UNIVERSIDAD AUTÓNOMA DEL ESTADO DE HIDALGO

ESCUELA SUPERIOR DE HUEJUTLA

LICENCIATURA EN CIENCIAS COMPUTACIONALES

GENERACIÓN Y VALIDACIÓN DE FIRMAS DIGITALES EN IMÁGENES A ESCALA DE GRISES

PRESENTA:

KEVING ALBERTO RAMÍREZ BARRAGÁN

ASESORES:

MTRO. VÍCTOR TOMÁS TOMÁS MARIANO

MTRO. FELIPE DE JESÚS NÚÑEZ CÁRDENAS

Noviembre 2017

ÍNDICE

TABLA DE ILUSTRACIONES

TABLA DE IMÁGENES DE LA INTERFAZ Y SU PROCESO

IMÁGENES DEL ALGORITMO DSA

TABLA DE IMÁGENES DE RESULTADOS OBTENIDOS

RESUMEN

El presente proyecto **Generación y Validación de Firmas Digitales en Imágenes a Escala de Grises,** permite elegir una imagen a color que es transformada a una escala de grises, para su posterior firma y haciendo uso del **Algoritmo de Firma Digital (DSA)**, le permite firmar la imagen y a su vez hacer una validación de la firma.

Dicho sistema esta creado en el lenguaje de programación Java con la herramienta de NetBeans, el sistema consta de tres etapas, las cuales son:

Etapa 1: *Elegir una Imagen*: Elige una imagen a color para su posterior transformación a una escala de grises, el resultado es una imagen a gris y un archivo (.txt), este archivo contiene valores comprendidos entre 0-255 como información de la imagen.

Etapa 2: *Firma de Imagen*: Este proceso hace uso del algoritmo **DSA**, el cual usa el archivo (.txt), que contiene los valores comprendidos entre 0-255, al firmar genera tres archivos (.txt), el primer archivo contiene la firma, el segundo una clave pública para ser compartida, y el tercer archivo contiene una clave privada la cual solo ara uso el usuario que realizo la firma.

Etapa 3: *Verificación de la Firma*: Este proceso hace uso de tres archivos, la clave pública, la firma, y la imagen a escala de grises.

El sistema ayuda a mantener la integridad de la información contenida en una imagen, asegura que un archivo no sea modificado en el envió de una red o internet.

Palabras clave: Seguridad, Integridad, Firma, Verificación, Generar, Procesar.

ABSTRACT

The present project **Generation and Validation of Digital Signatures in Gray Scale Images**, allows to choose a color image that is transformed to a gray scale, for later signature and use of the **Digital Signature Algorithm (DSA),** it allows you to sign the image and turn to make a validation of the signature.

This system is created in the Java programming language with the NetBeans tool, the system consists of three stages, which are:

Stage 1: *Choose an Image*: Choose a color for further transformation to a gray scale, the result is a gray image and a file (.txt), this file contains values between 0-255 as image information.

Stage 2: *Image Signature*: This process has use of the DSA algorithm, which uses the file (.txt), which contains the values between 0-255, when signing generates three files (.txt), the first file contains the signature, the second a public key to be shared, and the third file contains a private key that only the user who made the signature uses.

Stage 3: *Verification of the Firm*: This process makes use of three files, the public key, the signature, and the gray-scale image.
The system helps maintain the integrity of the information contained in an image, ensures that a file was not modified when sending a network or internet.

Key words: Security, Integrity, Signature, Verification, Generate, Process.

1 GENERALIDADES

1.1 INTRODUCCIÓN

Hoy en día las instituciones financieras y publicas afiliadas al Sistema Nacional de Pagos Electrónicos, entre otras, quieren un mejor método para firmar cualquier tipo de documento, pero a su vez quieren proteger la integridad de la información, en este proyecto se desarrollará un sistema como prototipo, la cual generara una firma por medio de una imagen digital a escala de grises, posteriormente esta firma se validará para saber que sea correcta.

El presente proyecto trata de firmar una imagen digital y a su vez validarla para su autenticidad, esto quiere decir que a dicha imagen se le aplicara un estado de grises, de manera que se puedan obtener valores entre 0-255 como información de la imagen transformada, ya obtenidos estos valores de la imagen la información servirá para su posterior firma y validación.

Con este prototipo de sistema creado en el lenguaje de programación Java utilizando la herramienta de NetBeans, se pretende a futuro que se implemente en aplicaciones móviles.

El desarrollo de este sistema se basará en la extracción de valores entre 0-255, los cuales se obtendrán al pasar la imagen a escala de grises. Este sistema hará uso del **Algoritmo de Firma Digital (DSA)**, para poder firmar una imagen y posteriormente verificar la firma de la misma.

El **DSA,** es un estándar del Gobierno Federal de los Estados Unidos de América o FIPS para firmas digitales. (Morales, s.f.)

1.2 PLANTEAMIENTO DEL PROBLEMA

Cada vez más personas y organizaciones usan documentos digitales en lugar de documentos en papel para llevar a cabo sus operaciones cotidianas. Al reducir la dependencia de documentos en papel, protegemos el medio ambiente y preservamos los recursos del planeta. Las firmas digitales respaldan este cambio, pues ofrecen garantías de la validez y protegen la integridad de la información.

Actualmente las firmas digitales son muy usadas por instituciones financieras y publicas afiliadas al Sistema Nacional de Pagos Electrónicos, entre otras, la firma digital es el equivalente a la cédula en Internet y genera los mismos compromisos que una firma personal en un papel. En los bancos permite ejecutar operaciones electrónicas, sin usar el *token* (dispositivo electrónico de autentificación) o la clave dinámica, así como la oficialización de trámites tales como un crédito.

Los pros y contras de los bancos que tienen incorporada la firma digital resaltan que es el mecanismo más seguro para validar la información por medio de la banca electrónica. (Nacion, 2013)

“Con este sistema conocemos al dueño del documento. Las transacciones que se firman digitalmente van encriptadas con los estándares más altos de seguridad”, enfatizó Luis Ángel González, gerente de Banca de Personas de Lafise.

Errol Gamboa, gerente de Medios Electrónicos del Banco de Costa Rica (BCR), resaltó que con el sistema el cliente puede firmar electrónicamente sus transacciones en la web como si lo hiciera en físico.

Por otro lado, Fernando Víquez, gerente de Banco Bansol, que aún no posee la firma digital, dijo que el mecanismo funciona con clientes que mueven altas sumas de dinero. “Para usarse el sistema requiere que la persona compre su tarjeta, tenga una computadora habilitada y un portal USB; si no, no puede utilizarlo”, detalló.

Por eso este proyecto pretende realizar un prototipo de sistema que sirva como guía para futuros en una aplicación móvil que pueda ser de utilidad en cualquier ámbito.

1.3 JUSTIFICACIÓN

El motivo por el cual se pretende hacer este proyecto es para ofrecer una mayor seguridad en perdida de información en instituciones financieras y publicas afiliadas al Sistema Nacional de Pagos Electrónicos, entre otras, para aumentar globalización de este mecanismo y proveer a los usuarios documentos auténticos, sin pérdida de información y validados por la persona correcta. Una de las maneras de lograr esto es por medio de la generación y validación de firmas digitales en imágenes a escala de grises, que se podrá implementar para cualquier ámbito relacionado con su funcionamiento. Ya que este sistema es capaz de generar una firma por medio de una fotografía a color y esta firma podrá ser validada para identificar si es correcta. A demás este sistema podría ser funcional en una penitenciaria de alta seguridad, para identificar a cada reo existente.

1.4 OBJETIVO GENERAL

Diseñar un sistema que permita la generación y validación de una firma digital a partir de una imagen a color pasando por una transformación a escala de grises, así como también que ofrezca integridad en la información.

1.5 OBJETIVO ESPECÍFICOS

- Realizar una investigación documental, para atender el tema a desarrollar.
- Realizar una primera versión del sistema en NetBeans que sirva como guía.
- Dicho sistema debe ofrecer integridad.

1.6 ANTECEDENTES

En la actualidad, así como existen sistemas que están basadas en las imágenes a escala de grises, también lo hay para la firma digital, todos estos sistemas están al alcance de cualquier persona, ya que se encuentran en, tabletas, celulares, correos electrónicos, transacciones electrónicas, en proveedores y consumidores, entre las aplicaciones que tienen cavidad en las firmas digitales se destaca la paquetería office, las firmas en los archivos de Microsoft Word, Microsoft Excel o Microsoft PowerPoint, que reciben una marca de tiempo de un servidor seguro, tienen, en determinadas circunstancias, la validez de una certificación.

En la Universidad Autónoma del Estado de Hidalgo, Escuela Superior Huejutla, dos alumnos de la Lic. Ciencias Computacionales, de octavo semestre realizaron como trabajo final un algoritmo de firmas digitales en archivos de texto, así como también un equipo de cuatro integrantes como proyecto final realizaron el reconocimiento de patrones por medio de una cámara web.

2 ESTADO DEL ARTE

2.1 PROCESAMIENTO DE IMÁGENES

El procesamiento de imágenes tiene como objetivo mejorar el aspecto de las imágenes y hacer más evidentes en ellas ciertos detalles que se desean hacer notar. La imagen puede haber sido generada de muchas maneras, por ejemplo, fotográficamente, o electrónicamente, por medio de monitores de televisión. El procesamiento de las imágenes se puede en general hacer por medio de métodos ópticos, o bien por medio de métodos digitales, en una computadora.

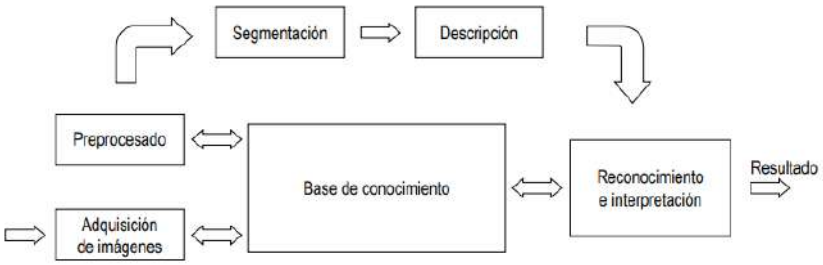


Ilustración 1. Procesado de la Imagen

El desarrollo de los métodos de procesamiento digital de imágenes tiene su origen en dos áreas principales de aplicación: el mejoramiento de la información pictórica para la interpretación humana, y el procesamiento de datos de la imagen para la percepción de máquina autónoma en el que se incluyen etapas de transmisión y/o almacenamiento de estos datos. La herramienta usada en el tratamiento digital de las imágenes son las matemáticas; los conceptos que se verán son básicos El término "imagen monocromática" o imagen simplemente, se refiere a una función de intensidad de luz bidimensional f(x,y). Una imagen digital puede ser considerada como una matriz cuyos índices de renglón y columna

identifican un punto. Los elementos de estos arreglos digitales son llamados elementos de imagen o pixeles.

En el tratamiento de imágenes se pueden distinguir tres etapas principales: (Ramírez, 2006) (Malacara, s.f.)

1 Adquisición de la imagen.
2 Procesamiento de la imagen.
3 Presentación al observador.

- El artículo de Procesamientos de Imágenes Digitales elaborado por el Dr. Rubén Wainschenker, Mg. Ing. José María Massa y Mg. Ing. Paula Tristan, da una explicación sobre las imágenes digitales, las herramientas utilizadas para la extracción de información de la imagen, así como también la lectura de bits. (Wainschenker, s.f.)

- El artículo de Formatos de Imagen Digital elaborado por el pasante de letras hispánicas Cristian Andrés Ordoñez Santiago, habla que durante la creación de una revista utilizan varios tipos de formatos de imágenes, esto para que este mejor presentado, y da a conocer varios formatos que utilizan las revistas y su representación en bits. (Santiago C. A., 2005)

- El artículo de Procesamiento y Visualización, Tridimensional de Imágenes, Biomédicas del Microscopio Confocal, elaborado por el Marta García Nuevo, habla de la reconstrucción de imágenes tridimensionales, partiendo de series de imágenes bidimensionales obtenidas con el microscopio con focal. Trataremos de encontrar un modo sencillo y eficaz de visualizar esta imagen tridimensional, e incluso mejorarla utilizando distintas técnicas de procesamiento digital de imágenes. (Nuevo, s.f.)

- El articulo Procesando Imagen Digital, tercera edición, elaborado por Rafael C. Gonzales, habla de proporcionar una introducción a los conceptos y metodologías básicas para el procesamiento de imagen digital, y para desarrollar una base que se puede utilizar como base para otros estudios e investigaciones en este campo. (Gonzales, s.f.)

- El articulo Procesado digital de imágenes de video para la detección de humo, elaborado por Rubén Llobregat Rubio, habla de detectar humo de ficheros de vídeo de incendios para ello obtendremos al final del proceso una imagen en la que se indicara las zonas de humo reconocidas. (Rubio, 2011)

2.2 SEGMENTACIÓN DE IMÁGENES

La segmentación subdivide una imagen en sus partes constituyentes u objetos, con el fin de separar las partes de interés del resto de la imagen. Los atributos básicos de segmentación de una imagen son: la luminancia en imágenes monocromáticos los componentes de color en imágenes en color, textura, forma, etc.

Los algoritmos de segmentación de imágenes monocromáticas generalmente se basan en una de las dos propiedades básicas de los valores del nivel de gris: discontinuidad y similaridad. En la discontinuidad el método consiste en dividir una imagen basándose en los cambios bruscos del nivel de gris. Los temas más importantes en la discontinuidad son:

- Detección de puntos aislados.
- Detección de líneas
- Detección de bordes de una imagen.

En la similaridad, se presenta la regularidad en los valores del nivel de gris, los principales métodos están basados en a) umbralización, b) crecimiento de región, y c) división y fusión de regiones. La segmentación de una imagen basado en la discontinuidad o en la similaridad de los valores del nivel de gris de sus pixeles es

aplicable tanto a las imágenes estáticas como a las dinámicas (variantes en el tiempo). (Palomino, s.f.)

Ilustración 2. Segmentación de la mitad de la Imagen.

- El articulo Diseño de Sistema de Reconocimiento de Placas utilizando MATLAB, elaborado por Paul Cárdenas Hidalgo, José Alfredo Flores Vargas, Jaime López Zabaleta, Pablo Martínez Moreno, habla del reconocimiento de matrices vehiculares capaz de procesar una imagen para tomar información de la misma y poder ser almacenada en una base de datos. (Hidalgo, s.f.)

- El articulo Una estrategia de segmentación de imágenes digitales de huellas dactilares latentes, elaborado por Ruíz E. María E., Morales S. Miguel y Hernández M. Yahir, habla sobre la detención de las huellas dactilares de un ser humano, y del proceso que pasa para su lectura de información. (Ruíz E. María E., s.f.)

- El articulo Segmentación de Imágenes de Color, elaborado por J.J. Baez Rojas, M.L. Guerrerob, J. Conde Acevedoc, A. Padilla Vivancoa y G. Urcid

Serranoa, que habla de las modificaciones del sistema de color, generando mapas temáticos con sus diferentes fases de color. (J.J. Baez Rojas, s.f.)

