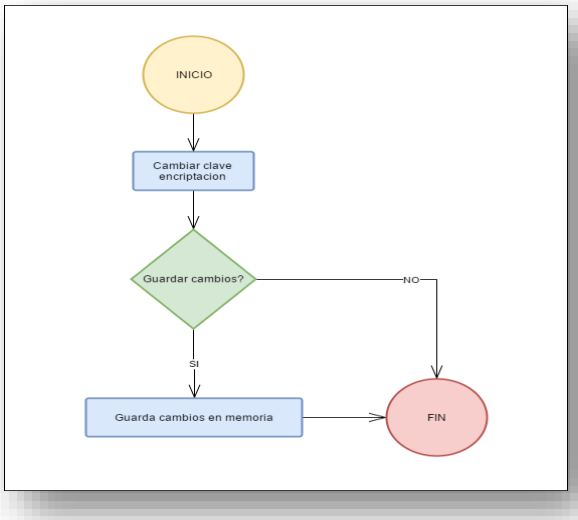


Figura 29. Diagrama de Actividad De cambio password de encriptación.

4.6 DISEÑO Y ARQUITECTURA DE SOTFWARE

4.6.1 Diagrama de Clases para la Aplicación Bóveda

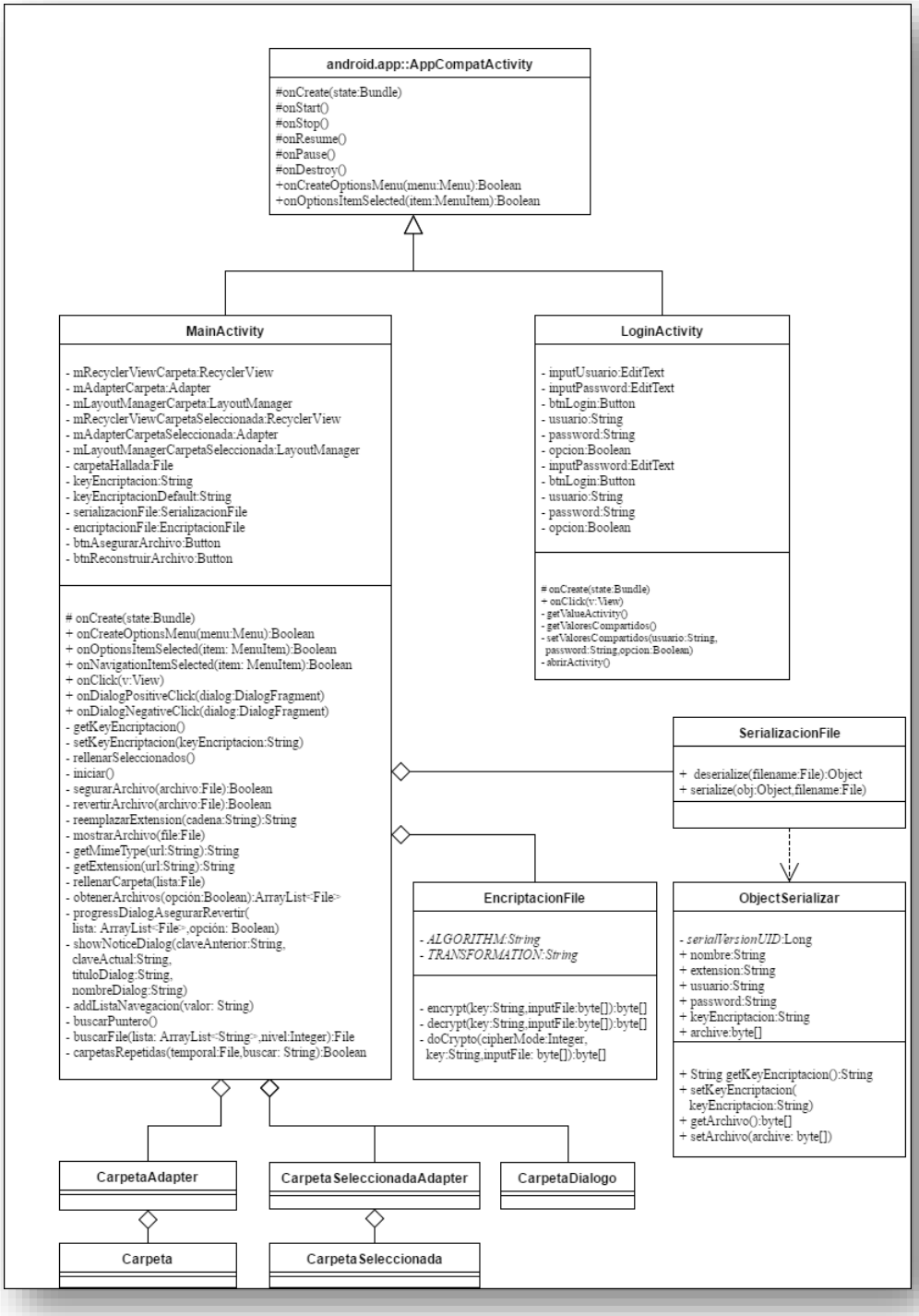


Figura 30. Diagrama De Clases Para La Aplicación Bóveda.

4.7 VERSIONES E ITERACIONES - Versiones 0.1

Iteración 1: El diseño de la base de datos es la parte primordial para empezar con el desarrollo de cualquier aplicación, ya que un buen diseño de ésta, es la base fundamental para el óptimo funcionamiento y éxito de cualquier aplicación.

Para definir el diseño se contó siempre con la ayuda de los clientes, quienes informaban que datos serian relevantes para el proyecto

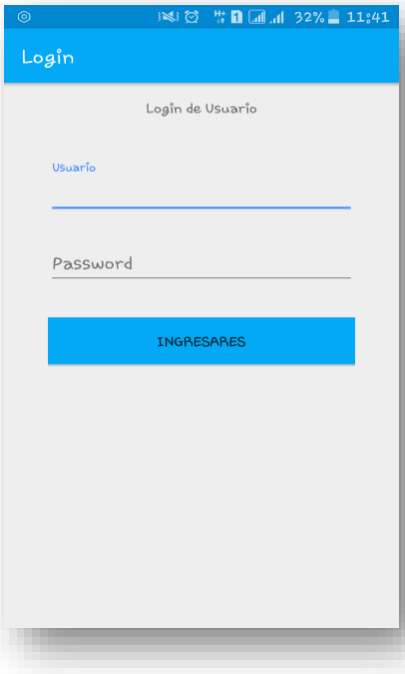


Figura 31. Versión de interación de Boveda 0.1

4.8 CODIFICACIÓN SEGÚN LA METODOLOGÍA XP

4.8.1 Usuario Siempre Disponible

XP recomienda como factor de éxito que el cliente esté involucrado en toda la etapa de desarrollo, esto se cumplió satisfactoriamente ya

que como se explicó anteriormente los desarrolladores eran al mismo tiempo clientes.

4.8.2 Estándares de Codificación

La metodología XP aconseja seguir estándares de codificación para que cualquier integrante del equipo de desarrollo pueda entender y asimilar fácilmente código por otro integrante.

4.8.3 Pruebas

XP recomienda diseñar las pruebas antes de codificar los módulos esta práctica representó un cambio drástico en la forma de desarrollar del equipo de trabajo ya que como no se había utilizado en proyectos anteriores se tuvo algunos inconvenientes y por esta razón se llevó más tiempo de lo deseado.

4.8.4 Especificación del Sistema Culminado

- Acabado el proceso de implementación de la aplicación, en niveles de software consideramos al Sistema de seguridad, pues ésta procesa información, encripta y serializa solo con la aplicación.

- De entre los tipos de aplicación conocidos, el sistema se le considera Sistema de seguridad, debido a que fue desarrollado para una tarea específica (protección de archivos).

4.9 VALIDACIÓN DE LA APP CON INFORMACIÓN CON PRUEBA ESTADÍSTICA

4.9.1 Prueba de Mcnemar

La prueba de McNemar se utiliza para decidir si puede o no aceptarse que determinado "tratamiento" induce un cambio en la respuesta dicotómica o dicotomizada de los elementos sometidos al mismo, y es aplicable a los diseños del tipo "antes-después" en los que cada elemento actúa como su propio control.

4.9.2 Prueba Hipótesis De Los Rangos Con Signo De Mcnemar

1) Planteamiento de Hipótesis

H_0: El usuario no queda satisfecho por el uso de la App.

H_1: El usuario queda satisfecho por el uso de la App.

Nivel de significancia (alfa) a=5% = 0,05

2) Prueba Estadística Rangos Con Signos De Mcnemar

$$X^2 = \frac{|A - D| - 1^2}{A + D}$$

$$X^2 = \frac{|34 - 3| - 1^2}{34 + 3}$$

$$X^2 = 24.32$$

3) Decisión:

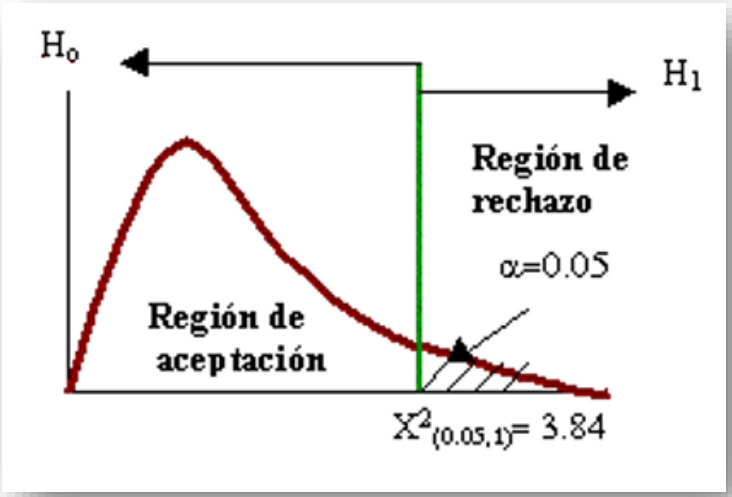


Gráfico 1. Resultado de nuestra hipótesis, Hipótesis nula esto quieres decir que los usuarios quedaron satisfechos con nuestra aplicación

Como $X^2 = 24.32 > X^2_{tabla} = 3.84$, Entonces rechazamos la Hipótesis Nula. Esto quiere decir que los usuarios quedaron satisfechos ante el uso de esta aplicación.