- El articulo Aplicaciones del Tratamiento Inteligente de Imágenes, elaborado por Silvia Kalbakdij Sánchez, Pablo Lebrero Villar, Salvador Sánchez Rodríguez y dirigido por Luís Garmendia Salvador, habla de cómo a partir del tratamiento de las imágenes obtenidas en tiempo real, por una cámara, se capta el movimiento de un objeto seleccionado para posteriormente llevar a cabo tres aplicaciones. (Salvador, s.f.)

- El articulo Procesamiento y Análisis Digital de Imágenes Mediante Dispositivos Lógicos Programables, elaborado por C.Manuel Alejandro Mendoza Manzano habla del diseño de una herramienta para el ámbito académico y de investigación, enfocada al ´área de procesamiento y análisis digital de imágenes sobre Circuitos Digitales Configurables (CDC), específicamente sobre un Arreglo de Compuertas Programable en Campo (FPGA, Field Programable Gate Array). (MANZANO, s.f.)

3 MARCO TEÓRICO

En este apartado se hablará de los temas sobre la generalización y validación de firmas digitales en imágenes digitales, con el objetivo de obtener un conocimiento más amplio a cerca de los métodos y técnicas que se pueden emplear para lograr lo que deseamos obtener.

Los siguientes términos y definiciones muestran las garantías que proporcionan las firmas digitales.

- **Autenticidad**: Confirmación del firmante como persona que firma el documento.
- **Integridad:** La firma digital permite garantizar que el contenido no se ha cambiado ni se ha manipulado desde que se firmó digitalmente.
- **No rechazo:** Prueba a todas las partes el origen del contenido firmado. Por rechazo se entiende el acto de un firmante de negar cualquier asociación con el contenido firmado.
- **Certificación**: Las firmas en los reciben una marca de tiempo de un servidor seguro, tienen, en determinadas circunstancias, la validez de una certificación.

Una imagen, a groso modo, se trata de una matriz de putos. Siempre que se desee acceder a un punto en concreto se debe utilizar una coordenada, (X, Y), donde ‘X’ representa el ancho de la imagen, ‘Y’ representa el alto de la misma, con esto cumple la coordinación y ampliación de F, F son valores discretos finitos que contienen un numero asignado a una posición, en cualquier par de coordenadas. La profundidad de bits es determinada por la cantidad de bits utilizados para definir cada píxel. Cuanto mayor sea la profundidad de bits, tanto mayor será la cantidad de tonos (escala de grises o color) que puedan ser representados. Las imágenes digitales se pueden producir en blanco y negro (en forma bitonal), a escala de grises o a color.

Una imagen bitonal está representada por píxeles que constan de 1 bit cada uno, que pueden representar dos tonos (típicamente negro y blanco), utilizando los valores 0 para el negro y 1 para el blanco o viceversa. Una imagen a escala de grises está compuesta por píxeles representados por múltiples bits de información, que típicamente varían entre 2 a 8 bits o más.

Ejemplo: “En una imagen de 2 bits, existen cuatro combinaciones posibles: 00, 01, 10 y 11. Si "00" representa el negro, y "11" representa el blanco, entonces "01" es igual a gris oscuro y "10" es igual a gris claro. La profundidad de bits es dos, pero la cantidad de tonos que pueden representarse es 2 2 ó 4. A 8 bits, pueden asignarse 256 (2-8) tonos diferentes a cada píxel.”

Una imagen a color está típicamente representada por una profundidad de bits entre 8 y 24 o superior a ésta. En una imagen de 24 bits, los bits por lo general están divididos en tres grupos: 8 para el rojo, 8 para el verde, y 8 para el azul. Para representar otros colores se utilizan combinaciones de esos bits. Una imagen de 24 bits ofrece 16,7 millones (2-24) de valores de color. Cada vez más, los escáneres están capturando 10 bits o más por canal de color y por lo general imprimen a 8 bits para compensar el "ruido" del escáner y para presentar una imagen que se acerque en el mayor grado posible a la percepción humana.

3.1 DIGITALIZACIÓN DE IMÁGENES

Una imagen digital es una foto electrónica tomada de una escena o escaneada de un documento: fotografías, manuscritos, textos impresos, e ilustraciones. Se obtiene mediante la confección de un mapa de la imagen en forma cuadricula de puntos, llamados pixeles, que definen los elementos de la figura. A cada pixel se le asigna un valor (negro, blanco, matrices de grises o color), el cual está representado en u código binario (ceros y unos). Este valor total para cada pixel está definido en bits, los cuales son almacenados por una computadora, en una secuencia y con

frecuencia se los recude en una representación matemática (comprimida). Luego la computadora interpreta y lee los bits para producir una versión analógica para su visualización o impresión.

3.2 SEGMENTACIÓN DE IMÁGENES

La segmentación de imágenes divide la imagen en sus partes constituyentes hasta un nivel de subdivisión en el que se aíslen las regiones u objetos de interés. Los algoritmos de segmentación se basan en una de estas dos propiedades básicas de los valores del nivel de gris: discontinuidad o similitud entre los niveles de gris de píxeles vecinos. (S/N, s.f.)

1. **Discontinuidad:** Se divide la imagen basándose en cambios bruscos de nivel de Segmentación de imágenes Similitud. Se divide la imagen basándose en la búsqueda de zonas que tengan valores similares, conforme a unos criterios prefijados:
    - Crecimiento de región: Procedimiento que agrupa los pixeles o subregiones de la imagen en regiones mayores basándose en un criterio prefijado. Normalmente se empieza con puntos “semillas” para formar una determinada región, añadiendo aquellos pixeles vecinos que cumplan la propiedad especificada.
      (Ej. que estén en un rango de nivel de grises determinados)

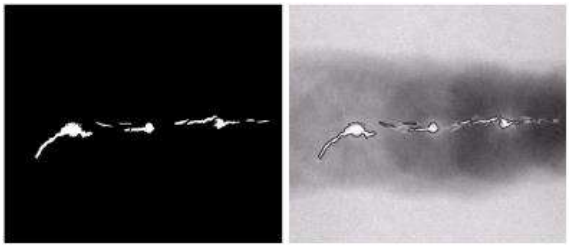


Ilustración 3. Crecimiento de Región (Esta en un rango de nivel de grises).

- Umbralización: Es uno de los más importantes métodos de segmentación. El objetivo es convertir una imagen en escala de grises a una nueva con sólo dos niveles, de manera que los objetos queden separados del fondo. Algunos métodos de umbralización son:
    - Método del valor medio.
    - Método del porcentaje de pixeles negros.
    - Método de los dos picos.
    - Umbralización adaptiva.

2. **Similitud entre los niveles de grises:** Se divide la imagen basándose en cambios bruscos de nivel de gris:
    - Detección de puntos aislados: Un punto aislado de una imagen tiene un tono gris que defiere significativamente de los tonos de gris de sus pixeles vecinos, es decir, de los ocho pixeles de su entorno. Una máscara (Laplaciano) para detectar un punto aislado es la siguiente.

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Ilustración 4. Mascara Laplaciano, para detectar un punto aislado.

Diremos que un pixel es un punto aislado y el resultado de aplicar la máscara sobre el pixel es mayor o igual que un cierto valor umbral.

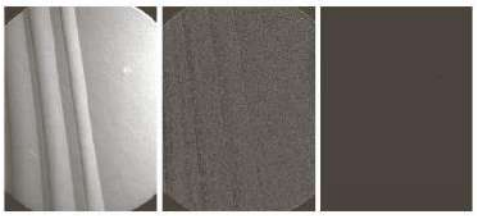


Ilustración 5. Aplicación de una máscara sobre un pixel, o cierto valor umbral.

- Detección de líneas: Aquí se utiliza la máscara de Laplaciano.
- Detección de bordes:

- Borde Ideal: Forma un camino de un pixel de ancho en la que se produce, perpendicularmente un camino en el nivel de grises.

Ilustración 6. Detención de bordes, hace un camino en el nivel de grises.

- Borde Rampa: Forma un conjunto de pixeles conexos en los que se produce en una determinada dirección, una variación gradual en el nivel de grises.

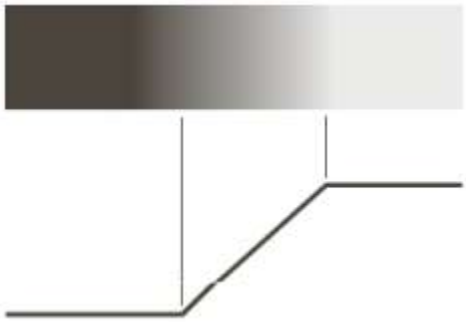


Ilustración 7. Borde Rampa, hace un conjunto de pixeles conexos.

3.3 ENTORNOS DE DESARROLLO

3.3.1 ¿QUÉ ES NETBEANS?

NetBeans es un proyecto exitoso de código abierto con una gran base de usuarios, una comunidad en constante crecimiento, y con cerca de 100 socios (¡y creciendo!)

en todo el mundo. Sun MicroSystems fundó el proyecto de código abierto NetBeans en junio 2000 y continúa siendo el patrocinador principal de los proyectos.
Al día de hoy hay disponibles dos productos: el NetBeans IDE y NetBeans Platform. (NetBeans, 2017)

3.3.2 EN TRABAJO FUTURO

3.3.2.1 ¿QUÉ ES ANDROID?

Android es un sistema operativo basado en Linux, inicialmente pensado para terminales móviles, al igual que BlackBerry OS o iOS (Apple).

Desarrollado por Android Inc. desde 2003. Hasta que fue comprado por Google en 2005. Se liberó el código bajo licencia Apache al crearse la Open Handset Alliance el 5 de noviembre de 2007. También considerado en cumpleaños de Android.

En 2008 se crean los primero chips compatibles y se lanza el primer teléfono Android, el HTC Dream.

Se empiezan a nombrar como dulces a las versiones de Android a partir de 2009.

Un sistema operativo Android tiene múltiples idiomas y tres tipos de denominar a las versiones de Android, aunque las tres hacen referencia a la misma versión:

1- La comercial con el nombre de postre. Por ejemplo: KitKat
2- La de los fabricantes (y también comercial) con la versión y subversión. Por ejemplo: 4.4
3- La de desarrollador con el nivel del API (ésta nos interesa mucho para desarrollar en Android): Por ejemplo: 19 Nombre Versión

Esta es la cuota a nivel mundial de los sistemas operativos para móviles. (Menéndez, s.f.)

	1º Trimestre 2012	2º Trimestre 2012	3º Trimestre 2012	4º Trimestre 2012	1º Trimestre 2013	2º Trimestre 2013	3º Trimestre 2013	4º Trimestre 2013	1º Trimestre 2014
Android	59,10%	69,10%	74,90%	70,30%	75,00%	79,30%	81,00%	78,10%	81,10%
iOS	23,10%	16,60%	14,40%	20,90%	17,30%	13,20%	12,90%	17,60%	15,20%
Windows	2,00%	3,10%	2,00%	2,60%	3,20%	3,70%	3,60%	3,00%	2,70%

Ilustración 8. Cuota nivel mundial de los S.O. en móviles.

Cuota en el mercado a nivel mundial de Android

Version	Codename	API	Distribution
2.2	Froyo	8	0.7%
2.3.3 - 2.3.7	Gingerbread	10	11.4%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	9.6%
4.1.x	Jelly Bean	16	25.1%
4.2.x		17	20.7%
4.3		18	8.0%
4.4	KitKat	19	24.5%

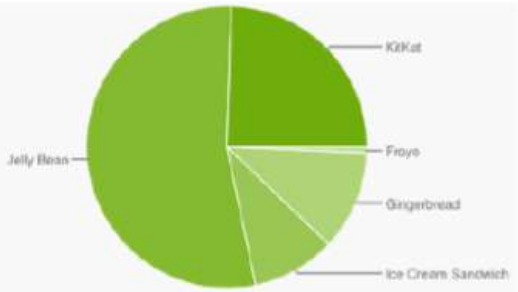


Ilustración 1 - Imagen obtenida de la web de http://developer.android.com/. Datos recogidos durante un periodo de 7 días al finalizar el 9 de Septiembre de 2014 (Todas las versiones con una distribución de menos de un 0,1% no se mostrarán)

Ilustración 9. Cuota mundial de Android.

3.3.3 ANDROID STUDIO

Android Studio es un entorno de desarrollo integrado (IDE), basado en IntelliJ IDEA de la compañía JetBrains, que proporciona varias mejoras con respecto al plugin ADT (Android Developer Tools) para Eclipse. Android Studio utiliza una licencia de software libre Apache 2.0, está programado en Java y es multiplataforma.

Fue presentado por Google el 16 de mayo del 2013 en el congreso de desarrolladores Google I/O, con el objetivo de crear un entorno dedicado en exclusiva a la programación de aplicaciones para dispositivos Android, proporcionando a Google un mayor control sobre el proceso de producción. Se trata pues de una alternativa real a Eclipse, el IDE recomendado por Google hasta la fecha, pero que presentaba problemas debido a su lentitud en el desarrollo de versiones que solucionaran las carencias actuales (es indispensable recordar que Eclipse es una plataforma de desarrollo, diseñada para ser extendida a través de plugins).

Android Studio se ha mantenido durante todo este tiempo en versión beta, pero desde el 8 de diciembre de 2014, en que se liberó la versión estable de Android Studio 1.0, Google ha pasado a recomendarlo como el IDE para desarrollar aplicaciones para su sistema operativo, dejando el plugin ADT para Eclipse de estar en desarrollo activo. Esta versión la puedes descargar desde la web de Android Developer. (Android, 2014)

4 MARCO METODOLÓGICO

4.1 MODELO ESPIRAL

Modelo Espiral, propuesto originalmente por Boehm, es un modelo de proceso de software evolutivo que conjuga la naturaleza iterativa de construcción de prototipos con los aspectos controlados y sistemáticos del modelo lineal secuencial. Proporciona el potencial para el desarrollo rápido de versiones incrementales del software.

En el modelo espiral, el software se desarrolla en una serie de versiones incrementales. Durante las primeras interacciones, la versión incremental podría ser un modelo en papel o un prototipo. Durante las últimas iteraciones, se producen versiones cada vez más completas del sistema diseñado. (EcuRed, 2017)

4.1.1 HISTORIA DEL MODELO ESPIRAL

El creador del modelo en espiral fue Barry Boehm quien recibió su grado de B.A. de Harvard en 1957, y sus grados de M.S. y de Ph.D. de UCLA en 1961 y 1964, todo en matemáticas.

Barry Boehm era un Programador-Analista en General Dynamics entre 1955 y 1959, sus intereses actuales de la investigación incluyen modelar de proceso del software, ingeniería de requisitos del software, las arquitecturas del software, métrica del software y los modelos del coste, los ambientes de la tecnología de dotación lógica, y tecnología de dotación lógica basada en el conocimiento. Sus contribuciones al campo incluyen el modelo constructivo del coste (COCOMO), el modelo espiral del proceso del software, el acercamiento de la teoría W (ganar-gane) a la determinación de la gerencia y de los requisitos del software y a dos ambientes

avanzados de la tecnología de dotación lógica: el sistema y el quántum de la productividad del software de TRW saltan el ambiente. (Modelo Espial, 2009)

4.2 TIPOS DEL MODELO ESPIRAL

- Modelo de cuatro regiones

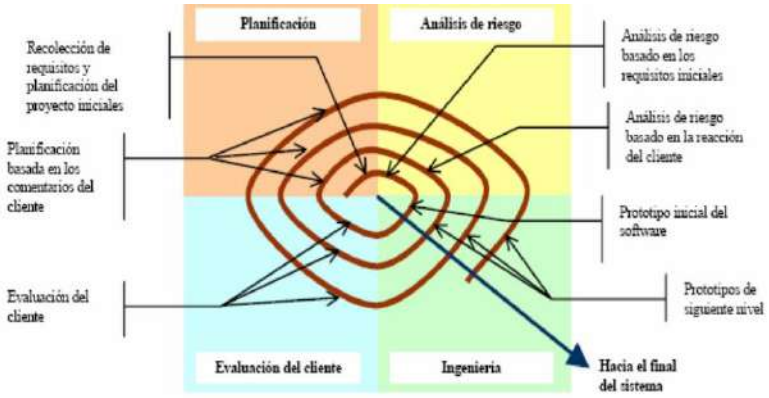


Ilustración 10. Modelo de cuatro regiones.

- Modelo de seis regiones

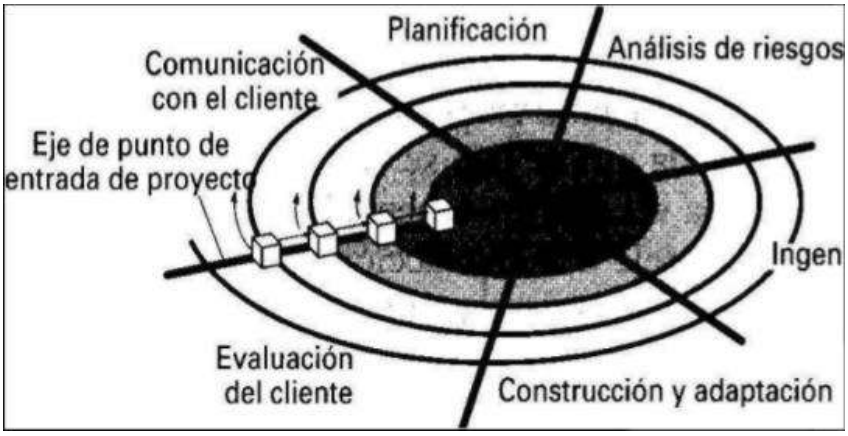


Ilustración 11. Modelo de seis regiones.

- Modelo WinWin: (Victoria Victoria)

Ilustración 12. Modelo WinWin.

4.3 HERRAMIENTAS

Fase	Herramientas
Comunicación con el cliente	En es la fase donde el usuario interactúa con la aplicación, desde la carga de la imagen, su transformación a escala de gris, su firma, y su verificación de la firma.
planificación	Planificar los recursos necesarios y establecer las fechas en la que se realizaran cada tarea durante el desarrollo de la aplicación.
Análisis de riesgos	Analizar a fondo y a largo plazo algunos problemas que podría ocurrir durante la implementación del sistema.
ingeniería	Se desarrolla en Photoshop cs6 una representación de la aplicación para visualizar como quedaría dicho modelo.
Construcción y acción	Es donde se lleva la documentación del sistema, así como su implementación y posteriormente llevarlo a la práctica.
Evaluación del cliente	Es en esta fase es donde el cliente evaluara la integridad o funcionamiento de la aplicación respecto a su opinión.

5 DESARROLLO

5.1 PROTOTIPO ALFA

5.1.1 ESPECIFICACIONES DE REQUERIMIENTOS

5.1.1.1 DEFINICIÓN DE LOS REQUERIMIENTOS DE SISTEMA

En este apartado se presentan los requisitos funcionales y no funcionales con los que cumple el desarrollo del sistema, los requisitos que se presentan a continuación fueron los que se necesitaron para el buen funcionamiento de nuestro sistema.

5.1.1.2 REQUERIMIENTOS FUNCIONALES

ID Requisito	Requisito	Descripción del requisito	Medio	Prioridad
RF-01	Selección de Imagen	El usuario debe elegir una imagen de cualquier dimensión, esta imagen debe ser a colores.	Interfaz en NetBeans	Alta
RF-02	Transformación a grises	Esta parte es importante, ya que la imagen seleccionada sufrirá una transformación a escala de grises, para este proceso la imagen será clonada, así el clon es la que resultara afectada, ahora en el imagen clone pasara por un proceso de media color, esto quiere decir que con la lectura RGB de la imagen se obtendrá el escalado a grises.	Interfaz en NetBeans	Alta
RF-03	Extracción de valores entre 0-255	Para esto la imagen a escala de grises pasara por una lectura de sus pixeles, cada pixel leído arrogara su número en bit de acuerdo al color que esta lea en la imagen a escala de grises. Los resultados se guardaran en una (.txt).	Archivos .txt	Alto
RF-04	Firma de la imagen	Se toma los resultados guardados en el archivo (.txt), en base a eso se genera la firma, y genera dos claves una privada que solo conocerá el emisor y una clave pública que conocerá tanto el emisor como el receptor, también geera la firma en archivo (.txt)	Interfaz en NetBeans	Alto

RF-05	Verificación de la firma	Tomando los archivos de la clave pública y la firma, junto con la ruta del archivo de RGB se procede a verificar la firma mandando un mensaje diciendo si es verdadera o falsa la firma.	Interfaz en NetBeans	Alto

Los requerimientos funcionales expresan el funcionamiento de dicho programa, su interacción con el usuario y los estados de funcionamiento que se ejecutan.

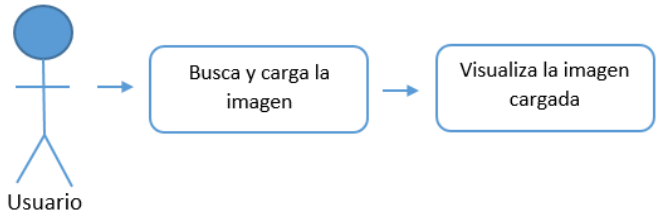
El primer requerimiento que tiene el programa es el de abrir una imagen a color de cualquier tamaño, esta imagen podrá visualizarse en una interfaz hecha en NetBeans, el segundo requerimiento es la transformación a escala de grises de la imagen cargada, en este requerimiento la imagen original se clonara ya que el clon es el que será únicamente afectada por esta transformación, al clonar pasara por un proceso que se llama media color, que se refiere a buscar el color a procesar por medio de los RGB, este color gris se aplicara al clon, y podrá visualizarse en la interfaz de NetBeans, teniendo de un lado la imagen original y al lado contrario la imagen transformada a escala de grises, en el tercer requerimiento es la lectura por cada pixel de la imagen a escala de grises, para este proceso se aplicara una lectura por cada pixel de la imagen transformada a grises, para el recorrido se tomara de referencia X,Y, como sus medidas de tamaño, al realizar esta lectura la información pasara por otro proceso que es ahora el de pedir el color correspondiente de cada pixel de la imagen, posteriormente se pasara a bit el resultado y se guardara en un archivo (.txt), con el nombre de “Resultados-RGB.txt”.

En el requerimiento de firma de la imagen hace uso del **Algoritmo de Firma Digital (DSA)**, con esto se toma la información del archivo “Resultados-RGB.txt”, generando dos archivos más con el nombre de “clv-publica-img.txt” y “firma-img.txt”, también genera una firma privada la cual solo conocerá el emisor, finalmente para el requerimiento de verificación de la firma nuevamente hace uso del **Algoritmo de Firma Digital (DSA)**, así se toman los archivos de clave pública y la firma, también hace uso de la ruta del archivo de RGB, teniendo esto se puede verificar la firma de

la imagen, mandando un mensaje diciendo si es verdadera o falta en caso de su modificación del archivo RGB.

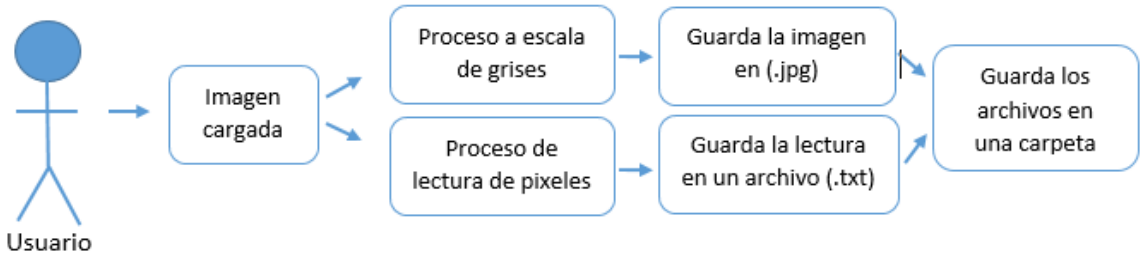
5.1.1.3 CASOS DE USO DE LOS REQUERIMIENTOS FUNCIONALES

5.1.1.3.1 CARGA DE LA IMAGEN



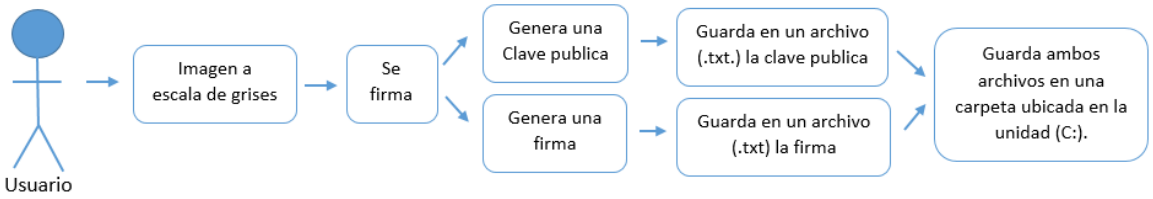
El primer requisito funcional del sistema es la carga de la imagen, ya que es fundamental para sus procesos siguientes, ya que al cargar la imagen nos mostrará datos de la misma, y con esto se podrá proceder a realizar sus demás funciones.

5.1.1.3.2 PROCESAMIENTO A ESCALA DE GRISES Y EXTRACCIÓN DE VALORES ENTRE 0-255.



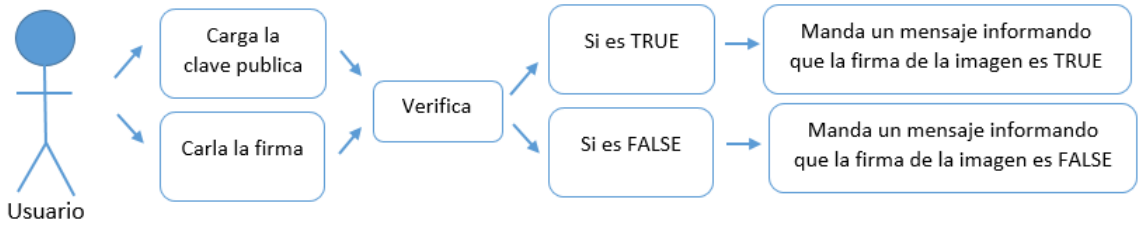
En el diagrama anterior muestra que el segundo requerimiento es la carga de la imagen, será necesario tener una imagen cargada para poder procesarla a escala de grises y al mismo tiempo realizar la lectura de pixeles de la imagen transformada, y finalizar tanto la imagen transformada de guarda (.jpg), y la lectura de pixeles se guardan en un archivo (.txt), ambos archivos se guardaran en una carpeta ubicada en la unidad (C:).

5.1.1.3.3 FIRMA DE LA IMAGEN



En este requerimiento es necesario tener la imagen transformada a escala de grises, posteriormente se firma y al hacerlo se generan dos archivos (.txt), en uno de ellos se guarda la clave pública generada, mientras que en el segundo archivo se guarda la firma, y ambos archivos se guardan en una carpeta con destino en la unidad (C:).

5.1.1.3.4 VERIFICACIÓN DE LA FIRMA DE LA IMAGEN



Por ultimo en este requerimiento, ya debe de haberse obtenido la clave pública como la firma, lo que ahora se hace es cargar ambos archivos en sus respectivos casos, se procede a verifica, al finalizar el proceso mandara un mensaje informando que la firma de la imagen es TRUE, en caso de ser verdadera, en caso contrario mandara un mensaje con FALSE.

5.1.1.4 REQUERIMIENTOS NO FUNCIONALES

ID Requisito	Requisito	Descripción del requisito	Medio	Prioridad
RNF-01	Tiempo en procesado de grises, verificación y validación.	En cada proceso que se lleva a cabo de la aplicación se va generando el tiempo que tarda dicho proceso desde su inicio hasta su fin.	Interfaz en NetBeans	Media

Las funciones no funcionales son todos aquellos requisitos que no forman parte del sistema.

El tiempo en el que tarda el proceso de cada función de la aplicación es un plus que se añadió que ayuda a determinar el máximo y mínimo tiempo de una imagen para su transformación a grises, firma y validación.

5.1.2 DISEÑO E INGENIERÍA

Los diseños se realizaron en Photoshop CS6

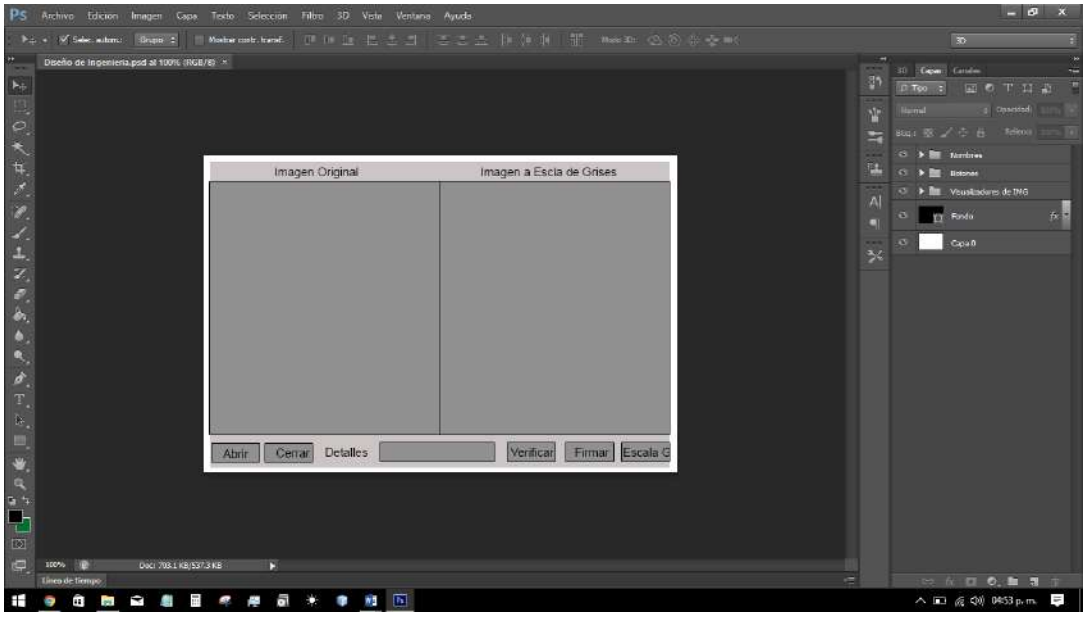


Imagen 1. Diseño de interfaz en Photoshop CS6.