Según la encuesta de los 100 usuarios, donde se redujo con nuestra formula finita se redujo a 48 usuarios, es aquí donde la investigación se centra donde el 69% de nuestros encuestados le dieron al buen uso de nuestra aplicación Bóveda.

4.10 RESULTADOS DE LA ENCUESTA

1. ¿Cómo considera usted el diseño de la interfaz para el usuario con la App?

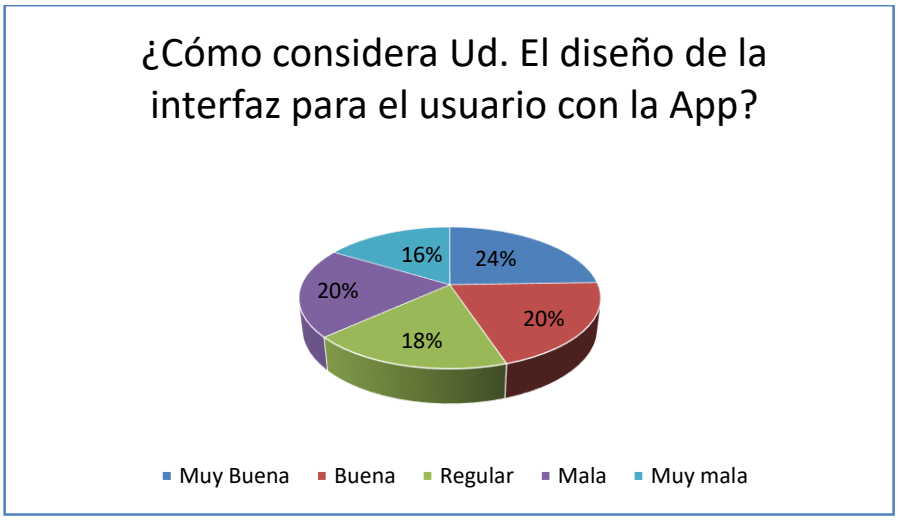


Gráfico 2. ¿Cómo considera usted el diseño de la interfaz para el usuario con la App? El 24% afirma que el diseño de la interface es muy buena, mientras que el 16% afirmó que el diseño interface es Muy mala. Elaboración propia.

2. ¿Cree usted que la App es rápido en su protección?

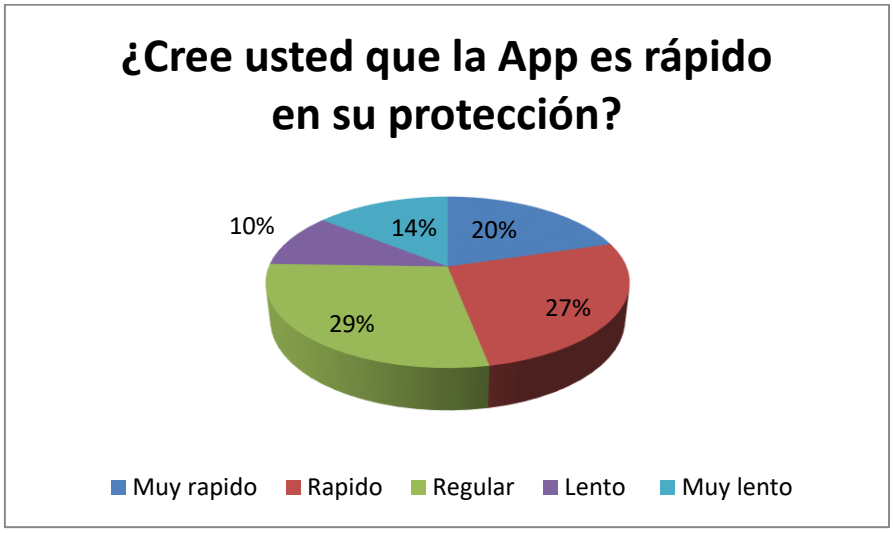


Gráfico 3. ¿Cree usted que la App es rápido en su protección? El 29% afirmó que el interfaz es rápido en su protección, mientras que el 10% afirmó que el interfaz es rápido en su protección. Elaboración propia.

3. ¿Cuáles son las probabilidades de que nos recomiende a otras personas?

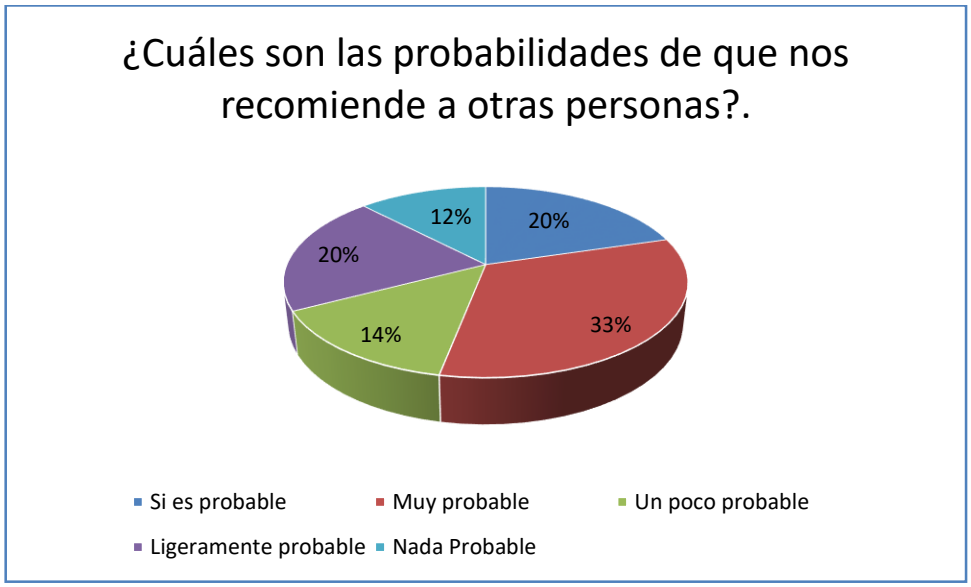


Gráfico 4. ¿Cuáles son las probabilidades de que nos recomiende a otras personas? El 12% afirmo que es nada probable recomendar la aplicación a otras personas, mientras que el 33% afirmo que es muy probable recomendar la aplicación a otras personas. Elaboración propia.

4. ¿Alguna vez perdió su celular?

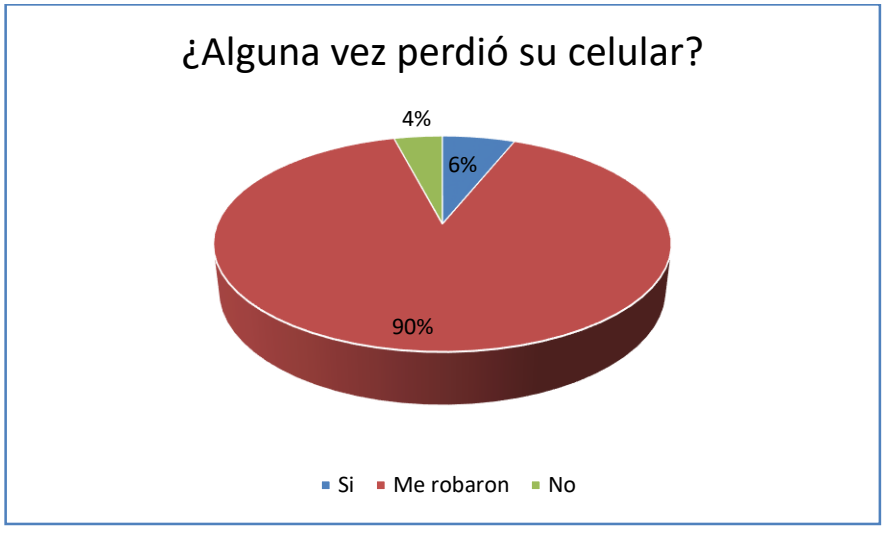


Grafico 5. ¿Alguna vez perdió su celular? El 4% afirmo que no perdió su celular, mientras que el 90% afirmo que les robaron el celular. Elaboración propia.

.

5. ¿Perdió información importante?

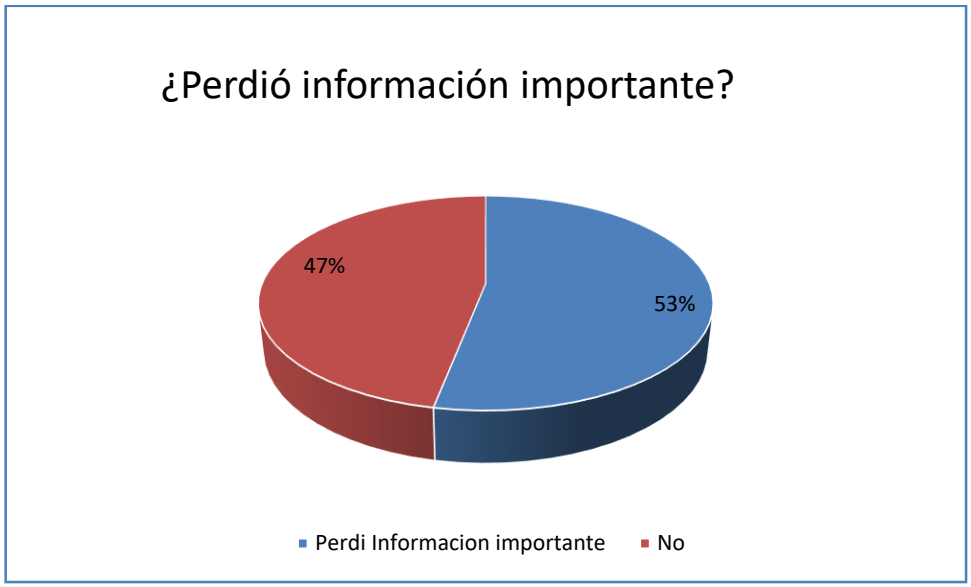


Gráfico 6. ¿Perdió información importante? El 47% no perdieron información importante, mientras que el 53% afirmó que perdieron información importante. Elaboración propia.

6. ¿Quedo satisfecho con muestra Aplicación?

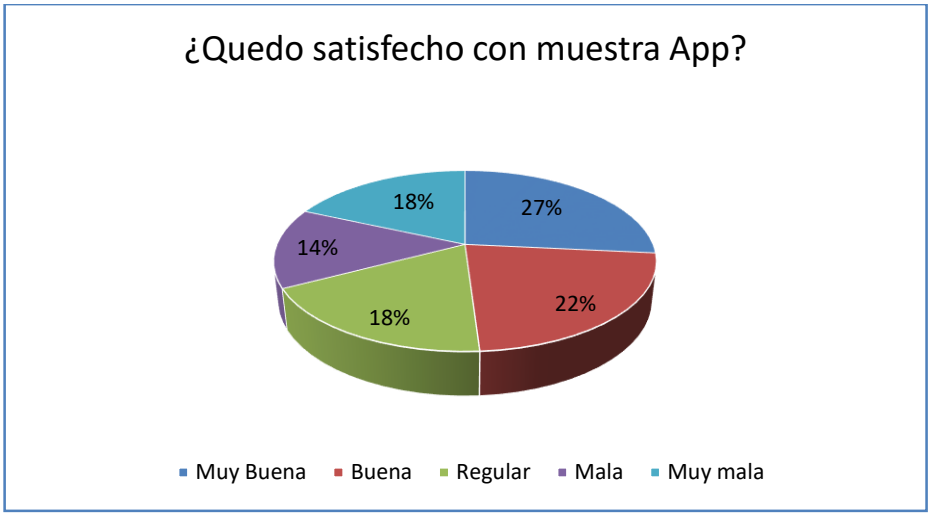


Gráfico 7. ¿Quedó satisfecho con muestra App? El 14% no quedó satisfecho con el App calificándolo como Malo, mientras que el 27% quedaron muy satisfechos con el App afirmando que el App es muy bueno. Elaboración propia.

CONCLUSIONES

La utilización de plataformas abiertas nos permite crear software de buena calidad sin preocuparnos de licencias, sus actualizaciones son constantes ya que se trata de software libre, además al reducir el presupuesto para un proyecto hace de Android una plataforma muy atractiva para el desarrollo de aplicaciones móviles.

Se logró diseñar desarrollar la aplicación **"BÓVEDA"** para teléfonos inteligentes con sistema Operativo Android que encriptado y serialización de objetos para la protección de archivos

Se ha logrado hacer buen uso de las herramientas SQLite contenidas en el framework de Android, se implementó con éxito la aplicación, demostrando así la facilidad de desarrollar aplicaciones para la plataforma Android.

El uso de XP como metodología de desarrollo, nos permite obtener resultados en poco tiempo dando mayor prioridad a los requerimientos, esto representa una ventaja ya que se prueba la aplicación y sus nuevas funcionalidades mucho más rápido que al desarrollarlo con otras metodologías.

Se implementó, para lo cual estudio estadístico donde nos indica que la aplicación fue un éxito con 69 % de aceptación, se logró hacer la prueba con rangos de MACNEMAR

El presente trabajo es un aporte para desarrollo social y tecnológico de la región, ya que mediante este documento podemos aprender una arquitectura que en el medio tiene muy buena demanda y que actualmente en nuestra región no se la está explotando.

RECOMENDACIONES

Para empezar a programar en Android es aconsejable tener experiencia básica en el lenguaje de programación Java y conocimientos en Programación Orientada a Objetos.

Si vamos a realizar un estudio de nuevas tecnologías se recomienda seguir la documentación provista por el desarrollador de esta tecnología, también es conveniente leer foros especializados ya que en los mismos se pueden encontrar respuestas a las inquietudes y problemas que puedan surgir en el proceso de aprendizaje.

Se recomienda desarrollar una aplicación utilizando una versión baja del API, esto permitirá que la aplicación sea compatible con una mayor cantidad de dispositivos.

Según las comparativas obtenidas, el sistema operativo Android es una excelente plataforma para desarrollar aplicaciones, por ser de software libre y estar respaldado por Google, que actualmente es uno de los gigantes de la informática.

Para que el aplicativo **"BÓVEDA"** trabaje de una manera precisa y rápida, se recomienda que el dispositivo cuente con plan de datos.

DISCUSIÓN

Según (Pacheco Veliz, 2016) a través de su tesis se puede notar que su información personal con la PC y también con los dispositivos móviles son expuestos a terceras personas a su vez son destructivas, hasta robo de información y chantaje, en lo que se respecta se puede decir que la tesis de investigación trata de proteger éstas situaciones es donde coincidimos que es necesario la protección de archivos.

Podemos notar por (Mejía Pacheco, 2016) qué tan importante es desarrollar la metodología para aplicar la seguridad y las vulnerabilidades actuales que han estado afectando a los dispositivos móviles, para la cual se tiene que salvaguardar nuestra información personal o de lugar de trabajo se puede notar que coincidimos con nuestro objetivo ser más precavidos con la información de cada usuario.

REFERENCIA BIBLIOGRAFICA

Alegsa, L. (2011). Diccionario de Informática *1, 2,* Recuperado http://www.alegsa.com.ar.

Alegsa, L. (2017). Definición de Login. Recuperado de http://www.alegsa.com.ar/Dic/login.php.

Androidfeed. (2012). Fujo de Trabajo en una Aplicacion. Recuperado de https://androidfeed.wordpress.com/2012/2009/2010/workflow/.

Gironés, J. T. (2012). El Gran Libro De Android Segunda Edición. *2da edicion*, 26-29.

Gómez-Biedma, S., Vivó, M., & Soria, E. <en blanco.pdf>.

Google. (s.f.-a). Arquitectura de la plataforma. Recuperado de https://developer.android.com/guide/platform/index.html?hl=es-419#hal.

Google. (s.f.-b). Cipher., Recuperado de https://developer.android.com/reference/javax/crypto/Cipher.html.

Google. (s.f.-c). Serializable. Recuperado de https://developer.android.com/reference/java/io/Serializable.html.

IntelliJ. (2014). Desarrolo de Android Studio., Recuperado de https://es.wikipedia.org/wiki/Android_Studio.

Jiménez-García, M., & Martínez-Ortega, M. d. l. Á. (2017). El Uso de una Aplicación Móvil en la Enseñanza de la Lectura. *Información tecnológica, 28*, 151-160.

Kendall, K. E. Y. K., Julie E. (2011). Analisis y Diseño de Sistemas 8va Edicion. 308.

Lucas, G. (2016). Historial de versiones de Android. (Recuperado de: https://es.wikipedia.org/wiki/Anexo:Historial_de_versiones_de_Android#cite_note-3).

Mattassi, J. (2015). Android Sigue Dominando El Mercado De Smartphones. (Recuperado de https://www.wayerless.com/2015/08/android-sigue-dominando-el-mercado-de-smartphones/).

Mejía Pacheco, Y. (2016). “Metodología De Seguridad Para El Manejo De Dispositivos Móviles Y La Vinculación Con Los Usuarios”. 3, 16, 98.

Netdreams. (2015). Dispositivos Móviles Perú Sell-In Primer Semestre 2015 . Recuperado de http://netdreams.pe/blog/mundo-movil/dispositivos-moviles-

peru-sell-in-primer-semestre-2015-half-year-mas-de-2013-2013-millones-de-smartphones-y-tablets-llegan-al-peru/

Pacheco Veliz, S. E. P. O., Carlos Damián. (2016). "Estudio y análisis de seguridad en dispositivos móviles. BYOD y su impacto en las organizaciones" (pp. 15-17): Universidad Nacional De La Plata.

Riquelme, S. (2013). ¿Qué es Android? , Recuperado de: http://aprendesobregalaxyyoung.blogspot.pe/2013/2008/que-es-android.html.

Risoto, D. V. (2013). Android: Su Historia. (Recuperado de: https://losmuertevideanos.wordpress.com/2013/10/24/android-su-historia/.

Torres, M. (2014). Serializable: Serialización De Objetos En Java. (recuperado de: http://michelletorres.mx/serializable-serializacion-de-objetos-en-java/.

Valencia, U. P. d. (2017). Máster en Desarrollo de Aplicaciones Android. . Recuperado de http://www.androidcurso.com/index.php/tutoriales-android/31-unidad-31-vision-general-y-entorno-de-desarrollo/98-comparativa-con-otras-plataformas.

Vélez, M. d. C. A. (2015). Integración Del Diseño Centrado En Usuario Con Metodologías Ágiles En El Desarrollo De Un Catálogo De Plantas. Un Estudio De Investigación - Acción. 27-29.

Wikipedia. (2016). Android. Recuperado de https://es.wikipedia.org/wiki/Android.