Cuando abra la aplicación el usuario aparecerá una interfaz así

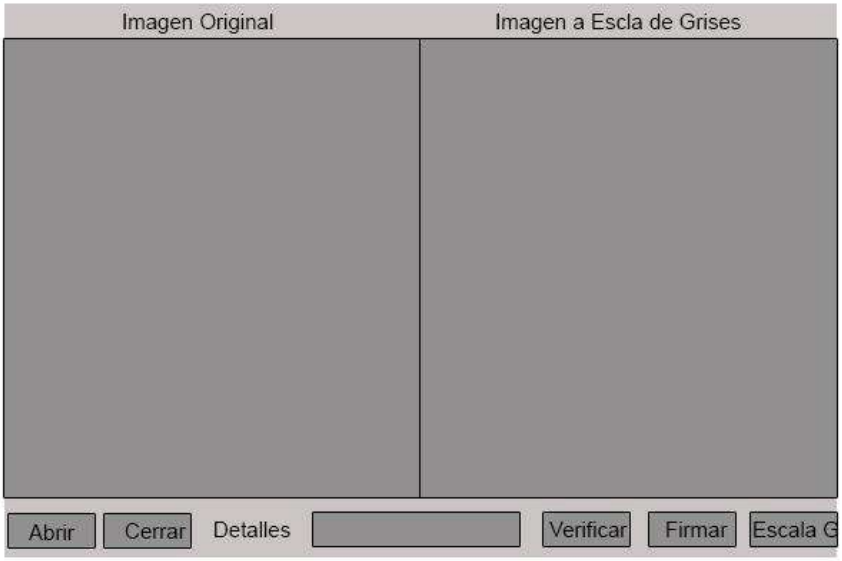


Imagen 2. Visualización de Diseño de Interfaz.

En la cual se podrá cargar una imagen, pasarla a escala de grises, firmar la imagen a escala de grises y verificar dicha firma de la imagen.

Imagen 3. Ventana para cargar una imagen.

Al presionar el botón de abrir de la aplicación, aparecerá una ventana en la cual podrá durarse al destino donde se encuentre la imagen, para así poderla cargar.

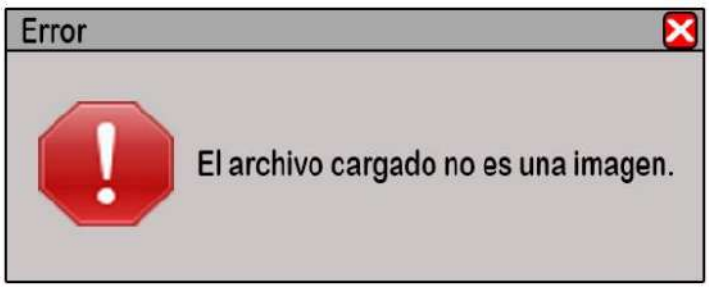


Imagen 4. Mensaje de error al cargar una imagen.

Si el archivo cargado no es imagen, aparecerá un mensaje de error.

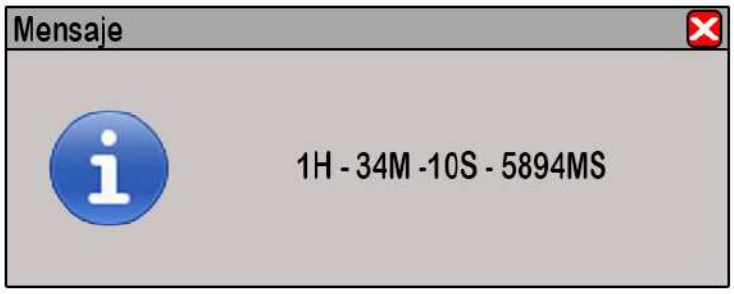


Imagen 5. Ventana de tiempo del proceso.

En cada proceso (Lectura de pixeles, firma y verificación) se cuenta el tiempo que se tarda de inicio hasta su fin, mostrado en una ventana donde muestra horas, minutos, segundos, milisegundos.

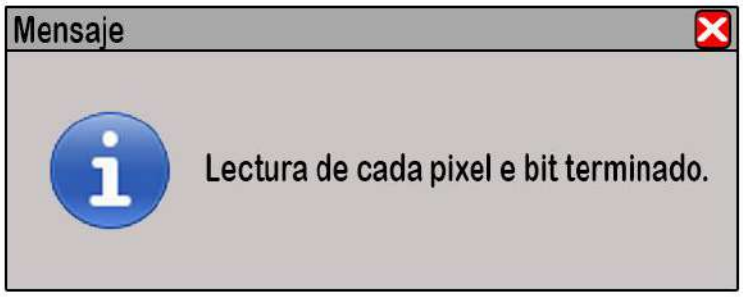


Imagen 6.Ventana de lectura de pixeles terminado.

Al terminado el proceso de lectura de pixeles mostrara un mensaje.

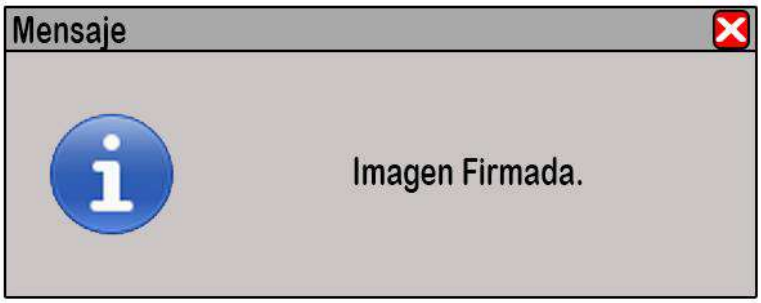


Imagen 7. Aviso de la imagen está firmada.

Al terminar el proceso de la firma de la imagen mostrara un mensaje.

Imagen 8. Mensaje que la firma de la imagen es correcta.

Imagen 9.. Mensaje que la firma de la imagen es incorrecta.

Al verificar a firma de la imagen mostrara un mensaje de TRUE en caso de que la firma sea correcta y en caso contrario mostrara un mensaje FALSE.

Tipo	Especificación
Botón Abrir	Carga una imagen donde quiera que se encuentre de cualquier tamaño.
Botón Cerrar	Cierra la ventana principal de la aplicación.
Recuadro de Detalles	Muestra al cargar la imagen el tamaño en pixeles de la misma.
Botón Verificar	Verifica la firma de la imagen a escala de grises.
Botón Firmar	Genera una firma en base a la información obtenida de la imagen a escala de grises, al haber leído cada pixel en bit de dicha imagen transformada.
Botón Escala Grises	Transforma la imagen cargada a una imagen en tonos grises.
Recuadro de Imagen Original	Visualiza la imagen cargada.
Recuadro de Imagen a Escala de Grises	Visualiza la imagen transformada a escala de grises.

5.1.3 CONSTRUCCIÓN

5.1.3.1 CARGA DE IMAGEN

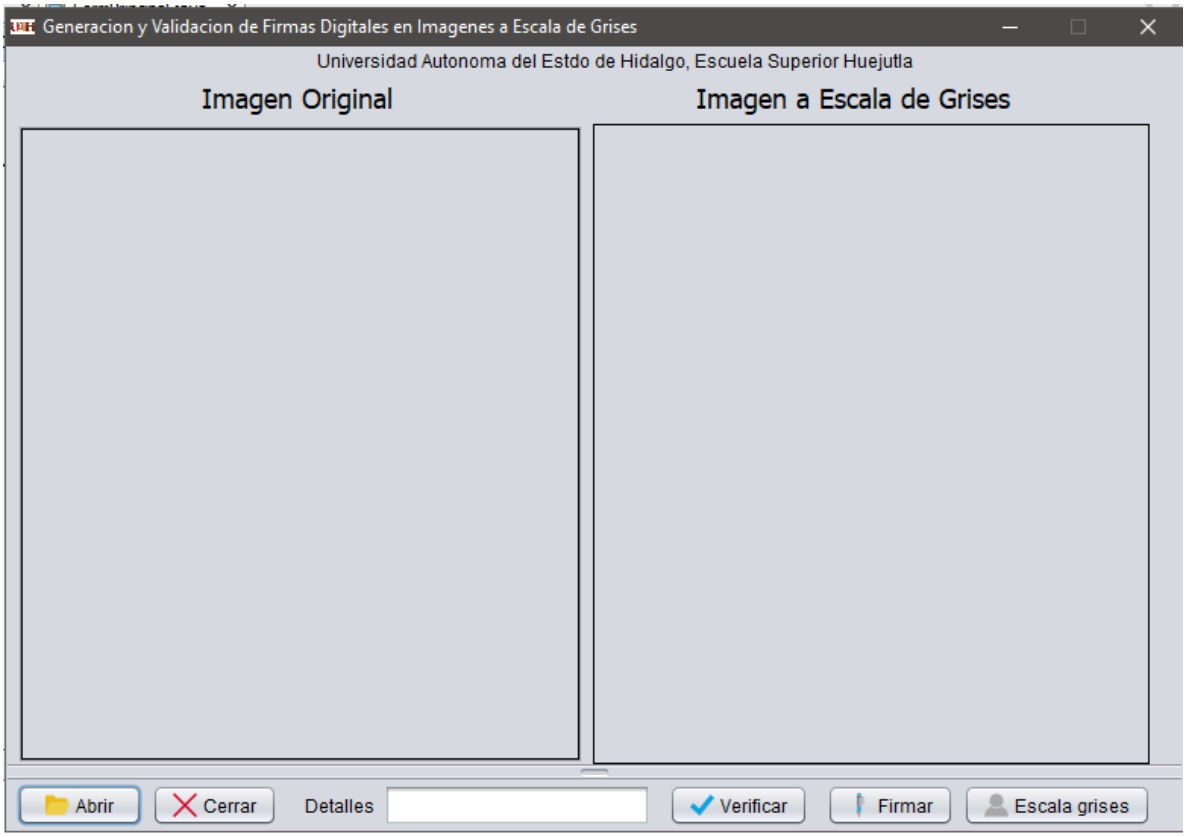


Imagen 10. Interfaz de la aplicación.

Al presionar el botón de abrir aparece una ventana, donde se elige el origen de la imagen que se cargara.

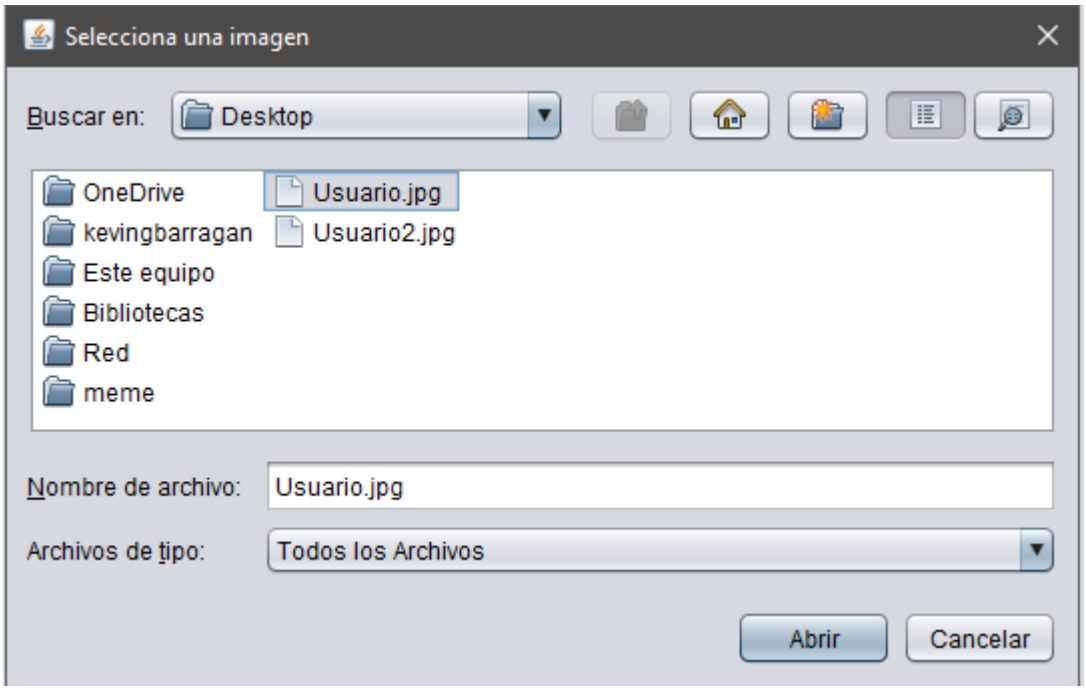


Imagen 11. Busca imagen para cargar.

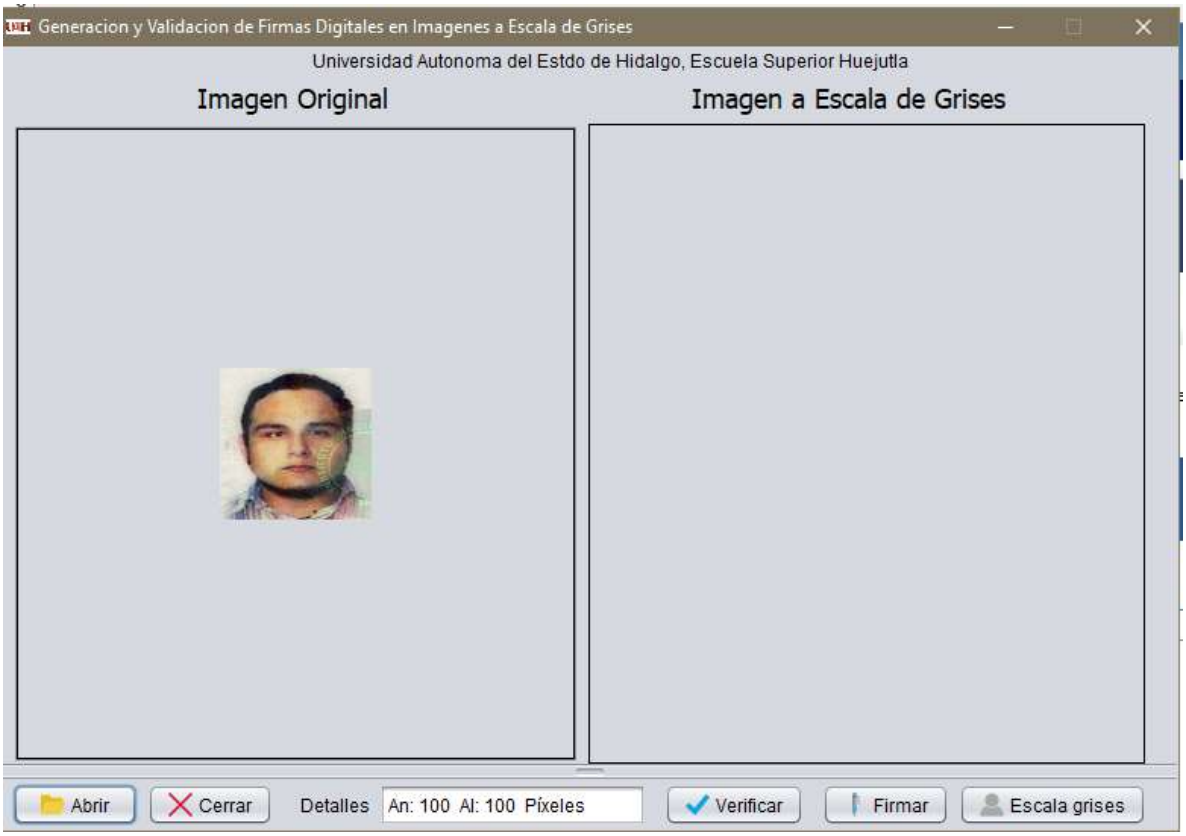


Imagen 12. Imagen cargada en la Interfaz.