GLOSARIO

A

API

Interfaz de programación de aplicaciones (IPA) o API (del inglés Aplicación Programming Interface) es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizada por otro software como una capa de abstracción. Son usadas generalmente en las bibliotecas (también denominadas “librerías").

APP

Tipo de programa informático diseñado como herramienta, para permitir a un usuario realizar diversos tipos de trabajos. Por dispositivo móvil, aplicación informática diseñada para teléfonos móviles inteligentes.

G

GOOGLE PLAY

Google Play Store o sólo Google Play (antes llamado Android Market) es una tienda de software en línea desarrollada por Google para los dispositivos Android. Es una aplicación ("app") que está preinstalada en la mayoría de los dispositivos Android y que permite a los usuarios buscar y descargar aplicaciones publicadas por desarrolladores terceros, alojada en Google Play. Los usuarios también pueden buscar y obtener información sobre aplicaciones en esta página web. En enero de 2012, Google Play disponía de más de 500.000 aplicaciones. El 6 de marzo de 2012, Android Market fue rebautizado como Google Play.

M

MÁQUINA VIRTUAL

En informática una máquina virtual es un software que emula a una computadora y puede ejecutar programas como si fuese una computadora real. Este software en un principio fue definido como "un duplicado eficiente y aislado de una máquina física". La acepción del término actualmente incluye a máquinas virtuales que no tienen ninguna equivalencia directa con ningún hardware real.

R

REFACTORIZACIÓN

La refactorización (del inglés refactoring) es una técnica de la ingeniería de software para reestructurar un código fuente, alterando su estructura interna, sin cambiar su comportamiento externo.

S

SDK

Un kit de desarrollo de software o SDK (siglas en inglés de software development kit) es generalmente un conjunto de herramientas de desarrollo de software que le permite al programador crear aplicaciones para un sistema concreto, por ejemplo ciertos paquetes de software, frameworks, plataformas de hardware, computadoras, videoconsolas, sistemas operativos, etc.

SQLite

SQLite es un sistema de gestión de bases de datos relacional compatible con ACID, contenida en una relativamente pequeña (~275 kib) biblioteca escrita en C. SQLite es un proyecto de dominio público creado por D. Richard Hipp. A diferencia de los sistemas de gestión de bases de datos cliente-servidor, el motor de SQLite no es un proceso independiente con el que el programa principal se comunica. En lugar de eso, la biblioteca SQLite se enlaza con el programa pasando a ser parte integral del mismo.

T

TDD (Test Driven Development)

Desarrollo guiado por pruebas de software, o Test-driven development (TDD) es una práctica de programación que involucra otras dos prácticas: Escribir las pruebas primero (Test First Development) y Refactorización (Refactoring). Para escribir las pruebas generalmente se utilizan las pruebas unitarias (unit test en inglés). En primer lugar, se escribe una prueba y se verifica que las pruebas fallen o no. A continuación, se implementa el código que hace que la prueba pase satisfactoriamente y seguidamente se refactoriza el código escrito. El propósito del desarrollo guiado por pruebas es lograr un código limpio que funcione. La idea es que los requisitos sean traducidos a pruebas, de este modo, cuando las pruebas pasen, se garantizará que el software cumple con los requisitos que se han establecido.

W

WI-FI

Wi-Fi es un mecanismo de conexión de dispositivos electrónicos de forma inalámbrica. Los dispositivos habilitados con Wi-Fi, tales como: un ordenador personal, una consola de videojuegos, un smartphone o un reproductor de audio digital, pueden conectarse a Internet a través de un punto de acceso de red inalámbrica. Dicho punto de acceso (o hotspot) tiene un alcance de unos 20 metros (65 pies) en interiores y al aire libre una distancia mayor. Pueden cubrir grandes áreas la superposición de múltiples puntos de acceso.

WS-I

Web Services Interoperability Organization - Organización para la Interoperabilidad de Servicios Web. Su objetivo es fomentar y promover la Interoperabilidad de Servicios Web (Web Services Interoperability - WS-I) sobre cualquier plataforma, sobre aplicaciones, y sobre lenguajes de programación. Su intención es ser un integrador de estándares para ayudar al avance de los servicios web de una manera estructurada y coherente. La WS-I ha organizado los estándares que afectan a la interoperabilidad de los servicios web en una pila basada en funcionalidades.

ANEXOS

Anexo A
Cuestionario para los Usuario Sobre Nuestra Aplicación

Se recomienda responder la presente encuesta con toda sinceridad y confianza

"Seguridad en archivos externos e internos para dispositivos con sistema operativo Android"

1. ¿Cómo considera usted El diseño de la interfaz para el usuario con la App?

a) . Muy buena.
b) . Buena.
c) . Regular.
d) . Mala.
e) . Muy mala.

2. ¿Cree usted que la App es rápida en su protección?

a) . Muy rápido.
b) . Rápido.
c) . Regular.
d) . Lento
e) . Muy Lento

3. ¿Cuáles son las probabilidades de que nos recomiende a otras personas?

a) . Si es probable.
b) . Muy probable.
c) . Un poco probable.
d) . Ligeramente probable.
e) . Nada probable.

4. ¿Alguna vez perdió su celular?

a) . Si
b) Me robaron
c) . No

5. ¿Perdió información importante?

a) . Perdí información Importante
b) . No

6. ¿Quedó satisfecho con nuestra App?

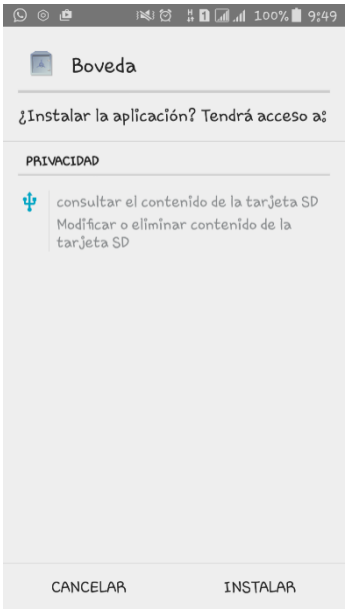
a) . Muy buena.
b) . Buena.
c) . Regular.
d) . Mala.
e) . Muy mala.

Anexo B

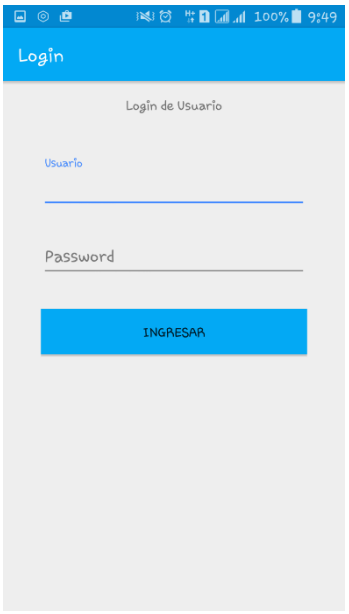
Manual de la App

Boveda

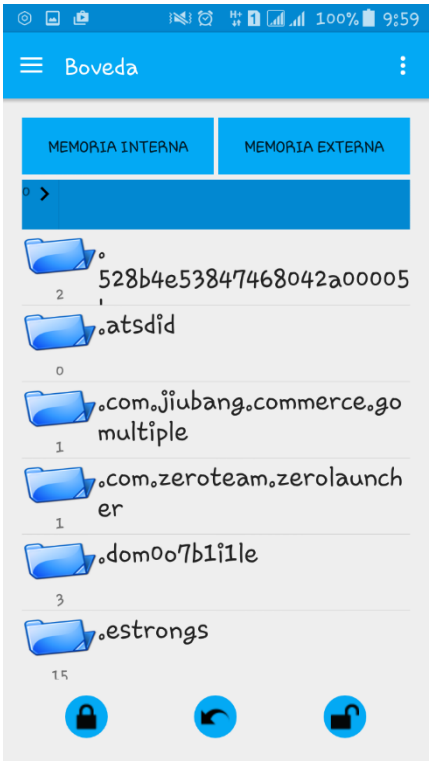
1. Instalamos BOVEDA App. (Puede descargar de esta dirección *https://drive.google.com/drive/folders/0B1p4_H8tBIvxOVUxWmp6WW5wWmM?usp=sharing* **)**



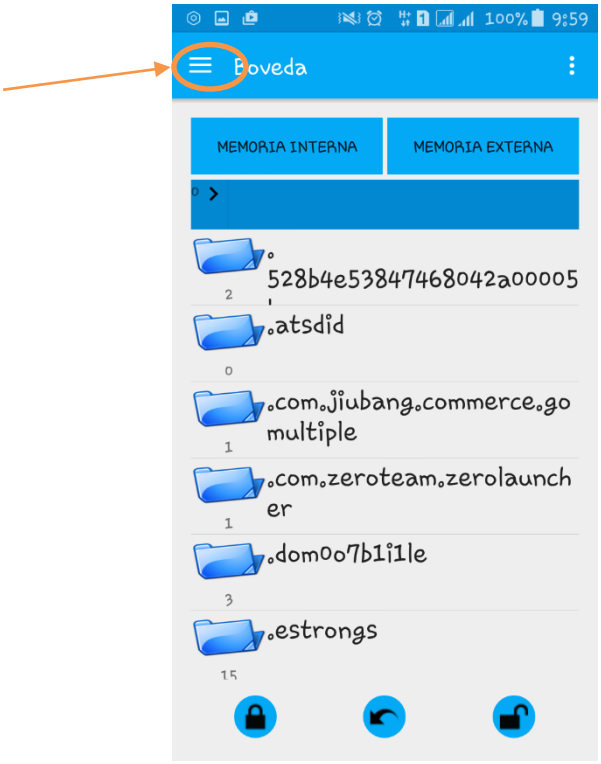
2. Abrir la aplicación por defecto es Usuario: admin Password: admin.

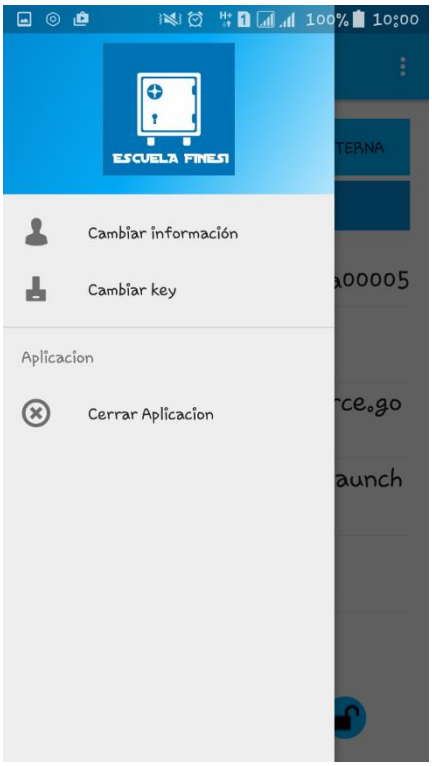


3. Visualizamos como tarjeta Externa e Interna.

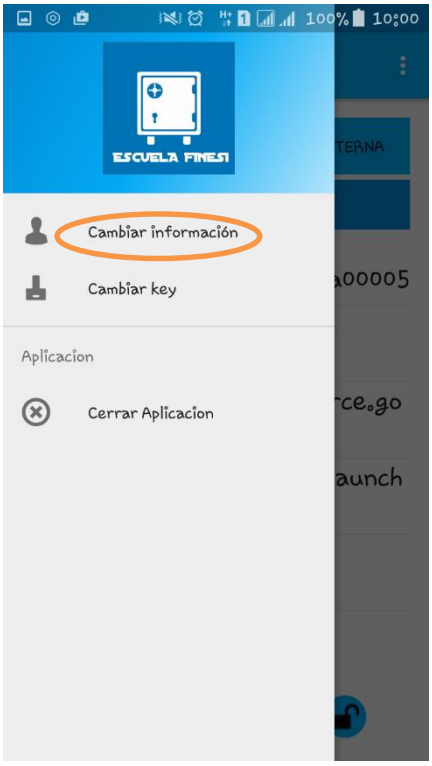


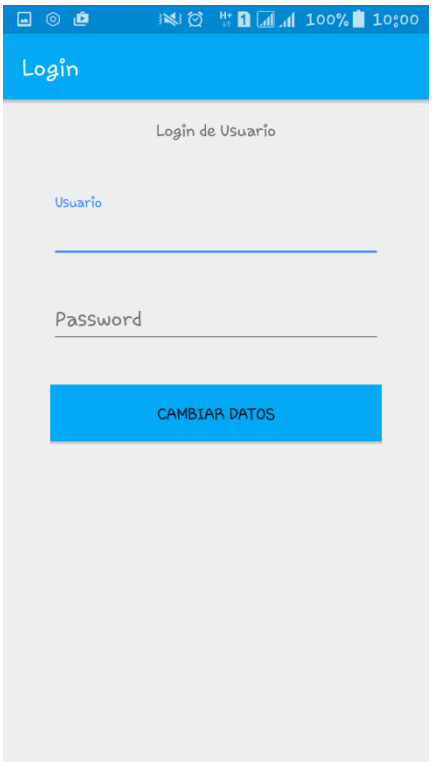
4. Vamos a opciones para cambiar contraseña de usuario y también contraseña maestra.



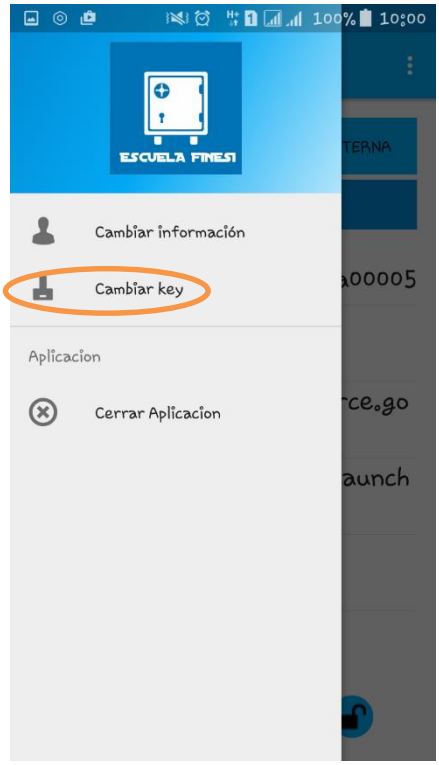


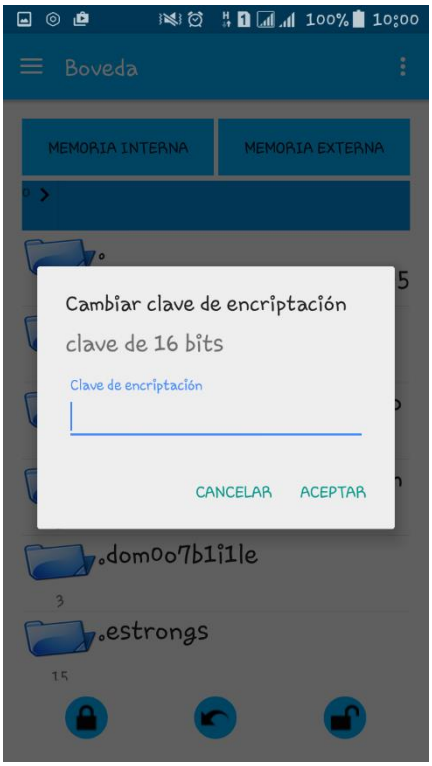
5. El usuario puede cambiar sus datos y clave.