Imagen 13. Mensaje de Error que no es una Imagen la que se desea cargar.

Si al cargar un archivo este no es una imagen manda un mensaje de error diciendo que el archivo cargado no es una imagen.

CÓDIGO

CLASE:

```
package ClasesImagenes;
/**
 *
 * @author kevingbarragan
 */
import java.awt.image.BufferedImage;
import java.net.URL;
import javax.imageio.ImageIO;
import javax.swing.JFileChooser;
import javax.swing.JOptionPane;

public class AbrirImagen extends BaseImagenes {
    private JFileChooser selectorImage;
    /**
```

```
     * Muestra un mensaje de error avisando que no se ha podido cargar la imagen
     */
    private void errorImagen(){
        JOptionPane.showMessageDialog(null,"El archivo cargado no es una
imagen","Error",JOptionPane.ERROR_MESSAGE);
    }

    /**
     * Carga el JFIleChooser para seleccionar imagen
     * @return devuelve false si la acción fue cancelada por el usuario y true si
     * ha seleccionado algún archivo
     */
    private boolean abrirJFileChooser(){
      this.selectorImage=new JFileChooser();
      this.selectorImage.setDialogTitle("Selecciona una imagen");
      int flag=this.selectorImage.showOpenDialog(null);
      if (flag==JFileChooser.APPROVE_OPTION){
          return true;
      }else{
         return false;
      }
   }

    /**
     * Muestra un cuadro de diálogo para abrir una imagen desde archivo
     * @return devuelve la imagen seleccionada o null si no se pudo cargar
     */
   public BufferedImage abrirImagenLocal(){
      BufferedImage imagenRetorno=null;
      if(this.abrirJFileChooser()==true){
          try {
              imagenRetorno = ImageIO.read(this.selectorImage.getSelectedFile());
              if (imagenRetorno!=null){
                  int alto = imagenRetorno.getHeight();//altura de la imagen Y
                  int ancho = imagenRetorno.getWidth();//ancho de la imagen  X
                  super.actualizarImagen(imagenRetorno, "Ancho: "+ancho+" "+" Alto:
"+alto+" "+" Píxeles");
              }else{
                  errorImagen();
              }
          } catch (Exception e) {
              errorImagen();
          }
      }
      return imagenRetorno;
   }
}
```

BOTÓN:

```
private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    BufferedImage imagen=ObjAbrir.abrirImagenLocal();
    if (imagen!=null){
        jLabel_Imagen.setIcon(ObjCambiarFormat.bufferedImageToIcon(imagen));
    }
    {
        detalles.setText(ObjBase.informacionImagenActual());
    }
}
```

5.1.3.2 TRANSFORMACIÓN A ESCALA DE GRISES, EXTRACCIÓN DE VALORES ENTRE 0-255 Y ALMACENAMIENTO DE LA IMAGEN A GRIS

Se presiona el botón correspondiente, en el proceso de transformación de grises en este apartado tuve que investigar sobre la extracción de los valores entre 0-255 de la imagen a escala de grises, así como también las diferentes transformaciones que pueda tener una imagen de acuerdo a su umbral y se investigó como recorrer una imagen con sus posiciones X, Y (ancho y alto).

La información de la extracción de los valores de la imagen a escala de grises se guarda automáticamente en un ".txt" con el nombre de "Resultados-RGB" y la imagen a grises también se guardará con el nombre "Img-Grises", estos archivos generados se alojarán en una carpeta que se tendrá que crear en la unidad "C:/" con el nombre de "FirmasDigitalesEnImagenes".

Imagen 14. Imagen transformada a escala de Grises.

Imagen 15. Resultados de la lectura de pixeles en RGB de la imagen a escala de Grises.

CÓDIGO

CLASE:

```
package ClasesImagenes;
/**
 *
 * @author kevingbarragan
 */
import java.awt.Color;
```

```
import java.awt.image.BufferedImage;
import java.io.FileNotFoundException;
import java.io.PrintStream;

public class TransformarImagen extends BaseImagenes{
   private int colorRGBaSRGB(Color colorRGB){
      int colorSRGB;
      colorSRGB=(colorRGB.getRed() << 16) | (colorRGB.getGreen() << 8) |
colorRGB.getBlue();
      return colorSRGB;
   }
   private BufferedImage clonarBufferedImage(BufferedImage bufferImage){
      BufferedImage copiaImagen=new BufferedImage
(bufferImage.getWidth(),bufferImage.getHeight(),bufferImage.getType());
      copiaImagen.setData(bufferImage.getData());
      return copiaImagen;
   }
   private int calcularMediaColor(Color color){
      int averageColor;
      averageColor=(int)((color.getRed()+color.getGreen()+color.getBlue())/3);
      return averageColor;
   }
   public BufferedImage escalaGrises(BufferedImage imagen) throws
FileNotFoundException{
      BufferedImage imagenRetorno=this.clonarBufferedImage(imagen);
      Color auxColor;
      int mediaColor;
      String  imprimirRGB="";
      for( int x = 0; x < imagenRetorno.getWidth(); x++ ){
         for( int y = 0; y < imagenRetorno.getHeight(); y++ ){
            auxColor=new Color(imagenRetorno.getRGB(x, y));
            mediaColor=this.calcularMediaColor(auxColor);
            imagenRetorno.setRGB(x, y,this.colorRGBaSRGB(new
Color(mediaColor,mediaColor,mediaColor,auxColor.getAlpha())));
             //Imprime los valores de RGB a BITS de la imagen
             int srcPixel = imagen.getRGB(y, x);//pide color depende de la posision
X, Y
             Color c = new Color(srcPixel);//apoyo de color para sacar el RGB
             int valR = +c.getRed();//componente de color ROJO
             int valG = +c.getGreen();//componente de color VERDE
             int valB = +c.getBlue();//componente de color AZUL
             imprimirRGB = imprimirRGB +" "+valR+" "+valG+" "+valB;
            System.out.println("R: " + valR + " G: " + valG + " B: " + valB);
            PrintStream DDescritor = new
PrintStream("C:\\FirmasDigitalesEnImagenes\\Resultados-RGB.txt");
            DDescritor.print(imprimirRGB);
         }
```

```
        }
        super.actualizarImagen(imagenRetorno,"Escala de Grises");
        return imagenRetorno;
    }
}
```

BOTÓN:

```
private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        BufferedImage
imagen=ObjCambiarFormat.iconToBufferedImage(jLabel_Imagen.getIcon());
        // Tiempo inicial del sistema en milisegundos
                long startTime = System.currentTimeMillis();
        try {
            imagen=ObjTransformaImg.escalaGrises(imagen);
        } catch (FileNotFoundException ex) {
            Logger.getLogger(FormPrincipal.class.getName()).log(Level.SEVERE, null,
ex);
        }
        jLabel_Imagen1.setIcon(ObjCambiarFormat.bufferedImageToIcon(imagen));
        detalles.setText(ObjBase.informacionImagenActual());
        try {
            ImageIO.write(imagen, "jpg", new
File("C:\\FirmasDigitalesEnImagenes\\Img-Grises.jpg"));
        } catch (IOException ex) {
            Logger.getLogger(FormPrincipal.class.getName()).log(Level.SEVERE, null,
ex);
        }
// Tiempo final del sistema en milisegundos
                long endTime = System.currentTimeMillis();
        // Se nuestra el resultado
                System.out.println("Tiempo MIlisegundos : " + (endTime - startTime));

            int num = (int) (endTime - startTime), seg, min,hor;
            seg=num/1000;
            min= (int)(seg /60);
            hor= (int)(seg/3600);
JOptionPane.showMessageDialog(this, hor+"h "+"- "+min+"m "+"- "+seg+"s "+"-
"+(endTime-startTime)+"ms ");
        JOptionPane.showMessageDialog(null, "Lectura de cada pixel en bit
terminado", "Aviso", JOptionPane.INFORMATION_MESSAGE);

    }
```

5.2 PROTOTIPO ALFA FINALIZADO

5.2.1 CONSTRUCCIÓN

5.2.1.1 FIRMA

Para la firma de la imagen hace uso del **Algoritmo de Firma Digital (DSA)**, con esto se toma la información del archivo “Resultados-RGB.txt”, generando dos archivos más con el nombre de “clv-publica-img.txt” y “firma-img.txt”, también genera una firma privada la cual solo conocerá el emisor.

Imagen 16. Imagen firmada, y su generación de clave pública con su firma.

CÓDIGO

CLASE:

```
package ClasesImagenes;

/**
 *
 * @author kevingbarragan
 */
import java.io.*;
import java.security.*;
import javax.swing.JOptionPane;

class firmar {
```

```
    public static void main(String[] args) {

    }

    public void firmas(String archivo) {
        try {

            /* Genera par de claves publica y privada */
            KeyPairGenerator keyGen = KeyPairGenerator.getInstance("DSA", "SUN");
            SecureRandom random = SecureRandom.getInstance("SHA1PRNG",
"SUN");

            keyGen.initialize(1024, random);

            KeyPair pair = keyGen.generateKeyPair();
            PrivateKey priv = pair.getPrivate();//almacena clave privada
            PublicKey pub = pair.getPublic();//almacena clave publica


            /* Crear un objeto Signature e inicializarlo con la clave privada */
            Signature dsa = Signature.getInstance("SHA1withDSA", "SUN");

            dsa.initSign(priv);

            /*Actualiza la firma y los datos */
            FileInputStream fis = new FileInputStream(archivo);

            BufferedInputStream bufin = new BufferedInputStream(fis);
            byte[] buffer = new byte[1024];
            int len;
            while (bufin.available() != 0) {
                len = bufin.read(buffer);
                dsa.update(buffer, 0, len);
            };

            bufin.close();

            /* Ahora que todos los datos a firmar han sido leídos,
                  Generar una firma para ello */
            byte[] realSig = dsa.sign();

            /* Guarda la firma en el archivo */
            FileOutputStream sigfos = new
FileOutputStream("C:\\FirmasDigitalesEnImagenes\\firma-img.txt");
            sigfos.write(realSig);
```

```
            sigfos.close();


            /* Guarda la clave publica en el archivo */
            byte[] key = pub.getEncoded();
            FileOutputStream keyfos = new
FileOutputStream("C:\\FirmasDigitalesEnImagenes\\clv-publica-img.txt");
            keyfos.write(key);

            keyfos.close();
            JOptionPane.showMessageDialog(null, "Imagen firmada", "Aviso",
JOptionPane.INFORMATION_MESSAGE);
        } catch (Exception e) {
            System.err.println("Excepcion Atrapada " + e.toString());
             JOptionPane.showMessageDialog(null, "No se pudo firmar", "Aviso",
JOptionPane.INFORMATION_MESSAGE);
        }

    }

}
```

BOTÓN:

```
     private void jMenuItem3ActionPerformed(java.awt.event.ActionEvent evt) {
       // TODO add your handling code here:
       // TODO add your handling code here:
       firmar firma = new firmar();
       // Tiempo inicial del sistema en milisegundos
                long startTime = System.currentTimeMillis();
       String ruta = "C:\\FirmasDigitalesEnImagenes\\Resultados-RGB.txt";
       if (ruta.length() == 0) {
       } else {
           firma.firmas(ruta);
       }
       // Tiempo final del sistema en milisegundos
                long endTime = System.currentTimeMillis();
       // Se nuestra el resultado
                System.out.println("Tiempo MIlisegundos : " + (endTime - startTime));

              int num = (int) (endTime - startTime), seg, min,hor;
              seg=num/1000;
              min= (int)(seg /60);
              hor= (int)(seg/3600);
JOptionPane.showMessageDialog(this, hor+"h "+"- "+min+"m "+"- "+seg+"s "+"-
"+(endTime-startTime)+"ms ");
    }
```

5.2.1.2 VERIFICACIÓN

Finalmente, para la verificación de la firma nuevamente hace uso del **Algoritmo de Firma Digital (DSA)**, así se toman los archivos de clave pública y la firma, también hace uso de la ruta del archivo de RGB, teniendo esto se puede verificar la firma de la imagen, mandando un mensaje diciendo si es verdadera o falta en caso de su modificación del archivo RGB.

Verificación FALSE (en caso de que se modifique el archivo RGB o la imagen)

Imagen 17. Verificación de la firma es falsa.

Verificación TRUE

Imagen 18. Verificación de la firma es correcta.