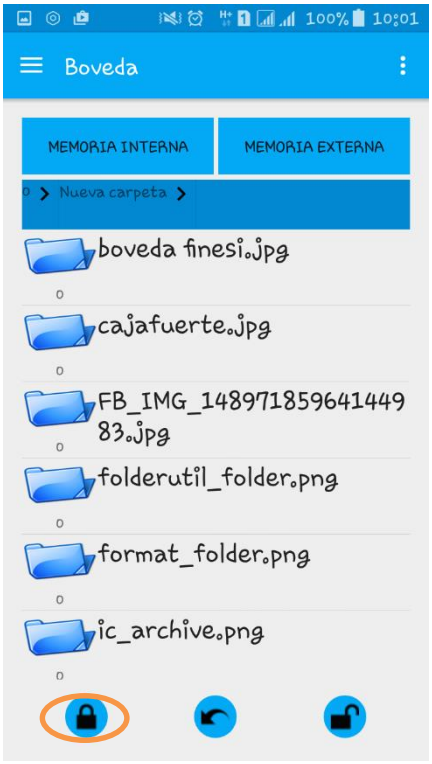


6. Acá el usuario puede poner una contraseña maestra solo para usuario avanzados.



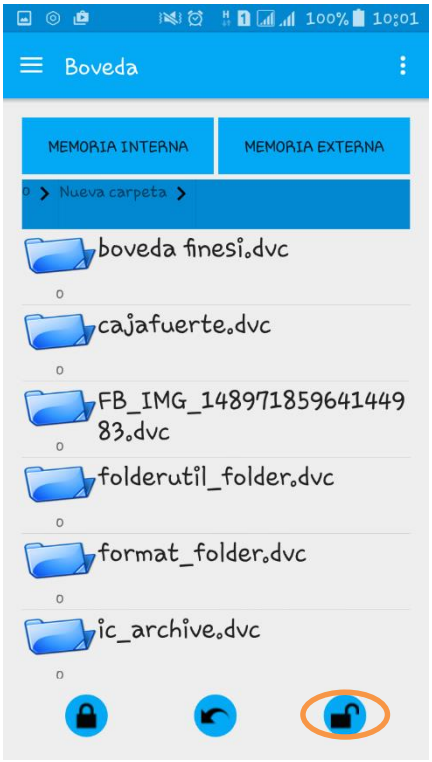


7. Visializamos nuestros archivos en jpg, etc.

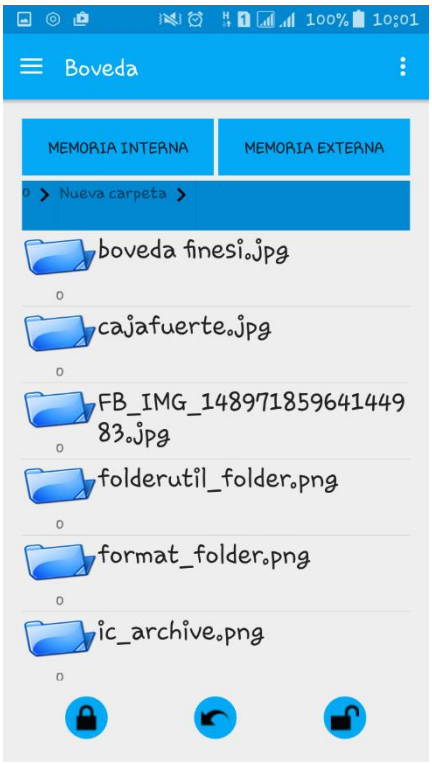


Luego ponemos en protección

8. Aplicamos la protección a los archivos, ahora estarán con la extensión .dvc.



Luego desprotegemos para ver de nuevo nuestros archivos



Anexo C

Código Fuente

```
import android.app.ProgressDialog;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.net.Uri;
import android.os.AsyncTask;
import android.os.Bundle;

import android.os.Environment;
import android.support.v4.app.DialogFragment;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.util.Log;
import android.view.View;
import android.support.design.widget.NavigationView;
import android.support.v4.view.GravityCompat;
import android.support.v4.widget.DrawerLayout;
import android.support.v7.app.ActionBarDrawerToggle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.Menu;
import android.view.MenuItem;
import android.view.ViewGroup;
import android.webkit.MimeTypeMap;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
```

```
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Date;

public class MainActivity extends AppCompatActivity
        implements
        NavigationView.OnNavigationItemSelectedListener,
        View.OnClickListener,
CarpetaDialogo.NoticeDialogListener{

    private int contador = 0;
private RecyclerView mRecyclerViewCarpeta;
    private RecyclerView.Adapter mAdapterCarpeta;
    private RecyclerView.LayoutManager mLayoutManagerCarpeta;
private ArrayList<Carpeta> listaCarpeta = new ArrayList<Carpeta>();

private RecyclerView mRecyclerViewCarpetaSeleccionada;
    private RecyclerView.Adapter mAdapterCarpetaSeleccionada;
    private RecyclerView.LayoutManager mLayoutManagerCarpetaSeleccionada;
    private ArrayList<CarpetaSeleccionada> listaCarpetaSeleccionada = new
ArrayList<CarpetaSeleccionada>();

private DrawerLayout drawer;
    private ActionBarDrawerToggle toggle;
    private NavigationView;
    private File carpetaHallada;

    private ArrayList<String> lista = new ArrayList<>();

private String inputDialogNombre;

    //variables relacionadas a la encriptacion.
```

```
private String nombre = "aes";
    private String extension = "dvc";
    private String usuario = "aes";
    private String password = "aes";
privatue String keyEncriptacion = "";
    private String keyEncriptacionDefault = "clave de 16 bits";

    //variable del objeto que realiza la serializacion.
    private SerializacionFile serializacionFile;
    //variable del objeto que realiza la encriptacion.
    private EncriptacionFile encriptacionFile;

    //botones para asegurar o reconstruir los archivos.
private Button btnAsegurarArchivo;
    private Button btnReconstruirArchivo;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

//obtener la clave de la encriptacion.
        this.getKeyEncriptacion();

        //Inicializando el objeto de encriptacion.
        encriptacionFile = new EncriptacionFile();
        //Inicializando el objeto de serializacion.
        serializacionFile = new SerializacionFile();

        //File lista = Environment.getExternalStorageDirectory();
        //obtiene la ruta raiz de la tarjeta SD.
lista.add(Environment.getExternalStorageDirectory().getName());
        //lista.add(this.getFilesDir().getName());
```

```
//referencia a los botones de asegurar y reconstruir archivo
        btnAsegurarArchivo = (Button)findViewById(R.id.btn_asegurar_archivo);
        btnReconstruirArchivo =
(Button)findViewById(R.id.btn_asegurar_reconstruir);

        //evento click de los botones.
        btnAsegurarArchivo.setOnClickListener(this);
        btnReconstruirArchivo.setOnClickListener(this);

        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
setSupportActionBar(toolbar);
        /*
        fab = (FloatingActionButton) findViewById(R.id.fab);
        fab.setOnClickListener(this);
        */
        drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
        toggle = new ActionBarDrawerToggle(
                this, drawer, toolbar, R.string.navigation_drawer_open,
R.string.navigation_drawer_close);
        drawer.setDrawerListener(toggle);

        navigationView = (NavigationView) findViewById(R.id.nav_view);
        navigationView.setNavigationItemSelectedListener(this);

        /*Lista de carpetas almacenadas*/
        mRecyclerViewCarpeta = (RecyclerView)
findViewById(R.id.recyclerViewCarpeta);
        mRecyclerViewCarpeta.addOnItemTouchListener(
                new RecyclerItemClickListener(
                        this,
                        mRecyclerViewCarpeta,
                        new RecyclerItemClickListener.OnItemClickListener(){
                            @Override public void onItemClick(View view, int position) {
                                // do whatever
```

```
                        ViewGroup v = (ViewGroup) view;
                        //TextView c =
(TextView)v.findViewById(R.id.textViewNombreCarpeta);
                        TextView c =
(TextView)view.findViewById(R.id.textViewNombreCarpeta);
                        Log.d("Recycler :","click "+c.getText()+" Id : "+v+" pos :
"+position);

//adicionar el elemento y actualizar el punto File
//ocultarFab();
                        addListaNavegacion(c.getText().toString());
                        iniciar();

                    }

                    @Override public void onLongItemClick(View view, int position) {

                    }
                }));

        // use this setting to improve performance if you know that changes
        // in content do not change the layout size of the RecyclerView
        mRecyclerViewCarpeta.setHasFixedSize(true);

        // use a linear layout manager
        mLayoutManagerCarpeta = new
LinearLayoutManager(this,LinearLayoutManager.VERTICAL,false);
mRecyclerViewCarpeta.setLayoutManager(mLayoutManagerCarpeta);

        //Establece la division entre cada item mostrado
mRecyclerViewCarpeta.addItemDecoration(new DividerItemDecoration(this,
LinearLayoutManager.VERTICAL));

        // specify an adapter (see also next example)
```

```
mAdapterCarpeta = new CarpetaAdapter(listaCarpeta);
        mRecyclerViewCarpeta.setAdapter(mAdapterCarpeta);

        /*Lista de niveles de las carpetas*/
        mRecyclerViewCarpetaSeleccionada = (RecyclerView)
findViewById(R.id.recyclerViewCarpetaSeleccionada);

mRecyclerViewCarpetaSeleccionada.addOnItemTouchListener(
                new RecyclerItemClickListener(
                        this,
                        mRecyclerViewCarpeta,
                        new RecyclerItemClickListener.OnItemClickListener(){
                            @Override public void onItemClick(View view, int position) {
                                // do whatever
                                ViewGroup v = (ViewGroup) view;
                                //TextView c =
(TextView)v.findViewById(R.id.textViewNombreCarpeta);
                                TextView c =
(TextView)view.findViewById(R.id.textViewSeleccionado);
Log.d("Recycler seleccionado :","click "+c.getText()+" pos :   "+position);

                                //posicionar el puntero en la posicion actual.
eliminarHijos(position);
                                rellenarSeleccionados();
                                iniciar();
                            }

                            @Override public void onLongItemClick(View view, int position) {
                                // do whatever
                                Log.d("Recycler :","Long Touch");
                            }
                        }));

        // use this setting to improve performance if you know that changes
```

```
        // in content do not change the layout size of the RecyclerView
        mRecyclerViewCarpetaSeleccionada.setHasFixedSize(true);

        // use a linear layout manager
        mLayoutManagerCarpetaSeleccionada = new
LinearLayoutManager(this,LinearLayoutManager.HORIZONTAL,false);
mRecyclerViewCarpetaSeleccionada.setLayoutManager(mLayoutManagerCarpet
aSeleccionada);

        //Establece la division entre cada item mostrado
mRecyclerViewCarpetaSeleccionada.addItemDecoration(new
DividerItemDecoration(this, LinearLayoutManager.HORIZONTAL));

        // specify an adapter (see also next example)
mAdapterCarpetaSeleccionada = new
CarpetaSeleccionadaAdapter(listaCarpetaSeleccionada);

mRecyclerViewCarpetaSeleccionada.setAdapter(mAdapterCarpetaSeleccionada);

        //rellenando el contenido de carpeta.
        iniciar();

        //rellenando la lista de niveles de carpetas
rellenarSeleccionados();

    }
    @Override
    protected void onPostCreate(Bundle savedInstanceState) {
        super.onPostCreate(savedInstanceState);
        // Sync the toggle state after onRestoreInstanceState has occurred.
        toggle.syncState();
    }

    private void getKeyEncriptacion(){
```

```
        SharedPreferences sharedPref =
this.getPreferences(Context.MODE_PRIVATE);
        this.keyEncriptacion =
sharedPref.getString("key",this.keyEncriptacionDefault);//por defecto es vacio
    }
    private void setKeyEncriptacion(String keyEncriptacion){
        SharedPreferences sharedPref =
this.getPreferences(Context.MODE_PRIVATE);
        SharedPreferences.Editor editor = sharedPref.edit();
        editor.putString("key",keyEncriptacion);
        editor.commit();
    }
    public void eliminarHijos(int posicion){
        int contador = this.lista.size() - (posicion + 1);
        for( int i = 0 ; i < contador ; i++){
            this.lista.remove(this.lista.size()-1);
Log.d("contador ",contador+"");
        }
        Log.d("Hijos eliminados  ",this.lista.size()+"");
    }
    public void rellenarSeleccionados(){

        //limpiamos los datos de listaCarpetaSeleccionada
this.listaCarpetaSeleccionada.clear();
        for( int i = 0; i < this.lista.size() ; i++){
this.listaCarpetaSeleccionada.add(new CarpetaSeleccionada(this.lista.get(i),i));
        }
        //notifica de los cambios realizados en la lista de carpetas.
        //mAdapterCarpetaSeleccionada.notifyDataSetChanged();
    }
public void iniciar()  {

File v = buscarFile(this.lista,this.lista.size() - 1);
        if(v != null){
```

```
            Log.d("Iniciar v :: ",this.lista.size()+" ");
if(v.isDirectory()){
                //notifica de los cambios realizados en la lista de carpetas.
                mAdapterCarpetaSeleccionada.notifyDataSetChanged();
                rellenarCarpeta(v);

            }else{

this.lista.remove(this.lista.size()-1);
//mostrar el archivo en un activity.

                if(v.isFile()){
                    Log.d("Archivo a mostrar :: ",v.getAbsolutePath());
                    mostrarArchivo(v);
                }

            }
        }
    }

    //leer File original -> obtener bytes[] -> encriptar los bytes[] ->
    //crear objeto serializable -> crear nuevo File -> serializar objeto -> eliminar File
original.
public Boolean asegurarArchivo(File archivo){

        /*
        FileInputStream inputStream;
        byte[] inputBytes = null;
        try {
            //leer File original
            inputStream = new FileInputStream(archivo);
            inputBytes = new byte[(int) archivo.length()];
            //obtener bytes[]
```

```
            inputStream.read(inputBytes);
            inputStream.close();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }


        //encriptar los bytes[]
        byte[] encriptBytes =
this.encriptacionFile.encrypt(this.keyEncriptacion,inputBytes);

//Realizar el proceso de renombrar el archivo a la nueva extension.
String cuerpoUrl = reemplazarExtension(archivo.getAbsolutePath());
        String extensionOriginal = getExtension(archivo.getAbsolutePath());
String renombrar = cuerpoUrl + this.extension;

        //creamos el objeto a serializar.
        ObjectSerializar objetoSerializar = new ObjectSerializar();
        objetoSerializar.setNombre(cuerpoUrl);
        objetoSerializar.setExtension(extensionOriginal);
        objetoSerializar.setUsuario(this.usuario);
        objetoSerializar.setPassword(this.password);
        objetoSerializar.setKeyEncriptacion(this.keyEncriptacion);
        objetoSerializar.setArchivo(encriptBytes);

        //crear nuevo File
        File nuevo = new  File(renombrar);

        //encriptando el objeto serializado.
try {
            Boolean crear = nuevo.createNewFile();
            if(crear){
```

```
//serializar objeto
            this.serializacionFile.serialize(objetoSerializar,nuevo);

            //eliminamos el archivo anterior
            archivo.delete();
            Log.d("cual create ::",""+nuevo.getAbsolutePath());
}

      } catch (IOException e) {
         e.printStackTrace();
      }
      */
      FileInputStream inputStream;
      byte[] inputBytes = null;
      try {
         //leer File original
         inputStream = new FileInputStream(archivo);
         inputBytes = new byte[(int) archivo.length()];
         //obtener bytes[]
         inputStream.read(inputBytes);
         inputStream.close();

         //encriptar los bytes[]
         byte[] encriptBytes =
this.encriptacionFile.encrypt(this.keyEncriptacion,inputBytes);

//Realizar el proceso de renombrar el archivo a la nueva extension.
String cuerpoUrl = reemplazarExtension(archivo.getAbsolutePath());
         String extensionOriginal = getExtension(archivo.getAbsolutePath());
String renombrar = cuerpoUrl + this.extension;

         //creamos el objeto a serializar.
         ObjectSerializar objetoSerializar = new ObjectSerializar();
         objetoSerializar.setNombre(cuerpoUrl);
```

```
            objetoSerializar.setExtension(extensionOriginal);
            objetoSerializar.setUsuario(this.usuario);
            objetoSerializar.setPassword(this.password);
            objetoSerializar.setKeyEncriptacion(this.keyEncriptacion);
            objetoSerializar.setArchivo(encriptBytes);

            //crear nuevo File
            File nuevo = new  File(renombrar);

            Boolean crear = nuevo.createNewFile();
            if(crear){
               //serializar objeto
               this.serializacionFile.serialize(objetoSerializar,nuevo);

               //eliminamos el archivo anterior
               archivo.delete();
               Log.d("cual create ::",""+nuevo.getAbsolutePath());
}


        } catch (FileNotFoundException e) {
            e.printStackTrace();
            return false;
        } catch (IOException e) {
            e.printStackTrace();
            return false;
        }
Log.d("cual fin :: ","Todo termino");
return true;
}

    //leer File original -> deserializar objeto -> obtener los bytes[] ->
    //desencriptar los bytes -> crear nuevo File -> eliminar File original.
public Boolean revertirArchivo(File archivo) {
```

```
/*
        ObjectSerializar objetoDeserializar = null;

        try {
            //deserializar objeto
            objetoDeserializar =
(ObjectSerializar)this.serializacionFile.deserialize(archivo);
} catch (IOException e) {
            e.printStackTrace();
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
}

        //obtener los bytes[]
        byte[] outputBytes = objetoDeserializar.getArchivo();

        //desencriptar los bytes
byte[] desencriptedBytes =
this.encriptacionFile.decrypt(this.keyEncriptacion,outputBytes);

//Realizar el proceso de renombrar el archivo a la nueva extension.
        String cuerpoUrl = reemplazarExtension(archivo.getAbsolutePath());
        String extensionOriginal = objetoDeserializar.getExtension();
        String renombrar = cuerpoUrl + extensionOriginal;

        Log.d("revert :: ",""+renombrar);
        //creammos un nuevo File, para almacenar la informacion.
        File nuevo = new  File(renombrar);
FileOutputStream outputStream;
        try {
            Boolean crear = nuevo.createNewFile();
            if(crear){
                outputStream = new FileOutputStream(nuevo);
```

```
                outputStream.write(desencriptedBytes);
outputStream.close();

//eliminamos el archivo original.
                archivo.delete();

                Log.d("revert crear :: ","+crear);
}
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
e.printStackTrace();
}
        */

        ObjectSerializar objetoDeserializar = null;

        try {
            //deserializar objeto
            objetoDeserializar =
(ObjectSerializar)this.serializacionFile.deserialize(archivo);

            //obtener los bytes[]
            byte[] outputBytes = objetoDeserializar.getArchivo();

            //desencriptar los bytes
byte[] desencriptedBytes =
this.encriptacionFile.decrypt(this.keyEncriptacion,outputBytes);

//Realizar el proceso de renombrar el archivo a la nueva extension.
            String cuerpoUrl = reemplazarExtension(archivo.getAbsolutePath());
            String extensionOriginal = objetoDeserializar.getExtension();
            String renombrar = cuerpoUrl + extensionOriginal;
```

```
            Log.d("revert :: ",""+renombrar);
            //creammos un nuevo File, para almacenar la informacion.
            File nuevo = new  File(renombrar);
FileOutputStream outputStream;

            Boolean crear = nuevo.createNewFile();
            if(crear) {
               outputStream = new FileOutputStream(nuevo);
               outputStream.write(desencriptedBytes);
               outputStream.close();

//eliminamos el archivo original.
               archivo.delete();

               Log.d("revert crear :: ", "" + crear);
}
        } catch (IOException e) {
            e.printStackTrace();
            return false;
        } catch (NullPointerException e){
            return false;
        }
        return true;
    }
    public String reemplazarExtension(String cadena){

        String parts[] = cadena.split("\\.");
        String retorno = "";
        for (int i = 0; i < parts.length - 1; i++) {
            retorno = retorno + parts[i] + ".";
        }
        return retorno;
    }
    public void mostrarArchivo(File file){
```

```
        Intent intent = new Intent();
        intent.setAction(android.content.Intent.ACTION_VIEW);
        //Uri uri = Uri.parse("file://" + file.getAbsolutePath());
        Uri uri = Uri.parse("file://" +file.getAbsolutePath());
        //intent.setDataAndType(uri,"image/*");
        intent.setDataAndType(uri,getMimeType(file.getAbsolutePath()));
        startActivity(intent);

    }

    public String getMimeType(String url)
    {
        //String parts[]=url.split("\\.");
        //String extension=parts[parts.length-1];
        String extension = getExtension(url);
        String type = null;
        if (extension != null) {
            MimeTypeMap mime = MimeTypeMap.getSingleton();
            type = mime.getMimeTypeFromExtension(extension);
        }

        return type;
    }

    public String getExtension(String url){

        String parts[]=url.split("\\.");
        String extension=parts[parts.length-1];

        Log.d("Extension : ",extension);

        return extension;
}
```

```
    public void  rellenarCarpeta(File lista){

        File[]archivos;
        Log.d("Archivo Interno ",lista.getName());
        Log.d("Archivo Padre ",lista.getParent());
Log.d("Archivo Path ",lista.getPath());
Log.d("Archivo Espacio Total
",lista.getTotalSpace()/(1024.0*1024.0*1024.0)+"GB");
        Log.d("Archivo Espacio Total
",lista.getUsableSpace()/(1024.0*1024.0*1024.0)+"GB");


        if(lista != null){

            listaCarpeta.clear();//Eliminando informacion restante.

archivos = lista.listFiles();
            for(int i=0;i<archivos.length;i++){
                Log.d("Lista archivos ::  ",archivos[i].getName()+"
"+archivos[i].lastModified()+"");
                File[] arc = archivos[i].listFiles();
                Integer tam = 0;
                if(arc != null){
                    tam = archivos[i].listFiles().length;
                }
                listaCarpeta.add(
                        new Carpeta(
                                archivos[i].getName(),
                                new Date(archivos[i].lastModified()),
                                tam
                        ));


                /*
```

```
            listaCarpeta.add(
                    new Carpeta(
                            archivos[i].getName(),
                            new Date(archivos[i].lastModified()),
                            archivos[i].listFiles().length
                    ));
                    */
        }
        mAdapterCarpeta.notifyDataSetChanged();
    }
}

@Override
public void onBackPressed() {
    DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
    if (drawer.isDrawerOpen(GravityCompat.START)) {
        drawer.closeDrawer(GravityCompat.START);
    } else {
        super.onBackPressed();
    }

}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.main, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
```

```
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    @SuppressWarnings("StatementWithEmptyBody")
    @Override
    public boolean onNavigationItemSelected(MenuItem item) {