CÓDIGO

CLASE:

```
package ClasesImagenes;

/**
 *
 * @author kevingbarragan
 */
import java.io.*;
import java.security.*;
import java.security.spec.*;
import javax.swing.JOptionPane;

class verificar {

    public static void main(String[] args) {

    }

    public void verificar(String clavep, String firma, String archivo) {
        try {

            /* import encoded public key */
            FileInputStream keyfis = new FileInputStream(clavep);
            byte[] encKey = new byte[keyfis.available()];
            keyfis.read(encKey);

            keyfis.close();

            X509EncodedKeySpec pubKeySpec = new
X509EncodedKeySpec(encKey);

            KeyFactory keyFactory = KeyFactory.getInstance("DSA", "SUN");
            PublicKey pubKey = keyFactory.generatePublic(pubKeySpec);

            /* input the signature bytes */
            FileInputStream sigfis = new FileInputStream(firma);
            byte[] sigToVerify = new byte[sigfis.available()];
            sigfis.read(sigToVerify);

            sigfis.close();

            /* create a Signature object and initialize it with the public key */
            Signature sig = Signature.getInstance("SHA1withDSA", "SUN");
            sig.initVerify(pubKey);
```

```
            /* Update and verify the data */
            FileInputStream datafis = new FileInputStream(archivo);
            BufferedInputStream bufin = new BufferedInputStream(datafis);

            byte[] buffer = new byte[1024];
            int len;
            while (bufin.available() != 0) {
                len = bufin.read(buffer);
                sig.update(buffer, 0, len);
            };

            bufin.close();

            boolean verifies = sig.verify(sigToVerify);

            System.out.println("signature verifies: " + verifies);

            JOptionPane.showMessageDialog(null, "La firma de la imagen es:
"+verifies, "Aviso", JOptionPane.INFORMATION_MESSAGE);
        } catch (Exception e) {
            System.err.println("Caught exception " + e.toString());
            JOptionPane.showMessageDialog(null, "Algun archivo no esta correcto",
"Aviso", JOptionPane.INFORMATION_MESSAGE);
        }

    }

}
```

BOTÓN:

```
        private void jMenuItem4ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        /**FormVerificar view = new FormVerificar();
        view.setVisible(true);**/
        verificar verificacion = new verificar();
// Tiempo inicial del sistema en milisegundos
                long startTime = System.currentTimeMillis();
        String ruta ="C:\\FirmasDigitalesEnImagenes\\Resultados-RGB.txt";
        String clvpp = "C:\\FirmasDigitalesEnImagenes\\clv-publica-img.txt";
        String firr = "C:\\FirmasDigitalesEnImagenes\\firma-img.txt";
        if (ruta.length() == 0 || clvpp.length() == 0 || firr.length() == 0) {
        } else {
            verificacion.verificar(clvpp, firr, ruta);
        }
// Tiempo final del sistema en milisegundos
                long endTime = System.currentTimeMillis();
        // Se nuestra el resultado
                System.out.println("Tiempo MIlisegundos : " + (endTime - startTime));
```

```
            int num = (int) (endTime - startTime), seg, min,hor;
            seg=num/1000;
            min= (int)(seg /60);
            hor= (int)(seg/3600);
JOptionPane.showMessageDialog(this, hor+"h "+"- "+min+"m "+"- "+seg+"s "+"-
"+(endTime-startTime)+"ms ");

   }
```

5.2.1.3 DISEÑO FINAL

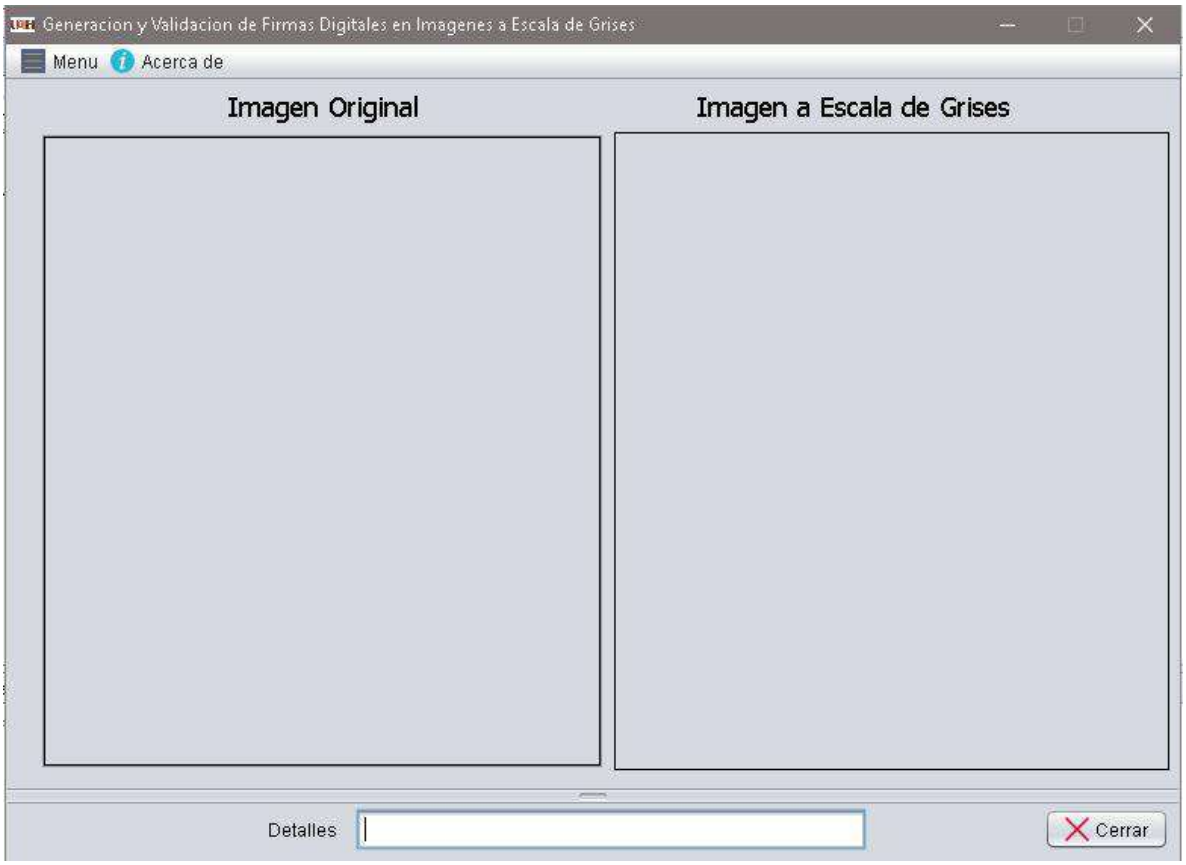


Imagen 19. Diseño Final de la Aplicación

5.2.1.4 EXPLICACIÓN PARA LA FIRMA Y VALIDACIÓN CON EL ALGORITMO DE FIRMA DIGITAL (DSA)

Genera Claves Publicas y Privadas.

Para poder crear una firma digital, se necesita una clave privada. (Se necesitará su clave pública correspondiente para verificar la autenticidad de la firma).

En algunos casos, el par de claves (clave privada y clave pública correspondiente) ya están disponibles en los archivos. En ese caso, el programa puede importar y usar la clave privada para firmar.

En otros casos, el programa necesita generar el par de claves. Se genera un par de claves utilizando la clase ***KeyPairGenerator*.**

Generará un par de claves públicas / privadas para el algoritmo de firma digital (DSA), las claves generadas serán de una longitud de 1024 bits.

Generar un par de claves requiere varios pasos:

Crear un generador de pares de llaves

El primer paso es obtener un objeto generador de pares de claves para generar claves para el algoritmo de firma DSA.

Como con todas las clases de motores, la forma de obtener un objeto ***KeyPairGenerator*** para un tipo particular de algoritmo es llamar al método ***getInstance*** factory factory en la clase ***KeyPairGenerator***. Este método tiene dos formas, ambas tienen un primer argumento de Algoritmo de cadena; una forma también tiene un segundo argumento del proveedor de cadenas.

Una persona que llama puede opcionalmente especificar el nombre de un proveedor, lo que garantizará que la implementación del algoritmo solicitado provenga del proveedor designado. El código de muestra de esta lección siempre especifica el proveedor SUN predeterminado integrado en el JDK.

Inicializar el generador de pares de claves.

El siguiente paso es inicializar el generador de pares de claves. Todos los generadores de pares clave comparten los conceptos de un tamaño de clave y una fuente de aleatoriedad. La clase ***KeyPairGenerator*** tiene un método de inicialización que toma estos dos tipos de argumentos.

El tamaño de clave para un generador de clave DSA es la longitud de la clave (en bits), que establecerá en 1024.

La fuente de aleatoriedad debe ser una instancia de la clase SecureRandom. Este ejemplo solicita uno que utiliza el algoritmo de generación de números pseudoaleatorios ***SHA1PRNG***, tal como lo proporciona el proveedor SUN incorporado. El ejemplo pasa esta instancia de SecureRandom al método de inicialización del generador de pares de claves

Genera el par de llaves.

El último paso es generar el par de claves y almacenar las claves en objetos de clave privada y clave pública.

```
/* Genera par de claves publica y privada */
KeyPairGenerator keyGen = KeyPairGenerator.getInstance("DSA", "SUN");
SecureRandom random = SecureRandom.getInstance("SHA1PRNG", "SUN");

keyGen.initialize(1024, random);

KeyPair pair = keyGen.generateKeyPair();
PrivateKey priv = pair.getPrivate();//almacena clave privada
PublicKey pub = pair.getPublic();//almacena clave publica
```

DSA - Firma & Verificación 1. Genera par de claves.

Firme los datos.

Ahora que ha creado una clave pública y una privada, está listo para firmar los datos. En este ejemplo, firmará los datos contenidos en un archivo. Obtiene el nombre de archivo de la línea de comando. Se crea (o verifica) una firma digital usando una instancia de la clase ***Signature***.

La firma de datos, generar una firma digital para esos datos, se realiza con los siguientes pasos.

Obtiene un objeto de firma.

A continuación, se obtiene un objeto ***Signature*** para generar o verificar firmas utilizando el algoritmo DSA, el mismo algoritmo para el cual el programa generó claves en el paso anterior, generar claves públicas y privadas.

Inicializar el objeto de firma.

Antes de que un objeto ***Signature*** pueda usarse para firmar o verificar, debe inicializarse. El método de inicialización para la firma requiere una clave privada. Utilice la clave privada colocada en el objeto ***PrivateKey*** llamado ***priv*** en el paso anterior.

```
/* Crear un objeto Signature e inicializarlo con la clave privada */
Signature dsa = Signature.getInstance("SHA1withDSA", "SUN");

dsa.initSign(priv);
```

DSA - Firma & Verificación 2. Crea objeto para inicializarlo con la clave privada.

Proporcione al objeto de firma los datos que se deben firmar.

Este programa usará los datos del archivo cuyo nombre se especifica como el primer (y único) argumento de la línea de comando. El programa leerá en los datos un búfer a la vez y lo suministrará al objeto ***Signature*** llamando al método de actualización.

```
/*Actualiza la firma y los datos */
FileInputStream fis = new FileInputStream(archivo);

BufferedInputStream bufin = new BufferedInputStream(fis);
byte[] buffer = new byte[1024];
int len;
while (bufin.available() != 0) {
    len = bufin.read(buffer);
    dsa.update(buffer, 0, len);
};

bufin.close();
```

DSA - Firma & Verificación 3. Actualiza la firma y los datos.

Generar la firma.

Una vez que se hayan proporcionado todos los datos al objeto ***Signature***, puede generar la firma digital de esos datos.

```
/* Ahora que todos los datos a firmar han sido leídos,
      Generar una firma para ello */
byte[] realSig = dsa.sign();
```

DSA - Firma & Verificación 4. Genera la firma.

Guarde la firma y la clave pública en los archivos.

Ahora que ha generado una firma para algunos datos, debe guardar los bytes de firma en un archivo y los bytes de clave pública en otro para poder enviar (a través de módem, disquete, correo, etc.) a otra persona.

Los datos para los cuales se generó la firma, y la clave pública.

El receptor puede verificar que los datos provienen de usted y no fueron modificados en tránsito ejecutando el programa para verificar que generará en los próximos pasos de Verificar una firma digital. Ese programa usa la clave pública para verificar que la firma recibida es la firma verdadera de los datos recibidos.

Llamada desde el paso Generar claves públicas y privadas que la clave pública se colocó en un objeto ***PublicKey*** llamado pub. Puede obtener los bytes clave codificados llamando al método ***getEncoded*** y luego almacenar los bytes codificados en un archivo. Puede nombrar el archivo como lo desee.

```
/* Guarda la firma en el archivo */
FileOutputStream sigfos = new FileOutputStream("C:\\FirmasDigitalesEnImagenes\\firma-img.txt");
sigfos.write(realSig);

sigfos.close();


/* Guarda la clave publica en el archivo */
byte[] key = pub.getEncoded();
FileOutputStream keyfos = new FileOutputStream("C:\\FirmasDigitalesEnImagenes\\clv-publica-img.txt");
keyfos.write(key);

keyfos.close();
```

DSA - Firma & Verificación 5. Guarda la firma y clave publica en archivo .txt.

Se ingresa y se convierten los bytes de clave pública codificada.

A continuación, para verificar necesita importar los bytes de clave pública codificados del archivo especificado como el primer argumento de línea de comando y convertirlos a ***PublicKey***. A ***PublicKeyes*** necesario porque eso es lo que el ***Signature initVerify*** método requiere para inicializar el ***Signatureobjeto*** para la verificación.

Primero, se leen los bytes de clave pública codificada.

Ahora la matriz de bytes ***encKeycontiene*** los bytes de clave pública codificados. Puede usar una ***KeyFactoryclase*** para crear una instancia de una clave pública de DSA a partir de su codificación. La ***KeyFactoryclase*** proporciona conversiones entre claves opacas (de tipo Key) y especificaciones de clave, que son representaciones transparentes del material clave subyacente. Con una clave opaca puede obtener el nombre del algoritmo, el nombre del formato y los bytes clave codificados, pero no el material clave, que, por ejemplo, puede consistir en la clave misma y los parámetros del algoritmo utilizados para calcular la clave. (Tenga en cuenta que ***PublicKey***, como se extiende Key, es en sí mismo Key).

Entonces, primero necesitas una especificación de clave. Puede obtener uno a través de lo siguiente, suponiendo que la clave se codificó de acuerdo con el estándar X.509, que es el caso, por ejemplo, si la clave se generó con el generador de pares de claves DSA incorporado suministrado por el proveedor SUN.:

Ahora necesitas un ***KeyFactoryobjeto*** para hacer la conversión. Ese objeto debe ser uno que funcione con claves DSA.

Finalmente, puede usar el ***KeyFactoryobjeto*** para generar a ***PublicKeypartir*** de la especificación de la clave.

Ingrese los bytes de la firma.

A continuación, ingrese los bytes de la firma desde el archivo especificado como el segundo argumento de línea de comando.

Ahora la matriz de bytes ***sigToVerifycontiene*** los bytes de la firma.

```
/* import encoded public key */
FileInputStream keyfis = new FileInputStream(clavep);
byte[] encKey = new byte[keyfis.available()];
keyfis.read(encKey);

keyfis.close();

X509EncodedKeySpec pubKeySpec = new X509EncodedKeySpec(encKey);

KeyFactory keyFactory = KeyFactory.getInstance("DSA", "SUN");
PublicKey pubKey = keyFactory.generatePublic(pubKeySpec);

/* input the signature bytes */
FileInputStream sigfis = new FileInputStream(firma);
byte[] sigToVerify = new byte[sigfis.available()];
sigfis.read(sigToVerify);

sigfis.close();
```

DSA - Firma & Verificación 6. Importa clave.

Verificar la firma

Inicializar el objeto de firma para la verificación

Al igual que con la generación de firmas, una firma se verifica mediante el uso de una instancia de la ***Signatureclase***. Debe crear un ***Signatureobjeto*** que utilice el mismo algoritmo de firma que se utilizó para generar la firma. El algoritmo utilizado fue el algoritmo SHA1withDSA del proveedor SUN.