        // Handle navigation view item clicks here.
        int id = item.getItemId();

        if (id == R.id.nav_camera) {
            //Cambiar usuario - password
            Intent intent = new Intent(this, LoginActivity.class);
intent.putExtra("opcion",false);//actualizar usuario password.
startActivity(intent);
            finish();

        } else if (id == R.id.nav_gallery) {

//Despliega el cuadro de dialogo
            showNoticeDialog(this.keyEncriptacion,"","Cambiar clave de
encriptación","dialogo");

} else if (id == R.id.closeApp) {
            this.finish();
        }
```

```
        DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
        drawer.closeDrawer(GravityCompat.START);
        return true;
    }
    /* Checks if external storage is available for read and write */
    public boolean isExternalStorageWritable() {
        if(Environment.isExternalStorageRemovable()){
Log.d("removible", "Removible");
        }else{
            Log.d("removible", " No Removible");
}
        String state = Environment.getExternalStorageState();
        Log.d("sd ","writable "+state);
        if (Environment.MEDIA_MOUNTED.equals(state)) {
            return true;
        }
        return false;
    }

    /* Checks if external storage is available to at least read */
    public boolean isExternalStorageReadable() {
        String state = Environment.getExternalStorageState();
        Log.d("sd ","readable "+ state);
        if (Environment.MEDIA_MOUNTED.equals(state) ||
                Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)) {
            return true;
        }
        return false;
    }
    @Override
    public void onClick(View v) {

        if(v.getId() == this.btnAsegurarArchivo.getId()){
```

```
//obtener la lista de archivos para asegurar.
ArrayList<File> tem = this.obtenerArchivos(true);
            for(int i = 0; i < tem.size(); i++){
                Log.d("item ",tem.get(i).getAbsolutePath());

            }
            this.progressDialogAsegurarRevertir(tem,true);
        }
        if(v.getId() == this.btnReconstruirArchivo.getId()){

//obtener la lista de archivos para reconstruir.
ArrayList<File> tem = this.obtenerArchivos(false);
            for(int i = 0; i < tem.size(); i++){
                Log.d("item ",tem.get(i).getAbsolutePath());

            }
            this.progressDialogAsegurarRevertir(tem,false);
        }

    }

    private ArrayList<File> obtenerArchivos(Boolean opcion){

        ArrayList<File> listaTemporal = new ArrayList<>();
        if(this.carpetaHallada != null){

            File[] v = this.carpetaHallada.listFiles();
            for(int i = 0; i < v.length; i++){
                if(v[i].isFile()){
                    //obtener la extension.
                    String extension = this.getExtension(v[i].getAbsolutePath());

if(opcion){//asegurar el archivo.
if(!extension.equals("dvc")){
```

```
                    listaTemporal.add(v[i]);
                }

            }else{//reconstruir el archivo.
                if(extension.equals("dvc")){
                    listaTemporal.add(v[i]);
                }
            }
        }
    }

    return listaTemporal;
}

private void progressDialogAsegurarRevertir(final ArrayList<File> lista,final
Boolean opcion){

//true = asegurar - false = revertir.
final String val = opcion ? "No hay elementos para asegurar" : "No hay elementos
para revertir";
    final String titulo = opcion ? "Asegurando los archivos" : "Reconstruyendo los
archivos";


if(lista.size() > 0){

        /*
        this.runOnUiThread(new Runnable() {
            @Override
            public void run() {
                Integer contador = 0;
                for(int i = 0;  i < lista.size(); i++ ){
```

```
if(opcion){//asegurar el archivo.
if(MainActivity.this.asegurarArchivo(lista.get(i))){
Log.d("ui asegurar ",lista.get(i).getAbsolutePath());
                    }

                    contador ++;
                    progressDialog.setProgress(contador);
                }else{//reconstuir el archivo.

if(MainActivity.this.revertirArchivo(lista.get(i))){
                        Log.d("ui reconstruir ",lista.get(i).getAbsolutePath());
}
                    contador++;
                    progressDialog.setProgress(contador);
                }
            }

            Log.d("ui fin ","fin del progress");
            //cerrando el progress dialog.
            progressDialog.dismiss();
}
});
        */
        new AsyncTask<String, Void, Boolean>() {

            private ProgressDialog progressDialog;

            @Override
            protected Boolean doInBackground(String... params) {

Integer contador = 0;
                Log.d("ui iniciar ","Iniciando el progress dialog");
for(int i = 0;  i < lista.size(); i++ ){
```

```
if(opcion){//asegurar el archivo.
if(MainActivity.this.asegurarArchivo(lista.get(i))){
Log.d("ui asegurar ",lista.get(i).getAbsolutePath());
                    }
                    contador ++;
                    progressDialog.setProgress(contador);
                }else{//reconstuir el archivo.

if(MainActivity.this.revertirArchivo(lista.get(i))){
                        Log.d("ui reconstruir ",lista.get(i).getAbsolutePath());
                    }
                    contador++;
                    progressDialog.setProgress(contador);
                }
            }
            return false;
        }
        protected void onPreExecute() {
            Log.d("ui preexecute ","preexecute");
            progressDialog = new ProgressDialog(MainActivity.this);
            progressDialog.setMax(lista.size());
            progressDialog.setProgress(0);
            progressDialog.setMessage("Procesando ...");
            progressDialog.setTitle(titulo);

progressDialog.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
progressDialog.setCancelable(false);
            progressDialog.show();
}
        protected void onPostExecute(Boolean valor) {
Log.d("ui fin ","onPostexecute "+valor);
            //cerrando el progressdialog.
            progressDialog.dismiss();
```

```
                ////actualiza las vistas.
                MainActivity.this.iniciar();
}
            protected void onProgressUpdate(Void... val) {

            }
        }.execute("iniciar");

    }else{
        Toast.makeText(this,val,Toast.LENGTH_LONG).show();
    }
  }

  //parte del dialogo
  public void showNoticeDialog(String claveAnterior,String claveActual,String
tituloDialog,String nombreDialog) {
    // Create an instance of the dialog fragment and show it
/*
    DialogFragment dialog = new CarpetaDialogo();
    dialog.show(getSupportFragmentManager(), "CarpetaDialogo");
    */
    DialogFragment dialog =
CarpetaDialogo.newInstance(claveAnterior,claveActual,tituloDialog);
    dialog.show(getSupportFragmentManager(), nombreDialog);
  }

  //adicionar un elmento a la lista de navegacion
public void addListaNavegacion(String valor){
this.lista.add(valor);
    rellenarSeleccionados();
  }


  //busqueda de Puntero a raiz del archivo
```

```
public void buscarPuntero(){



        Boolean isSDPresent =
Environment.getExternalStorageState().equals(Environment.MEDIA_MOUNTED);
        if(isSDPresent)
        {
            Log.d("sd ","presente");
            String path =
getApplicationContext().getExternalCacheDir().getAbsolutePath();
            Log.d("sd dir",path);
            File lista  = Environment.getExternalStoragePublicDirectory(
                    Environment.DIRECTORY_PICTURES);
            File[] archivos;
            if(lista != null){
                archivos = lista.listFiles();
                for(int i=0;i<archivos.length;i++){
                    Log.d("Lista archivos ::  ",archivos[i].getName()+"
"+archivos[i].getAbsolutePath()+"");

                }
            }



        }
        else
        {
            Log.d("sd","ausente");
}

//devuelve un File a la raiz de los archivos.
        //carpetaHallada = this.getFilesDir();
        carpetaHallada = Environment.getExternalStorageDirectory();
```

```
        Log.d("sd other",carpetaHallada.getAbsolutePath());

        if(!carpetaHallada.exists()){//La raiz no existe
            carpetaHallada = null;

        }

    }
    //Busca la carpeta especifica en una lista de rutas, en profundidad hasta cierto
nivel
    //Inicialmente se busca de que
public File buscarFile(ArrayList<String> lista, int nivel){

buscarPuntero();

        //Encontrar el puntero raiz.
        if(carpetaHallada != null){
            File temporal = carpetaHallada;
for(int i = 1; i <= nivel ; i++){
                carpetaHallada = new File(temporal,lista.get(i));
//carpeta existe
                if(carpetaHallada.exists()){
                    temporal = carpetaHallada;
}else{
                    return null;
                }
            }
        }

        return carpetaHallada;
    }

//Encontrar el puntero al archivo y verificar que la carpeta no exista.
```

```
public boolean carpetasRepetidas(File temporal, String buscar){

        File v = new File(temporal,buscar);
//la carpeta existe y no puede crearse
if(v.exists()){
            return false;
        }
        return true;
    }

    @Override
    public void onDialogPositiveClick(DialogFragment dialog) {

        EditText v =
(EditText)dialog.getDialog().findViewById(R.id.editTextCarpetaNombre);
this.inputDialogNombre = v.getText().toString();

        //tamaño de 16 caracteres
if(this.inputDialogNombre.isEmpty()){
            String cadena = new String(this.keyEncriptacionDefault);
            this.setKeyEncriptacion(cadena);
            this.getKeyEncriptacion();
        }else{

//maximo numero de carateres para la clave.
char[] array = new char[this.keyEncriptacionDefault.length()];

            for(int i = 0; i < this.keyEncriptacionDefault.length(); i++){
//reemplazando los valores.
array[i] = this.keyEncriptacionDefault.charAt(i);

}

            Log.d("encrip default"," "+this.keyEncriptacionDefault);
```

```
            for(int i = 0; i < this.inputDialogNombre.length(); i++){
                //reemplazando los valores.
                array[i%this.keyEncriptacionDefault.length()] =
this.inputDialogNombre.charAt(i);

            }
            Log.d("encrip input"," "+this.inputDialogNombre);
            String cadena = new String(array);
Log.d("encrip cadena"," "+cadena);
            this.setKeyEncriptacion(cadena);
            this.getKeyEncriptacion();
        }
    }
    @Override
    public void onDialogNegativeClick(DialogFragment dialog) {
        Log.d("Dialogo","cancelar");
    }

}
```