Luego, necesitas inicializar el ***Signatureobjeto***. El método de inicialización para la verificación requiere la clave pública.

Suministrar el objeto de firma con los datos que se verificarán

Ahora debe proporcionar al ***Signatureobjeto*** los datos para los que se generó una firma. Estos datos están en el archivo cuyo nombre se especificó como el tercer

argumento de línea de comando. Como lo hizo al firmar, lea los datos de un búfer a la vez y entréguelo al ***Signatureobjeto*** llamando al ***updatemétodo***.

Verificar la firma

Una vez que haya suministrado todos los datos al ***Signatureobjeto***, puede verificar la firma digital de esos datos e informar el resultado. Recuerde que la supuesta firma se leyó en una matriz de bytes llamada ***sigToVerify***.

El ***verifiesvalor*** será true si la supuesta firma (***sigToVerify***) es la firma real del archivo de datos especificado generado por la clave privada correspondiente a la clave pública ***pubKey***. (Algoritmo, s.f.)

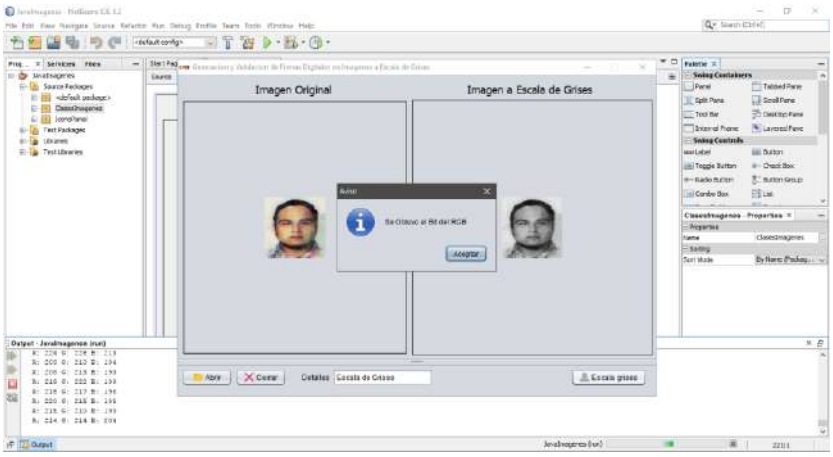
```
/* Update and verify the data */
FileInputStream datafis = new FileInputStream(archivo);
BufferedInputStream bufin = new BufferedInputStream(datafis);

byte[] buffer = new byte[1024];
int len;
while (bufin.available() != 0) {
    len = bufin.read(buffer);
    sig.update(buffer, 0, len);
};

bufin.close();

boolean verifies = sig.verify(sigToVerify);

System.out.println("signature verifies: " + verifies);
```

DSA - Firma & Verificación 7. Actualiza y verifica los datos.

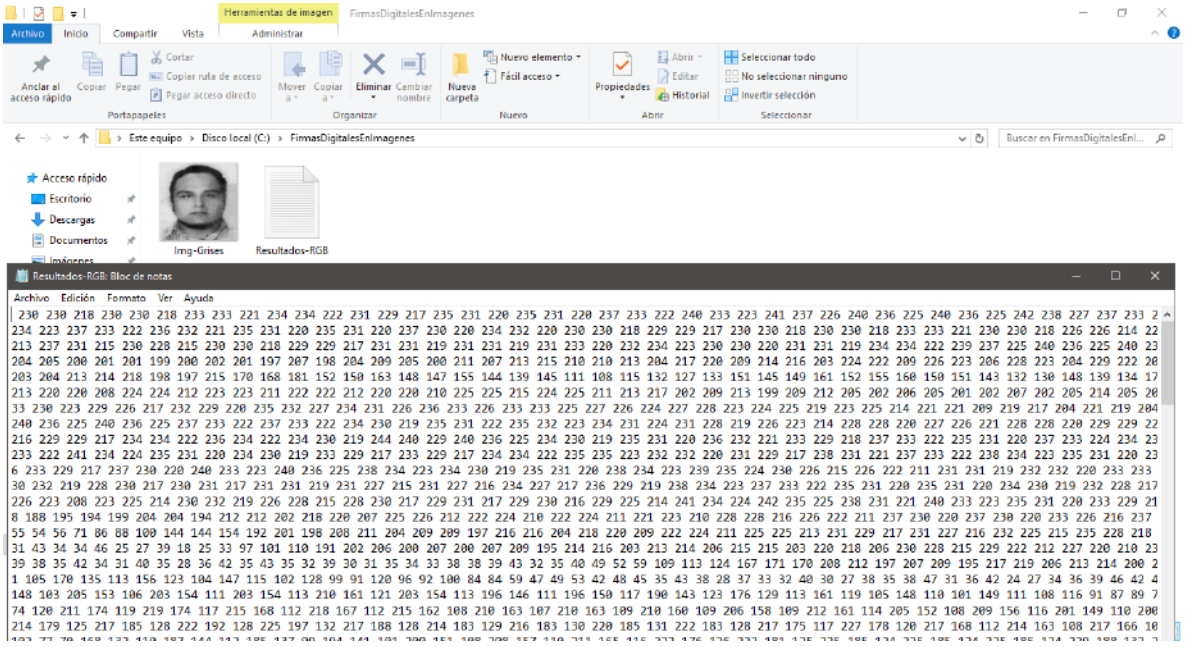
5.2.2 RESULTADOS OBTENIDOS

5.2.2.1 PRUEBA 1, IMAGEN DE IFE-KEVING AL 100X100PX

Prueba 1. IFE-KEVING 100px x 100px



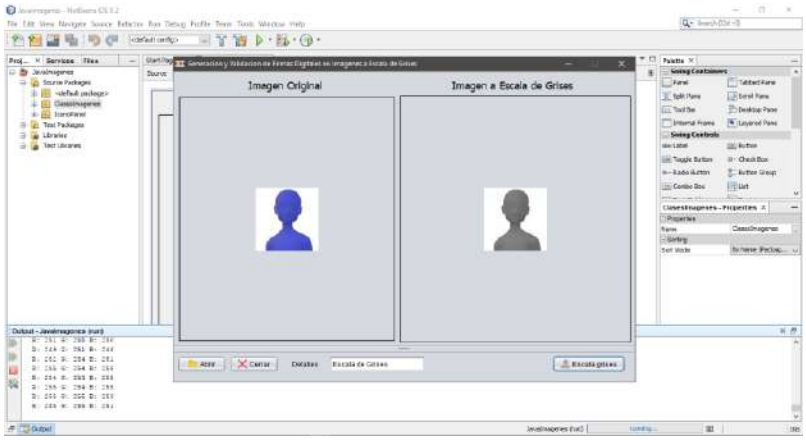
Prueba 2. IFE-KEVING 100px x 100px, escala de gris



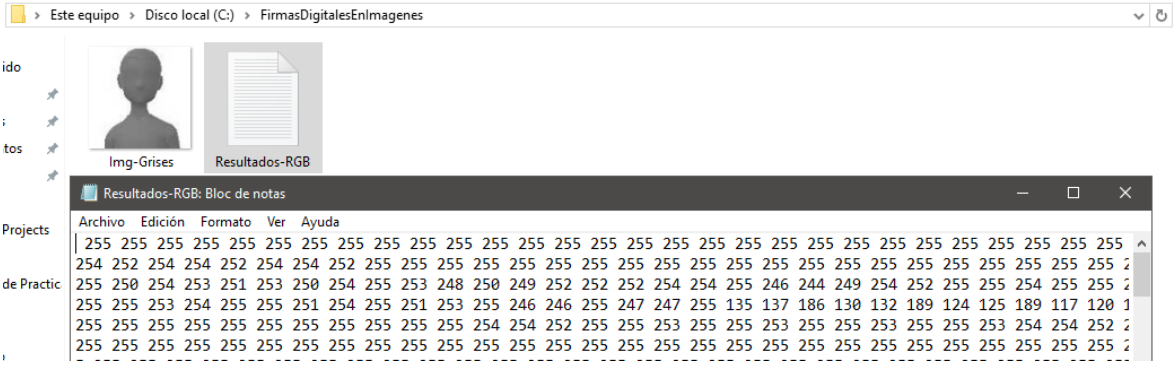
Prueba 3. IFE-KEVING 100px x 100px, Obtenido valores entre 0-255

5.2.2.2 PRUEBA 2, IMAGEN DE COLOR SOLIDO 107X107PX

Prueba 4. COLOR SOLIDO 107px x 107px

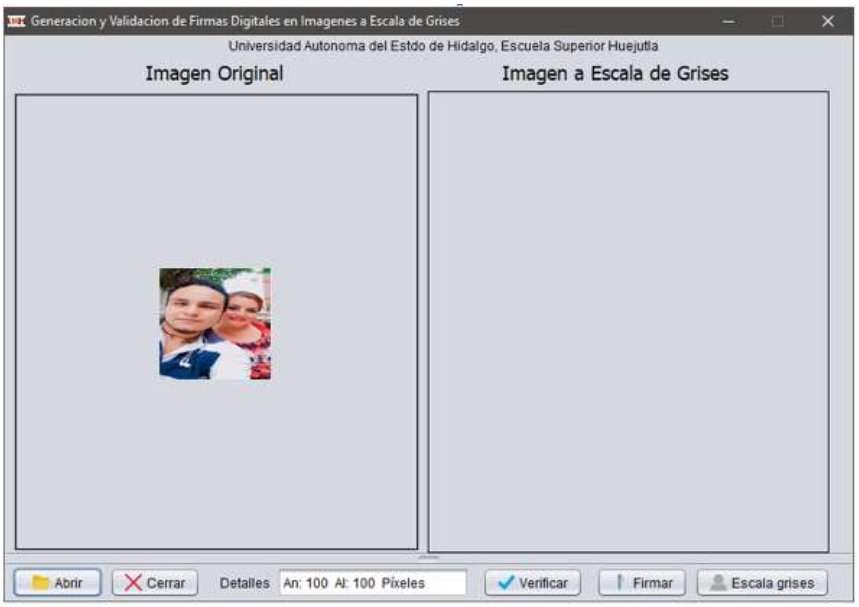


Prueba 5. COLOR SOLIDO 107px x 107px, Escala de gris



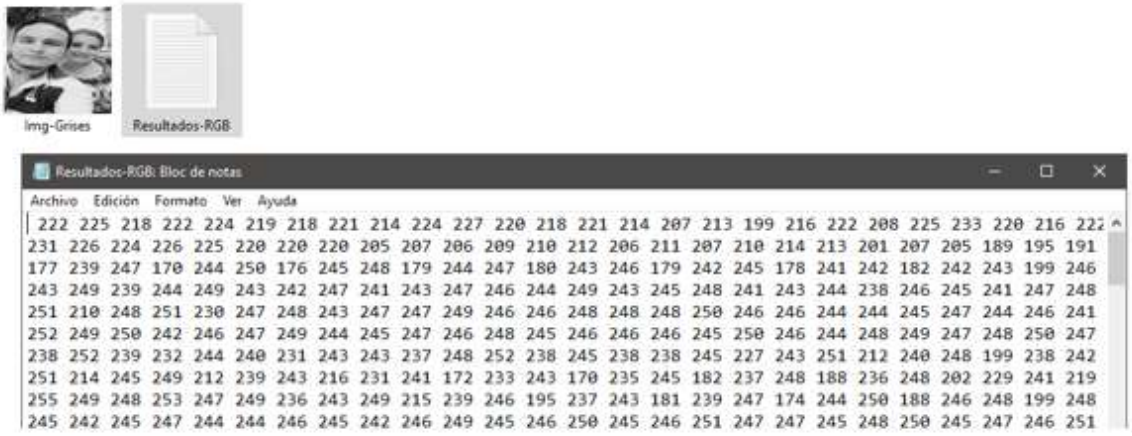
Prueba 6. COLOR SOLIDO 107px x 107px, Obtenido Valores entre 0-255

5.2.2.3 PRUEBA 3, IMAGEN FOTO AL 100X100



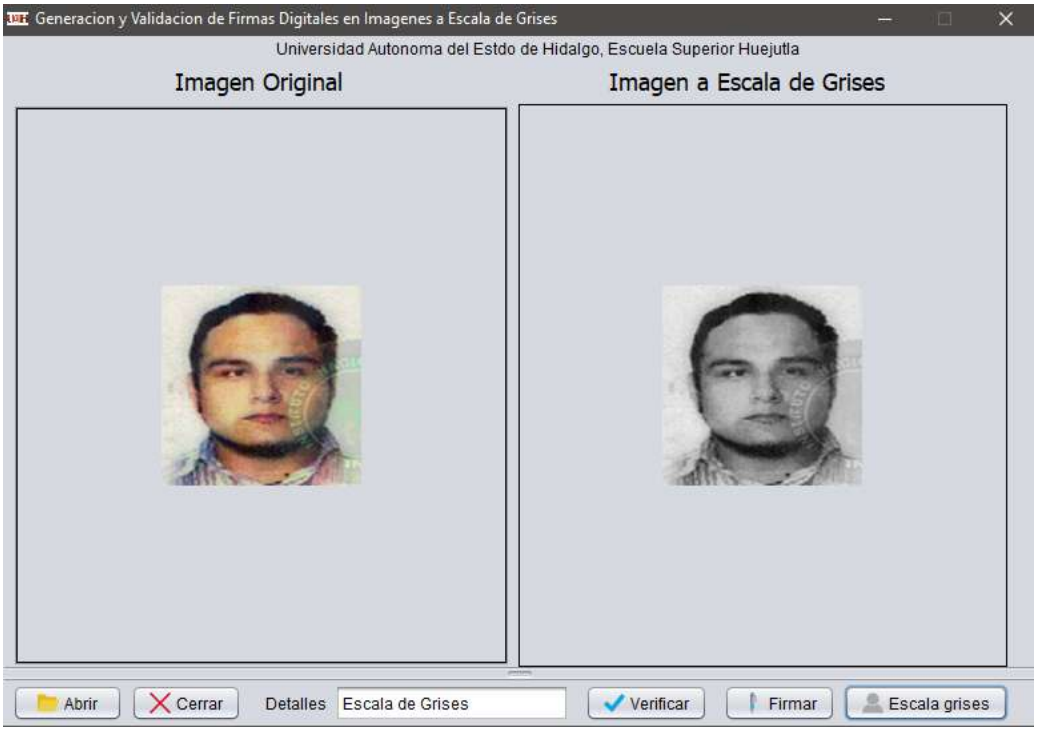
Prueba 7. FOTO 100px x 100px

Prueba 8. FOTO 100px x 100px, Escala de gris

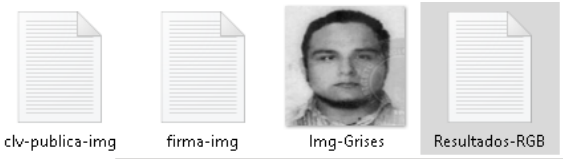


Prueba 9. FOTO 107px x 107px, Obtenido valores entre 0-255

5.2.2.4 PRUEBA 4, IMAGEN IFE-KEVING AL 150X150 PX



Prueba 10. IFE-KEVING 150px x 150px, escala de gris



Resultados-RGB: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
230 230 218 230 230 218 230 230 218 233 233 221 234 234 222 233 233 221 232 230 218 232
218 208 210 213 204 211 214 205 213 216 207 215 220 213 217 224 216 224 225 217 223 225 2
221 235 231 222 237 233 224 240 235 229 240 235 229 238 234 225 237 233 222 240 236 225 2
236 230 216 234 231 216 230 226 214 231 229 217 231 231 219 231 231 219 232 232 220 234 2
226 212 229 230 216 233 234 220 236 236 224 237 237 225 232 232 220 227 228 214 224 225 2
225 241 237 226 240 236 225 240 233 223 238 231 221 235 231 220 236 232 221 236 234 222 2
203 206 189 202 205 188 195 198 187 196 199 192 204 207 198 206 209 200 204 207 200 202 2
230 218 227 227 215 225 225 213 229 229 217 230 230 218 231 231 219 233 233 221 234 234 2
183 204 200 197 202 198 195 210 206 205 214 213 211 210 210 208 203 208 202 202 207 201 2
236 232 221 237 233 222 237 233 222 239 235 226 238 234 225 238 233 227 235 230 224 234 2
209 224 222 207 221 220 202 224 223 203 226 225 207 223 224 210 225 225 213 229 229 217 2
227 227 215 221 221 209 220 220 208 223 224 210 224 224 212 223 223 211 222 222 212 221 2
232 221 235 231 220 234 230 219 236 229 219 236 227 218 236 227 218 233 229 218 233 231 2
```

Prueba 11. IFE-KEVING 150px x 150px, Obtenido valores entre 0-255

5.2.3 TRABAJOS FUTUROS

En este proyecto se pretende de acuerdo a este prototipo de sistema de **Generación de Firmas Digitales en Imágenes a Escala de Grises**, implementarlo para su mejor funcionamiento, algunas mejoras son:

1. Integrar un botón que abra una cámara y esta a su vez capture una foto en un tamaño cualquiera.

2. En el momento de generar o verificar la firma de la imagen, que se agregue una contraseña que realice dichos procesos, por el cual solo el usuario único ara uso de esa contraseña.

3. Que al inicio del sistema pida un nombre de usuario solo para llevar un control cada vez que hagan uso del sistema.

4. Que genere un reporte semanal o mensual con nombre de usuario, fecha, hora y numero de trabajo por cada vez que hagan uso del sistema.

5. Que este prototipo de sistema sirva como una guía para implementarlo en aplicaciones móviles.

5.2.4 EVALUACIÓN

Num.	Nombre	Tamaño	Peso	Especificaciones
1	IMG-IFE-KEVING	100x100px	29.8 KB	Carga la imagen a la perfección, también la procesa a escala de grises sin ningún problema, así como también guarda la lista de bits correspondientes a cada pixel.
2	IMG-FOTO	100x100px	29.1 KB	Carga la imagen a la perfección, también la procesa a escala de grises sin ningún problema, así como también guarda la lista de bits correspondientes a cada pixel.
3	Usuario	107x107px	3.58 KB	Carga la imagen a la perfección, también la procesa a escala de grises sin ningún problema, así como también guarda la lista de bits correspondientes a cada pixel.
4	IMG-IFE-KEVING-150px	150x150px	34.2 KB	Carga la imagen a la perfección, también la procesa a escala de grises sin ningún problema, así como también guarda la lista de bits correspondientes a cada pixel.
5	IMG-FOTO-150px	150x150px	39.0 KB	Carga la imagen a la perfección, también la procesa a escala de grises sin ningún problema, la lista de bits correspondientes a cada pixel la guarda en chino.

Núm.	Proceso	Tiempo		Especificaciones
		De	Hasta	
1	Proceso a Escala Gris	38,69 Seg.	44, 48 Seg.	Aproximadamente
	Guardar Imagen a Escala de Gris	Al Instante		
	Generar Firma	,90 Milisegundos	,99 Milisegundos	Aproximadamente
	Verificar Firma	,90 Milisegundos	,99 Milisegundos	Aproximadamente

Núm.	Proceso	Tiempo		Especificaciones
		De	Hasta	
2	Proceso a Escala Gris	31,39 Seg.	28,53 Seg.	Aproximadamente
	Guardar Imagen a Escala de Gris	Al instante		
	Generar Firma	,90 Milisegundos	,99 Milisegundos	Aproximadamente
	Verificar Firma	,90 Milisegundos	,99 Milisegundos	Aproximadamente

Núm.	Proceso	Tiempo		Especificaciones
		De	Hasta	
3	Proceso a Escala Gris	37,95 Seg.	35,9 Seg.	Aproximadamente
	Guardar Imagen a Escala de Gris	Al instante		
	Generar Firma	,90 Milisegundos	,99 Milisegundos	Aproximadamente
	Verificar Firma	,90 Milisegundos	,99 Milisegundos	Aproximadamente

Núm.	Proceso	Tiempo		Especificaciones
		De	Hasta	
4	Proceso a Escala Gris	2 Min, 34 Seg	2 Min, 45 Seg	Aproximadamente
	Guardar Imagen a Escala de Gris	Al instante		
	Generar Firma	,90 Milisegundos	,99 Milisegundos	Aproximadamente
	Verificar Firma	,90 Milisegundos	,99 Milisegundos	Aproximadamente

Núm.	Proceso	Tiempo		Especificaciones
		De	Hasta	
5	Proceso a Escala Gris	2 Min, 21 Seg.	2 Min, 33 Seg.	Aproximadamente
	Guardar Imagen a Escala de Gris	Al instante		
	Generar Firma	,90 Milisegundos	,99 Milisegundos	Aproximadamente
	Verificar Firma	,90 Milisegundos	,99 Milisegundos	Aproximadamente

6 CONCLUSIÓN

Como objetivo de mantener una mejor seguridad en la integridad de la información, se pensó en realizar este proyecto haciendo uso de las tecnologías actuales y que gracias a estos se pueda desarrollar sistemas que permitan controlar el acceso de personas mediante las firmas digitales en imágenes, este sistema tiene la capacidad firmar una imagen de un tamaño mínimo de 100x100px a un tiempo en proceso de grises de 30 segundos, y firmando-verificando a un tiempo de 90 milisegundos, hasta un tamaño de 150px a un tiempo de proceso de grises de 2min con 21 segundos y firmando-verificando a un tiempo de 90 milisegundos, esto no es exactamente el tiempo es un aproximado ya que varía por segundos, pero mientras más grande sea el tamaño de la imagen el tiempo de proceso de grises aumente muy considerablemente mientras que al firmar-verificar el tiempo se mantiene a menos de 1 segundo.

Para llevar a cabo el funcionamiento de este sistema se realizaron múltiples investigaciones para poder comprender los procesos de escala a grises y firma-verificación y el funcionamiento de la implementación del **Algoritmo de Firma Digital (DSA)**. Toda investigación documentada fue de mucha ayuda para llevar a cabo este proyecto porque de esas investigaciones se sustentaron los resultados que se obtuvieron durante las etapas.

Tanto para el algoritmo como para las pruebas se utilizaron imágenes de color solido e imágenes normales con múltiples colores, estas imágenes fueron de diferentes dimensiones variando su peso, también se fueron capturando los tiempos de cada proceso en las diferentes imágenes.

Al utilizar imágenes de color solido e imágenes normales con múltiples colores y de dimensiones y pesos distintos, ayudo a realizar una comparativa de cuánto tiempo como mínimo y máximo tarde en procesar cada imagen a gris y posteriormente cuanto tiempo tarda en firmar-verificar.

En conclusión, puedo decir con certeza que las pruebas son buenas satisfactorias ya que me dieron a conocer diferentes tiempos en los distintos procesos, obteniendo el resultado esperado con la aplicación.

Si se deseara mejorar este sistema es importante que se tenga en cuenta los procesos fundamentales, carga de imagen, proceso de grises, lectura de pixeles (extracción de valores entre 0-255), firma de la imagen, verificación de la imagen, también en la generación única de dos archivos que son la clave pública y la firma. Ya que si le hace falta uno de estos procesos no será posible dar seguimiento al resultado esperado, porque cada proceso va dela mano del que le lleva la delantera.

7 BIBLIOGRAFÍA

Algoritmo, A. (s.f.). Obtenido de http://leo.ugr.es/elvira/devel/Tutorial/Java/security1.2/apisign/step2.html

Android, A. (11 de 12 de 2014). *Academia Android*. Obtenido de https://academiaandroid.com/android-studio-v1-caracteristicas-comparativa-eclipse/

EcuRed. (14 de 08 de 2017). *EcoRed*. Obtenido de https://www.ecured.cu/Modelo_Espiral

Fotolia. (13 de 07 de 2011). *Fotolia*. Obtenido de https://blog.fotolia.com/es/2011/07/13/historia-tecnica-de-la-fotografia-la-era-digital-5-5/

Gonzales, R. C. (s.f.). Obtenido de http://web.ipac.caltech.edu/staff/fmasci/home/astro_refs/Digital_Image_Processing_3rd Ed_truncated.pdf

Hidalgo, P. C. (s.f.). Obtenido de http://tesis.ipn.mx/jspui/bitstream/123456789/7376/1/ice%20237.pdf

J.J. Baez Rojas, M. G. (s.f.). *Segmentacion de Imagen de Color*. Obtenido de http://www.ejournal.unam.mx/rmf/no506/RMF50605.pdf

Malacara, D. (s.f.). *Optica Tradicional y Moderna*. Obtenido de http://bibliotecadigital.ilce.edu.mx/sites/ciencia/volumen2/ciencia3/084/htm/sec_9.htm

Manzano, C. M. (s.f.). Obtenido de http://jupiter.utm.mx/~tesis_dig/10726.pdf

Menéndez, R. I. (s.f.). *Android 100%*. Obtenido de https://mega.nz/#!agVjyJAS!_MfO-NopphsIuJ4mu-Udk3nVFEDoxKP7JW_gaQah-Lk

Modelo Espial. (08 de 08 de 2009). Obtenido de http://webcache.googleusercontent.com/search?q=cache:http://modeloespiral.blogspot.com/&gws_rd=cr&ei=BKCRWZTmH4rXjwTJrLjoBg

Morales, L. A. (s.f.). *DSA*. Obtenido de http://es.calameo.com/read/000578884c567a67a7b98

Nacion, L. (26 de Febrero de 2013). *Firma digital se usa en una de cada tres entidades*. Obtenido de http://www.nacion.com/archivo/Firma-digital-usa-entidades_0_1326067448.html

NetBeans. (2017). Obtenido de https://netbeans.org/index_es.html

Nuevo, M. G. (s.f.). *Procesamiento y Visualización*. Obtenido de http://www.elai.upm.es/webantigua/spain/Investiga/GCII/personal/mgarcia/Procesamiento.pdf

Palomino, D. N. (s.f.). *Tecnicas de Segmentacion de Imagenes Digitales* . Obtenido de http://sisbib.unmsm.edu.pe/BibVirtual/Publicaciones/risi/2009_n2/v6n2/a02v6n2.pdf

Ramírez, D. B. (08 de 2006). *Procesamiento Digital de Imagenes* . Obtenido de http://verona.fi-p.unam.mx/boris/teachingnotes/Introduccion.pdf

Rubio, R. L. (2011). Obtenido de https://riunet.upv.es/bitstream/handle/10251/13863/Proyecto%20Rub%C3%A9n%20Llobregat.pdf?sequence=1

Ruíz E. María E., M. S. (s.f.). *RIEE&C*. Obtenido de http://www.itson.mx/publicaciones/rieeyc/Documents/v9/art1vf_una_estrategia_de_segmentacion_de_imagenes_digitales_de_huellas_dactilares_latentes.pdf

S/N. (s.f.). Obtenido de http://alojamientos.us.es/gtocoma/pid/tema4.pdf

Salvador, L. G. (s.f.). *Aplicaciones del tratamiento inteligente de imagenes*. Obtenido de http://eprints.ucm.es/9773/1/Memoria.pdf

Santiago, C. A. (10 de mayo de 2005). Obtenido de http://www.revista.unam.mx/vol.6/num5/art50/may_art50.pdf

Santiago, C. A. (10 de Mayo de 2005). Obtenido de http://www.revista.unam.mx/vol.6/num5/art50/may_art50.pdf

Wainschenker, D. R. (s.f.). *Procesamiento Imagenes Digitales*. Obtenido de http://www.exa.unicen.edu.ar/catedras/pdi/FILES/TE/CP1.pdf

8 ANEXOS

Universidad Autónoma del Estado de Hidalgo, Escuela Superior Huejutla

Generación & Validación de Firmas Digitales en Imágenes

Keving Alberto Ramírez Barragán

Documentación		Agosto				Septiembre				Octubre				Noviembre			
Fase Inicio (20%)		Semana 1	Semana 2	Semana 3	Semana 4	Semana 1	Semana 2	Semana 3	Semana 4	Semana 1	Semana 2	Semana 3	Semana 4	Semana 1	Semana 2	Semana 3	Semana 4
1 Introducción 2%	P																
	R																
	%	2%															
2 Estado de Arte 2%	P																
	R																
	%	2%															
3 Marco Teórico 2%	P																
	R																
	%	1%	1%														
4 Marco Metodológico 3%	P																
	R																
	%		1%	2%													
5 Desarrollo 5%	P																
	R																
	%			0.50%	0.50%	0.50%	0.50%	0.50%	0.50%	0.50%	0.50%	0.50%	0.50%				
6 Resultados 2%	P																
	R																
	%												1%	1%			
7 Conclusión 2%	P																
	R																
	%														2%		
8 Bibliografía 1%	P																
	R																
	%															1%	
9 Anexos 1%	P																
	R																
	%															1%	

Desarrollo Espiral		Agosto				Septiembre				Octubre				Noviembre			
Fase Alfa (30%)		Semana 1	Semana 2	Semana 3	Semana 4	Semana 1	Semana 2	Semana 3	Semana 4	Semana 1	Semana 2	Semana 3	Semana 4	Semana 1	Semana 2	Semana 3	Semana 4
1 Objetivos 5%	P																
	R																
	%	5%															
2 Análisis de Riesgo 10%	P																
	R																
	%	5%	5%	5%													
3 Desarrollar, verificar y validar. 30%	P																
	R																
	%			3.75%	3.75%	3.75%	3.75%	3.75%	3.75%	3.75%	3.75%						
4 Planificar 5%	P																
	R																
	%											5%					
Fase Beta (30%)																	
1 Objetivos 5%	P																
	R																
	%												5%				
2 Análisis de Riesgo 5%	P																
	R																
	%												2.50%	2.50%			
3 Desarrollar, verificar & validar 15%	P																
	R																
	%													5%	5%	5%	
4 Planificar 5%	P																
	R																
	%																5%